kolektiv autorů

Linux

Dokumentační projekt

Computer Press
Praha
2001

EDICE

OPERAČNÍ SYSTÉMY

2. aktualizované vydání

PROFI

Podrobná dokumentace
k operačnímu systému a jeho aplikacím

LINUX

Dokumentační projekt

- Příručka uživatele
- Příručka správce operačního systému — *Update*
- Příručka správce sítě — *Update*
- Praktické návody: sítě, firewall, proxy server, e-mail, Apache, DNS, jádro, NFS, Bash, konfigurace systému, vypalování CD — *Nové*

computer press

VŠECHNY CESTY
K INFORMACÍM

Linux – Dokumentační projekt

kolektiv autorů

Druhé, aktualizované vydání.
Vydavatelství a nakladatelství Computer Press®,
Hornocholupická 22, 143 00 Praha 4, http://www.cpress.cz

ISBN 80-7226-503-2

Prodejní kód: K0506

Překlad: Jiří Veselský, Ludvík Roubíček, Marek Kocan
Odborná korektura: Pavel Janík
Jazyková korektura: Josef Novák
Vnitřní úprava: Petr Klíma
Sazba: Petr Klíma
Rejstřík: Tomáš Kuchař

Obálka: Jaroslav Novák
Komentář na zadní straně obálky: Ivo Magera
Technická spolupráce: Pavlína Bauerová
Odpovědný redaktor: Ivo Magera
Technický redaktor: Petr Klíma
Produkce: Petr Baláš

Tisk: Tiskárny Havlíčkův Brod

Tato publikace je volně dostupná na adrese www.cpress.cz/knihy/ldp2

Veškeré dotazy týkající se distribuce směřujte na:

Computer Press Brno, náměstí 28. dubna 48, 635 00 Brno-Bystrc,
tel. (05) 46 12 21 11, e-mail: distribuce@cpress.cz

Computer Press Bratislava, Hattalova 12, 831 03 Bratislava, Slovenská republika,
tel.: +421 (7) 44 45 20 48, 44 25 17 20, e-mail: distribucia@cpress.sk

Nejnovější informace o našich publikacích naleznete na adrese:
http://www.cpress.cz/knihy/bulletin.html.

Máte-li zájem o pravidelné zasílání bulletinu do Vaší e-mailové schránky, zašlete nám jakoukoli, i prázdnou zprávu na adresu bulletin@cpress.cz.

Obsah

ČÁST I

Příručka uživatele **1**

Poděkování 1
Typografické konvence použité v této části 2

kapitola 1

Úvod **3**

Kdo by si měl přečíst tuto knihu 3
Kdy se můžete čtení této knihy vyhnout 4
Jak číst tuto knihu 4
Dokumentace k operačnímu systému Linux 5
Operační systémy 6

kapitola 2

Co je to Unix **7**

Historie operačního systému Unix 7
Historie operačního systému Linux 8

kapitola 3

Začínáme pracovat s operačním systémem Linux **13**

Jak Linux přebírá kontrolu nad počítačem 14
Činnost uživatele 16
Hlášení jádra systému 18

kapitola 4

Příkazový interpret operačního systému Unix **23**

Příkazy operačního systému Unix 23
Pomozte si sami 25
Ukládání informací 26
Manipulace se soubory 30

kapitola 5

Systém X Window **35**

Spuštění a ukončení systému X Window 35
Co je systém X Window? 36
Co se nachází na pracovní ploše X Window 36
Správce oken 38
Atributy systému X Window 40
Společné vlastnosti 41

kapitola 6

Práce s operačním systémem Unix **45**

Pseudoznaky 45
Jak ušetřit čas pomocí příkazu bash 47
Standardní vstup a standardní výstup 48
Současný běh úloh 50
Virtuální konzoly 55

kapitola 7

Malé a výkonné programy **57**

V čem spočívá síla operačního systému Unix 57
Práce se soubory 57
Systémová statistika 58
Co obsahují soubory 60
Informační příkazy 61

kapitola 8

Editování souborů pomocí editoru Emacs **64**

Co je to Emacs? 64
Používání editoru pod systémem X Window 67
Editování více souborů současně 68
Ukončení práce s editorem 69
Klíče Meta 69
Práce s bloky textu 70
Vyhledávání a náhrada řetězců 71
Vnitřní funkce editoru Emacs72
Nápověda v editoru Emacs 73
Pracovní módy editoru Emacs 73
Programovací módy 74
Jak zvýšit efektivitu práce s editorem Emacs 76
Konfigurace editoru Emacs 76
Kde získat další informace 81

kapitola 9

Konfigurace operačního systému Unix **81**

Konfigurace příkazového interpretu bash 81
Inicializační soubory systému X Window 90
Ostatní inicializační soubory 97
Kde si můžete prohlédnout některé příklady 99

kapitola 10

Komunikace s ostatními systémy **101**

Elektronická pošta 101
Jak vyhledat uživatele sítě 103
Používání systémů vzdálenými počítači 104
Přenášení souborů 105
Putování po stránkách WWW 105

kapitola 11

Zábavné příkazy **107**
Příkaz find 107
Archivační program tar 113
Program dd 119

kapitola 12

Chyby, skryté závady a další nepříjemnosti **123**
Jak předcházet chybám 123
Chyba není ve vás 124

příloha A

Technické informace **127**
Stručná historie editoru vi 127
Stručný výklad příkazů editoru Ed 128
Stručný výklad příkazů editoru vi 130
Pokročilejší techniky práce s editorem vi 132
Kopírování a přesouvání částí textu 137
Vyhledávání a nahrazování textů 139

ČÁST II

Příručka správce operačního systému **143**
Zdrojová podoba a předformátované verze 143

kapitola 1

Úvod **145**
Linux – dokumentační projekt 147

kapitola 2

Přehled operačního systému Linux **149**
Důležité části jádra systému 150
Nejdůležitější služby v unixovém systému 151

Kapitola 3

Přehled adresářové struktury **155**
Základy 155
Souborový systém `root` 157
Souborový systém `/usr` 160
Souborový systém `/var` 161
Souborový systém `/proc` 162

kapitola 4

Disky a jiná média **165**
Dva druhy zařízení 165
Pevné disky 166
Diskety 169
Jednotky CD-ROM 170
Pásky 171

Formátování 171
Diskové oblasti 173
Souborové systémy 177
Disky bez souborových systémů 189
Přidělování diskového prostoru 189

Kapitola 5

Správa paměti 193
Co je virtuální paměť? 193
Vytvoření odkládacího prostoru na disku 194
Využívání odkládacího prostoru 195
Sdílení odkládacího prostoru s jinými operačními systémy 196
Přidělování odkládacího prostoru 196
Vyrovnávací paměť 197

Kapitola 6

Spouštění a zastavování systému 199
Zavádění a ukončení práce systému – přehled 199
Zavádění podrobněji 200
Podrobněji o zastavení systému 202
Znovuzavedení systému 203
Jednouživatelský režim 204
Záchranné zaváděcí diskety 204

Kapitola 7

init 205
Proces init přichází první 205
Konfigurace procesu init pro spouštění programu getty – soubor /etc/inittab 206
Úrovně běhu systému 207
Zvláštní konfigurace v souboru /etc/inittab 208
Zavádění systému v jednouživatelském režimu 208

Kapitola 8

Přihlašování a odhlašování 211
Přihlašování přes terminály 211
Přihlášení prostřednictvím sítě 213
Co dělá program login 214
X a xdm 214
Řízení přístupu 214
Spouštění interpretu příkazů 215

Kapitola 9

Správa uživatelských účtů 217
Co je to účet? 217
Vytváření uživatelských účtů 217
Změny vlastností uživatelských účtů 220
Zrušení uživatelského účtu 220
Dočasné zablokování uživatelského účtu 221

Kapitola 10

Zálohování **223**

Jak je důležité míti zálohy 223
Výběr média pro zálohování 224
Výběr nástroje pro zálohování 224
Jednoduché zálohování 225
Víceúrovňové zálohování 228
Co zálohovat 229
Komprimované zálohy 230

Kapitola 11

Časové údaje **231**

Časové zóny 231
Hardwarové a softwarové hodiny 232
Zobrazení a nastavování času 232
Co když jdou hodiny špatně 233

Slovníček **235**

ČÁST III

Příručka správce sítě **237**

Zdrojová podoba a předformátované verze 237

Úvod **239**

Účel a orientace této knihy 240
Zdroje informací 240
Standardy souborového systému 244
Linux Standard Base 245
O této knize 245
Oficiální tištěná verze 246
Přehled 247
Konvence používané v této části 248
Oznamování změn 249
Poděkování 249

Kapitola 1

Úvod do sítí **251**

Historie 251
Sítě TCP/IP 252
Sítě UUCP 260
Sítě v Linuxu 260
Údržba vašeho systému 262

Kapitola 2

Problematika sítí TCP/IP **265**

IP adresy 265
Rozlišování adres 266
Směrování protokolu IP 267
Protokol ICMP 272
Rozlišení jmen hostitele 273

Kapitola 3

Konfigurace síťového hardware **275**

Konfigurace jádra 277
Prohlídka síťových zařízení v Linuxu 283
Instalace Ethernetu 284
Ovladač PLIP 286
Ovladače SLIP a PPP 288
Další typy sítí 288

Kapitola 4

Konfigurace sériových zařízení **289**

Komunikační software pro modemové linky 289
Úvod k sériovým zařízením 290
Přístup k sériovým zařízením 290
Sériový hardware 293
Použití konfiguračních nástrojů 293
Sériová zařízení a výzva login: 297

Kapitola 5

Konfigurace sítí TCP/IP **301**

Připojování souborového systému /proc 30
Instalace binárních souborů 302
Nastavení jména hostitele 303
Přiřazení IP adresy 303
Vytváření podsítí 304
Vytvoření souborů hosts a networks 305
Konfigurace rozhraní pro protokol IP 307
Vše o příkazu ifconfig 314
Příkaz netstat 316
Kontrola tabulek ARP 318

Kapitola 6

Služby jmen a konfigurace resolveru **321**

Knihovna resolveru 321
Jak DNS funguje 327
Provozování programu named 333

Kapitola 7

Linka SLIP **347**

Použití protokolu SLIP 347
Práce s privátními IP sítěmi 349
Použití nástroje dip 350
Spuštění v režimu serveru 355

Kapitola 8

Protokol PPP **357**

Nastavení konfigurace protokolu IP 362
Obecné bezpečnostní úvahy 366
Autentikace protokolem PPP 367
Složitější konfigurace protokolu PPP 371

Kapitola 9

TCP/IP Firewall **375**

Metody útoků 376
Co je to firewall? 377
Co je to filtrování? 378
Vytvoření firewallu na Linuxu 378
Tři způsoby realizace filtrace 380
Původní IP firewall (jádra 2.0) 381
IP Firewall Chains (jádra 2.2) 387
Netfilter a IP Tables (jádra 2.4) 396
Manipulace s bity typu služby 403
Testování konfigurace firewallu 405
Příklad konfigurace firewallu 407

Kapitola 10

IP účtování **415**

Konfigurace jádra pro účtování 415
Konfigurace IP účtování 415
Vyhodnocení výsledků účtování 419
Vymazání pravidel 423
Pasivní shromažďování účtovacích dat 423

Kapitola 11

IP maškaráda a překlad síťových adres **425**

Vedlejší efekty, výhody a nevýhody 426
Konfigurace jádra pro maškarádu 427
Konfigurace maškarády 428
Obsluha dotazů na jmenné servery 430
Další informace o překladu síťových adres 430

Kapitola 12

Důležité síťové aplikace **433**

Superserver inetd 433
Řízení přístupu wrapperem tcpd 435
Soubory services a protocols 437
Vzdálené volání procedur 438
Konfigurace vzdáleného přihlašování a spouštění 440

Kapitola 13

Network Information System **447**

Seznámení se systémem NIS 448
Systém NIS versus systém NIS+ 450
NIS na straně klienta 450
Provozování serveru NIS 450
Bezpečnost serveru NIS 452
Nastavení NIS klienta pomocí GNU libc 453
Výběr vhodných map 454
Použití map passwd a group 456
NIS a stínová hesla 458

Kapitola 14

Network File System **459**

Příprava systému NFS 460
Připojení svazku systému NFS 460
Démony systému NFS 462
Soubor 463
Podpora serveru NFSv2 v jádře 465
Podpora serveru NFSv3 v jádře 465

Kapitola 15

IPX a souborový systém NCP **475**

Xerox, Novell a historie 467
IPX a Linux 468
Konfigurace jádra pro IPX a NCPFS 469
Konfigurace IPX rozhraní 470
Konfigurace IPX směrovače 472
Připojení vzdáleného svazku NetWare 475
Práce s dalšími nástroji IPX 478
Tisk do front NetWare 479

Kapitola 16

Správa programu Taylor UUCP **483**

Historie 483
Přenos souborů a vzdálené spouštění 484
Konfigurační soubory protokolu UUCP 487
Řízení přístupu k funkcím UUCP 498
Nastavení systému pro příchozí volání 501
Nízkoúrovňové protokoly protokolu UUCP 504
Hledání a odstraňování problémů 506
Logovací soubory a ladění 508

Kapitola 17

Elektronická pošta **511**

Co je to poštovní zpráva? 512
Jak se pošta doručuje? 514
Adresy elektronické pošty 515
Jak pracuje směrování pošty? 516
Konfigurace programu elm 521

Kapitola 18

Program sendmail **525**

Instalace programu sendmail 525
Konfigurační soubory – přehled 526
Soubory sendmail.cf a sendmail.mc 526
Vygenerování souboru sendmail.cf 531
Interpretace a vytváření přepisovacích pravidel 531
Konfigurace nastavení programu sendmail 535
Některá užitečná nastavení sendmailu 536
Konfigurace virtuálních poštovních hostitelů 541
Testování konfigurace 543
Spuštění programu sendmail 546
Tipy a triky 547

Kapitola 19

Nastavení a spuštění programu Exim **551**

Spuštění programu Exim 552
Když pošta nefunguje 553
Překlad programu Exim 554
Režimy doručování pošty 554
Různé konfigurační volby 555
Směrování a doručování zpráv 556
Ochrana před spammingem 559
Nastavení pro UUCP 560

Kapitola 20

Síťové news **563**

Historie Usenetu 563
Co je to vlastně Usenet? 564
Jak Usenet obsluhuje news? 565

Kapitola 21

C News **569**

Doručování news 569
Instalace 571
Soubor sys 572
Soubor active 574
Dávkové zpracování článků 575
Vypršení platnosti news 577
Různé soubory 580
Řídicí zprávy 581
C News v prostředí NFS 583
Nástroje pro údržbu 583

Kapitola 22

Protokol NNTP a démon nntpd **585**

Protokol NNTP 586
Instalace serveru NNTP 592
Omezení přístupu k NNTP serveru 592
Autorizace protokolem NNTP 593
Interakce serveru nntpd a systému C News 594

Kapitola 23

Internet News **595**

Podrobnosti o programu INN 595
Čtení zpráv a INN 597
Instalace balíku INN 597
Konfigurace programu INN: Základní nastavení 598
Konfigurační soubory balíku INN 598
Spuštění programu INN 609
Správa INN: Příkaz ctlinnd 610

Kapitola 24

Konfigurace klienta pro čtení zpráv 615
Konfigurace programu tin 615
Konfigurace programu trn 616
Konfigurace programu nn 617

Dodatek A

Příklad sítě virtuálního pivovaru 619
Připojení sítě virtuální pobočky 620

Dodatek B

Užitečná zapojení kabelů 621
Paralelní kabel PLIP 621
Null-modemový sériový kabel 622

Dodatek C

SAGE: The System Administrators Guild 623

ČÁST IV

Praktické návody 625

Kapitola 1

Linux v sítích 627
Úvod 627
Historie dokumentu 627
Jak tento dokument používat 628
Obecné informace o používání Linuxu na sítích 629
Informace k běžné konfiguraci sítě 630
Informace vztahující se k Ethernetu 653
Informace o protokolu IP 655
Transparentní proxy 665
Pokročilé síťové funkce v jádře 2.2 682
Typický PC hardware 685
Další síťové technologie 699
Kabely a kabeláž 706
Některé termíny používané v tomto dokumentu 708
Autorská práva 709

Kapitola 2

Linux Intranet Server 711
Úvod 711
Vysvětlení činnosti firewallů 712
Architektura firewallu 715
Vytvoření filtrujícího firewallu 716
Požadavky 717
Příprava linuxového systému 718
Testování sítě 721

Zabezpečení firewallu 722
Nastavení IP filtrování (IPFWADM) 723
Nastavení IP filtrování (IPCHAINS) 724
Instalace transparentního SQUID proxy serveru 726
Instalace proxy serveru TIS 726
SOCKS proxy server 730
Složitější konfigurace 734
Příklady skriptů 736
VPN RC skript pro RedHat 743

Kapitola 3

Pošta z pohledu správce 745
Úvod, autorská práva a zodpovědnost za správnost 745
Další zdroje informací 746
Jak elektronická pošta funguje 746
Požadavky 753
Volba transportního agenta 753
Instalace transportního agenta 755
Uživatelské doručovací programy 774
Vzdálená práce s poštou 775
Poděkování 778

Kapitola 4

Pošta z pohledu uživatele 779
Úvod 779
Poštovní klienti 780
Pokročilá témata 782
Další zdroje informací 784
Administrativa 786

Kapitola 5

DNS 789
Předmluva 789
Úvod 790
Caching-only DNS server 791
Další vylepšení 794
Jednoduchá doména 795
Upozornění 805
Příklad skutečné domény 806
Údržba 811
Přechod z verze 4 na verzi 8 812
Otázky a odpovědi 813
Jak se stát lepším správcem DNS 815

Kapitola 6

Jádro Linuxu 817
Úvod 817
Stručný přehled překladu jádra 818
Důležité otázky a odpovědi 822
Jak nastavit jádro 823
Překlad jádra 828
Opravy jádra 829

Další balíčky 831
Některé léčky 832
Poznámky k inovaci jádra na verze 2.0.x a 2.2.x 835
Moduly 835
Tipy a triky 837
Další užitečné dokumenty 838
Různé 838
Další formáty tohoto dokumentu 840

Kapitola 7

NFS 843
Preambule 843
Úvod 844
Nastavení serveru NFS 845
Nastavování klienta NFS 851
Optimalizace 853
Zabezpečení a NFS 855
Odstraňování problémů
861

Kapitola 8

Úvod do programování v BASH 869
Úvod 869
Velmi jednoduché skripty 870
Roury 872
Podmínky 873
Smyčky for, while a until 874
Uživatelské rozhraní 876
Různé 877
Tabulky 878
Další skripty 882
Když něco nefunguje (ladění) 885
O tomto dokumentu 885

Kapitola 9

Apache 887
Úvod 887
Apache 887
Apache Software Foundation 888
Vývoj webových aplikací pro Apache 888
Správa výkonu a šířky pásma 890
Virtuální hostitelé 890
Vyrovnávání zátěže 891
Zabezpečené transakce 890
SNMP 890
Moduly pro ověřování 892
Grafická rozhraní pro Apache 892
Vytváření modulů pro Apache 892
Knihy o serveru Apache 893
WebDAV 893
Projekty jazyka Java 893
Projekty XML 897

PHP 900
Python 901
Tcl 901
Moduly pro další jazyky 901
Apache 2.0 902
Přechod z webových serverů Netscape (iPlanet) 902
Přechod z Microsoft IIS 902
Odkazy 902
Kontakt na autora 902

Kapitola 10

Konfigurace systému 905
Úvod 905
Obecné nastavení systému 906
Obvyklé úlohy správy 915
Na konec 937

Kapitola 11

Vypalujeme CD 939
Úvod 939
Nastavení systému Linux pro vytváření CD-ROMů 943
Vypalování CD-R 949
Drahý Winfriede,... 954
Odstraňování problémů 962
Zásluhy 964

ČÁST V

GNU general public licence – český překlad 967
Preambule 969
Ustanovení a podmínky pro kopírování, distribuci a modifikaci 969
Jak uplatnit tato ustanovení na vaše nové programy 973

ČÁST VI

České sdružení uživatelů operačního systému LINUX (CZLUG) 975
CZLUG - Czech Linux User Group 977

Rejstřík 979

ČÁST I

Příručka uživatele

Tato příručka obsahuje vše, co potřebujete znát pro práci s operačním systémem Linux, což je volně šířitelná obdoba operačního systému Unix pro osobní počítače. Popisuje základní příkazy operačního systému Unix a rovněž se zabývá některými specifickými vlastnostmi Linuxu. Manuál je určen zejména pro začátečníky, avšak zkušenější uživatelé zde rovněž naleznou spoustu užitečných informací.

Poděkování

Autor by rád poděkoval následujícím lidem za jejich neocenitelnou pomoc při psaní a korekci této příručky. Jsou to:

Linus Torvalds, autor operačního systému Linux, který poskytl řadu podkladů.

Karl Fogel mi pomohl s vytvářením dokumentace k operačnímu systému Linux a napsal podstatné části kapitol 8 a 9.

Maurizio Codogno napsal podstatnou část kapitoly 11.

David Channon napsal dodatek A věnovaný editoru `vi`.

Společnost **Yggdrasil Computing, Inc.** poskytla štědrou podporu pro realizaci této příručky.

Společnost **Red Hat Software** rovněž dobrovolně podpořila realizaci této příručky.

Typografické konvence použité v této části

Tučné písmo — Používá se pro nové pojmy, varovné poznámky a klíčová slova u programovacích jazyků.

Italika — Používá se ke zdůraznění textu.

Skloněné neproporcionální písmo — Používá se k označení meta-proměnných, zejména v příkazovém řádku. Například „`ls -l` *foo*“, kde foo by mělo být nahrazeno jménem souboru, jako například `/bin/cp`.

`Neproporcionální písmo` — Používá se k reprezentaci textů vypisovaných na obrazovce. Dále se používá při výpisu kódů (zpravidla vytvořených v jazyku C), skriptů a výpisu obecných souborů, jako jsou konfigurační soubory. Aby nemohlo dojít k nedorozumění, budou tyto výpisy uváděny v rámečcích.

Výrazy v rámečku — Označují klávesy, které se mají stisknout. Uvidíte je nejčastěji ve formě: „Pro pokračování stiskněte klávesu Enter.“

Tento symbol bude signalizovat zvláštní upozornění nebo nebezpečí. Odstavce takto označené čtěte se zvláštní pozorností.

Tento symbol bude označovat text týkající se zvláštních pokynů pro uživatele systému X Window.

Tímto symbolem jsou označeny odstavce, které obsahují důležité informace a měly by být čteny obzvlášť pečlivě.

KAPITOLA 1

Úvod

Kdo by si měl přečíst tuto knihu

Patříte mezi ty, kteří by si měli přečíst tuto knihu? Pokud neznáte přímou odpověď, pokuste se odpovědět na jiné otázky. Získali jste někde operační systém Linux a nevíte, co máte dělat dál? Jste uživateli jiného operačního systému než Unix a uvažujete o používání systému Linux? Chcete se dozvědět, co lze od tohoto operačního systému očekávat?

Máte-li k dispozici tuto knihu a přitom jste na některou z předcházejících otázek odpověděli „ano“, pak byste si ji měli přečíst. Každý, kdo si nainstaloval na svůj osobní počítač operační systém Linux (což je volně šiřitelná obdoba operačního systému Unix vytvořená panem Linusem Torvaldsem) a kdo neví, jak jej má efektivně využívat, by si měl tuto knihu přečíst. Je v ní je popsána většina základních příkazů operačního systému Unix a rovněž jsou zde uvedeny pokročilejší techniky pro práci se systémem. Rovněž se zmíníme o výkonném editoru GNU Emacs a několika dalších důležitých aplikacích.

Co byste měli udělat, než začnete tuto knihu číst

V této knize se předpokládá, že máte zajištěn přístup k operačnímu systému typu Unix a že tímto operačním systémem je Linux, tedy operační systém běžící na osobních počítačích s procesorem Intel.* Posledně jmenovaný předpoklad však není nezbytný – budeme upozorňovat na případy, kdy se ostatní unixové operační systémy od systému Linux liší.

Operační systém Linux je dostupný v mnoha formách, jež se nazývají distribuce. Předpokládá se, že máte nainstalovánu kompletní distribuci, jako je SlackWare, Redhat nebo MCC-Interim. Mezi jednotlivými distribucemi jsou jisté rozdíly, ale většinou nejsou důležité. Může se stát, že příklady uvedené v této knize se budou lišit od vaší distribuce. Ničeho se neobávejte, většinou půjde o odlišnosti zanedbatelné, se kterými si snadno poradíte. Pokud byste narazili na závažné rozdíly, neváhejte a napište o nich autorovi.

Pokud máte přístupová práva jako superuživatel (tedy například jako správce operačního systému nebo ten, kdo provedl instalaci), měli byste si pro sebe vytvořit normální uživatelský účet.

Potřebné informace najdete v instalační příručce. Jestliže nemáte přístupová práva jako super uživatel, pak požádejte o vytvoření účtu správce systému.

* Poznámka korektora: Operační systém Linux je v současné době k dispozici nejen pro procesory Intel, ale i SPARC, Alpha, ARM, PowerPC, pro stanice SGI a jiné.

Na čtení této knihy byste měli mít dostatek času a trpělivosti. Naučit se pracovat s operačním systémem Linux není jednoduché – většina lidí považuje ostatní operační systémy za jednodušší (jako například Macintosh Operating System). Jakmile se však naučíte základům, budete pracovat velmi snadno a efektivně. Unix je výkonný operační systém a umožňuje snadno provádět i velmi složité úlohy.

Navíc se v této knize předpokládá, že jste seznámeni se základní počítačovou terminologií. I když není tento předpoklad nezbytným, bude se vám kniha číst snadněji. Měli byste například vědět, co je to spustitelný program nebo co je to textový soubor. Pokud to nevíte, budete potřebovat někoho, kdo vám při studiu tohoto operačního systému poradí a pomůže.

Kdy se můžete čtení této knihy vyhnout

Jakýkoliv počítačový program se nejlépe naučíte tak, že si jej budete zkoušet na počítači. Většina lidí přijde na to, že čtení manuálů bez možnosti bezprostředně program nebo operační systém používat není příliš užitečné. Operační systém Unix se nejlépe naučíte tak, že jej budete používat. Nevyhýbejte se experimentování – můžete sice leccos pokazit, ale stále máte možnost provést novou instalaci. Většině havárií způsobených vaší nezkušeností se můžete vyhnout důsledným zálohováním.

Operační systém Unix nelze používat tak intuitivně, jako většinu ostatních operačních systémů. Proto byste si měli přečíst alespoň kapitoly 4, 5 a 6.

Jednu z možností, jak se vyhnout čtení této knihy, představují tzv. manuálové stránky, jež poskytují dokumentaci on-line.

Jak číst tuto knihu

Při čtení této knihy vám doporučujeme, abyste si bezprostředně vyzkoušeli to, co si právě přečtete. Pokud jste s některou tématikou již seznámeni, můžete příslušné odstavce přeskočit. Některá témata vám budou připadat zajímavá, jiná triviální, či dokonce nudná. Možná máte tolik sebejistoty, že budete některé příkazy zkoušet přesto, že přesně nevíte, jakou mají funkci. I to je způsob, jak se naučit pracovat s operačním systémem, i když poněkud riskantní.

Mnozí lidé ztotožňují operační systém Unix s příkazovým interpretem. Příkazový procesor je program, kterým lze řídit vše, co se v operačním systému Unix odehrává. Operační systém Unix je ve skutečnosti velmi složitý – většina jednoduše zadaných příkazů spustí spoustu procesů, o nichž uživatel zpravidla ani neví. V této knize probereme základní pravidla pro používání příkazového procesoru a také se zmíníme o důležitých programech, které jsou součástí operačního systému Unix nebo Linux.

Tato kapitola je spíše pseudokapitolou, která předkládá informace o obsahu knihy. Další kapitoly se zabývají tématy dle následujícího seznamu:

Druhá kapitola se zabývá vznikem operačních systému Unix a Linux. Dále předkládá informace o nadaci Free Software Foundation a o projektu GNU.

Ve **třetí kapitole** je vysvětleno, jak začít a skončit práci s počítačem, co se stane při startování a při ukončení činnosti operačního systému. Většina těchto informací není pro práci s operačním systémem Linux důležitá. Jsou to ale informace užitečné a zajímavé.

Kapitola 4 představuje úvod k práci s příkazovým procesorem operačního systému Unix. Jedná se o prostředí, v němž se provádějí příkazy a spouštějí programy. Dozvíte se zde o základních příkazech a programech, které musíte znát, pokud máte používat operační systém Unix.

V **kapitole 5** se budeme zabývat systémem X Window. Ten představuje primární grafické uživatelské rozhraní pro operační systémy typu Unix a některé distribuce jej používají jako základní uživatelské prostředí.*

V **kapitole 6** se budeme zabývat pokročilejšími technikami při práci s příkazovým procesorem, zejména těmi, které vaši práci zefektivní.

V **sedmé kapitole** jsou stručně popsány příkazy operačního systému Unix. Čím více nástrojů budete umět používat, tím efektivnější bude vaše práce.

Kapitola 8 popisuje věhlasný editor Emacs. Jedná se o velký program, který většinu nástrojů operačního systému Unix integruje do jediného prostředí.

Devátá kapitola je věnována konfiguraci operačního systému Unix. Pomocí této konfigurace si můžete nastavit vlastnosti systému tak, aby se vám co nejlépe pracovalo.

V **desáté kapitole** se zmíníme o možnostech, které má uživatel operačního systému Unix k dispozici, chce-li komunikovat s ostatními počítači, případně se sítí Internet. Najdete zde informace o elektronické poště a o systému World Wide Web.

Jedenáctá kapitola popisuje některé komplikovanější příkazy.

V **kapitole 12** se dozvíte, jak se při práci s operačním systémem Unix nebo Linux snadno vyhnout chybám.

Dokumentace k operačnímu systému Linux

Tato uživatelská příručka (původní název zní: *„The Linux Users' Guide"*) je určena začátečníkům. V rámci projektu *„Linux Documentation Project"* se však připravují knihy pro pokročilejší uživatele.

Další knihy týkající se operačního systému Linux

Dokumentace k operačnímu systému Linux dále obsahuje tyto knihy: *„Installation and Getting Started"* – příručka popisující instalaci systému Linux. *„The Linux System Administrator's Guide"* – dokumentace týkající se správy operačního systému Linux. *„The Linux Kernel Hackers' Guide"* – kniha popisující jádro systému Linux a způsoby jeho modifikace. *„The Linux Network Administration Guide"* – příručka týkající se instalace, konfigurace a používání síťových spojení.

Soubory HOWTO

Kromě knih napsaných v rámci projektu *„Linux Documentation Project"* byla vytvořena řada krátkých dokumentů ve formě souborů týkajících se jednotlivých témat kolem operačního systému Linux. Například soubor SCSI-HOWTO popisuje řešení možných komplikací s používáním rozhraní SCSI.

Soubory HOWTO jsou dostupné v několika formách – jako ucelené knihy (například *„The Linux Bible"* nebo *„Dr. Linux"*), nebo je můžete získat prostřednictvím diskusní skupiny comp.os.linux.answers. Soubory se rovněž nacházejí na mnoha serverech FTP nebo WWW. Centrálním místem obsahujícím informace o operačním systému Linux je *http://www.linux.org*.

* Poznámka korektora: V současné době používá systém X Window většina používaných distribucí operačního systému Linux.

Co je to „Linux Documentation Project"?

Cílem projektu *„Linux Documentation Project"* je shromáždit, utřídit a v jednotné formě prezentovat dokumentaci týkající se operačního systému Linux. Pracují na něm lidé po celém světě. Impuls k projektu dal Lars Wirzenius a dnes jej koordinuje Matt Welsh s pomocí pana Michaela K. Johnsona.

Předpokládá se, že v rámci projektu *„Linux Documentation Project"* budou postupně vydány knihy týkající se všech témat kolem operačního systému Linux. Pokud budete mít jakékoliv náměty na zlepšení této dokumentace, dejte prosím vědět autorovi této knihy (prostřednictvím adresy *leg+@andrew.cmu.edu*) a/nebo panu Mattu Welshovi (prostřednictvím adresy *mdw@cs.cornell.edu*).

Operační systémy

Primární funkce operačního systému spočívá v tom, že poskytuje podporu pro realizaci počítačových programů. Proto například můžete používat svůj oblíbený editor a vytvářet dokumenty. Bez podpory operačního systému by editor nemohl pracovat – editor potřebuje ke své činnosti interakci s vaším terminálem, s vašimi soubory a dalším technickým vybavením počítače.

Nyní si položme otázku: Proč je nutné psát a číst knihy o operačních systémech, které jsou pouhou podporou vašich aplikací? Každý operační systém obsahuje spoustu programů pro správu, konfiguraci, řízení spojení s ostatními systémy, zálohování a podobně. Pokud jde o operační systém Linux, najdete zde řadu „miniaplikací", jež vám pomohou pracovat efektivněji. Jestliže tedy nepoužíváte váš počítač pouze k práci s jednou nebo několika málo aplikacemi, je znalost operačního systému velmi důležitá.

Operační systém (nejčastěji označován zkratkou „OS") může být jednoduchý a minimalizovaný (například DOS) nebo velký a složitý (například OS/2 nebo VMS). Operační systémy typu Unix patří ke středně velkým systémům. Poskytují o něco více zdrojů a prostředků než první jednoduché operační systémy, a to v takové formě, aby s jejich pomocí bylo možné řešit prakticky všechny úlohy.

Původně byl operační systém Unix navržen jako zjednodušení operačního systému Multics. Filosofie operačního systému Unix spočívá v tom, že by se veškeré funkce měly rozdělit do malých částí, tedy relativně jednoduchých programů.[1] Nové funkční vlastnosti lze získat vhodnou kombinací těchto jednoduchých programů. Samozřejmě se stále objevují nové a nové obslužné programy, které lze snadno integrovat do vašich nástrojů, a tak můžete váš operační systém neustále rozšiřovat. Když jsem například psal tento dokument, používal jsem následující programy: `fvwm` pro obsluhu a konfiguraci systému X Window, editor `Emacs` pro vytvoření textu, `xdvi` pro prohlížení textu před tiskem, `LATEX` pro formátování textu, `dvips` pro přípravu tisku a `lpr` pro vlastní tisk. Kdyby například existoval jiný program pro prohlížení textu před tiskem, pak bych jej mohl použít, aniž bych změnil ostatní programy. V tomto okamžiku na mém počítači běží současně třicet osm programů (většina z nich se nachází v neaktivním stavu „sleep", dokud nebudou aktivovány k provedení nějakého úkolu).

Při používání operačního systému Unix budete jistě chtít minimalizovat množství práce potřebné k realizaci každého běžně prováděného úkolu. Operační systém Unix vám pro tento účel nabízí spoustu nástrojů. Abyste je mohli efektivně používat, budete muset poměrně přesně vědět, jakou funkci každý nástroj má. Pokud byste nad uskutečněním každého jednoduchého úkolu strávili ho-

1 Tato filosofie byla ve skutečnosti určena technickým vybavením počítačů, na kterých běžely první verze operačního systému Unix. Časem se ukázalo, že Unix dobře funguje i na počítačích s vyspělejším technickým vybavením. I dnes, po dvaceti pěti letech, je původní filosofie operačního systému Unix zcela vyhovující.

dinu nebo dokonce více, pak by vaše práce nebyla příliš produktivní. V této knize se dozvíte, jak a v které situaci využívat nástroje obsažené v operačním systému Unix a jak tyto nástroje kombinovat za účelem řešení složitějších úloh.

Klíčovou částí operačního systému je tzv. **jádro**. Ve většině operačních systémů, jako je Unix, OS/2 nebo VMS, plní jádro systému takové funkce, jako je spouštění programů, přidělování systémových zdrojů, přidělování času procesoru současně běžícím programům a podobně. Samozřejmě platí, že i jádro systému je program. Ten běží na vašem počítači jako první program po jeho nastartování, kdy realizuje všechny konfigurační funkce, a jako poslední program před vypnutím počítače, kdy realizuje všechny potřebné funkce k zastavení systému.

KAPITOLA 2

Co je to Unix

Historie operačního systému Unix

V roce 1965 pracovaly společnosti Bell Telephone Laboratories (divize AT&T) a General Electric na projektu „MAC of MIT“, jehož cílem bylo vytvořit operační systém Multics. Později se společnost Bell Telephone Laboratories rozhodla od spolupráce odstoupit, ale v důsledku toho neměla k dispozici kvalitní operační systém.

Pánové Ken Thompson a Dennis Ritchie se rozhodli navrhnout operační systém, který by společnosti Bell Telephone Laboratories vyhovoval. Ken Thompson tento návrh realizoval při vytváření vývojového prostředí na počítači PDP-7. Další výzkumný pracovník společnosti Bell Telephone Laboratories, pan Brian Kernighan, dal novému operačnímu systému název Unix.

Později zveřejnil pan Dennis Ritchie programovací jazyk C. V roce 1973 byl Unix kompletně přepsán do jazyka C (původní systém byl vytvořen v assembleru[1]). V roce 1977 byl operační systém Unix převeden z počítače PDP na nový počítač s použitím procesu, jež se nazývá „**porting**“. Tato akce byla uskutečnitelná právě proto, že byl operační systém Unix přepsán v jazyce C.

Koncem sedmdesátých let byla společnosti AT&T protimonopolním úřadem zakázána činnost v oblasti počítačového průmyslu. Proto se společnost rozhodla za velmi výhodných finančních podmínek převést licenci na operační systém Unix na některé university. Unix se tedy stal populárním především v akademických kruzích, avšak postupem času se začal prosazovat i v komerční sféře. Dnešní podoba Unixu se zcela liší od verze z roku 1970. Existují dvě základní varianty: System V od společnosti USL (Unix System Laboratories, dnes jej vlastní Novell[2]) a BSD (Berkeley Software Distribution).

Poslední verze USL má označení SVR4[3] (čtvrtá verze), zatímco poslední verze od BSD má označení 4.4. Kromě těchto základních verzí však existuje spousta dalších verzí operačního systému Unix. Komerční verze jsou zpravidla odvozeny od jedné z verzí USL nebo BSD. Existuje však spousta verzí operačního systému Unix, které kombinují vlastnosti obou základních verzí.

Ceny současných komerčních verzí operačního systému Unix pro počítače s procesorem Intel se pohybují od 500 do 2000 dolarů.

1 Assembler je základní programovací jazyk, který je úzce spjat s konkrétním typem počítače. Programy napsané v jazyce assembler zpravidla nejsou přenositelné mezi různými počítačovými platformami.

2 Tato verze operačního systému Unix byla nedávno prodána společnosti Novell. Předtím byla verze USL vlastněna společností AT&T.

3 SVR4 je zkratkou pro „System V“ verze 4. (System V Release 4)

Historie operačního systému Linux

Autorem operačního systému Linux je pan Linus Torvalds. Jeho původní verze byla v průběhu asi deseti let zdokonalována bezpočtem lidí na celém světě. Linux představuje verzi operačního systému Unix pro osobní počítače s procesorem Intel, Alpha a další a byl kompletně vytvořen znovu – na jeho vývoji se společnosti Unix System Laboratories a Berkeley Software Distribution vůbec nepodílely. Zajímavé je, že se na vývoji operačního systému Linux podíleli lidé na celém světě – od Austrálie po Finsko a všichni doufali, že se Linux podaří uvést do podoby schopné konkurovat ostatním operačním systémům typu Unix.

Vlastní projekt operačního systému Linux začal výzkumem vlastností procesoru 386. Systém je navržen tak, aby maximálně využíval všech vlastností tohoto procesoru.

Na operační systém Linux se vztahují licenční podmínky GPL (GNU General Public Licence). Tato licence se vztahuje na veškeré programové vybavení produkované nadací Free Software Foundation a jejím cílem je zabránit komukoliv omezovat distribuční práva ostatních. Jinými slovy, podle licenčních podmínek GPL si můžete účtovat za distribuci programového vybavení GNU kolik chcete, ale nesmíte nikomu nařizovat, za jaký poplatek je má distribuovat dál. Dále musíte s každou distribucí programového vybavení GNU zpřístupnit zdrojové kódy[4], což je velmi užitečné pro programátory. Pak si totiž každý může například modifikovat operační systém Linux a dále distribuovat tuto modifikovanou verzi – opět za předpokladu, že ji bude distribuovat podle licenčních podmínek GPL.

Operační systém Linux podporuje většinu programového vybavení napsaného pro Unix, včetně systému X Window. Tento systém byl vytvořen na Massachusetts Institute of Technology tak, aby umožňoval operačním systémům Unix vytvářet grafická okna a interaktivně spolupracovat mezi sebou. Dnes platí, že je systém X Window implementován pro každou verzi operačního systému Unix.

Kromě standardů, které představují verze System V a BSD, existuje sada standardizačních dokumentů publikovaná úřadem pro standardizaci IEEE, která má označení **POSIX**. Operační systém Linux splňuje většinu podmínek uvedených v dokumentech POSIX-1 a POSIX-2. Přitom má v mnoha aspektech vlastnosti podobné systému BSD, ale má také vlastnosti, které najdete u systému System V. Stručně řečeno, pokud jde o standard, představuje operační systém Linux kombinaci (a většina lidí se domnívá, že velmi dobrou) všech tří standardů.

Většina programového vybavení dodávaná s operačním systémem Linux pochází od nadace Free Software Foundation a je součástí projektu GNU. Hlavním cílem projektu GNU je jednak zpřístupnit programové vybavení původně napsané pro operační systém Unix uživatelům ostatních operačních systémů, jednak vytvořit přenositelný operační systém, který bude mít prakticky stejné vlastnosti jako Unix. Přenositelný znamená, že tento operační systém poběží na všech počítačových platformách (ať jsou vybaveny procesorem Intel, PowerPc, Alpha či jiným). Tento operační systém má označení Hurd. Hlavní rozdíl mezi operačním systémem Linux a operačním systémem Hurd nespočívá v uživatelském rozhraní, ale v programátorském rozhraní – Hurd představuje moderní operační systém, zatímco Linux je operačním systémem založeným na filosofii stařičkého Unixu.

4 Zdrojový kód programu je soubor ASCII obsahující příkazy pro konkrétní programovací jazyk. Příslušným překladačem jej lze přeložit do strojového kódu pro konkrétní počítač. Strojový kód je soubor, který již není snadno čitelný a modifikovatelný tak jako zdrojový kód.

Předcházející odstavce by mohly vzbudit dojem, že se o operační systém Linux stará pouze pan Linus Torvalds. V raných dobách existence tohoto systému se například o překladač GCC (základní překladač jazyka C pro Linux) a o knihovny Linux C staral H. J. Lu. V každé distribuci operačního systému Linux naleznete v adresáři `/usr/src/linux/` soubor `CREDITS` obsahující seznam všech lidí, kteří se významnou měrou podíleli na jeho vývoji.

Dnešní podoba operačního systému Linux

První číslo ve verzi operačního systému Linux představuje číslo hlavní revize. V době, kdy byla napsána tato kniha (únor 1996) je k dispozici teprve první verze. Druhé číslo představuje méně podstatné revize. Pokud je druhé číslo sudé, jedná se o stabilní verzi. Jestliže je liché, jedná se o vývojovou verzi. Ve vývojových verzích bývá spousta chyb, které „odvážní“ uživatelé postupně odhalují a programátoři postupně opravují. Jakmile jsou všechny závažné nedostatky z vývojové verze odstraněny, prohlásí se vývojová verze za stabilní a začne se pracovat na nové vývojové verzi. V únoru 1996 měla stabilní verze číslo 1.2.11 a poslední vývojová verze 1.3.61.*

Operační systém Linux je velkým systémem obsahujícím spoustu chyb, které se postupně odstraňují. Ve stabilních verzích se však vyskytují chyby velmi zřídka a souvisejí hlavně s nestandardním technickým vybavením počítače. Doposud jsme hovořili pouze o chybách jádra systému. Operační systém však zdaleka není jen jádro systému, ale obsahuje spoustu obslužných programů.

Ani velmi zkušený uživatel často není schopen určit, zda je původ chyby v obslužném programu nebo v jádře systému. Také je velmi nesnadné rozeznat, zda jsou některé „divné“ věci chybou operačního systému, nebo novou vlastností. Doufáme, že vám tato kniha pomůže orientovat se právě v takových situacích.

Několik málo otázek a odpovědí na ně

Než se pustíme do další kapitoly, odpovězme si na několik důležitých otázek.

Otázka: Jak se má vyslovovat Linux?

Odpověď: Podle autora operačního systému Linux se má samohláska „i“ vyslovovat krátce (jako například ve slově minimum). Linux by se měl rýmovat se slovem Minix, což je další verze operačního systému Unix. Samohláska „u“ by se měla vyslovovat ostře, jako například ve slově „rule“, a nikoliv měkce, jako například ve slově „ducks“. Obecně by se mělo slovo Linux rýmovat se slovem „cynics“. V našich zemích se Linux vyslovuje „linuks“.

Otázka: Proč by se mělo pracovat s operačním systémem Linux?

Odpověď: Proč ne. Linux je obecně levnější než ostatní operační systémy a je méně problematický než většina komerčních operačních systémů. Nemusí být pravda, že zrovna Linux je tím nejlepším operačním systémem pro vaši konkrétní aplikaci. Pokud má však někdo zájem o používání operačního systému typu Unix na osobním počítači a o aplikace vytvořené pro Unix, pak je Linux pravděpodobně nejlepším vysoce výkonným systémem.

Komerční programové vybavení pro operační systém Linux

Pro operační systém Linux existuje spousta komerčního programového vybavení. Nejdůležitější roli hraje systém Motif, což je uživatelské rozhraní pro systém X Window připomínající Microsoft Windows. Na základě systému Motif byla vytvořena řada komerčního programového vybavení.

* Poznámka korektora: V současné době (srpen 1998) je poslední stabilní verze 2.0.34 a vývojová verze 2.1.108.

Dnes můžete koupit prakticky cokoliv ve verzi pro operační systém Linux – od oblíbeného textového procesoru Word Perfect až po Maple, univerzální nástroj pro matematické a fyzikální modelování.

Možná se budete divit, jak lze skloubit operační systém Linux, jenž je distribuován podle licenčních podmínek GPL (GNU General Public Licence, viz dodatek B), s komerčním programovým vybavením. Podmínky GPL platí pouze pro jádro systému, zatímco na ostatní aplikace vytvořené pro Linux se vztahují jiné podmínky GNU, *„Library General Public Licence"* (viz dodatek B). Podle těchto podmínek smějí poskytovatelé programového vybavení prodávat své aplikace a přitom nemusejí distribuovat zdrojové kódy.

Uvědomte si prosím, že uvedené dva dokumenty představují něco jako autorská práva, ale v žádném případě nepředstavují licenci na užívání. Tyto dokumenty nikterak nevymezují způsoby, kterými můžete dané programové vybavení používat, ale vymezují pravidla, kterými se musíte řídit při distribuci tohoto programového vybavení. V tom spočívá hlavní myšlenka filosofie nadace Free Software Foundation a platí i pro Linux: na používání operačního systému Linux neexistuje žádná licence.

KAPITOLA 3

Začínáme pracovat s operačním systémem Linux

Pravděpodobně máte jisté zkušenosti s používáním jednouživatelského operačního systému, jako je DOS nebo OS/2. Chcete-li pracovat v těchto operačních systémech, nemusíte se identifikovat – předpokládá se, že jste jediným uživatelem operačního systému, který má přístup ke všem jeho zdrojům. Na druhé straně, operační systém Unix je víceuživatelským systémem. To znamená, že v daném okamžiku může mít k systému přístup více uživatelů najednou.

Aby mohl operační systém Unix komunikovat s každým uživatelem odděleně, musí jej identifikovat procesem, který se nazývá „**přihlášení se**“ (logging in). Když zapnete počítač, proběhne spousta inicializačních procesů a až potom je počítač schopen komunikovat s uživatelem. Protože je tato příručka věnována operačnímu systému Linux, vysvětlíme si, co se odehrává v průběhu zavádění tohoto operačního systému Linux.

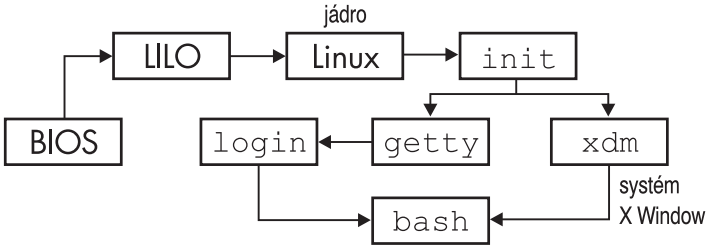
Pokud nepoužíváte operační systém Linux na osobním počítači s procesorem Intel, pak se vás nebudou některé informace v této kapitole týkat. Většinu užitečných informací najdete v oddílu Zapnutí počítače.

Jestliže se pouze zajímáte o používání vašeho počítače, pak kromě oddílu Činnost uživatele nemusíte tuto kapitolu číst.

Zapnutí počítače

Po zapnutí počítače s procesorem Intel se jako první provede program zvaný BIOS. BIOS je zkratkou pro Basic Input/Output System. Tento program je trvale uložen ve zvláštní paměti počítače. Realizuje některé základní testy a identifikuje technické vybavení, jako je pevný disk či disketová jednotka. Po identifikaci disku najde tzv. zaváděcí sektor (boot sector) a spustí kód, který v něm nalezne. Jestliže byl nalezen disk, ale žádný zaváděcí sektor, vypíše BIOS následující hlášení:

```
Non-system disk or disk error
```



Obrázek 3.1 – Cesta, po které dospěje osobní počítač s procesorem Intel k příkazovému řádku. Proces `init` může, ale také nemusí realizovat zavedení systému X Window. Pokud systém X Window zavádí, spustí nejdříve program `xdm`. Jinak spustí příslušný počet procesů `getty`.

Pak ovšem stačí disk bez zaváděcího sektoru vyjmout, stisknout kteroukoliv klávesu a zaváděcí proces bude pokračovat.

Pokud není v disketové jednotce vložena disketa, BIOS vyhledá hlavní zaváděcí záznam MBR (master boot record) na pevném disku (závisí na nastavení BIOSu). Ten obsahuje kód (zaváděcí program), jehož úkolem je zavést operační systém. Pro operační systém Linux se tento program jmenuje LILO (LInux LOader). Linux Loader je nejpoužívanějším bootovacím programem pro Linux, ale je možno použít i GRUB Loader, SYSLinux nebo i NT Loader. To znamená, že program LILO je uložen právě v diskové oblasti MBR. Nyní budeme předpokládat, že se zavádí operační systém Linux. (U vaší konkrétní distribuce může proces zavádění operačního systému Linux probíhat poněkud odlišně – proto si pečlivě pročtěte dokumentaci k vaší distribuci. Další užitečné informace naleznete v dokumentaci k zaváděcímu programu LILO.)

Jak Linux přebírá kontrolu nad počítačem

Nejdříve tedy předá BIOS kontrolu zaváděcímu programu LILO a ten pak předá kontrolu **jádru** operačního systému Linux (Linux kernel). Jádro je centrálním programem celého systému a řídí všechny ostatní programy. V prvním kroku jádro realizuje přepnutí procesoru do tzv. chráněného módu. Procesor 80386[1], jež řídí váš počítač, může pracovat ve dvou módech: v tzv. „reálném módu“ a v tzv. „chráněném módu“. Operační systém DOS, stejně jako BIOS, funguje v reálném módu. Vyspělejší operační systémy však pracují v chráněném módu. Proto platí, že Linux po svém zavedení zcela nahradí BIOS.

Pokud jde o počítače s jinými procesory, probíhá tato fáze zavádění systému Linux poněkud jinak. Zpravidla nedochází k přepínání do chráněného módu a jen několik málo procesorů vyžaduje komplikované zavádění operačního systému podobné kombinaci BIOS – LILO. Po zavedení jádra pracuje Linux na všech počítačích stejně bez ohledu na procesor.

V dalším kroku se Linux pokouší identifikovat technické vybavení počítače, na kterém běží. Musí znát informace o pevném disku, zda je nebo není k dispozici myš, zda je nebo není váš počítač připojen k počítačové síti a podobně. Linux si není schopen po vypnutí počítače tyto údaje pamatovat, proto je pokaždé zjišťuje znovu. Naštěstí nemusíte údaje o technickém vybavení vašeho počítače zadávat – Linux si je zjistí sám.

1 Pod pojmem procesor 80386 budeme dále rozumět všechny 32bitové procesory Intel, tedy i 80486, Pentium, Pentium Pro, Pentium MMX a Pentium II. Místo 80386 budeme také používat označení 386.

V průběhu zavádění vypisuje operační systém Linux řadu hlášení. Podrobněji se jimi budeme zabývat v oddílu Hlášení jára systému. Asi by se vám nelíbilo, pokud byste museli při každém startu odpovídat na spoustu dotazů týkajících se technického vybavení a konfigurace vašeho počítače. Na několik důležitých dotazů tohoto druhu však budete muset odpovědět při instalaci systému. Budete-li mít jakékoliv problémy, pročtěte si dokumentaci k vaší distribuci.

Jak jsme se již zmínili, jádro řídí všechny ostatní programy. Jestliže tedy úspěšně identifikuje veškeré technické vybavení počítače, spustí jiný program, který již začne dělat něco užitečného. Tento program má název `init`. (Všimněte si, že pro název programu jsme použili jiný font. `Neproporcionální font` budeme používat k označení programů, adresářů i obecných souborů.) Po nastartování programu `init` se jádro stane jakýmsi „manažerem“, avšak již není aktivním programem.

Jestliže chceme vědět, co dělá počítač po zavedení jádra, musíme studovat program `init`. Ten prochází mnoha komplikovanými stádii, která nejsou na všech počítačích stejná. Operační systém Linux má mnoho verzí programu `init` – každá z nich realizuje své cíle různými způsoby. Také záleží na tom, zda je váš počítač zapojen do sítě a jakou distribuci operačního systému Linux používáte. Nyní si uveďme některé základní procesy, jež program `init` realizuje:

- Zpravidla proběhne kontrola souborového systému. Co je to souborový systém? Je to systém je způsob organizace a ukládání souborů na pevném disku. Linux při kontrole souborového systému zjišťuje, které části disku jsou již použity a které jsou volné. Souborový systém se podobá rejstříku nebo štítkovému katalogu v knihovně. Bohužel se stává, že například při výpadku elektrické energie dojde k poškození některých souborů a pak je používání některých částí pevného disku konfliktní. Program `init` proto spouští další program označený jako `fsck`, jenž se pokouší vadné soubory a konfliktně obsazené části pevného disku opravit.
- Pokud je váš počítač připojen k síti, spustí se několik speciálních programů, jež zajišťují komunikaci vašeho počítače s jinými počítači.
- Na pevném disku mohou zůstat některé dočasné soubory, které jsou jinak neužitečné. Ty mohou být programem `init` zrušeny.
- Zpravidla také dojde k aktualizaci systémového času. Operační systém Unix předpokládá, že je čas definován jako UCT (Universal Coordinated Time), známý také jako střední greenwichský čas. Přitom je čas ve vašem počítači pravděpodobně nastaven na lokální – to znamená, že některý program musí přečíst lokální čas nastavený ve vašem počítači a korigovat jej na čas UCT.

Jakmile program `init` ukončí všechny uvedené aktivity, přejde do stavu, jenž lze označit jako „rodič“ všech procesů v systému Unix. Pod pojmem proces si lze jednoduše představit běžící program. Protože jeden program může být spuštěn dvakrát nebo vícekrát, mohou existovat dva procesy nebo více procesů realizujících konkrétní program.

V operačním systému Unix tedy platí, že proces je instancí programu. Vytváří se systémovým voláním (službou poskytovanou jádrem systému) nazvaným „fork“ (rozvětvení). Pojem „fork“ se používá proto, že se původně jeden proces při spuštění dalšího procesu rozdělí na dva oddělené procesy. Typickým příkladem programu, který běží jako několikanásobný proces, je program `getty`. `getty` je důležitý program, jenž voláním programu `login` umožňuje uživateli přihlásit se do systému.

Činnost uživatele

Jak se přihlásit do systému

Než začnete používat operační systém Unix, musíte se identifikovat. Procesu identifikace se říká „**přihlášení do systému**“ (`login`). Jedná se o proces, ve kterém se operační systém přesvědčí, zda má uživatel oprávnění systém používat. Systém se vás zeptá na jméno účtu (account name) a heslo (password). Jméno účtu je zpravidla podobné vašemu jménu – pravděpodobně jste jej obdrželi od vašeho systémového správce nebo jste si vytvořili vlastní, pokud jste sami systémovým správcem. (Informace o vytváření jména účtu naleznete v příručce *„Příručka správce operačního systému Linux“.)*

Po dokončení všech zaváděcích procesů byste měli na svém monitoru spatřit hlášení podobné následujícímu. Nejnovější verze Linuxu se hlásí takto (první řádek je pouze uvítací zprávou – může zde být cokoliv jiného):

```
RedHat Linux release 5.1 (Manhattan) with all updates. Kernel 4.1.108 on an Intel
mousehouse login:
```

Je však možné, že vás systém uvítá něčím úplně jiným. Místo nudné textové obrazovky se vám může představit nějaká grafická obrazovka. Avšak i v tomto případě budete vyzváni k přihlášení se do systému a proběhnou stejné procedury identifikace uživatele. Pokud se vám objevila zmíněná grafická obrazovka, pak jste zřejmě uživateli systému X Window. To znamená, že budete pracovat v systému oken, o kterém se podrobněji zmíníme v kapitole 5. Všimněte si, že veškerý výklad o systému X Window bude v této knize označován velkým písmenem X na levém okraji textu.

Jako jméno účtu budeme používat smyšlené jméno `larry`. Kdykoliv uvidíte v následujícím textu jméno `larry`, měli byste místo něj dosadit své vlastní jméno účtu. Jména účtu by měla být založena na skutečném jménu uživatele; ve větších operačních systémech typu Unix se zpravidla používají příjmení uživatelů, kombinace jména a příjmení nebo kombinace jména a číslic. Pro uživatele se jménem Larry Greenfield by se mohly použít například tyto kombinace: `larry`, `greenfie`, `lgreenfi` nebo `lg19`.

Pro bližší vysvětlení, jméno `mousehouse` je „jménem“ počítače, na kterém pracuji. Je pravděpodobné, že jste při instalaci operačního systému Linux použili jiné, podobně duchaplné jméno. Jméno počítače není důležité; v této knize budeme používat již uvedené jméno `mousehouse`, případně `lionsden`. Jestliže tedy zadáte své jméno účtu a stisknete klávesu Enter, objeví se výzva k zadání hesla:

```
mousehouse login: larry
Password:
```

Na druhém řádku očekává operační systém Linux zadání hesla (password). Při zadávání hesla neuvidíte na obrazovce, co píšete, proto zadávejte heslo pečlivě. Chybně zadaný znak můžete zrušit, ale opět nebudete vidět, který znak jste zrušili. Jestliže se na vás zrovna někdo dívá, nezadávejte heslo příliš pomalu. Mohlo by tak dojít k vyzrazení vašeho hesla a nepovolaná osoba by jej mohla zneužít. Jestliže zadáte heslo chybně, budete mít možnost zadat je znovu.

Pokud jste zadali heslo správně, objeví se zpravidla na vaší obrazovce nějaká zpráva, jež se většinou nazývá „zpráva dne“. Tato zpráva může obsahovat cokoliv – její znění zadává systémový správce.

Jako další se objeví tzv. výzva příkazového řádku (prompt). Výzva příkazového řádku je řetězec indikující skutečnost, že systém očekává zadání příkazu. Mohla by vypadat takto:

```
/home/larry#
```

Pokud jste uživateli systému X Window, pak zřejmě uvidíte výzvu podobnou výše uvedené v nějakém okně na obrazovce. Okno je obdélníková orámovaná oblast obrazovky, která v tomto případě imituje terminál. Pokud chcete zadat příkaz v příkazovém řádku v okně, musí být okno aktivní – stačí například do tohoto okna přesunout kurzor myši.

Jak ukončit práci s počítačem

Chcete-li skončit práci s počítačem, pak rozhodně nestačí pouze počítač vypnout. Pokud to uděláte v operačním systému Linux, s největší pravděpodobností ztratíte nějaká data nebo poškodíte nějaké důležité soubory.

Na rozdíl od operačního systému DOS, kde lze počítač vypnout vypínačem nebo restartovat tlačítkem Reset, je v operačním systému Linux nutné provést řadu kroků, které zajistí bezpečné ukončení systému. Uveďme si hlavní důvod. Operační systém Linux používá tzv. **paměť cache** pro diskové operace. To znamená, že jistá část souborů uložených na pevném disku je umístěna do paměti RAM[3]. K synchronizaci mezi pamětí RAM a pevným diskem dochází přibližně každých třicet sekund. Jestliže se má vypnout nebo restartovat počítač, pak je nutné, aby se část paměti RAM obsahující disková data uložila.

Máte-li tedy ukončit práci s počítačem a přitom jste normálně přihlášeni (t.j. zadali jste jméno uživatelského účtu a heslo), pak se musíte odhlásit. K tomuto účelu slouží příkaz `logout`. Napište v příkazovém řádku `logout` a stiskněte klávesu Enter. Klávesou Enter se odesílá každý příkaz. Než tuto klávesu stisknete, můžete chybně zadané znaky opravovat, případně celý příkaz zrušit a zadat nový.

```
RedHat Linux release 5.1 (Manhattan) with all updates.
Kernel 4.1.108 on an Intel
mousehouse login:
```

Nyní se může do systému přihlásit další uživatel.

Vypnutí počítače

Jste-li jediným uživatelem počítače, pak po odhlášení můžete počítač odpojit od zdroje elektrické energie.[4] V takovém případě se přihlašujte jako uživatel se speciálním jménem účtu root. Účet root je účtem systémového správce, který má zajištěn přístup ke každému souboru v systému. Pokud hodláte vypnout počítač, pak musíte obdržet heslo od systémového správce. (Jste-li jediným uživatelem počítače, pak jste systémovým správcem právě vy! Proto nikdy nezapomeňte heslo!)

3 Rozdíl mezi pamětí RAM a pevným diskem lze přirovnat ke krátkodobé a dlouhodobé paměti. Po vypnutí počítače se veškeré informace v paměti RAM ztratí, zatímco informace uložené na pevném disku zůstávají uchovány. Na druhé straně platí, že přístup k informacím na pevném disku je nesrovnatelně pomalejší, než přístup k informacím v paměti RAM.

4 Chcete-li šetřit některé komponenty technického vybavení vašeho počítače, pak vypínejte počítač jen když nebudete na počítači pracovat delší dobu (například večer). Časté zapínání a vypínání počítače zbytečně namáhá některé jeho součásti.

Přihlášení a odhlášení by mohlo vypadat takto:

```
mousehouse login: root
Password:
Last login: Sun 5 11:14:57 on tty1
/# shutdown -h now

Broadcast message from root (tty1) Sun 14:52:32 1998...

The system is going down for system halt NOW!
INIT:Switching to runlevel: 0
INIT: sending processes the TERM signal
The system is halted
System Halted
```

Po zadání příkazu `shutdown -h now` proběhnou jisté procedury, které připraví systém na vypnutí nebo reset.

Na závěr ještě uveďme krátkou poznámku pro pohodlnější uživatele. Místo procesu přihlašování a odhlašování jako uživatel root lze použít příkaz `su`. Jako běžný uživatel zadejte v příkazovém řádku příkaz `su` a stiskněte klávesu Enter. Systém si vyžádá heslo uživatele root a pak vám přidělí všechna jeho privilegia. Nyní můžete bez problémů systém ukončit příkazem `shutdown -h now`.

Hlášení jádra systému

Když poprvé startujete počítač, objeví se na obrazovce spousta hlášení popisující technické vybavení vašeho počítače. Uvedená hlášení vypisuje jádro operačního systému Linux. V tomto oddílu si některá důležitá hlášení podrobněji popíšeme.

Přirozeně platí, že se hlášení jádra systému budou lišit, což je dáno jednak typem počítače a jednat distribucí operačního systému Linux. V následujícím textu budou popsána hlášení platná pro můj počítač. Některá mají obecnou platnost, jiná jsou více specifická. Musím poznamenat, že můj počítač má minimální konfiguraci, pokud jde o technické vybavení, proto dále neuvidíte příliš mnoho informací týkajících se speciálního technického vybavení. Následující hlášení pocházejí z verze operačního systému Linux 1.3.55. Tato verze patří v době, kdy píši tuto knihu, k nejnovějším.

1. V prvním kroku operační systém Linux identifikuje grafickou kartu, aby mohl vybrat nejvhodnější font a jeho velikost. (Čím menší font je zvolen, tím větší počet hlášení se vejde na jednu obrazovku.) Linux se vás může zeptat, jaký font má používat, nebo použije standardní „kompilovaný“ font (compiled font).[5]

   ```
   Console: 16 point font, 400 scans
   Console: colour VGA+ 80x25, 1 virtual console (max 63)
   ```

   V předcházejícím příkladu se vlastník počítače v době kompilace rozhodl pro větší, standardní font. Také si všimněte zastaralého tvaru slova „colour“. Linus se evidentně naučil špatnou verzi angličtiny.

2. V dalším kroku zobrazí jádro operačního systému zprávu o výkonnosti vašeho počítače. Výkonnost se měří v jednotkách „BogoMIPS“. Zkratka MIP (million instructions per second)

5 Kompilace je proces, při kterém se instrukce srozumitelné člověku přeloží do instrukcí srozumitelných počítači. Pojmem kompilovaný se zde rozumí „začleněný do programu“.

označuje počet instrukcí, které je počítač schopen provést za sekundu. „BogoMIP“ je zkratkou „bogus MIP“ (bogus=podvodný, falešný) a označuje, kolikrát za sekundu neudělá počítač absolutně nic. Protože cyklus, kterým se výkonnost počítače měří, nedělá ve skutečnosti nic, nelze uvedené číslo považovat za objektivní ukazatel výkonnosti.* Operační systém Linux toto číslo používá v případě, kdy potřebuje vyčkat na odezvu nějakého zařízení.

```
Calibrating delay loop.. ok - 33.28 BogoMIPS
```

3. V třetím kroku vypíše jádro systému informace o použití paměti:

```
Memory: 23180k/24576k available (544 kernel code, 384k reserved,468k data)
```

Z této informace vyplývá, že má počítač 24 MB operační paměti. Část této paměti byla rezervována pro jádro. Zbytek může být využíván ostatními programy. Uvedené informace se týkají paměti **RAM**, která může být používána k uchovávání dat, jen když je počítač zapnut. Počítač však má k dispozici permanentní paměť zvanou **pevný disk**. Obsah pevného disku zůstává uchován i tehdy, když je váš počítač vypnut.

4. V průběhu zavádění operačního systému provádí Linux řadu testů technického vybavení a o výsledcích těchto testů vypisuje průběžné zprávy.

```
This processor honours the WP bit even when in supervisor mode. Good.
```

5. V dalším kroku se Linux zabývá konfigurací sítě. Následující zpráva by měla být popsána v manuálu *„Příručka správce sítě“*.

```
Swansea University Computer Society NET3.033 for Linux 1.3.50
IP Protocols: ICMP, UDP, TCP
```

Popis konfigurace sítě přesahuje rámec této dokumentace, a proto se jím nebudeme zabývat.

6. Operační systém Linux podporuje jednotku pro výpočet v pohyblivé řádové čárce FPU (floating point unit). Jedná se o speciální integrovaný obvod (nebo přímo součást procesoru 80486DX), jenž realizuje aritmetické operace s neceločíselnými operandy. Některé z těchto integrovaných obvodů mohou být špatné a když se je operační systém Linux pokusí detekovat, dojde k jeho zhroucení – počítač přestane být funkční. Pokud nastane tento případ, uvidíte na obrazovce zprávu:

```
Checking 386/387 coupling...
```

V případě, že používáte procesor 486DX, uvidíte tuto zprávu:

```
Checking 386/387 coupling... Ok, fpu using exception 16 error reporting.
```

Máte-li starší procesor 386 s matematickým koprocesorem 387, uvidíte zprávu:

```
Checking 386/387 coupling... Ok, fpu using irq13 error reporting.
```

7. Nyní proběhne další test, jenž testuje instrukci „halt“.

* Poznámka korektora: Velikost tohoto čísla může být např. ovlivněna typem procesoru, jeho schopností předvídat cykly a také např. místem v paměti, kde se testovací cyklus provádí.

```
Checking 'hlt' instruction... Ok.
```

8. Po dokončení počáteční konfigurace vypíše operační systém Linux řádek, kterým identifikuje sám sebe. Na tomto řádku je informace o verzi systému, verzi překladače GNU C Compiler, kterým bylo jádro přeloženo, a kdy byl překlad realizován.

```
Linux version 1.3.55 (root@mousehouse) (gcc version 2.7.0)
Sun Jan 7 14:56:26 EST 1996
```

9. Jako další se nastartuje ovladač sériového portu, který zjistí informace o příslušném technickém vybavení. **Ovladač** (driver) je součást jádra operačního systému, která řídí nějaké zařízení, zpravidla periferní. Ovladač je odpovědný za detaily komunikace mezi procesorem a periferním zařízením. Ovladače umožňují, aby se programátoři mohli při psaní aplikací soustředit na vlastní aplikaci a nemuseli se starat o takové problémy, jako je například tisk na tiskárně.

```
Serial driver version 4.11 no serial optins enabled
tty00 at 0x03f8 (irq = 4) is a 16450
tty01 at 0x02f8 (irq = 3) is a 16450
tty02 at 0x03e8 (irq = 4) is a 16450
```

V tomto případě byly nalezeny tři sériové porty. Sériový port je ekvivalentní portu COM v operačním systému DOS. Jedná se o zařízení běžně používané ke komunikaci s modemy nebo myší.

Uvedený výpis sděluje, že sériový port s pořadovým číslem 0 (`COM1`) má adresu `0x03f8`. Když port přeruší činnost jádra (zpravidla proto, aby sdělil systému, že chce přenášet data), použije přerušení IRQ 4. Přerušení IRQ představuje další prostředek pro komunikaci mezi programovým vybavením a periferním zařízením. Každý sériový port má také svůj řídící integrovaný obvod. Často se používají obvody 16450, mohou se však také vyskytnou obvody 16550 nebo 820.**

10. Nyní přichází na řadu paralelní port. Paralelní port se nejčastěji připojuje k tiskárně a v operačním systému Linux začínají jména paralelních portů dvojicí písmen lp. lp je zkratkou pro Line Printer (řádková tiskárna), i když v poslední době by se mělo spíše jednat o zkratku pro Laser Printer (laserová tiskárna). Samozřejmě platí, že je operační systém Linux schopen komunikovat s každým typem paralelní tiskárny – jehličkovou, bublinkovou i laserovou.***

```
lp0 at 0x03bc, (polling)
```

Uvedená zpráva říká, že v systému byl nalezen jeden paralelní port a že pro něj bude použit standardní driver.

11. V dalším kroku identifikuje operační systém Linux ovladače pevných disků. Podle následující zprávy byly identifikovány dva pevné disky IDE:s

```
hda: WDC AC2340, 325MB, w/127KB Cache, CHS=1010/12/55
hda: WDC AC2850F, 814MB, w/64KB Cache, LBA, CHS=827/32/63
```

---

* Poznámka korektora: V novějších jádrech Linuxu je obsažen i test na procesory, které obsahují i tzv. chybu „F00F".

** Poznámka korektora: V novějších počítačích je téměř vždy obvod 1655A na 16550A.

*** Poznámka korektora: V poslední době se objevují i tzv. GDI-tiskárny, jejichž podpora není ještě zcela dokonalá, protože jejich výrobci nejsou ochotni poskytovat programátorům specifikaci komunikačního protokolu.

12. Dále provede jádro kontrolu disketových jednotek. Podle následujícího příkladu má počítač dvě disketové jednotky: jednotku „A“ pro diskety 5 1/4 palce a jednotku „B“ pro diskety 3 1/2 palce. Operační systém Linux přidělí disketové jednotce „A“ jméno `fd1` a disketové jednotce „B“ jméno `fd0`.

```
Floppy drive(s): fd0 is 1.44M, fd1 is 1.2M
floppy: FDC 0 is a National Semiconductor PC87306
```

13. Dalším ovladačem v případě mého počítače je ovladač SLIP. Vypíše následující zprávu o své konfiguraci:

```
SLIP: version 0.8.3-NET3.019-NEWTTY (dynamic channels, max=256)
  (6 bit encapsulation enabled)
CSLIP: code copyright 1989 Regents of the University of
  California
```

14. Dále jádro ověří všechny pevné disky, které byly identifikovány. Prohledá různé diskové oddíly a identifikuje ty části, na kterých se nachází operační systém, aby zabránil případné interferenci s jinými operačními systémy. V následujícím příkladě jsou identifikovány dva pevné disky: `hda` se čtyřmi oddíly a `hdb` s jedním oddílem:

```
Partition check:
hda: hda1 hda2 hda3 hda4
hdb: hdb1
```

15. V konečné fázi provede operační systém Linux připojení (mount) hlavního souborového systému platného pro kořenový oddíl (root partition). Kořenový oddíl je část pevného disku, ve které je umístěn operační systém. Když operační systém připojuje hlavní souborový systém, zpřístupní jej uživateli:

```
VFS: Mounted root (ext2 filesystem) readonly.
```

KAPITOLA 4

Příkazový interpret operačního systému Unix

Příkazy operačního systému Unix

Když se poprvé přihlásíte do operačního systému Unix, objeví se na obrazovce **výzva příkazového řádku** (prompt), která by mohla vypadat takto:

```
/home/larry#
```

Podle názvu „výzva příkazového řádku“ lze usuzovat, že systém na tomto místě očekává zadání příkazu. Každý příkaz operačního systému Unix představuje posloupnost písmen, čísel a znaků. Nepatří sem však mezery. Platnými příkazy operačního systému Unix jsou například: `mail`, `cat` nebo `CMU_is_Number-5`. Některé znaky není v příkazech povoleno používat – zmíníme se o nich později. Poznamenejme, že Unix má vlastnost označovanou jako „**case-sensitive**“ (doslova citlivý na malá a velká písmena).[1] To znamená, že `cat` a `Cat` jsou různé příkazy.

Příkazový řádek je zobrazován speciálním programem, který se nazývá **příkazový interpret** (shell). Příkazový interpret přijímá příkazy a realizuje je. Příkazové interprety mají svůj vlastní programovací jazyk a programy napsané v tomto jazyce se nazývají skripty příkazového interpretu (shell scripts).

Pro operační systémy Unix existují dva hlavní typy příkazových interpretůs: Bourne shell a C shell. (V současnosti existuje více než desítka různých příkazových interpretů – např. v distribuci Red Hat Linux 5.1 je pro každého uživatele k dispozici pět příkazových interpretů – `ash`, `bash`, `tcsh`, `zsh` a `pdksh` – ze kterých si může vybrat příkazem `chsh`). Příkazový procesor Bourne shell byl nazván podle svého autora, kterým je Steven Bourne. Ten vytvořil původní příkazový procesor pro operační systém Unix a většina příkazových procesorů vytvořených od té doby končí písmeny sh. Tím se indikuje, že další příkazové procesory jsou rozšířením původního příkazového interpretu. Dnes existuje spousta implementací původního příkazového interpretu Bourne shell.

1 Některé operační systémy, jako je OS/2 nebo Windows NT, sice zachovávají v názvech velká a malá písmena, ale jinak mezi nimi nerozlišují. V praxi platí, že se v operačních systémech Unix nepoužívají příkazy nebo obecná jména, která by se lišila pouze malými a velkými písmeny (například různé příkazy cat a Cat).

Jiný typ představují tzv. C shelly (původně je navrhl pan Bill Joy), jež se rovněž hojně používají. Obecně platí, že příkazové interprety typu Bourne shell se používají z důvodu kompatibility s originálním příkazovým interpretem, zatímco C shell se používá pro interaktivní zpracování příkazů.

Příkazové interprety typu C shell se vyznačují tím, že mají lepší vlastnosti pro interaktivní práci, ale mají poměrně nepružné vlastnosti z hlediska programátorského.

Operační systém Linux se distribuuje s příkazovým interpretem `bash`, který byl vytvořen nadací Free Software Foundation. `bash` je zkratkou pro Bourne Again Shell, což tradičně představuje spíše špatnou slovní hříčku typickou pro operační systém Unix. Jedná se o vylepšený příkazový procesor Bourne shell. Má podobné programátorské vlastnosti jako Bourne shell a také má spoustu dobrých vlastností převzatých z příkazového interpretu C shell. `bash` je implicitním příkazovým interpretem používaným v operačním systému Linux.

Když se přihlásíte do systému Linux, pak je to právě program `bash`, který zobrazuje výzvu příkazového řádku. Příkazový procesor `bash` vás bude provázet po celou dobu práce s počítačem.

Typické příkazy operačního systému Unix

Prvním příkazem, se kterým se seznámíme, je příkaz `cat`. Máte-li jej použít, napište `cat` a stiskněte klávesu Enter.

```
/home/larry# cat
```

Jestliže je nyní kurzor na následujícím řádku, pak jste zadali příkaz `cat` správně. Existuje několik způsobů, jak příkaz `cat` zadat. Některé jsou správné, jiné nikoliv.

- Předpokládejme, že jste se spletli:

  ```
  /home/larry# ct
  ct: command not found
  /home/larry#
  ```

Tímto způsobem vás příkazový interpret upozornil na to, že nemůže najít program, který by se jmenoval „`ct`“. Nabídne vám další výzvu příkazového řádku a očekává další příkaz. Připomeňme, že operační systém Unix je citlivý na velká a malá písmena. Příkaz `CAT` je rovněž chybný.

- Před příkaz můžete uvést libovolný počet mezer, například:[2]

  ```
  /home/larry# _ cat
  ```

  Takto zadaný příkaz je v pořádku a program `cat` se spustí.

- V příkazovém řádku také můžete pouze stisknou klávesu Enter. Ničeho se neobávejte, nic se nestane.

Předpokládejme, že jste spustili program `cat`. Asi jste zvědavi, co dělá. Pokud si myslíte, že se jedná o nějakou hru, pak asi budete zklamáni. Tento program je velice užitečným nástrojem, i když se to na první pohled nezdá. Napište jakýkoliv text a stiskněte Enter. Pak na obrazovce uvidíte:

```
/home/larry# cat
Help! I'm stuck in a Linux program!
Help! I'm stuck in a Linux program!
```

2 _ indikuje mezeru.

(Skloněným písmem je označeno to, co jsem v programu `cat` napsal.) Zdá se, že program pouze kopíruje text. To je někdy užitečné, ale program `cat` toho umí mnohem více. Pro tento okamžik program cat ukončíme a později si uvedeme příklady, kdy je mnohem užitečnější.

K ukončení většiny příkazů operačního systému Unix se používá kombinace kláves Ctrl+D[3]. Touto kombinací odešlete znak konce souboru EOF (end-of-file). Alternativně lze znak EOF považovat za konec textu, avšak v této knize jej budeme považovat za konec souboru. Znak EOF signalizuje programům operačního systému Unix, že jste ukončili zadávání dat. Pro program `cat` to znamená, že další data nemá zpracovávat a ukončí se.

Také si můžete vyzkoušet program sort. Jak napovídá jeho název, jedná se o program pro třídění. Jestliže zapíšete několik řádků a pak stisknete Ctrl+D, zobrazí se vám tyto řádky uspořádány. Programy tohoto druhu se nazývají **filtry**. Fungují tím způsobem, že nejdříve akceptují nějaký text, provedou s ním nějaké operace (například třídění) a pak modifikovaný text vypíší na obrazovce (nebo na jiném výstupním zařízení). Programy cat a sort jsou poněkud neobvyklé filtry, cat je neobvyklý v tom, že přečte text a neprovádí v něm žádné změny. Program `sort` je neobvyklý v tom, že čte řádky a neprovádí nic, dokud nepřečte znak EOF. Mnoho filtrů zpracovává data řádek po řádku – přečtou řádek, provedou nějaké modifikace a zobrazí řádek modifikovaný.

Pomozte si sami

V operačních systémech Unix existuje příkaz `man`,[4] jenž zobrazuje tzv. manuálové stránky. Originální manuálové stránky jsou pouze v anglickém jazyce, ale na jejich překladu se již pracuje. Zadejte například příkaz:

```
/home/larry# man cat
CAT(1)                                                    CAT(1)
NAME
   cat - concentrate files and print on the standard output
SYNOPSIS
   cat [-benstuvAET] [--number] [--number-nonblank]
   [--squeeye-blank] [--show-nonprinting] [--show-ends]
   [--show-tabs] [--show-all] [--help] [--version] [--file...]
DESCRIPTION
   This manual page documents the GNU version of cat.       cat
```

Uvedený výpis představuje asi jednu stránku informací o příkazu `cat`. Nepředpokládejte, že budete okamžitě všemu rozumět. Manuálové stránky zpravidla předpokládají, že má jejich čtenář jisté znalosti o operačním systému Unix, tedy znalosti, které vy zatím nemáte. Když čtete manuálovou stránku, pak se v dolním řádku objeví blok s textem jako „`--more--`“ nebo „`Line 1`“. Jedná se o nápovědu, že k danému tématu existuje více informací.

Man po zobrazení první stránky vyčkává, co se rozhodnete dělat dále. Chcete-li pokračovat dále v prohlížení manuálových stránek, stiskněte klávesu Spacebar – pak se vám zobrazí následující stránka. Pokud chcete program man ukončit, stiskněte klávesu Q. Pak se vrátíte do příkazového řádku příkazového procesoru, který bude očekávat zadání dalšího příkazu.

---

3 Podržte stisknutou klávesu Ctrl a stiskněte klávesu D.

4 Program `man` také zobrazí informace o systémovém volání, podprogramu, formátu souboru a mnoho dalších informací. V originálních verzích operačního systému Unix obsahuje manuálová stránka stejné informace jako tištěná dokumentace. Pro tento okamžik se spokojíme s nápovědou k syntaxi příkazu.

Programu man lze také předat jisté argumenty, jež řídí jeho funkce. Předpokládejme například, že si chcete přečíst informace o všem, co souvisí s formátem Postscript (Postscript je řídící jazyk pro tisk na tiskárnách od firmy Adobe). Pak zadejte příkaz `man -k ps` nebo `man -k Postscript`. Zobrazí se vám seznam všech příkazů, systémových volání a další dokumentace pojednávající o tomto tématu. Tento postup je velmi užitečný v případě, kdy chcete nalézt nějaký nástroj, ale nevíte, kde jej hledat, nebo nevíte, zda vůbec existuje.

Ukládání informací

Programy patřící do kategorie filtrů jsou velmi užitečné, zejména pokud patříte mezi pokročilejší uživatele. Zůstává zde však jeden problém. Jak ukládat, uchovávat a aktualizovat informace? Za tímto účelem nabízí operační systém Unix **soubory** a **adresáře**.

Adresář je něco jako pořadač, který místo svazků papíru obsahuje soubory. Velké pořadače mohou obsahovat několik dalších menších pořadačů. V operačním systému Unix se systém adresářů a souborů nazývá souborový systém. Zpočátku je každý souborový systém tvořen jediným adresářem, který se nazývá kořenový nebo také hlavní adresář. Uvnitř tohoto adresáře se mohou vyskytovat další adresáře a uvnitř nich se opět mohou vyskytovat adresáře (ty se někdy nazývají podadresáře). V každém adresáři mohou být uloženy soubory.[5]

Každý soubor a každý adresář má své vlastní jméno. Jméno může být buď krátké a může se shodovat se jménem adresáře nebo souboru uloženého jinde, nebo dlouhé (tzv. jméno včetně cesty), které je v rámci souborového systému na daném disku unikátní. Příkladem krátkého jména může být `joe`, zatímco odpovídajícím dlouhým jménem (t.j. jménem včetně cesty) je `/home/larry/joe`. Posloupnost znaků v dlouhém jménu ukončená znakem / se nazývá cesta. Cesta může být dekódována do posloupnosti adresářů. Následující příklad ilustruje, jak systém dospěje k souboru `/home/larry/joe`:

```
/home/larry/joe
```

První znak / indikuje kořenový adresář.

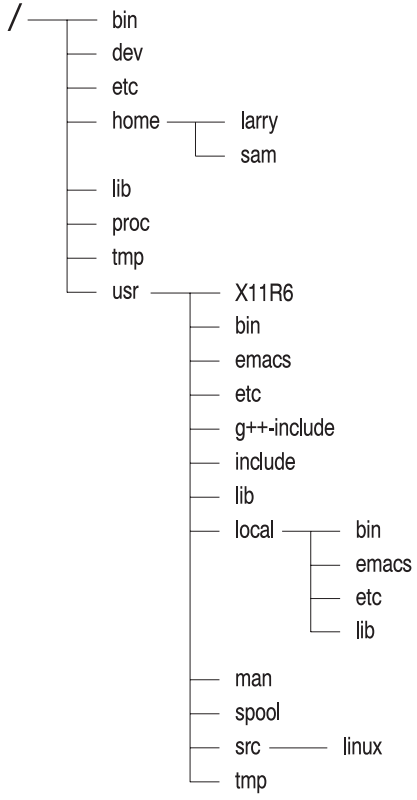
Následuje adresář se jménem `home`, který je podadresářem kořenového adresáře.

Další je adresář `larry`, který je podadresářem adresáře `home`.

Soubor `joe` je uvnitř adresáře `larry`. Cesta může odkazovat buď na adresář, nebo na soubor – `joe` tedy může být jak adresářem, tak i souborem. Všechny položky před krátkým jménem musejí být adresáře.

Existuje jednoduchý způsob, jak strukturu adresářů vizualizovat. Tato vizualizace se nazývá stromový diagram a většina uživatelů asi zná stromové diagramy z operačního systému DOS (vytvářejí je známé programy pro správu souborů, jako je NORTON COMMANDER). Stromový diagram můžete vytvořit programem tree nebo v programu Midnight Commander, Linuxové obdobě Norton Commandera. Typickou stromovou strukturu adresářů používanou v operačním systému Linux znázorňuje obrázek 4.1. Poznamenejme, že zde uvedený diagram není kompletní. Úplná struktura operačního systému Linux obsahuje přes 8000 souborů! Zrovna tak platí, že zde mohou být uvedeny adresáře, jež se nevyskytují ve vaší instalaci a naopak, ve vaší instalaci mohou být adresáře, které v uvedeném obrázku uvedeny nejsou.

5 Počet podadresářů může být omezen, ale v některých systémech nemusí. V operačním systému Linux jsem bez problémů vytvořil podadresáře až do úrovně 10.



Obrázek 4.1 – Typická (zkrácená) stromová struktura adresářů v operačním systému Unix

Prohledávání adresářů pomocí příkazu ls

Nyní máte jistě představu o tom, jak jsou v operačním systému Unix organizovány adresáře a soubory. Samozřejmě budete potřebovat nějaké nástroje, pomocí nichž budete moci se soubory a adresáři manipulovat. Prvním důležitým nástrojem je příkaz `ls`. Jedná se o příkaz, který zobrazuje seznam souborů. Nyní zadejte příkaz `ls` na úrovni příkazového řádku:

```
/home/larry# ls
/home/larry#
```

Žádný seznam se nezobrazil, což je v pořádku. Operační systém Unix pracuje přesně. Příkazu `ls` nebyly předány žádné parametry, proto se žádný seznam nezobrazil.

Jak jsme se již zmínili, při instalaci operačního systému Linux se nainstaluje více než 8000 souborů. Kde jsou tyto soubory uloženy? Abychom pochopili používání příkazu `ls`, musíme si vysvětlit pojem aktuální adresář. Podle nápovědy příkazového řádku poznáte, že aktuálním adresářem je `/home/larry`. Zde však nejsou uloženy žádné soubory, proto příkaz `ls` nezobrazil žádný seznam. Zkuste si zobrazit seznam souborů uložených v hlavním adresáři:

```
/home/larry# ls /
bin     etc     lostfound       proc    tmp
boot    home    misc            root    usr
dev     lib     mnt             sbin    var
/home/larry#
```

V předcházejícím příkazu „ls /“ jsme programu ls předali parametr „/“. Prvním slovem v příkazu je jméno příkazu a další slova jsou parametry. Obecně platí, že parametry modifikují funkce příkazů. V případě příkazu ls se jako první parametr uvádí jméno adresáře, jehož obsah má příkaz ls zobrazit. Některé příkazy mají speciální parametry, kterým se říká volby (options) nebo přepínače (switches). Vyzkoušejte si následující příkaz:

```
/home/larry# ls -F /
bin/    etc/    lostfound/      proc/   tmp/
boot/   home/   misc/           root/   usr/
dev/    lib/    mnt/            sbin/   var/
/home/larry#
```

Parametr `-F` se nazývá **volba**. Volba je speciální parametr, který začíná pomlčkou a modifikuje způsob provádění programu. V případě příkazu `ls` je parametr `-F` určen k rozlišení položek seznamu na adresáře, speciální soubory, programy a ostatní soubory. Každá položka končící znakem / je adresář. Později budeme hovořit o příkazu ls, který je opravdu velmi univerzální, podrobněji.

Nyní byste si měli udělat malé cvičení. Především se naučte, jak příkaz ls pracuje. Vyzkoušejte si zobrazit obsahy jiných adresářů podle obrázku 4.1. Některé z nich budou prázdné, jiné budou obsahovat spoustu souborů. Doporučuji, abyste si vyzkoušeli příkaz ls s parametrem -F i bez něj. Například adresář /usr/local by mohl vypadat takto:

```
/home/larry# ls /usr/local
Acrobat3        com     etc     include lib     man     share   teTeX
bin             doc     games   info            libexec sbin    src
/home/larry#
```

Druhé cvičení bude více obecné. Spousta příkazů v operačním systému Unix se spouští podobně jako příkaz `ls`. Také mají nějaké parametry a nějaké volby, jež se zpravidla uvádějí jako jednoznakové řetězce za pomlčkou. Na rozdíl od příkazu `ls` však některé příkazy vyžadují parametry a/nebo volby. K vyjádření syntaxe příkazů budeme používat následující formu:

`ls [-aRF] [`*`adresář`*`]`

Když budeme popisovat nový příkaz, použijeme k jeho definici podobný syntaktický zápis. Prvním slovem je příkaz (v tomto případě `ls`). To, co následuje za příkazem, jsou parametry. Volitelné parametry jsou uzavřeny do hranatých závorek. Tzv. meta-proměnné jsou vyznačeny skloněným písmem – jsou to slova, která musejí být nahrazena skutečnými parametry. V uvedeném příkladu je meta-proměnnou *`adresář`*. Ta by měla být nahrazena jménem skutečného adresáře.

V případě voleb platí jiná pravidla. V syntaktické definici příkazu jsou uzavřeny v hranatých závorkách. V příkazu můžete použít kteroukoliv volbu nebo kteroukoliv jejich kombinaci. V případě příkazu `ls` máte tedy možnost použít osm kombinací voleb. Porovnejte výstup příkazů `ls -R` a `ls -F`.

Aktuální adresář a příkaz cd

`pwd`

Používání adresářů může být poněkud těžkopádné – asi by se vám nelíbilo, kdybyste museli pokaždé vypisovat celou cestu, kdykoliv byste si chtěli zpřístupnit nějaký soubor. V operačním systému Unix je zaveden pojem aktuálního adresáře. V příkladech, které jsme doposud použili, je aktuálním adresářem `/home/larry`. Pokud chcete zjistit, jaký je skutečný aktuální adresář, zadejte příkaz `pwd` (print working directory). V některých případech zobrazuje nápověda příkazového řádku i jméno počítače. To je užitečné pouze tehdy, když je počítač zapojen do sítě, ve které se vyskytuje více počítačů.

```
mousehouse> pwd
/home/larry
mousehouse>
```

`cd [`*`adresář`*`]`

Jak jste se mohli přesvědčit, příkaz `pwd` zobrazí aktuální adresář.[6] Příkaz `pwd` je tedy velmi jednoduchý. Většina příkazů v operačním systému Unix funguje implicitně v aktuálním adresáři, který lze změnit pomocí příkazu `cd`. Vyzkoušejte si například příkazy:

```
/home/larry# cd /home
/home# ls -F
larry/ sam/ shutdown/ steve/ user1/
/home#
```

Pokud vynecháte volitelný parametr *`adresář`*, pak se navrátíte do vašeho domovského adresáře, kterým je v našem případě `/home/larry`. Jinak se dostanete do adresáře specifikovaného prostřednictvím parametru. Například:

```
/home# cd
/home/larry# cd /
/# cd home
/home# cd /usr
/usr# cd local/bin
/usr/local/bin#
```

Jak jste si asi všimli, příkaz `cd` umožňuje specifikovat buď **absolutní cestu,** nebo **relativní cestu**. Absolutní cesta začíná znakem / a musí obsahovat všechny adresáře vedoucí k výslednému adresáři, do kterého se chcete přepnout. Relativní cesta se vztahuje k vašemu aktuálnímu adresáři. V předcházejícím příkladu jsme v adresáři `/usr` zadali relativní adresář `local/bin` – `local` je adresář pod adresářem `usr` a `bin` je adresář pod adresářem `local`. Podobně jsme použili relativní adresář v příkazu `cd home`.

V operačním systému Unix (podobně jako v operačním systému DOS) existují dva speciální adresáře, jež se používají pouze jako relativní. Jsou to adresáře „`.`“ a „`..`“. Adresář „`.`“ specifikuje aktuální adresář a adresář „`..`“ specifikuje nadřazený adresář. Jedná se tedy o zkratky, které existují v každém adresáři, ale ne vždy mají přesně ten význam, jak jsme jej popsali. Například hlavní adresář má jako nadřazený adresář sebe sama.

6 Někdy se také používá termín pracovní adresář. V této knize se budeme držet termínu aktuální adresář.

Soubor `./chapter-1` v aktuálním adresáři by měl být identický se souborem `chapter–1`. Některé příkazy vyžadují zadání typu `./chapter-1`, ale to není častý případ. Ve většině případů jsou `./chapter-1` a `chapter-1` identické specifikace souboru.

Adresář „`..`“ je velmi užitečný při „pohybu“ v adresářové struktuře směrem nahoru:

```
/usr/local/bin# cd ..
/usr/local# ls -F
archives/      bin/    emacs@ etc/    ka9q/   lib/    tcl@
/usr/local# ls -F ../src
cweb/          linux/ xmris
/usr/local#
```

V tomto případě jsme se přepnuli do nadřazeného adresáře pomocí `cd ..` a pak jsme zobrazili obsah adresáře `/usr/src` z adresáře `/usr/local` pomocí příkazu `ls -F ../src`.

Pro domovský adresář existuje zkratka ~. Vyzkoušejte si následující příkaz:

```
/usr/local# ls -F ~/
/usr/local#
```

Opět jsme se přesvědčili, že v domovském adresáři nejsou žádné soubory. Užitečnost zkratky ~ pro domovský adresář vynikne zejména v případě, kdy začneme se soubory manipulovat.

Vytváření a odstraňování adresářů

`mkdir` *`adresář1`* [*`adresář2 ... adresářN`*]

Vytváření vlastních adresářů je v operačním systému Unix opravdu jednoduché a představuje velmi užitečný nástroj pro organizaci a údržbu souborů. K vytvoření adresáře slouží příkaz `mkdir` (make directory).

Uveďme si jednoduchý příklad, abychom pochopili, jak příkaz `mkdir` funguje.

```
/home/larry# ls -F
/home/larry# mkdir report-1993
/home/larry# ls -F
report-1993/
/home/larry# cd report-1993
/home/larry/report-1993#
```

Příkaz `mkdir` může akceptovat více než jeden parametr. Předpokládá, že každý z nich představuje jméno adresáře, který se má vytvořit. Opět můžete specifikovat celou cestu nebo relativní adresář. V předcházejícím příkladu jsme použili relativní adresář `report-1993`.

```
/home/larry/report-1993# mkdir /home/larry/report-1993/chap1
  ~/report-1993/chap2
/home/larry/report-1993# ls -F
chap1/  chap2/
/home/larry/report-1993#
```

`rmdir` *`adresář1`* [*`adresář2 ... adresářN`*]

Jakýmsi opakem k příkazu `mkdir` je příkaz `rmdir` (remove directory). `rmdir` funguje přesně jako `mkdir`. Podívejte se na následující příklad:

```
/home/larry/report-1993# rmdir chap1 chap3
rmdir: chap3: No such file or directory
/home/larry/report-1993# ls -F
chap2/
/home/larry/report-1993# cd ..
/home/larry# rmdir report-1993
rmdir: report-1993: Directory not empty
/home/larry#
```

Jak jste si zřejmě všimli, příkaz `rmdir` odmítnul odstranit adresář, který buď neexistuje nebo není prázdný. Uvědomte si, že adresář `report-1993` obsahuje podadresář `chap2`, proto nelze adresář `report-1993` odstranit.

Manipulace se soubory

Práce s adresáři je sice zajímavá, ale stále neřeší náš problém s ukládáním, aktualizací a udržováním informací. Tento problém se řeší prostřednictvím **souborů**.

Vytváření a editování souborů se budeme věnovat v několika následujících kapitolách.

Základními příkazy pro manipulaci se soubory jsou v operačním systému Unix příkazy `cp` (copy), `mv` (move) a `rm` (remove).

Příkaz pro kopírování souborů cp

```
cp [-i] zdroj cíl
cp [-i] soubor1 soubor2 ... souborN cílový adresář[7]
```

Příkaz `cp` je v operačním systému Unix velmi důležitým a výkonným příkazem. V jedné sekundě vám umožní přemístit tolik informací, kolik jich středověký mnich přemístil za celý rok.

Jestliže nemáte na svém pevném disku dostatek prostoru, buďte při používání příkazu `cp` opatrní. Nikdo nechce vidět zprávu „Disk full“, když pracuje s důležitými soubory. Příkaz `cp` je dále schopen přepsat důležité soubory bez jakéhokoliv varování – tomuto nebezpečí se budeme věnovat podrobněji.

Nejprve se zaměříme na první definici příkazu `cp`. Prvním parametrem příkazu `cp` je soubor, který se má kopírovat (tzv. zdrojový soubor, source file). Druhým parametrem je soubor, který má být kopií prvního (tzv. cílový soubor, destination file). Při kopírování můžete vytvářet soubory s novým jménem nebo můžete soubory se stejným jménem kopírovat do jiných adresářů. Vyzkoušejte si následující příklady:

```
/home/larry# ls -F /etc/passwd
/etc/passwd
/home/larry# cp /etc/passwd .
/home/larry# ls -F
passwd
/home/larry# cp passwd frog
/home/larry# ls -F
frog passwd
/home/larry#
```

7 Příkaz `cp` má v předloze dva řádky, neboť význam druhého parametru může být rozdílný v závislosti na počtu parametrů.

Prvním příkazem `cp` se z adresáře `/etc` zkopíroval soubor `passwd` (jenž obsahuje jména všech uživatelů systému Unix spolu se zakódovanými hesly) do mého domovského adresáře. Toto platí pouze za předpokladu, že nepoužíváme tzv. shadow passwords, kdy jsou zakódovaná hesla uložena v souboru `/etc/shadow`. Příkaz `cp` neruší zdrojový soubor, proto jsem neudělal nic, co by nějak poškodilo systém. Nyní existují v mém systému dvě kopie souboru `passwd`, jedna v adresáři `/etc` a druhá v mém domovském adresáři `/home/larry` a obě kopie mají jméno `passwd`.

Třetí kopii jsem vytvořil příkazem `cp passwd frog`. Nyní jsou v mém systému tři kopie souboru: `/etc/passwd`, `/home/larry/passwd` a `/home/larry/frog`. Posledně vytvořená kopie má jiné jméno, ale obsahuje stejná data.

Příkaz `cp` je schopen kopírovat soubory mezi adresáři, a to v tom případě, kdy je prvním parametrem jméno souboru a druhým parametrem je jméno adresáře. Pak zůstane jméno cílového souboru shodné se jménem zdrojového souboru.

Kopírovat soubory a přitom měnit jejich jména lze jen tehdy, když se v příkazu `cp` jako parametry uvedou jména souborů. S příkazem `cp` je spojeno jedno nebezpečí. Když například zadáte příkaz `cp /etc/passwd /etc/group`, vytvoří příkaz `cp` nový soubor `group`, jehož obsah bude identický s obsahem souboru `/etc/passwd`. Pokud by však soubor `/etc/group` již existoval, pak jej přepíšete bez jakéhokoliv varování a bez možnosti starou verzi souboru `/etc/group` obnovit.

Podívejme se na další příklad použití příkazu `cp`:

```
/home/larry#   ls -F
frog passwd
home/larry#    mkdir passwd_version
home/larry#    cp frog passwd passwd_version
home/larry#    ls -F
frog           passwd  passwd_version/
home/larry#    ls -F passwd_version
frog passwd
home/larry#
```

Jak jsem nyní použil příkaz `cp`? Je evidentní, že příkaz `cp` může akceptovat více než dva parametry (nyní jsem příkaz `cp` použil podle druhé definice). Příkaz `cp` v předcházejícím příkladu zkopíroval všechny uvedené soubory (`frog` a `passwd`) do adresáře `passwd_version`, který je uveden jako poslední parametr. V podstatě platí, že příkaz `cp` je schopen akceptovat jakýkoliv počet parametrů, přičemž prvních n-1 považuje za jména souborů a poslední považuje za jméno adresáře, do kterého se mají uvedené soubory kopírovat.

Jestliže kopírujete v daném okamžiku více než jeden soubor, pak jej nemůžete přejmenovat – kopírované soubory si uchovají svá původní jména. Co se stane, když zadáte příkaz `cp frog passwd toad`, kde `frog` a `passwd` jsou soubory a přitom `toad` není adresář? Vyzkoušejte si takový příkaz a uvidíte.

Odstranění souborů pomocí příkazu rm

`rm [-i] soubor1 soubor2 ... souborN`

Nyní, když jsme se naučili, jak kopírovat a vytvářet soubory (později si probereme jiné způsoby vytváření souborů), bude jistě užitečné vědět, jak soubory rušit. Ve skutečnosti je to velmi jednoduché. Příkaz pro odstraňování souborů má název `rm` a zruší všechny soubory, jejichž jména se uvedou jako parametry.

Například:

```
/home/larry# ls -F
frog passwd passwd_version/
/home/larry# rm frog toad passwd
rm: toad: No such file or directory
/home/larry# ls -F
passwd_version/
/home/larry#
```

Jak asi vidíte, příkaz `rm` je velmi nevlídný. Nejen, že se vás nezeptá, zda má uvedené soubory skutečně zrušit, ale provede zrušení existujících souborů, i když vlastně příkaz nebyl správně zadán (soubor `toad` neexistoval). Příkaz `rm` může být skutečně nebezpečný. Podívejme se na rozdíl mezi následujícími dvěma posloupnostmi příkazů:

```
/home/larry# ls -F
toad frog/
/home/larry# ls -F frog
toad
/home/larry# rm frog/toad
/home/larry#
```

```
/home/larry# rm frog toad
rm: frog is a directory
/home/larry# ls -F
frog/
/home/larry#
```

Pátý řádek v první posloupnosti a první řádek v druhé se liší jediným znakem. Uvedený příklad má ilustrovat, že opomenutí jediného znaku může v operačním systému Unix způsobit, že provedete něco úplně jiného, než jste původně chtěli. Proto je velmi důležité, abyste raději několikrát zkontrolovali zapsaný příkaz, než stisknete klávesu Entr.

Přesouvání souborů pomocí příkazu mv

`mv [-i] *starý nový*`
`mv [-i] *soubor1 soubor2 ...souborN adresář*`

Nakonec se podívejme na příkaz pro tzv. přesouvání souborů. Příkaz má název `mv` a je podobný příkazu `cp`. Na rozdíl od příkazu `cp` však zdrojové soubory po jejich zkopírování odstraňuje. Příkaz `mv` je tedy jakousi kombinací příkazů `cp` a `rm`. Uveďme si na následujícím příkladu, jak příkaz `mv` funguje:

```
/home/larry# cp /etc/passwd .
/home/larry# ls -F
passwd
/home/larry# mv passwd frog
/home/larry# ls -F
frog
/home/larry# mkdir report
/home/larry# mv frog report
/home/larry# ls -F
```

```
report/
/home/larry# ls -F report
frog
/home/larry#
```

Jak jste si pravděpodobně všimli, příkaz `mv` soubor přejmenuje, pokud je druhým parametrem jméno souboru. Jestliže je druhým parametrem jméno adresáře, pak se soubor přesune do nového adresáře a přitom zůstane uchováno jeho původní jméno.

Podobně jako v případě příkazu `cp` musíte být při používání příkazu `mv` velmi opatrní. Příkaz neprovádí kontrolu, zda cílový soubor existuje nebo ne. Kdyby například v mém adresáři `report` již existoval soubor `frog`, pak by se příkazem `mv frog report` nejdříve zrušil soubor `/report/frog` a pak by se nahradil obsahem souboru `~/frog`.

Ve skutečnosti existuje možnost, jak přimět příkazy `rm`, `cp` a `mv`, aby vás upozorňovali na možnost přepsání či zrušení souborů. Všechny tyto příkazy akceptují volbu `-i`. Po uvedení této volby bude uživatel před každým zrušením souboru upozorněn. Dokonce můžete použít tzv. **alias** (druhé jméno) a zařídit, aby vás například příkaz `rm` upozorňoval na zrušení souboru, aniž uvedete volbu `-i`. O tom však budeme diskutovat až v oddíle Vytváření druhých jmen v kapitole 9.

KAPITOLA 5

Systém X Window

Tato kapitola se týká pouze uživatelů systému X Window. Jestliže máte před sebou barevnou grafickou obrazovku s mnoha okny a kurzorem, kterým lze pohybovat pouze prostřednictvím myši, pak jste zřejmě uživateli systému X Window. Pokud však máte před sebou obrazovku s bílým textem na černém pozadí, pak momentálně systém X Window nemáte aktivován. Budete-li chtít systém X Window aktivovat, přečtěte si následující oddíl.

Spuštění a ukončení systému X Window

Spuštění systému X Window

Systém X Window můžete spustit i v případě, kdy se automaticky nenastartuje ihned po zavedení operačního systému. Existují dva příkazy, kterými lze nastartovat systém X Window: `startx` a `xinit`. Nejdříve vyzkoušejte příkaz `startx`. Pokud příkazový procesor ohlásí, že takový příkaz neexistuje, pak zkuste `xinit`. Pokud ani v tomto případě nedojde ke spuštění systému X Window, pak jej pravděpodobně nemáte nainstalován. Prostudujte si příslušnou dokumentaci.

Jestliže se příkaz spustí, ale po nějakém čase se opět vrátíte do příkazového řádku příkazového procesoru, pak je systém X Window sice instalován, ale není nakonfigurován. I v tomto případě si budete muset důkladně přečíst dokumentaci k instalaci systému X Window.

Ukončení systému X Window

V závislosti na tom, jak je váš systém X Window konfigurován, existují dva možné způsoby, jak systém X Window ukončit. První závisí na tom, zda váš systém X Window řídí správce oken či nikoliv. Pokud ano, můžete systém X Window ukončit prostřednictvím nabídky (viz oddíl Nabídky). Chcete-li si zobrazit nabídku, klepněte myší na pozadí pracovní plochy.

Pak se vám mezi jinými objeví položka „Exit Window Manager“ nebo „Exit X“ nebo jiná položka obsahující slovo „Exit“. Pokuste se najít tuto položku (počítejte s tím, že zde může existovat více než jedna nabídka – vyzkoušejte různá tlačítka myši) a ukončete systém X Window.

Druhá metoda ukončení systému X Window spočívá v tom, že se k řízení systému X Window využije speciálního okna `xterm`. V takovém případě byste měli najít okno nazvané „login“ nebo „system `xterm`“. Přesuňte kurzor myši do tohoto okna a zadejte příkaz `exit`.

Jestliže se systém X Window automaticky spustil, když jste se přihlásili do systému, pak by vás jedna z výše uvedených metod ukončení systému X Window měla automaticky odhlásit. Jestliže jste však systém X Window nastartovali z příkazového řádku, pak se do tohoto příkazového řádku po ukončení systému X Window opět vrátíte (pokud se chcete odhlásit, zadejte příkaz `logout`).

Co je systém X Window?

Systém X Window je distribuované grafické uživatelské prostředí původně vyvinuté v akademické instituci Massachusetts Institute of Technology. Později byl předán skupině distributorů s názvem „The X Consortium“. Zde se dodnes udržuje a nadále vyvíjí.

Systém X Window (někdy zkracován jako „X“[1]) se každých několik málo let distribuuje v nové verzi, uváděné jako „release“. Poslední verze má označení X11R6, tedy „release“ 6. Číslo 11 značí hlavní verzi systému X Window, avšak toto označení je trvalé, protože se nová verze neplánuje.

V souvislosti se systémem X Window existují dva důležité pojmy, se kterými byste se měli seznámit. Program, jenž běží pod systémem X Window, se nazývá **klient**. Například `xterm` je klient, který se spustí v okamžiku, kdy se přihlašujete do systému. Program, který poskytuje služby ostatním programům typu klient, se nazývá **server**. Server například kreslí okno pro program `xterm` a zajišťuje komunikaci s uživatelem.

Protože jsou server a klient dva odlišné programy, je možné zajistit, aby server běžel na jednom počítači a klient na jiném. Vzhledem k tomu, že grafické uživatelské rozhraní dodržuje jistý standard, můžete v systému X Window spustit program na vzdáleném počítači (třeba na druhém konci světa) a přitom můžete grafické rozhraní tohoto programu vidět na svém počítači.

Dalším důležitým termínem, se kterým byste se měli seznámit, je **správce oken** (window manager). Jedná se o speciální program typu klient, který určuje pozici jednotlivých oken na pracovní ploše systému X Window a řídí způsoby, kterými uživatel například přemisťuje okna. Samotný server v podstatě pro uživatele nic nedělá. Pouze představuje rozhraní mezi uživatelem a klientem.

Co se nachází na pracovní ploše X Window

Když poprvé nastartujete systém X Window, zároveň se nastartuje řada dalších programů. Jako první se nastartuje server. Další v pořadí se nastartují některé programy typu klient – jejich pořadí a typ závisí na distribuci a není standardizován. Je však pravděpodobné, že mezi těmito klienty je správce oken (buď program `fvwm` nebo `twm`), terminálové okno `xterm` a hodiny `xclock`.

Program XClock

`xclock [-digital] [-analog] [-update` *seconds*`] [-hands` *color*`]`

Program `xclock` (hodiny) funguje přesně tak, jak asi očekáváte. Zobrazuje ciferník s vteřinovou ručičkou a malou i velkou ručičkou v malém okně.

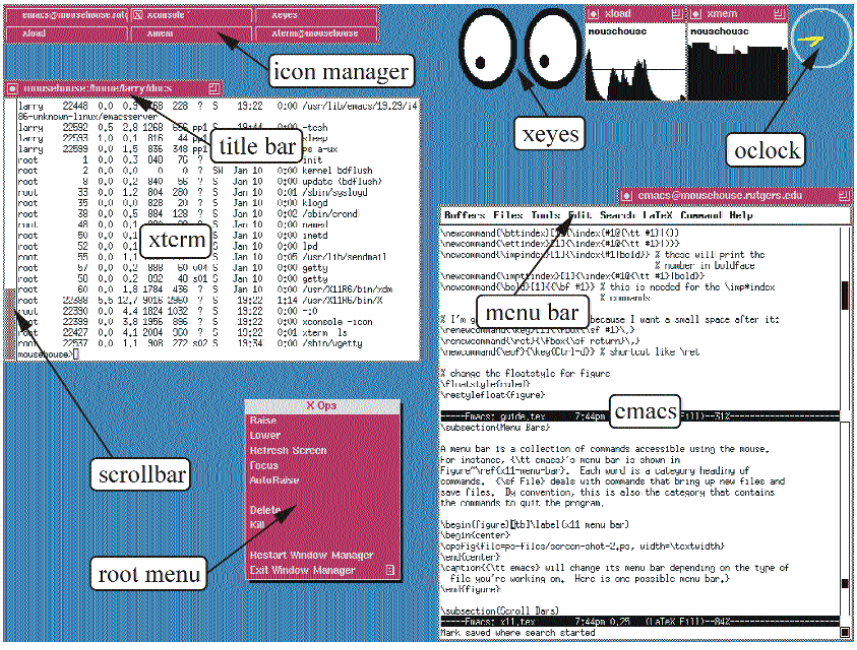
Prostřednictvím myši nemůžete vlastnosti okna s hodinami příliš modifikovat. Nanejvýš můžete měnit jeho velikost. Pokud však program xclock spustíte z příkazového řádku, pak můžete prostřednictvím jeho parametrů nastavit různé vlastnosti. Když zadáte příkaz `xclock -digital`, zobrazí se digitální hodiny. Pomocí parametru `-update` můžete nastavit, zda se má vteřinová ručička aktualizovat každou sekundu (`-update 1`) nebo například každých pět sekund (`update -5`).

Chcete-li vědět více o programu `xclock` a jeho parametrech, podívejte se do manuálových stránek. Stačí v příkazovém řádku zadat `man xclock`. Budete-li chtít spustit více programů `xclock` najednou, přečtěte si oddíl Současný běh úloh, ve kterém budeme diskutovat o provozování více úloh současně.

Jestliže máte program `xclock` spuštěn na popředí (což je běžný způsob v provozování programů) a chcete jej ukončit, stiskněte kombinaci kláves Ctrl-C.

---

1 Existuje několik přijatelných způsobů, jak nazývat systém X Window. Nejčastěji se vyskytuje zkratka „X“ nebo název „X Window“.



Obrázek 5.1 – Příklad standardní obrazovky systému X Window. V tomto příkladu uživatel spustil program twm. Standardní hodiny byly nahrazeny jiným programem zvaným `oclock`.

Program xterm

Okno s příkazovým řádkem uvnitř (tedy v našem případě okno s výzvou `/home/larry#`) je řízeno programem, jenž se nazývá xterm. Což je zvláštní a přitom komplikovaný program. Na první pohled se zdá, že toho moc neumí, ale jeho prostřednictvím lze vykonat spoustu užitečné práce. Emuluje terminál, a proto v něm lze spouštět všechny textově orientované aplikace operačního systému Unix. Také obsahuje vyrovnávací paměť uchovávající jistou množinu příkazů – proto se můžete snadno vracet k dříve zadaným příkazům (podrobněji se tímto tématem budeme zabývat v oddílu Posuvné lišty).

Celá tato kniha je zaměřena na příkazy zadávané v příkazovém řádku terminálu, proto je program xterm v systému X Window tak důležitý. Chcete-li něco zadávat v okně tohoto programu xterm, musí být toto okno aktivní. To znamená, že musíte do okna terminálu přesunout kurzor myši. Způsob, kterým se nastavuje aktivní okno obecně, závisí na správci oken.

Program xterm představuje jednu z možností, jak v systému X Window spouštět více programů současně. Programy běžící v systému X Window jsou standardními programy operačního systému Unix, proto mohou být spuštěny z terminálového okna. Protože dlouhodobé programy spouštěné z terminálového okna zablokují program `xterm` po celou dobu svého běhu, spouští se takové programy zpravidla na pozadí. Více informací o tomto tématu uvedeme v oddílu Současný běh úloh.

Správce oken

V operačním systému Linux existují dva běžně používané programy pro správu oken. První z nich, twm, je zkratkou pro „Tab Window Manager“. Druhý program je menší a má název fvwm („F(?) Virtual Window Manager“). Oba programy jsou konfigurovatelné, což znamená, že si uživatel může definovat význam ovládacích klíčů a způsob práce s myší.

Konfiguraci programu twm je věnován oddíl Konfigurace programu twm a konfiguraci programu fvwm probereme v oddílu Konfigurace programu fvwm.

Jak se vytvářejí nová okna

Při vytvoření nového okna existují tři možné varianty, jež realizuje správce oken. V poslední době se objevuje spousta nových správců oken, např. `fvwm95`, `gnome` nebo `kwm`. Správce oken může být nakonfigurován tak, že se zobrazí rám nového okna a uživatel má možnost okno umístit kamkoliv na obrazovku. Tento způsob umístění okna se nazývá **ruční umístění** (manual placement). Jestliže se před vámi objeví rám okna, jednoduše jej pomocí kurzoru myši nastavte na požadovanou pozici a stiskněte levé tlačítko.

Je také možné, že správce oken umístí nové okno na pracovní ploše systému X Window dle „svého uvážení“. Takové umístění okna se nazývá **náhodné umístění** (random placement).

Třetí varianta spočívá v tom, že se okno náležící jisté aplikaci pokaždé otevře na stejné pozici. Správce oken může být nakonfigurován tak, aby pro vybrané aplikace otevíral okna s předem definovanou velikostí na předem definované pozici.

Jako příklad může sloužit program `xclock`, kdy asi budete chtít zobrazovat hodiny v horním levém rohu pracovní plochy systému X Window.

Aktivní okno

Správce oken řídí spoustu důležitých nastavení. Vás bude určitě zajímat, jak nastavit dané okno (například terminálové okno) tak, abyste v něm mohli zadávat příkazy. Aktivní okno je nejčastěji určeno pozicí kurzoru myši. Jestliže je kurzor myši umístěn na jednom z terminálových oken,[2] pak v tomto okně můžete zadávat příkazy z klávesnice. V tom spočívá rozdíl od jiných operačních systémů, jako je OS/2, Microsoft Windows nebo Macintosh, kde musíte na dané okno (chcete-li, aby bylo aktivní) klepnout myší. V systému X Window obvykle platí, že když kurzor myši opustí rámec okna, pak v tomto okně již nemůžete zadávat příkazy.

Poznamenejme, že oba správce oken (tedy program twm a fvwm) lze konfigurovat tak, že se aktivace oken bude realizovat stejně jako například v operačním systému OS/2. Popis konfigurace správce oken najdete v příslušné dokumentaci, nebo můžete požadované konfigurace dosáhnout metodou pokusů a omylů.

Přemisťování oken

V systému X Window lze rovněž konfigurovat způsob, kterým mají být okna přemisťována. Moje osobní konfigurace programu twm obsahuje tři způsoby, jak posouvat okna po pracovní ploše. Nejznámější způsob spočívá v tom, že se kurzor myši umístí na titulní (hlavní) lištu okna, stiskne kterékoliv tlačítko (levé, pravé nebo střední[3]), a pak se okno přesune na novou pozici. Je ovšem velmi pravděpodobné, že je váš systém X Window konfigurován tak, aby se okna přemisťovala pomocí levého tlačítka (tak, jak je tomu u ostatních operačních systémů).

---

2 V daném okamžiku můžete spustit několik programů `xterm` současně.

3 Některé osobní počítače mají dvoutlačítkovou myš. Pak můžete prostřední tlačítko simulovat současným stisknutím pravého a levého tlačítka.

Jiný způsob přemisťování oken může spočívat v tom, že se využívá některé klávesy na klávesnici. Například moje vlastní konfigurace umožňuje stisknout klávesu Alt, nastavit kurzor myši kamkoliv do rámce okna a pak po stisknutí levého tlačítka okno přesunout na novou pozici.

I zde platí, že si konfiguraci správce oken můžete měnit metodou pokusů a omylů, nebo si prostudujte příslušnou dokumentaci. Budete-li se pokoušet interpretovat konfigurační soubor správce oken, přečtěte si oddíl Konfigurace programu twm nebo oddíl Konfigurace programu fvwm.

Překrývání oken

Protože v systému X Window může být otevřeno několik oken najednou, má smysl hovořit o překrývání oken. Přestože okna a samotná pracovní plocha systému X Window jsou dvourozměrné, mohou se okna navzájem překrývat, jako by představovaly třírozměrné útvary. To znamená, že vybrané okno může být zobrazeno v popředí a částečně nebo úplně překrývat okna na pozadí. V souvislosti s překrýváním oken existuje několik operací:

- **Vyzvednutí** (raising) okna do popředí. Toho lze zpravidla dosáhnout klepnutím jistým tlačítkem myši na titulní lištu vybraného okna. Podle konfigurace správce oken to může být kterékoliv tlačítko; také je možné dosáhnout vyzvednutí okna do popředí pomocí více než jednoho tlačítka.
- **Zasunutí** (lowering) okna do pozadí. Toho lze obvykle dosáhnout klepnutím jiného tlačítka myši na titulní lištu okna. Dokonce lze nakonfigurovat správce oken tak, že k vyzvednutí i zasunutí okna se použije totéž tlačítko myši – je-li okno v popředí, pak se zasune do pozadí a je-li v pozadí, vyzvedne se do popředí.
- **Cyklické** procházení všech oken. Správce oken může být konfigurován tak, že po stisknutí jisté klávesy se postupně objevuje ve stanoveném pořadí každé otevřené okno v popředí.

Transformace okna do ikony

Nyní se podívejme na další operace, které umožňují transformovat (nebo také skrýt) okno do ikony. V závislosti na konfiguraci správce oken lze tohoto efektu dosáhnout několika různými způsoby. Při používání správce oken twm si většina lidí nakonfiguruje tzv. **správce ikon** (icon manager). Jedná se o speciální okno, které obsahuje seznam jmen všech ostatních oken otevřených na pracovní ploše. Pokud klepnete jistým tlačítkem myši na konkrétní jméno, dané okno se transformuje na ikonu. Okno je stále aktivní, ale nevidíte jej. Pomocí jiného tlačítka můžete provést inverzní operaci – ikona se zpět transformuje na okno.

Uvedené vlastnosti mohou být velmi užitečné. Předpokládejme například, že máte otevřenu spoustu terminálových oken pro příležitostnou komunikaci se vzdálenými počítači. Kdyby zůstala všechna okna otevřena, měli byste na pracovní ploše nepřehledno. Protože však s těmito terminálovými okny pracujete pouze příležitostně, můžete je transformovat do ikon a získat tak větší přehled. Jediné malé nebezpečí spočívá v tom, že zapomenete na některé okno transformované do ikony a zbytečně si otevřete pro stejný účel nové okno.

Pokud jde o umisťování ikon, lze správce oken nakonfigurovat tak, aby se ikony automaticky řadily na spodní okraj pracovní plochy.

Jak měnit velikost oken

V systému X Window existuje několik způsobů, jak měnit velikost oken. Tyto způsoby opět závisejí na konfiguraci správce oken. První a asi nejpoužívanější metoda spočívá v tom, že klepnete na rámeček ohraničující okno levým tlačítkem myši, držíte je stisknuté a rozměr okna nastavíte na požadovanou velikost (tento způsob je běžný i u ostatních operačních systémů, jako je Microsoft Windows nebo OS/2).

Další metoda je založena na speciálním tlačítku umístěném v titulní liště okna. Na obrázku 5.1 si můžete všimnout malého tlačítka na pravé straně titulní lišty každého okna. Stačí klepnout levým tlačítkem myši na toto tlačítko a nastavit rozměr okna na požadovanou velikost. Jakmile má okno příslušné rozměry, levé tlačítko myši uvolněte.

Nastavení maximální velikosti okna

Většina správců oken podporuje tzv. maximalizaci oken, tedy nastavení rozměrů okna tak, aby pokrylo celou obrazovku. Při použití správce oken `twm` lze dokonce nastavit takovou konfiguraci, aby bylo možné maximalizovat pouze vertikální rozměr, horizontální rozměr nebo oba rozměry najednou. Tento proces je v programu `twm` označován jako „zooming" (zvětšování), ale častěji se dává přednost termínu „maximization". Různé aplikace reagují na změnu rozměrů okna různě. Například `xterm` vám poskytne větší pracovní prostor a nezvětšuje přitom velikost fontů.

Obecně lze říci, že proces maximalizace oken není bohužel v systému X Window standardizován.

Nabídky

Dalším úkolem správce oken je řídit funkci nabídek a umožnit tak uživateli rychle realizovat úkony, které se stále dokola opakují. Například lze vytvořit nabídku, která umožní automaticky a rychle spustit editor emacs nebo terminál `xterm`. Obecně platí, že programy běžící v systému X Window můžete spouštět buď z terminálového okna, což je pomalejší a nepohodlné, nebo z vhodně nakonfigurovaných nabídek, což je rychlé a pohodlné.

Různé nabídky mohou být zpřístupněny klepnutím myši v základním okně (často se pro toto okno používá označení pracovní plocha), tedy okně, které nelze přemisťovat a které obsahuje všechna ostatní okna. Pozadí základního okna je implicitně šedé[4]. Chcete-li vyvolat nabídku, stačí umístit kurzor myši na pozadí základního okna a stisknout tlačítko. V nabídce si opět pomocí kurzoru myši vyberte požadovanou položku (aniž uvolníte tlačítko myši) a uvolněním tlačítka ji spusťte.

Atributy systému X Window

Existuje spousta programů, jež využívají vlastností systému X Window. Některé programy, jako je editor emacs, mohou být spuštěny jako textově orientované aplikace a zároveň mohou být spuštěny v prostředí systému X Window. Existuje však řada programů, které mohou běžet pouze v prostředí X Window.

Geometrie

Existuje několik aspektů, které mají programy běžící pod systémem X Window společné. První z nich lze označit jako geometrii. Atributy geometrie popisují velikost a umístění okna a tvoří je čtyři komponenty.

- Horizontální rozměr, zpravidla měřený v grafických bodech. (Grafický bod je nejmenší jednotka na obrazovce, které lze přiřadit barvu. Systém X Window běžně pracuje s rozlišovací schopností 1024 bodů horizontálně a 768 bodů vertikálně.) Některé aplikace (`xterm` nebo `emacs`) měří velikost okna ve znacích. Například šířka okna je dána počtem znaků na řádku (nejčastěji osmdesát), který může aplikace zobrazit.

4 Existuje program xfishtank, který na pozadí základního okna simuluje akvárium a také spousta dalších programů pro změnu pozadí v systému X Window.

- Vertikální rozměr, opět obvykle měřený v grafických bodech. Rovněž je možné stanovit vertikální rozměr ve znacích.
- Horizontální vzdálenost od okrajů obrazovky. Například číslo `+35` znamená, že levý okraj okna bude vzdálen 35 grafických bodů od levého okraje obrazovky. Na druhé straně číslo `-50` bude znamenat, že pravý okraj okna bude vzdálen padesát grafických bodů od pravého okraje obrazovky. Obecně platí, že není možné inicializovat okno mimo rámec obrazovky, i když je pak možné okno mimo rámec obrazovky přesunout.
- Vertikální vzdálenost od horního nebo dolního okraje obrazovky. Kladná vertikální vzdálenost je měřena od horního okraje obrazovky a záporná je měřena od spodního okraje obrazovky. Všechny čtyři komponenty definující geometrii okna se uvádějí prostřednictvím jediného řetězce. Například `503x73-78+0` znamená, že okno bude mít šířku 503 grafických bodů, výšku 73 grafických bodů a že bude umístěno vpravo bezprostředně pod horním okrajem obrazovky. Obecný tvar řetězce pro specifikaci geometrie okna je tedy: `hsizexvsize+hplace+vplace`.

Display

Každé aplikaci v systému X Window je přiřazen tzv. display, což je jméno obrazovky, kterou řídí server systému X Window. Displej se skládá ze tří komponent:

- Jméno počítače, na kterém server běží. Při samostatné instalaci operačního systému Linux běží server vždy na stejném systému jako klient. Ve většině případů může být jméno počítače vynecháno.
- Číslo serveru, který na daném počítači běží. Na každém jednotlivém počítači může běžet několik serverů systému X Window současně (v případě operačního systému Unix to není příliš častý případ). Pak každý z nich musí mít unikátní číslo.
- Číslo obrazovky. Systém X Window podporuje servery řídící v daném okamžiku více nežjednu obrazovku. Některé aplikace je účelné provozovat na více než jednom monitoru a pak je nutné, aby server řídil více monitorů. Není totiž efektivní, aby byl pro každý monitor spuštěn zvláštní server.

Uvedené tři atributy lze specifikovat prostřednictvím jediného řetězce v následujícím formátu: *`machine:server-number.screen-number`*.

Například moje aplikace mají nastavenu hodnotu pro display jako `:0.0`. To znamená, že systém pracuje s první obrazovkou, řídí jej první server a běží na lokálním počítači. Kdybych však používal vzdálený počítač, pak bude display nastaven jako `mousehouse:0.0`.

Implicitně se řetězec definující hodnotu display čte ze systémové proměnné `DISPLAY` (viz oddíl Systémové proměnné) a může být předefinován parametry příkazového řádku pro spuštění systému X Window (viz tabulka 5.2). Chcete-li se přesvědčit, zda je systémová proměnná nastavena, zadejte příkaz `echo $DISPLAY`.

Společné vlastnosti

Systém X Window představuje uživatelské grafické rozhraní, které lze prakticky ve všech aspektech konfigurovat. Je téměř nemožné říci, jak jeho jednotlivé komponenty fungují, protože to především závisí na konfiguraci.

jméno	následováno čím	příklad
`-geometry`	geometrií okna	`xterm -geometry 60x24+0+90`
`-display`	display, ve kterém se má program objevit	`xterm -display lionsden:0.0`
`-fg`	primární barva popředí	`xterm -fg yellow`
`-bg`	primární barva pozadí	`xterm -bg blue`

Tabulka 5.2 – Standardní volby pro programy běžící pod systémem X Window

Každá vlastnost může být překonfigurována, změněna nebo úplně nahrazena jinou. Proto je obtížné jednoznačně popsat chování jednotlivých prvků tvořících toto rozhraní. Zatím jsme popsali to, co popsat lze: různé správce oken a možnosti jejich konfigurace.

Situaci ještě více komplikuje skutečnost, že různé aplikace jsou založeny na různých grafických knihovnách. Pro tyto knihovny se vžil název „knihovny přípravků" – widget sets. Spolu se systémem X Window se distribuuje „Athena", grafický systém vyvinutý v Massachusetts Institute of Technology. Systém Athena se běžně používá ve volně šiřitelném programovém vybavení. Má tu nevýhodu, že nevytváří příliš dobře vyhlížející grafické objekty a jeho používání je poměrně těžkopádné.

Další populární knihovnou je „Motif". Motif patří ke komerčním produktům a má podobné vlastnosti jako uživatelské grafické rozhraní používané v operačním systému Microsoft Windows. Tuto knihovnu využívá spousta komerčních a některé volně šiřitelné aplikace. Populární program pro prohlížení stránek WWW Netscape také využívá knihovnu Motif.

Pokusme se nyní popsat některé společné vlastnosti systému X Window.

Tlačítka

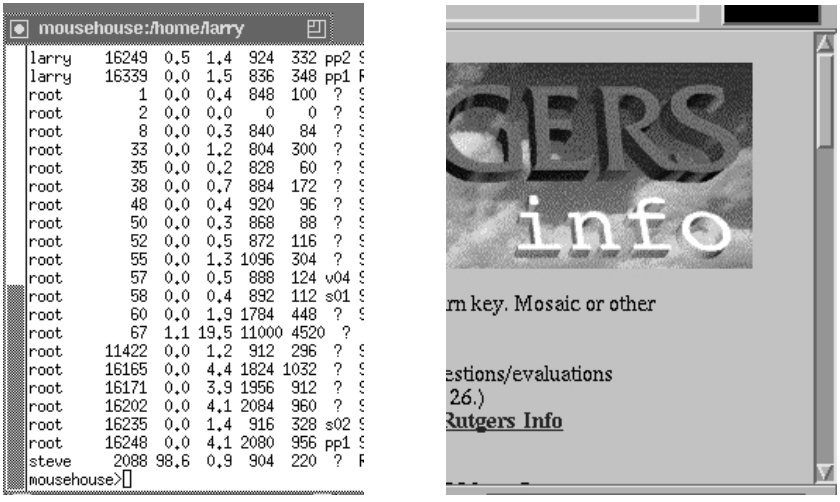
Tlačítka jsou prvky grafického rozhraní, jež se nejsnadněji používají. Stačí na tlačítko přesunout kurzor myši a klepnout levým tlačítkem. Tlačítka systémů Athena a Motif fungují stejně, rozdíly mezi nimi jsou pouze kosmetické.

Nabídkové lišty

Nabídková lišta je soubor příkazů zpřístupněných pomocí myši. Na obrázku 5.3 je například zobrazena nabídka editoru `emacs`. Každé slovo v nabídce představuje záhlaví jisté množiny dalších příkazů. Například položka `File` obsahuje příkazy pro zavádění a ukládání souborů. Podle zažité konvence zde najdete i příkaz k ukončení editoru.

Chcete-li zpřístupnit některý příkaz z nabídky, přesuňte kurzor myši na příslušnou nabídku (například `File`) a stiskněte levé tlačítko myši (držte je stále stisknuté). Pak se vám zobrazí další nabídka příkazů. Chcete-li si jeden z nich vybrat, přesuňte kurzor myši na požadovaný příkaz a uvolněte tlačítko myši. Některé nabídkové lišty vyžadují, abyste na požadované položce klikli myší. Pak program vyčkává, až klepnete na nějaký příkaz. Také můžete klepnout mimo nabídku a tím dáte programu najevo, že žádný příkaz realizovat nemá.

Obrázek 5.3 – Editor Emacs mění nabídkovou lištu podle toho, s jakým typem souboru zrovna pracujete. Zde je ukázka lišty obsahující nabídky.



Obrázek 5.4 – Na levé straně tohoto terminálového okna je ukázka posuvné lišty vytvořené v systému Athena. Na druhé straně je ukázka posuvné lišty (v okně programu Netscape) vytvořené systémem Motif.

Posuvné lišty

Posuvná lišta je nástroj umožňující zobrazit pouze část dokumentu, zatímco zbytek dokumentu je mimo okno. Například na obrázku 5.4 zobrazuje terminálové okno dolní třetinu textu. Díky posuvným lištám se také snadno určí, jaká část dokumentu je právě zobrazena. Tmavá část posuvné lišty ukazuje relativní pozici právě zobrazené stránky textu a zároveň její relativní velikost. Jestliže se celý text vleze na obrazovku, je celá posuvná lišta zobrazena tmavě. Jestliže je na obrazovce zobrazena polovina textu, pak bude zobrazena tmavě polovina posuvné lišty.

Vertikální posuvné lišty mohou být zobrazeny jak v pravém okraji okna, tak i v levém. Horizontální posuvné lišty bývají zpravidla zobrazeny u dolního okraje okna.

Posuvné lišty v systému Athena

Posuvné lišty v systému Athena fungují jinak, než u ostatních systémů. Každé tlačítko myši má jinou funkci. Má-li se text rolovat nahoru (tj. má-li se zobrazit text nad momentálně zobrazeným textem), pak stačí klepnout pravým tlačítkem myši, přičemž je kurzor myši umístěn kdekoliv v posuvné liště. Pokud chcete text rolovat dolů, pak klepněte levým tlačítkem myši.

Dále si můžete pomocí prostředního tlačítka myši zobrazit text v konkrétní pozici. Stačí klepnout prostředním tlačítkem myši na odpovídající pozici v posuvné liště.

Posuvné lišty v systému Motif

Posuvné lišty v systému Motif fungují téměř stejně jako v operačním systému Windows nebo Macintosh. Příklad posuvné lišty je na obrázku 5.4. Všimněte si, že na začátku a konci posuvné lišty jsou šipky. Ty jsou určeny pro „jemné“ rolování textu. Pokud na jedné z těchto šipek klepnete levým nebo prostředním tlačítkem myši, posune se text jen o několik málo řádků. Pravé tlačítko myši v tomto případě neplní žádnou funkci.

Pokud jde o používání myši uvnitř posuvné lišty, existují významné rozdíly mezi systémy Athena a Motif. Pravé tlačítko myši nemá žádný význam. Klepnutí levým tlačítkem myši nad momentální pozicí tmavé části posuvné lišty způsobí posun textu nahoru. Podobně platí, že klepnutí levým tlačítkem myši pod momentální pozicí tmavé části lišty způsobí posun textu dolů. Dále lze zobrazovaným textem přímo manipulovat tak, že se klepne levým tlačítkem myši na tmavou část posuvné lišty, drží se tlačítko stisknuté a nastaví se pozice tmavé části na požadované místo. Jakmile se tlačítko uvolní, požadovaná pozice se zobrazí.

Prostřední tlačítko má podobnou funkci jako u systému Athena. Zobrazí se ta část textu, která svou pozicí relativně odpovídá pozici kurzoru myši v posuvné liště.

KAPITOLA 6

Práce s operačním systémem Unix

Operační systém Unix je velmi výkonný nástroj pro ty, kdo jej umí používat. V této kapitole si popíšeme pokročilejší techniky práce s příkazovým procesorem `bash`.

Pseudoznaky

V předcházející kapitole jsme se naučili pracovat s příkazy pro manipulaci se soubory. Příležitostně se může stát, že budete chtít pracovat s více soubory najednou. Například budete chtít zkopírovat všechny soubory začínající posloupností „data“ do adresáře `~/backup`. Můžete to udělat tak, že použijete několikrát příkaz `cp`, nebo v jednom příkazu `cp` uvedete seznam všech souborů, které se mají kopírovat. Oba způsoby jsou těžkopádné a jistě vám seberou hodně času. Navíc máte velkou šanci udělat chybu.

Na následujícím příkladě si uvedeme daleko elegantnější postup:

```
/home/larry/report# ls -F
1993-1 1994-1 data1 data5
1993-2 data-new data2
/home/larry/report# mkdir ~/backup
/home/larry/report# cp data* ~/backup
/home/larry/report# ls -F ~/backup
data-new data1 data2 data5
/home/larry/report#
```

Z uvedeného příkladu vidíte, že po zadání hvězdičky zkopíroval příkaz `cp` všechny soubory začínající řetězcem data do adresáře `~/backup`. Zkuste uhodnout, jak bude fungovat příkaz `cp d*w ~/backup`

Co se ve skutečnosti stalo?

Rozhodně dobrá otázka. Příkazový interpret `bash` je schopen interpretovat jisté znaky, kterým se říká pseudoznaky (wildcards). Znak hvězdička (*) je interpretován tak, že za něj dosadí jakoukoliv posloupnost znaků. Proto příkaz z našeho příkladu `cp data* ~/backup` byl interpretován jako příkaz `cp data-new data1 data2 data5 ~/backup`.

Abychom ještě lépe ilustrovali interpretaci znaku hvězdička, uvedeme si nový jednoduchý příkaz `echo`, který zkopíruje své parametry jako text na obrazovce.

```
/home/larry# echo Hello!
hello!
/home/larry# echo How are you?
How are you?
/home/larry# cd report
/home/larry/report# ls -F
1993-1 1994-1 data1 data5
1993-2 data-new data2
/home/larry/report# echo 199*
1993-1 1993-2 1994-1
/home/larry/report# echo *4*
1994-1
/home/larry/report# echo *2*
1993-2 data2
/home/larry/report#
```

Jak sami vidíte, příkazový procesor rozšířil hvězdičku, vybral všechny soubory vyhovující specifikaci s hvězdičkou a předal je programu, který byl zadán ke spuštění. Naskýtá se přirozená otázka. Co se stane, když specifikaci s hvězdičkou nevyhovuje žádný soubor? Vyzkoušejte si například příkaz `echo /rc/fr*og` a uvidíte – příkazový procesor předá specifikaci přesně tak, jak byla zadána (žádnou interpretaci hvězdičky neprovede).

Ostatní příkazové procesory, například tcsh, se chovají jinak – nepředávají přesnou specifikaci, ale vypíší zprávu `No match`. Uveďme si příklad s příkazovým procesorem `tcsh`:

```
mousehouse>echo /rc/fr*og
echo: No match.
mousehouse>
```

Také vás asi napadá otázka, zda je možné pomocí příkazu `echo` zobrazit například `data*` (a nikoliv seznam souborů vyhovujících specifikaci `data*`). Řešení je jednoduché, stačí požadovaný text uzavřít do uvozovek. Například:

```
/home/larry/report# echo "data*"
data*
/home/larry/report#
```

```
mousehouse>echo "data*"
nebo    data*
mousehouse>
```

Znak otazník

Kromě znaku hvězdičky je příkazový procesor schopen interpretovat jako speciální znak také znak otazník. Uvede-li se ve specifikaci otazník, pak to znamená, že má být nahrazen právě jedním znakem. Například příkaz `ls /etc/??` zobrazí všechny soubory v adresáři `/etc`, jejichž jména se skládají právě ze dvou znaků.

Jak ušetřit čas pomocí příkazu bash

Editování příkazového řádku

Někdy se stane, že napíšete v příkazovém řádku příkazového interpretu `bash` dlouhý příkaz a pak si všimnete, že jste udělali chybu. Samozřejmě můžete postupně smazat všechny znaky, až se dostanete k chybně zadanému znaku, ale pak budete muset celý zbytek příkazu znovu psát. Příkazový interpret `bash` umožňuje používat kurzorové klávesy (šipka vlevo a šipka vpravo), pomocí kterých si můžete nastavit kurzor na chybně zadaný znak a opravit jej.

Dále existuje řada speciálních klávesových zkratek, pomocí nichž můžete příkazový řádek editovat. Tyto klávesové zkratky jsou většinou podobné příkazům používaným v editoru Emacs. Například kombinace Ctrl+T provede výměnu dvou sousedních znaků.[1] Popis většiny důležitých klávesových zkratek najdete v kapitole 8, která je věnována editoru Emacs.

Doplňování příkazů a jmen souborů

Další vynikající vlastnost příkazového procesoru bash spočívá v tzv. doplňování příkazových řádků. Podívejme se na následující příklad.

```
/home/larry# ls -F
this-is-a-long-file
/home/larry# cp this-is-a-long-file shorter
/home/larry# ls -F
shorter         this-is-a-long-file
/home/larry#
```

Samozřejmě je velmi nepohodlné a namáhavé vypisovat každý znak jména souboru `this-is-a-long-file` pokaždé, když chcete tento soubor zpřístupnit. Vytvořte si takový soubor, například příkazem `cp /etc/passwd this-is-a-long-file`. Nyní si ukážeme, jak zadat předcházející příkaz rychleji a s menším rizikem, že uděláme chybu.

Místo vypisování celého jména souboru zadejte `cp th` a pak stiskněte klávesu Tab. Jako mávnutím kouzelného proutku se vám celé jméno souboru objeví v příkazovém řádku a vám již stačí napsat `shorter`. Příkazový procesor `bash` neumí bohužel číst vaše myšlenky, proto jméno souboru `shorter` musíte napsat sami.

Když stisknete klávesu Tab, příkazový interpret `bash` se pokusí najít soubor, který začíná znaky, jež jste doposud napsali. Když například napíši `/usr/bin/ema` a pak stisknu Tab, příkazový procesor bash najde soubor `/usr/bin/emacs`. Když však napíši `/usr/bin/ld` a pak stisknu klávesu Tab, příkazový interpret pípne, aby mě upozornil, že našel více souborů. Skutečně, v adresáři `/usr/bin` jsou soubory `ld`, `ldd` a `ld86`.

Jestliže se pokoušíte doplnit jméno souboru a příkazový procesor pípne, pak můžete znovu stisknout klávesu Tab a objeví se vám seznam všech nalezených souborů. Proto nemusíte znát přesně jména svých souborů, příkazový procesor vám je vždy tímto způsobem připomene.

1 Zkratka Ctrl+T znamená, že máte stisknout klávesu Ctrl, držet ji stisknutou, a pak stisknout klávesu T.

Standardní vstup a standardní výstup

Zkuste si provést příkaz, který vypíše seznam všech souborů v adresáři `/usr/bin`: `ls /usr/bin`. Adresář `/usr/bin` obsahuje velké množství souborů, proto po zobrazení jejich jmen zůstane na obrazovce jen několik a ostatní „utečou“ nahoru. Jak tento problém řešit?

Standardní vstup a výstup

Operační systém Unix nabízí programátorům velmi jednoduchý způsob zápisu na terminál. Pokud nějaký program něco vypisuje na vaši obrazovku, pak používá tzv. **standardní výstup** (standard output). Standardní výstup se zkracuje zkratkou stdout a slouží k předávání informací uživateli. Na druhé straně existuje **standardní vstup** (standard input), který slouží k předávání informací od uživatele. Samozřejmě je možné, aby program komunikoval s uživatelem bez použití standardního vstupu a výstupu, ale většina programů, kterými se zabýváme v této knize, standardní vstup a standardní výstup používá.

Například příkaz `ls` vypisuje seznam adresářů a souborů na standardní výstup, jenž je normálně spojen s terminálem. Příkazový interpret `bash` čte vámi zadané příkazy ze standardního vstupu.

Programy také mohou zapisovat do tzv. **standardního chybového výstupu** (standard error). Standardní chybový výstup je téměř výlučně spjat s terminálem, proto mohou být uživatelé informováni o případných chybách.

V následujících odstavcích si ukážeme, jak využívat standardní vstup a standardní výstup v konstrukcích, kterým se říká přesměrování vstupu, přesměrování výstupu a roura vstupu/výstupu.

Přesměrování výstupu

Velmi důležitou vlastností operačního systému Unix je schopnost přesměrovat výstup. Tato vlastnost vám umožní uložit výsledky příkazu do souboru nebo vytisknout na tiskárně. Jestliže chcete například přesměrovat výstup z příkazu `ls /usr/bin` do souboru, přidejte na konec příkazu znak `>` a za něj uveďte jméno souboru (nejlépe neexistujícího). Například:

```
/home/larry# ls
/home/larry# ls -F /usr/bin > listing
/home/larry# ls
listing
/home/larry#
```

Jak vidíte, seznam souborů se nezobrazil na terminálu, ale místo toho se vytvořil ve vašem domovském adresáři nový soubor `listing`. Zkusme se na tento soubor podívat pomocí příkazu `cat`. Jestli si vzpomínáte, příkaz `cat` se jevil jako zbytečný příkaz, který zkopíroval na obrazovku (standardní výstup) to, co jste napsali (standardní vstup). Příkaz `cat` také umí vypsat obsah souboru na standardní výstup, pokud tento soubor uvedete jako parametr:

```
/home/larry# cat listing
 ...
/home/larry#
```

Soubor `listing` nyní obsahuje přesný výstup příkazu `ls /usr/bin`, tedy seznam všech souborů v adresáři `/usr/bin`. To je v pořádku, ale stále to neřeší náš původní problém.[2]

2 Pro netrpělivé čtenáře poznamenejme, že zde mohou použít program `more`. Než se však k němu dostaneme, musíme ještě několik věcí vysvětlit.

Nyní je příkaz `cat` o něco zajímavější – lze jej použít spolu s přesměrováním výstupu. Co asi udělá příkaz `cat listing > newfile`? Přesměrování `> newfile` pro příkazový interpret znamená toto: „vezmi celý výstup z příkazu a ulož jej do nového souboru“. Výstup z příkazu `cat listing` je ovšem soubor `listing`. Jinými slovy, pravě jsme popsali jeden ze způsobů kopírování souborů, i když ne příliš efektivní.

Co asi udělá příkaz `cat > fox`? Příkaz `cat` čte data ze standardního vstupu (v tomto případě terminálu) a kopíruje je na standardní výstup, dokud nenarazí na znak konce souboru Ctrl+D. V tomto případě bude standardní vstup přesměrován do souboru `fox`. Nyní nám příkaz `cat` slouží jako základní editor:

```
/home/larry# cat > fox
The quick brown fox jumps over the lazy dog.
```

Stiskni Ctrl+D

Právě jste vytvořili soubor se jménem `fox` obsahující větu „`The quick brown fox jumps over the lazy dog.`“ Další důležitou funkcí příkazu `cat` je spojování souborů dohromady. Příkaz `cat` bude vypisovat každý soubor, který mu byl předán jako parametr. Proto například příkaz `cat listing fox` nejprve vypíše seznam souborů adresáře `/usr/bin` a nakonec vypíše naši hloupou větu. Příkaz `cat listing fox > listandfox` vytvoří nový soubor tvořený obsahem souborů `listing` a `fox`.

Přesměrování vstupu

Tak, jako je možné přesměrovat standardní výstup, je možné přesměrovat i standardní vstup. Místo toho, aby program četl z klávesnice, bude číst ze souboru. Protože se přesměrování vstupu vztahuje k přesměrování výstupu, je přirozené pro něj vyhradit znak `<`. Tento znak se také uvádí za příkazem, který chcete realizovat.

Přesměrování vstupu je užitečné zejména v případě, kdy máte soubor obsahující data a program, který data čte se standardního vstupu. Na druhé straně platí, že většina programů vyžaduje specifikaci souboru se vstupními daty, proto se přesměrování vstupu nepoužívá tak často, jako přesměrování výstupu.

Roura

Řada příkazů v operačním systému Unix produkuje velké množství informací. Jak jsme si již ukázali, příkaz `cp /usr/bin` vyprodukuje příliš mnoho informací, než aby mohly být zobrazeny na terminálu. Abyste si mohli prohlédnout všechny informace z příkazů, jako je `ls /usr/bin`, budete muset použít další důležitý příkaz operačního systému Unix, který má název `more`.[3] Příkaz `more` způsobí pozastavení výpisu na obrazovku, jakmile se celá obrazovka zaplní. Například příkaz `more < /etc/rc` zobrazí stejným způsobem soubor `/etc/rc` jako příkaz `cat`, ale s tím rozdílem, že výpis pozastaví po každém naplnění obrazovky.

Příkaz `more` také můžete použít ve tvaru `more /etc/rc`, což představuje normální způsob jeho použití.

Co je však platné, že příkaz `more` umožňuje zobrazit více informací, než se vleze na obrazovku? Příkaz `more < ls /usr/bin` nebude fungovat, protože přesměrování vstupu funguje pouze se soubory a nikoliv s příkazy. Budete tedy muset postupovat takto:

3 Název `more` pochází z nápovědy tohoto programu, kterou byl původně řetězec --more--. V mnoha verzích operačního systému Linux se vyskytuje identický příkaz, který postupně programátoři zdokonalovali.

```
/home/larry# ls /usr/bin > temp-ls
/home/larry# more temp-ls
...
/home/larry# rm temp-ls
```

Naštěstí nabízí operační systém Unix mnohem elegantnější způsob. Stačí, když zadáte příkaz `ls /usr/bin | more`. Znak | indikuje tzv. **rouru**. Roura v operačním systému Unix řídí tok dat.

Užitečnost roury ještě více vzrůstá ve spojení s dalšími nástroji, kterým se říká **filtry**. Filtr je program, který čte standardní vstup, nějakým způsobem jej modifikuje a odešle jej na standardní výstup. Příkladem filtru je právě příkaz `more` – čte data ze standardního vstupu, zobrazuje je na standardní výstup (obrazovku) a umožňuje vám prohlédnout celý soubor. `more` však není úplně dokonalý filtr, protože jeho výstup není vhodný pro to, aby mohl být předán jako vstup jinému programu.

Mezi další filtry patří příkazy `cat`, `sort`, `head` a `tail`. Chcete-li například přečíst pouze prvních deset řádků výstupu z příkazu `ls`, můžete použít příkaz `ls /usr/bin | head`.

Současný běh úloh

Jak řídit úlohy

Řízení úloh (job control) představuje možnost zajistit, aby daný proces (proces není v podstatě nic jiného, než běžící program) běžel na pozadí nebo naopak, aby běžel na popředí. Jinými slovy, zřejmě potřebujete nástroje k tomu, abyste mohli dlouhodobě běžící program přesunout do pozadí a tak mohli pracovat na jiných věcech, a přitom měli možnost do programu běžícího na pozadí kdykoliv zasahovat. Tímto nástrojem je v operačním systému Unix příkazový procesor. Ukážeme si, jak jej využívat k řízení úloh.

V souvislosti s řízením úloh jsou v operačním systému Unix vyhrazena dvě důležitá klíčová slova. První z nich je `fg` pro běh programů v popředí a druhé je `bg` pro běh programů na pozadí. Abychom vyzkoušeli, jak fungují, použijeme příkaz `yes`:

```
/home/larry# yes
```

Po spuštění příkazu `yes` se na levé straně obrazovky vypisuje velkou rychlostí písmeno „y“.[4] Pokud byste chtěli příkaz `yes` zastavit, stiskli byste klávesu Ctrl+C. Místo toho však stiskněte klávesu Ctrl+Z. Nyní se vám bude zdát, že se provádění příkazu `yes` zastavilo. Na obrazovce se však objeví následující zpráva:

```
[1]+ Stopped          yes
```

To znamená, že proces `yes` byl pozastaven. Pozastavený proces můžete obnovit pomocí příkazu `fg`, který jej aktivuje v popředí a program poběží dále. Zatímco je program pozastaven, můžete realizovat další příkazy. Než zadáte příkaz `fg`, zkuste si zadat několikrát jiný příkaz, například `ls`.

Jakmile se příkaz `yes` „vrátí“ do popředí, bude pokračovat ve výpisu písmen „y“ stejnou rychlostí jako na počátku po jeho spuštění. Nemusíte mít obavy, že se výstup z programu, který byl pozastaven, někam ztratí. Je-li program pozastaven uvedeným způsobem, pak až do okamžiku, kdy jej obnovíte, žádný výstup neprodukuje. Nyní stiskněte klávesu Ctrl+C a program `yes` zrušte.

4 Možná se vám zdá příkaz `yes` divný. Řada příkazů však potřebuje potvrdit nějakou volbu, proto v operačním systému Unix takový příkaz existuje.

Nyní se vraťme k předcházející zprávě:

```
[1]+ Stopped          yes
```

Číslo v hranatých závorkách je **pořadové číslo úlohy**, na které se můžete odvolávat, kdykoliv bude chtít na běhu této úlohy něco změnit. Číslo úlohy se zavádí z toho důvodu, aby existovalo nějaké rozlišení mezi více současně běžícími úlohami. Znak „+“ za hranatými závorkami indikuje, že uvedená úloha je aktuální úlohou, tedy úlohou, jež byla jako poslední převedena z popředí na pozadí. Jestliže zadáte příkaz `fg`, pak do popředí převedete právě tu úlohu, která je označena znakem „+“. Slovo `Stopped` znamená, že úloha byla pozastavena a nachází se ve zvláštním stavu – není zrušena, ale také není aktivní. Operační systém Linux ji uvedl do speciálního stavu, kdy může pokračovat v realizaci, jakmile bude zadán příslušný příkaz. Jako poslední je uvedeno jméno procesu, tedy v našem případě `yes`.

Než pokročíme dále, zrušme tuto úlohu a nastartujme ji, tentokrát jiným způsobem. Příkaz ke zrušení úlohy se jmenuje `kill` (doslova zabít, zničit) a používá se následujícím způsobem:

```
/home/larry# kill %1
[1]+ Stopped yes
/home/larry#
```

Zpráva, která se objevila na obrazovce (říká, že proces byl pozastaven) je zavádějící. Abychom zjistili, v jakém stavu se proces nachází (t.j. zda je aktivní, zmrazen nebo pozastaven), zadáme příkaz `jobs`:

```
/home/larry# jobs
[1]+ Terminated yes
/home/larry#
```

Až zde máte napsáno, v jakém stavu se úloha `yes` skutečně nachází – byla ukončena a nikoliv pozastavena. Je také možné, že zadáte příkaz `jobs` a vůbec žádná zpráva se nevypíše. To znamená, že na pozadí opravdu neběží žádná úloha. Jestliže jste právě zrušili nějakou úlohu a příkaz `jobs` nic nevypíše, pak jste ji zrušili úspěšně. Obvykle se ale objeví zpráva, že úloha byla ukončena (`Terminated`).

Nyní spusťme příkaz `yes` znovu, tentokrát následujícím způsobem:

```
/home/larry# yes > /dev/null
```

Pokud jste četli oddíl o přesměrování vstupu a výstupu, pak asi tušíte, že příkaz `yes` bude odesílat svůj výstup do souboru `/dev/null`. Soubor `/dev/null` má v operačním systému Unix speciální význam. Představuje „černou díru“, do které se ztratí jakékoliv množství informací, aniž by například hrozilo, že se zaplní váš pevný disk. Nemáte-li dostatek fantazie, pak si představte, že je ve vašem počítači vyvrtána díra, kterou proudí informace ven a ztrácejí se v hlubinách vesmíru.

Po zadání uvedeného příkazu se vám nevrátí nápověda příkazového řádku, ani se nebudou na obrazovce vypisovat písmena „y“. I když je výstup z příkazu `yes` přesměrován do souboru `/dev/null`, úloha stále běží na popředí. Jako obvykle ji můžete pozastavit pomocí klávesy Ctrl+Z. Pak se vám samozřejmě znovu objeví výzva příkazového řádku:

```
/home/larry# yes > /dev/null
[Příkaz "yes" běží, nyní stiskněme Ctrl+Z]
[1]+ Stopped          yes > /dev/null
/home/larry#
```

Je nějaká možnost přesunout úlohu do pozadí tak, aby dále běžela a aby se nám objevila výzva příkazového řádku? K tomuto účelu slouží příkaz `bg`:

```
/home/larry# bg
[1]+ yes > /dev/null &
/home/larry#
```

Nyní byste měli věřit následujícímu tvrzení: po zadání příkazu `bg` běží proces `yes > /dev/null` dále, a to na pozadí. Poznáte to podle toho, že když v příkazovém řádku zadáte nějaký příkaz, například `ls` nebo `stuff`, bude jeho realizace poněkud zpomalena. Jiný efekt úlohy běžící na pozadí nemají. Nadále můžete v příkazovém řádku zadávat jakékoliv příkazy a proces `yes > /dev/null` poběží na pozadí a přitom bude výstup odesílat do „černé díry".

Existují dvě možnosti, jak proces běžící na pozadí zrušit (doslova „zabít"). Buď použijete příkaz `kill`, nebo přesunete proces do popředí a ukončíte jej pomocí klávesy Ctrl+C. Vyzkoušíme si druhý způsob, abychom lépe pochopili vztah mezi příkazy `fg` a `bg`:

```
/home/larry# fg
yes > /dev/null
[Nyní běží proces opět v popředí. Stiskněte klávesu Ctrl+C a ukončete jej.]
/home/larry#
```

Jako další si vyzkoušíme spustit současně více procesů, například:

```
/home/larry# yes > /dev/null &
[1] 1024
/home/larry# yes | sort > /dev/null &
[2] 1026
/home/larry# yes | uniq > /dev/null
[nyní stiskněte klávesu Ctrl+Z]
[3]+ Stopped yes | uniq > /dev/null
/home/larry#
```

Určitě jste si všimli, že jsme na konec prvních dvou příkazů zapsali znak `&`. Pokud na konec příkazu zadáte znak `&`, příkazový procesor spustí úlohu hned od počátku na pozadí. Je to mnohem jednodušší, než spustit úlohu normálním způsobem, stisknout klávesu Ctrl+Z a pak zadávat příkaz `bg`. První dvě úlohy jsme tedy přímo spustili na pozadí. Třetí úloha je v tomto okamžiku pozastavena a tedy neaktivní. Jistě jste zpozorovali, že počítač nyní reaguje na další příkazy pomaleji, protože dvě běžící úlohy spotřebovávají jistý čas procesoru.

Zrušme nyní druhou úlohu, protože ta patrně výrazně snižuje výkon vašeho počítače. Mohli bychom použít příkaz `kill %2`, ale to by bylo příliš jednoduché. Místo toho zadejme následující příkazy:

```
/home/larry# fg %2
yes | sort > /dev/null
[Stiskněte klávesu Ctrl+C]
/home/larry#
```

Uvedený příklad demonstruje, že příkaz `fg` akceptuje parametry začínající znakem `%`. Ve skutečnosti jsme mohli použít následující posloupnost příkazů:

```
/home/larry# %2
yes | sort > /dev/null
[Stiskněte klávesu Ctrl+C]
/home/larry#
```

První příkaz bude opět fungovat, protože příkazový procesor interpretuje číslo úlohy jako požadavek, že má být úloha přenesena do popředí. Čísla úloh odlišuje od ostatních čísel právě pomocí znaku %. Nyní zadejte příkaz `jobs`, abyste věděli, které úlohy momentálně běží:

```
/home/larry# jobs
[1]- Running yes > /dev/null &
[3]+ Stopped yes | uniq > /dev/null
/home/larry#
```

Znak „-“ znamená, že úloha číslo 1 byla do pozadí přesunuta jako první v pořadí a bude jako druhá v pořadí přesunuta do popředí, pokud zadáte příkaz `fg` bez parametrů. Znak „+“ znamená, že úloha číslo 3 bude do popředí přesunuta jako první v pořadí, pokud zadáte příkaz `fg` bez parametrů. Uvedené implicitní pořadí můžete změnit takto:

```
/home/larry# fg %1
yes > /dev/null
[nyní stiskněte klávesu Ctrl+Z]
[1]+ Stopped yes > /dev/null
/home/larry#
```

Pomocí uvedených příkazů jste pozastavili úlohu číslo 1 a změnili jste také pořadí priorit všech úloh. K jakým změnám došlo? Na to vám odpoví příkaz `jobs`:

```
/home/larry# jobs
[1]+ Stopped yes > /dev/null
[3]- Stopped yes | uniq > /dev/null
/home/larry#
```

Nyní jsou obě úlohy pozastaveny (v obou případech jsme použili klávesu Ctrl+Z). Úloha číslo 1 je nyní na pozici první, pokud jde o pořadí, v jakém budou úlohy implicitně přenášeny do popředí. Je tomu tak z toho důvodu, že jsme úlohu nejprve přesunuli do popředí a pak ji pozastavili. Znak „+“ vždy označuje tu úlohu, která byla jako poslední pozastavena, když předtím běžela na popředí. Úlohu číslo 1 můžeme opět aktivovat tímto způsobem:

```
/home/larry# bg
[1]+ yes > /dev/null &
/home/larry# jobs
[1]- Running yes > /dev/null &
[3]+ Stopped yes | uniq > /dev/null
/home/larry#
```

Všimněte si, že úloha 1 nyní běží a u druhé úlohy se objevil znak „+“. Nakonec obě úlohy zrušíme, protože se už nemůžeme dále dívat na to, jak snižují výkon počítače:

```
/home/larry# kill %1 %2
[3] Terminated yes | uniq > /dev/null
/home/larry# jobs
[1]+ Terminated yes > /dev/null
/home/larry#
```

Jistě jste si všimli zpráv upozorňujících na ukončení úloh – jak se zdá, nic neumírá tiše. Tabulka 6.1 obsahuje stručný přehled o příkazech, které se vztahují k řízení úloh.

Teorie řízení úloh

Především je nutné pochopit, že úlohy řídí příkazový procesor. Ve skutečnosti v systému neexistují programy jako fg, bg, jobs, & nebo kill. Tyto programy jsou vnitřními příkazy příkazového procesoru a jsou jeho součástí. (Výjimku někdy tvoří příkaz `kill`, který je v některých operačních systémech Unix implementován jako externí program. Pokud však jde o Linux, příkaz `kill` je integrován v příkazovém procesoru bash). Tento mechanismus řízení úloh je logický. Každý uživatel chce mít k realizaci úloh svůj prostor. Protože každý uživatel pracuje se svým příkazovým procesorem, je logické, aby příkazový procesor úlohy řídil. Odtud dále vyplývá, že čísla úloh se vztahují právě k jednomu uživateli. Moje úloha číslo 1 bude zřejmě zcela odlišná od vaší úlohy číslo 1, a to i v případě, kdy budeme oba přihlášeni k témuž systému. Ve skutečnosti platí, že přihlásíte-li se do systému více než jednou, bude každý spuštěný příkazový procesor vlastnit unikátní datové struktury pro řízení úloh. Jako jediný uživatel tedy můžete mít několik zcela různých úloh s pořadovým číslem 1 běžících v různých příkazových interpretech.

`fg %job`	Tento příkaz je příkazem příkazového procesoru a přesouvá úlohu do popředí. Chcete-li zjistit, která úloha se přesune implicitně jako první, zadejte příkaz `jobs` a podívejte se, která úloha je označena znakem „+". Parametry: volitelným parametrem je číslo úlohy. Implicitní je úloha označená znakem „+".
`&`	Pokud se znak „&" uvede na konec příkazového řádku, bude úloha spuštěna automaticky přímo na pozadí. Pro takto spuštěný proces platí všechny metody řízení úloh, které jsme zde uvedli.
`bg %job`	Tento příkaz je příkazem příkazového procesoru a přesouvá úlohu do pozadí. Chcete-li zjistit, která úloha se přesune implicitně jako první, zadejte příkaz `jobs` a podívejte se, která úloha je označena znakem „+". Parametry: volitelným parametrem je číslo úlohy. Implicitní je úloha označená znakem „+".
`kill %job PID`	Tento příkaz je příkazem příkazového procesoru a ukončuje úlohu, ať běží na pozadí nebo byla pozastavena. Pokaždé byste měli použít jako parametr číslo úlohy, kterému předchází znak `%`. Parametry: buď číslo úlohy, kterému předchází znak `%`, nebo PID (identifikační číslo procesu), pak znak `%` není nutné uvádět. Na jednom příkazovém řádku může být uvedeno více úloh, které se mají ukončit.
`jobs`	Tento příkaz je příkazem příkazového procesoru. Vypisuje informace o běžících úlohách a o úlohách, které byly pozastaveny.
Ctrl+C	Klávesová zkratka vyhrazená k bezpodmínečnému ukončení úlohy. Běžně se používá k ukončení úlohy běžící v popředí. Je však nutné upozornit na skutečnost, že ne všechny programy na tuto klávesovou zkratku reagují.
Ctrl+Z	Klávesová zkratka vyhrazená k pozastavení úlohy. Opět platí, že některé programy jej ignorují. Jakmile je jednou úloha pozastavena, může být znovu přesunuta do popředí nebo ukončena.

Tabulka 6.1 – Přehled příkazů a kláves používaných k řízení úloh

Jediný bezpečný způsob, jak identifikovat proces, představuje identifikační číslo procesu (process identification number, zkratka PID). Každá úloha běžící v systému má svoje unikátní identifikační číslo procesu. Dva různí uživatelé mohou k odkazu na proces použít totéž identifikační číslo a pak mají jistotu, že se jedná o tentýž proces (samozřejmě za předpokladu, že jsou přihlášeni k témuž systému).

Podívejme se na další příkaz, který vám pomůže lépe pochopit význam identifikačních čísel procesů. Příkaz `ps` vypíše seznam všech běžících nebo pozastavených procesů, včetně vašeho příkazového procesoru. Tento příkaz má také několik voleb, z nichž nejdůležitější jsou `a`, `u` a `x`. Zadáte-li volbu `a`, pak se vám zobrazí seznam všech procesů, t.j. procesů, které spustili i ostatní uživa-

telé. Pomocí volby `x` se zobrazí seznam těch procesů, které nejsou nijak svázány s terminálem.[5] Poslední volba `u` zajistí, že se vám zobrazí další užitečné informace o běžících nebo pozastavených procesech.

Jestliže chcete získat komplexní představu, co se ve vašem systému odehrává, zadejte příkaz `ps -aux`. (V novějších verzích je možno znaménko „–“ vynechat a použít pouze příkaz `ps aux`.) Pak se můžete podívat, kolik paměti každý proces potřebuje (sloupec `%MEM`) a jak zatěžuje procesor (sloupec `%CPU`). Ve sloupci `TIME` je uveden celkový čas procesoru, který spuštěný proces spotřeboval.

Ještě jedna poznámka o identifikačním číslu procesu. Kromě parametru `%číslooúlohy#` můžete v příkazu `kill` použít identifikační číslo procesu. Spusťte příkaz `yes > /dev/null` na pozadí, pak spusťte příkaz `ps` a podívejte se na identifikační číslo náležící procesu `yes`. Toto identifikační číslo můžete také použít ke zrušení procesu `yes`.[6]

Jestliže začnete programovat v jazyku C, pak se brzy dozvíte, že příkazy pro řízení úloh jsou interaktivní verze systémových volání `fork` a `execl`. Jedná se o příliš složité mechanismy, než abychom je zde mohli popsat. Pokud budete vytvářet programy, které jsou schopny spouštět více procesů, pak se s těmito systémovými funkcemi budete muset seznámit v příslušné dokumentaci.

Virtuální konzoly

Operační systém Linux podporuje tzv. virtuální konzoly. Jedná se o metodu, která budí dojem, že váš počítač není jedním počítačem ale několika počítači najednou. Linux je schopen spustit několik terminálů současně a přitom jsou všechny spojeny s jedním jádrem. Naštěstí je používání virtuálních konzol jednou z nejjednodušších záležitostí v operačním systému Linux. Chcete-li si je vyzkoušet, stiskněte klávesu Alt a pak klávesu F2.[7]

Najednou se vám objeví nová výzva k přihlášení se do systému. Nepodléhejte panice. Nyní pracujete s virtuální konzolou číslo 2. Přihlaste se a proveďte několik příkazů, abyste se přesvědčili, že nově nastartovaný příkazový procesor skutečně funguje. Budete-li se chtít vrátit do virtuální konzoly číslo 1, stiskněte opět klávesu Alt a pak F1. Nebo můžete vytvořit třetí konzolu pomocí kláves Alt a F3.

Operační systém Linux má implicitně povoleno šest virtuálních konzol. Pokud budete chtít vědět jak, prostudujte si manuál *„Příručka správce operačního systému Linux“*. Budete muset udělat nějaké úpravy v souborech v adresáři `/etc`. Šest virtuálních konzol však většině uživatelů stačí.

Začnete-li používat virtuální konzoly, brzy zjistíte, že jsou ideálním nástrojem k realizaci mnoha úkonů současně. Na první konzole můžete například provozovat editor Emacs, na druhé programy pro komunikaci s Internetem a na třetí můžete mít spuštěn příkazový procesor, kdybyste chtěli spustit ještě nějaký další program.

---

5 To má smysl jen v případě jistých systémových programů, které nekomunikují s uživatelem prostřednictvím klávesnice.

6 Obecně je mnohem jednodušší zrušit úlohu pomocí jejího čísla.

7 Ujistěte se, že pracujete s textovou konzolou. V případě systému X Window by uvedená kombinace kláves nemusela fungovat.

KAPITOLA 7

Malé a výkonné programy

V čem spočívá síla operačního systému Unix

Síla operačního systému Unix spočívá v používání malých a jednoduchých příkazů, které se sami o sobě zdají neužitečné. Pokud se je naučíte spojovat dohromady, vytvoříte systém, jenž je mnohem pružnější a výkonnější než všechny ostatní operační systémy. V této kapitole se budeme zabývat příkazy `sort`, `grep`, `more`, `cat`, `wc`, `spell`, `diff` a `tail`. Z názvů těchto programů nelze bohužel intuitivně předpokládat, jakou funkci plní.

Nejdříve se budeme věnovat každému příkazu odděleně a nakonec uvedeme několik příkladů, jak je spojovat a používat dohromady.[1]

Práce se soubory

Kromě příkazů `cd`, `mv` a `rm`, o kterých jsme se zmínili v kapitole 4, existují další příkazy určené k manipulaci se soubory (a nikoliv s daty, jež soubory obsahují). Patří mezi ně `touch`, `chmod`, `du` a `df`. Žádný z těchto příkazů se nestará o obsah souborů – pouze mění některé jejich atributy, se kterými operační systém Unix pracuje.

Uveďme si seznam atributů, se kterými uvedené příkazy manipulují:

- Časové údaje. S každým souborem jsou spjaty tři časové údaje.[2] První časový údaj obsahuje informaci o době vytvoření souboru, druhý o době jeho poslední modifikace a třetí o době, kdy byl naposledy zpřístupněn.
- Vlastník souboru. Každý soubor v operačním systému Unix je vlastněn některým uživatelem.
- Skupina. Každý soubor je také spojen se skupinou uživatelů. Nejčastěji se tato skupina nazývá `users`, což znamená, že soubor je sdílen každým uživatelem přihlášeným do systému.
- Přístupová práva. Každý soubor má přístupová práva (pro která se někdy používá označení privilegia). Jedná se o mechanismus, jenž určuje, kdo může k danému souboru přistupovat, kdo může spouštět dané programy a podobně. Každé z přístupových práv může být odděleně přiřazeno vlastníku souboru, skupině nebo všem uživatelům operačního systému.

1 Poznamenejme, že výklad uvedených příkazů nebude zcela vyčerpávající. Podrobnosti naleznete v manuálových stránkách.

2 Starší souborové systémy v operačním systému Linux uchovávaly pouze jeden časový údaj, protože byly odvozeny od operačního systému Minix. Pokud máte takový souborový systém, pak pro vás budou některé informace neaktuální.

`touch` *`soubor1 soubor2 ... souborN`*

Příkaz `touch` provede aktualizaci časových údajů každého souboru uvedeného jako parametr – nastaví čas vytvoření na aktuální čas. Pokud soubor uvedený jako parametr neexistuje, program touch jej vytvoří. Je také možné specifikovat čas, který má být souborům přiřazen. Podrobnosti si nastudujte v manuálových stránkách.

`chmod [-Rfv]` *`mód soubor1 soubor2 ... souborN`*

Příkaz umožňující změnit přístupová práva se nazývá `chmod` (change mode). Než se pustíme do jeho popisu, musíme si probrat, jaká přístupová práva operační systém Unix bere v úvahu. Každý soubor je spojen se skupinou přístupových práv. Podle těchto přístupových práv operační systém Unix určuje, zda může být soubor čten, zda do něj může být zapisováno, nebo, jedná-li se o spustitelný program, může být spuštěn. V následujících odstavcích budeme hovořit o přístupových právech uživatelů. Každý program, jenž uživatel spustí, má shodná přístupová práva. Pokud přesně nevíte, co program dělá, mohou nastat problémy s bezpečností systému.

Operační systém Unix rozlišuje tři různé typy uživatelů. Prvním z nich je vlastník souboru. Tím je osoba, která má právo v souvislosti s tímto souborem používat příkaz `chmod`. Druhý typ představuje skupina. Většina souborů ve vašem systému by měla mít přiřazen atribut „`users`“, a tak by měla být přístupna každému normálnímu uživateli. Chcete-li znát hodnotu atributu skupina, zadejte příkaz `ls -l` *`file`*.

Nakonec operační systém identifikuje ty osoby, které nejsou ani vlastníky žádného souboru, ani členové žádné skupiny. Pro ty je vyhrazen atribut „`other`“ (ostatní).

Typické nastavení přístupových práv je: pro vlastníka právo číst soubor a zapisovat do souboru, pro skupinu právo číst soubor a pro ostatní jsou všechna práva potlačena. Může se však stát, že skupina má právo soubor číst i do něj zapisovat a přitom vlastník nemá žádná práva.

Nyní si vyzkoušejme, jak pomocí příkazu `chmod` změnit některá přístupová práva. Nejdříve si vytvořte nový soubor, řekněme pomocí příkazu `cat` nebo pomocí editoru Emacs. Implicitně je vám přiděleno právo soubor číst i právo do něj zapisovat. Práva ostatních uživatelů jsou určena podle toho, jak je nastaven váš systém a jak je nastaven proces přihlašování. Ujistěte se, že nově vytvořený soubor jste schopni číst pomocí příkazu `cat`. Nyní si odeberte právo číst tento soubor pomocí příkazu `chmod u-r` *`filename`*. Parametr `u-r` se interpretuje jako „user minus read“. Když se nyní pokusíte soubor přečíst, obdržíte chybové hlášení `Permission denied!`. Chcete-li získat zpět právo soubor číst, zadejte příkaz `chmod u+r` *`filename`*.

Přístupová práva k adresářům používají stejné atributy (pro čtení, zapisování a spouštění), ale poněkud odlišným způsobem je interpretují. Právo číst znamená, že uživatel, skupina nebo ostatní mohou vypisovat seznam souborů v adresáři. Právo zapisovat znamená, že uživatel, skupina nebo ostatní mohou přidávat soubory do adresáře nebo rušit soubory. Právo spouštět znamená, že uživatel může zpřístupňovat soubory v adresáři i v jeho podadresářích. Pokud nemá uživatel žádná práva, nemůže ani použít příkaz `cd`.

Při používání příkazu `chmod` se specifikuje, kdo má k danému souboru přístupová práva práva (uživatel, skupina, ostatní nebo všichni). Dále se specifikují atributy, které definují, co se s daným souborem může dělat. Znaménko plus indikuje udělení daného práva, znaménko mínus popření. Pomocí znaménka rovná se (=) se specifikují přesná přístupová práva. Přípustná přístupová práva jsou read (čtení), write (zápis) a execute (spouštění).

Pokud se použije v příkazu chmod volba R, pak se změna přístupových práv týká všech souborů v adresáři a všech podadresářů (R je označením pro rekurzivní). Jestliže se použije volba f, pak se příkaz chmod pokusí změnit přístupová práva práva i v případě, že uživatel není vlastníkem daného souboru. Zadá-li se volba v, pak bude příkaz chmod podrobně vypisovat informace o všech krocích, které realizuje.

Systémová statistika

Příkazy uvedené v tomto oddílu jsou určeny k výpisu informací o systému nebo o jeho částech.

`du [-abs] [cesta1 cesta2 ... cestaN]`

Příkaz du je zkratkou pro „disk usage“ (využívání disku). Zobrazuje informaci o velikosti diskového prostoru, který je přidělen danému adresáři a všem jeho podadresářům. Samotný příkaz du zobrazuje seznam, jehož položky uvádějí u každého adresáře velikost diskového prostoru (který adresář obsazuje), kolik prostoru obsazuje aktuální adresář a jako poslední udává velikost diskového prostoru spotřebovanou aktuálním adresářem a všemi jeho podadresáři. Pokud příkazu du předáte nějaké parametry (jméno souboru nebo adresáře), pak vám vypíše uvedené informace o tomto souboru či adresáři.

Použije-li se volba a, pak se vypíší uvedené informace jak o souborech, tak i o adresářích. Po zadání volby b se informace o velikosti diskového prostoru nebudou uvádět v kilobajtech, ale přímo v bajtech. Jeden bajt je ekvivalentní jednomu znaku v textovém souboru.

Pokud se použije volba s, pak příkaz du vypíše zmíněné informace o adresářích uvedených jako parametry a nikoliv o jejich podadresářích.

`df`

Příkaz df je zkratkou pro „disk filling“. Sumarizuje množství použitého diskového prostoru. Pro každý souborový systém (připomeňme, že souborový systém zaujímá buď samostatný disk, nebo samostatný diskový oddíl) vypíše informaci o celkovém množství diskového prostoru, informaci o velikosti použité části, o volné kapacitě a nakonec o celkové kapacitě souborového systému.

Paradoxně se může stát, že se vám zobrazí kapacita větší než 100 %. Je to způsobeno tím, že operační systém Unix rezervuje pro každý souborový systém jistý prostor pro superuživatele. Proto je zajištěno, že i když uživatel zaplní celý disk, zůstává na disku něco málo volného místa, aby se mohly realizovat některé operace.

Pro většinu uživatelů nemá příkaz df žádné užitečné volby.

`uptime`

Příkaz uptime dělá přesně to, co byste podle jeho názvu očekávali. Vypisuje celkový čas, který uplynul od posledního zavedení operačního systému.

Dále příkaz uptime uvádí informaci o aktuálním čase a o tzv. „load average“. Termínem „load average“ se míní průměrný počet úloh čekajících na spuštění v průběhu daného časového intervalu. Příkaz uptime uvádí tyto hodnoty pro poslední minutu, posledních pět minut a posledních deset minut. Jestliže se hodnota „load average“ blíží k nule, pak to znamená, že je systém téměř nevytížen. Hodnota blízká k jedničce znamená, že je systém plně vytížen, ale není zahlcen. Vysoké hodnoty jsou výsledkem skutečnosti, že v systému běží několik programů současně.

Příkaz uptime patří k několika málo programům operačního systému Unix, které nemají parametry a žádné volby. (V nových verzích programu uptime, který je součástí balíku procps, je akceptována volba V, která je určena pro zobrazení verze balíku procps).

`who`

Příkaz `who` slouží k zobrazení seznamu momentálně přihlášených uživatelů systému. Pokud se k příkazu přidají parametry `am i`, pak se zobrazí informace o aktuálním uživateli.

`w [-f] [`*`uživatelské jméno`*`]`

Příkaz `w` zobrazuje informace o momentálních uživatelích operačního systému a informace o tom, co dělají. V podstatě tento příkaz kombinuje příkazy `uptime` a `who`. Záhlaví výstupu z příkazu `w` je přesně stejné jako v případě příkazu `uptime` a každý řádek obsahuje informace o uživateli, například kdy se přihlásil. Kolonka `JCPU` ukazuje celkový čas procesoru, který uživatel spotřeboval, zatímco kolonka `PCPU` ukazuje celkový čas procesoru spotřebovaný jejich momentálně běžící úlohou.

Jestliže u příkazu `w` uvedete volbu `f`, pak se nezobrazí vzdálený systém, ze kterého je daný uživatel přihlášen (ve výpisu bude chybět kolonka `FROM`).

Co obsahují soubory

V operačním systému Unix existují dva hlavní příkazy pro výpis obsahu souborů – `cat` a `more`. Již jsme o nich hovořili v kapitole 6.

`cat [-nA] [`*`soubor1 soubor2 ... souborN`*`]`

Příkaz `cat` nepatří mezi uživatelsky přátelské příkazy. Nečeká, až si přečtete obsah souboru a spíše se používá v souvislosti s rourami. Má však několik užitečných voleb. Například volba `n` zajistí, že všechny řádky vypisovaného souboru budou číslovány. Při zadání volby `A` se budou řídící znaky zobrazovat jako normální znaky a ne jako podivné sekvence paznaků. Opět připomínáme, že kompletní seznam voleb akceptovatelných příkazem `cat` naleznete v manuálových stránkách. Pokud se v příkazovém řádku neuvede ani jeden parametr, použije příkaz `cat` standardní vstup.

`more [-l] [+`*`linenumber`*`] [`*`file1 file2 ... fileN`*`]`

Příkaz `more` je mnohem užitečnější a budete jej často používat zejména k zobrazování souborů ve formátu ASCII. Jedinou zajímavou volbou je `l`, po jejímž uvedení příkazu `more` sdělíte, že nechcete interpretovat znak Ctrl-L jako znak „nová stránka“. `more` zahájí zobrazení od specifikovaného čísla řádku (*`linenumber`*).

Protože je more interaktivním příkazem, uvádíme v následujícím seznam příkazů, kterými lze jeho činnost řídit.

Mezerník	**Zobrazí se následující stránka.**
`d`	**Text se posune o 11 řádků, což je přibližně polovina stránky.**
`/`	**Vyhledání regulárního výrazu. Zatímco regulární výraz může být opravdu komplikovaný, můžete zadat textový řetězec, který se má vyhledat. Zadáte-li například `/toad` Enter, vyhledá se v celém textu řetězec „`toad`“. Jestliže uvedete jen / Enter, pak se vyhledá další výskyt posledně zadaného řetězce.**
`n`	**Také tento příkaz je určen k vyhledání posledně zadaného regulárního výrazu.**
`: n`	**Pokud jste zadali prostřednictvím parametrů více než jeden soubor, zahájí se prohlížení následujícího souboru.**
`: p`	**Prohlížení se nastaví na předcházející soubor.**
`q`	**Program `more` se ukončí.**

`head [-lines] [file1 file2 ... fileN]`

Příkaz `head` zobrazí prvních deset řádků každého z uvedených souborů nebo prvních deset řádků ze standardního vstupu, pokud žádný soubor není jako parametr v příkazovém řádku uveden. Jakákoliv numerická hodnota uvedená jako volba změní implicitní nastavení deset. Například `head -15 frog` zobrazí prvních patnáct řádků souboru `frog`.

`tail [-lines] [file1 file2 ... fileN]`

Podobně jako příkaz `head` zobrazuje příkaz `tail` jen část souboru. Normálně zobrazuje posledních deset řádků souboru nebo posledních deset řádků ze standardního vstupu. Také akceptuje volbu pro nastavení počtu zobrazených řádků.

`file [file1 file2 ... fileN]`

Příkaz `file` se pokouší identifikovat formát souborů uvedených v seznamu v příkazovém řádku. Protože ne všechny soubory mají příponu charakterizující formát souboru nebo neobsahují posloupnosti znaků, podle kterých by se dal formát identifikovat, pokouší se příkaz `file` provádět některé základní testy a tak odhadnout, co soubor obsahuje.

Buďte opatrní, často se může stát, že je formát souboru identifikován chybně.

Informační příkazy

Následující oddíl je věnovaný příkazům, které mění soubor, provádějí jisté operace se souborem nebo zobrazují některé statistické informace o souboru.

`grep [-nvwx] [-number] expression [file1 file2 ... fileN]`

Jeden z nejužitečnějších příkazů operačního systému Unix je příkaz `grep` (generalized regular expression parser). Jeho jméno se zdá být příliš nadsazené na to, že jde o program, jenž umí pouze vyhledávat regulární výrazy v textu. Následující příklad demonstruje nejjednodušší způsob používání příkazu grep:

```
/home/larry# cat animals
Animals are very interesting creatures. One of my favorite animals is
the tiger, a fearsome beast with large teeth.
I also like the lion---it's really neat!
/home/larry# grep iger animals
the tiger, a fearsome beast with large teeth.
/home/larry#
```

Z uvedeného příkladu vyplývá jedna nevýhoda. I když příkaz vypsal všechny řádky obsahující hledané slovo, neuvedl, kde se dané slovo v souboru vyskytuje (neuvedl číslo řádku).

Někdy to může stačit, v závislosti na tom, co potřebujete udělat. Když například hledáte chyby ve výstupu z programu, stačí zadat příkaz `a.out | grep error`, kde `a.out` je jméno vašeho programu.

Jestliže však chcete přesně vědět, kde se hledaný regulární výraz v souboru nachází, zadejte volbu `n`. Pak bude příkaz `grep` zobrazovat i pořadová čísla řádků. Volbu `v` použijte tehdy, když budete chtít zobrazit seznam všech řádků neobsahujících daný regulární výraz.

K dalším vlastnostem příkazu `grep` patří to, že implicitně vyhledává neúplná slova. V předcházejícím příkladu jsme zadali neúplné slovo `iger` a `grep` našel slovo `tiger`. Jestliže má program `grep` pracovat pouze s celými slovy, zadejte volbu `w`. Po zadání volby `x` bude `grep` pracovat s celými řádky.

Pokud neuvedete v příkazu `grep` žádné parametry, bude prohledávat standardní vstup.

`wc [-clw] [soubor1 soubor2 ... souborN]`

`wc` je zkratka pro „word count" (počet slov). Tento příkaz spočítá slova, znaky a řádky v souborech uvedených v seznamu. Pokud se v příkazovém řádku neuvedou jako parametry žádné soubory, pracuje příkaz `wc` se standardním vstupem.

Pro příkaz `wc` existují tři volby: `c` (character – znak), `l` (line – řádek) a `w` (word – slovo). Pomocí volby se specifikuje, co má program `wc` počítat. Pokud například zadáte příkaz `wc -cw`, pak spočítá znaky a slova, ale nikoliv řádky. Příkaz zadaný bez voleb spočítá vše – znaky, slova i řádky.

Jedna z aplikací příkazu `wc` spočívá v tom, že lze s jeho pomocí určit počet souborů v adresáři: `ls | wc -w`. Pokud chcete například vědět, kolik souborů končí s příponou `.c`, pak zadejte příkaz `ls *.c | wc -w`.

`spell [soubor1 soubor2 ... souborN]`

`spell` je velmi jednoduchý program operačního systému Unix pro kontrolu pravopisu textových souborů. Zpravidla kontroluje pravopis americké angličtiny.[3] Program `spell` je tedy filtr, který prochází soubor ve formátu ASCII a na výstupu vypisuje slova, která považuje za pravopisně nesprávná. `spell` pracuje se soubory uvedenými v příkazovém řádku jako parametry, jinak zpracovává standardní vstup.

Pravděpodobně máte k dispozici i dokonalejší program pro kontrolu pravopisu `ispell`. Program `ispell` navíc nabízí možnosti opravy chybně napsaného slova. Pokud se v příkazovém řádku specifikuje soubor, pak se program `ispell` ovládá pomocí nabídky, jinak pracuje jako klasický filtr.

Chcete-li se dozvědět o programu ispell více, podívejte se do manuálových stránek.

`cmp soubor1 [soubor2]`

Příkaz `cmp` porovnává obsahy dvou souborů. První musí být uveden jako parametr v příkazovém řádku, druhý je buď specifikován jako parametr, nebo se čte ze standardního vstupu. Příkaz `cmp` je velmi jednoduchý a jeho úkolem je ukázat, kde se dva soubory liší.

`diff soubor1 soubor2`

Jedním z nejkomplikovanějších standardních programů operačního systému Unix je příkaz `diff`. GNU verze programu `diff` má více než dvacet voleb! Jedná se o zdokonalenou verzi programu `cmp`, která ukazuje nejen kde se soubory liší, ale také v čem se liší.

Nebudeme probírat kompletní možnosti programu `diff`, protože to překračuje rámec této knihy. Místo toho se zaměříme na základní operace. Krátce řečeno, program `diff` akceptuje jako parametry dva soubory a zobrazí rozdíly mezi nimi na principu „řádek po řádku". Uveďme si příklad:

```
/home/larry# cat frog
Animals are very interesting creatures. One of my favorite
  animals is
the tiger, a fearsome beast with large teeth.
I also like the lion---it's really neat!
```

3 I když existuje několik verzí tohoto programu pro evropské jazyky, distribuce operačního systému Linux většinou obsahuje kontrolu pravopisu americké angličtiny.

```
/home/larry# cp frog toad
/home/larry# diff frog toad
/home/larry# cat dog
Animals are very nteresting creatures. One of my favorite
  animals is
the tiger, a fearsome beast with large teeth.
I also like the lion---it's really neat!
/home/larry# diff frog dog
1c1,2
< Animals are very interesting creatures. One of my favorite
  animals is
---
> Animals are very nteresting creatures. One of my favorite
  animals is
>
3c4
< I also like the lion---it's really neat!
---
> I also like the lion---it's really neat!
/home/larry#
```

Jak vidíte v našem příkladu, program `diff` neprodukuje žádný výstup, pokud jsou soubory identické. Pokud se porovnávaly dva různé soubory, pak řádek `1c1,2` říká, že byl porovnán první řádek levého souboru (`frog`) s prvním a druhým řádkem pravého souboru (`dog`) a že zde byly nalezeny rozdíly. Pak byly porovnány řádky 3 (v souboru `frog`) a 4 (v souboru `dog`) a i zde byl nalezen rozdíl. Může se zdát podivné, že se nejdříve porovnávají řádky s různým pořadovým číslem. Je to však z toho důvodu, aby program pracoval efektivněji.

`gzip [-v#] [`*`soubor1 soubor2 ... souborN`*`]`
`gunzip [-v] [`*`soubor1 souborX (xÎ{1,2,...,N})`*`]`
`zcat [`*`soubor1 soubor2 ... souborN`*`]`

Tyto tři programy se používají ke kompresi a dekompresi dat.

Příkaz `gzip` (GNU zip) je program, který čte původní soubory a produkuje soubory menší. Přitom soubory specifikované v příkazovém řádku ruší a nahrazuje je komprimovanými soubory. Komprimované soubory mají stejná jména, ale navíc mají příponu „`gz`“.

`tr` *`řetězec1 řetězec2`*

Příkaz `tr` (translate characters) pracuje pouze se standardním vstupem – neakceptuje žádné soubory jako parametry. Jeho dvěma parametry jsou dva řetězce. Příkaz nahradí všechny výskyty znaků řetězce *`řetězec1`* na vstupu odpovídajícím znakem z řetězce *`řetězec2`*. Kromě relativně jednoduchého tvaru tohoto příkazu, například `tr frog toad`, může příkaz `tr` akceptovat poměrně složité příkazy. Uvedeme si příklad, kdy se pomocí příkazu `tr` převádějí malá písmena na velká:

```
/home/larry# tr [:lower:] [:upper:]
this is WEIRD sentence.
THIS IS WEIRD SENTENCE.
/home/larry#
```

Program `tr` je dosti složitý a často se využívá v malých programech pro příkazové procesory.

Editování souborů pomocí editoru Emacs

Co je to Emacs?

Abyste mohli dělat něco rozumného s počítačem, potřebujete mít možnost ukládat texty do souborů a měnit texty, které jsou v souborech uloženy. Takové úkony vám umožní realizovat textový editor. Emacs je jeden z nejpopulárnějších editorů na světě – zejména proto, že i začátečníkům umožňuje bez problémů pracovat s textovými soubory. Klasický editor dodávaný s operačním systémem Unix, `vi`, probereme v dodatku A.

Než se začnete učit pracovat s editorem Emacs, musíte si najít nějaký soubor obsahující obyčejný text a zkopírovat si jej do domovského adresáře.[1] Na začátek by bylo riskantní editovat originální soubor, mohl by obsahovat důležité informace. Editor Emacs se spouští jednoduše:

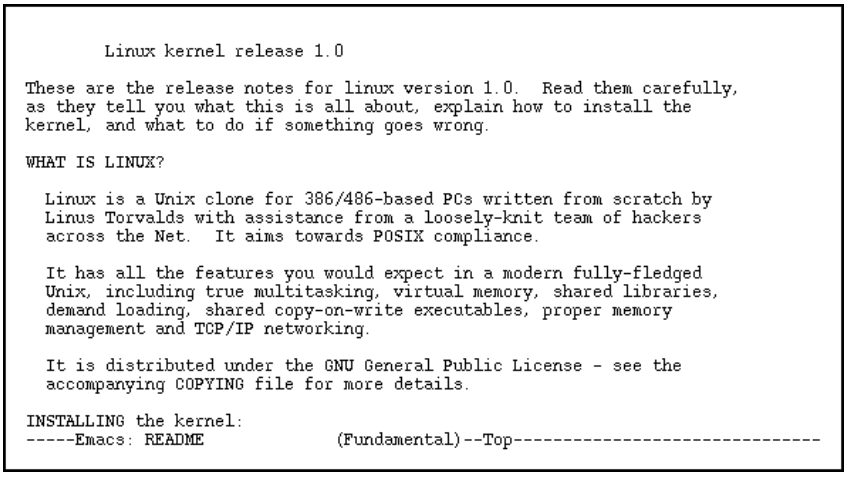
```
/home/larry# emacs README
```

Pokud jste se rozhodli zkopírovat soubor `/etc/rc`, `/etc/inittab` nebo jiný, pak samozřejmě jako parametr uvedete jméno tohoto souboru – například `emacs rc`.

Spuštění editoru Emacs může mít různý efekt. Záleží to na tom, v jakém prostředí jej spouštíte. V textovém terminálu zobrazujícím pouze textové znaky vytvoří nastartovaný editor Emacs okno přes celou obrazovku. Pokud jej spustíte v systému X Window, vytvoří si editor své vlastní okno. Budeme předpokládat, že jste spustili editor Emacs v textovém terminálu. Vše, co zde budeme uvádět, bude platit i pro editor Emacs spuštěný v okně systému X Window. V systému X Window nezapomínejte přesunout kurzor myši na okno obsahující editor, jinak nebudete moci nic psát.

Vaše obrazovka (nebo okno v systému X Window) by měla vypadat přibližně tak, jak je zobrazena na obrázku 8.1. Většina obrazovky obsahuje text vašeho souboru. Obzvláště důležité jsou poslední dva řádky, zejména chcete-li se naučit pracovat s tímto editorem. Řádek obsahující dlouhé posloupnosti pomlček je řádek specifikující mód editoru (modeline).

1 **Například** `cp /usr/src/linux/README ./README`



Obrázek 8.1 – Editor Emacs byl právě nastartován příkazem `emacs README`

Na uvedeném obrázku vidíte v předposledním řádku slovo „Top“. Může zde být také uvedeno slovo „All“ a mohou se zde také vyskytovat malé rozdíly, což záleží na verzi editoru. Někteří uživatelé mají v tomto řádku zobrazen aktuální čas. Pod uvedeným řádkem se nachází informační a **příkazový řádek**, někdy označovaný jako „minibuffer“, jindy jako „echo area“. Informační řádek slouží editoru ke komunikaci s uživatelem, k vypisování zpráv a někdy zde jeho prostřednictvím budete zadávat příkazy. Budou se zde objevovat takové pokyny, jako: „`For information about GNU project and its goals, type C-h C-p.`“ Zatím takové pokyny ignorujte, budeme se jimi zabývat později.

Než opravdu změníte text v nějakém souboru, musíte se naučit „pohybovat“ textem a pohybovat kurzorem. Kurzor by měl být umístěn na začátku souboru v levém horním rohu obrazovky. Chcete-li kurzor posunout o jeden znak dopředu, stiskněte kombinaci kláves C-F (stiskněte klávesu Ctrl, držte ji stisknutou a stiskněte klávesu F). Kurzor se posune o jeden znak dopředu. Pokud budete obě klávesy stále držet stisknuty, bude se pohyb kurzoru opakovat. Všimněte si, že když kurzor dospěje na konec řádku, automaticky se přesune na začátek následujícího řádku. Opačného směru pohybu kurzoru dosáhnete pomocí kláves C-B. Pokud chcete posunout kurzor o jeden řádek dolů, použijte klávesy C-N, pokud jej chcete posunout o jeden řádek nahoru, stiskněte klávesy C-P[2]

Používání klávesy Ctrl zajišťuje nejrychlejší způsob, jak při editování textu pohybovat kurzorem. Editor Emacs používá takové kombinace kláves, které při psaní udržují vaše ruce nad klávesnicí. Jestliže jste však zvyklí používat kurzorové klávesy, budou také fungovat.

Pokud pracujete v systému X Window, můžete k nastavení pozice textového kurzoru použít kurzor myši. Stačí kurzor myši nastavit na požadovanou pozici v textu a klepnout levým tlačítkem. Tento způsob je však velmi pomalý (jednou rukou musíte opustit klávesnici, uchopit myš, nastavit kurzor a pak se na klávesnici zase vrátit) a nedoporučujeme si na něj zvykat. Většina uživatelů, kteří pracují s editorem Emacs dlouhodobě, používá k pohybu kurzoru výhradně klávesnici.

2 Zajisté jste si již všimli, že příkazy editoru Emacs jsou většinou realizovány pomocí kombinace dvou kláves.

Pomocí kláves C-P a C-B nastavte kurzor opět na začátek souboru, tedy horní levý roh obrazovky. Nyní podržte klávesy C-B stisknuty poněkud déle. Uslyšíte pípnutí a v informačním řádku uvidíte zprávu „`Beginning of buffer`“.

V tomto okamžiku se asi podivíte. Co je to buffer?

Když editor Emacs zpracovává soubor, nepracuje ve skutečnosti se souborem přímo. Zkopíruje si jeho obsah do speciální pracovní oblasti, která se nazývá **buffer** – zde můžete obsah souboru modifikovat. Až ukončíte editaci souboru, dáte příkaz, aby editor buffer uložil. Do té doby zůstane obsah původního souboru nezměněn a změny se provádějí pouze uvnitř pracovní oblasti editoru.

Nyní jsme tedy připraveni obsah bufferu modifikovat. Vše, co jsme dělali doposud, bylo „nedestruktivní“, což znamená, že jsme obsah bufferu neměnili. Předpokládejme, že chcete do editovaného souboru zapsat znak „X“. Jakmile stisknete klávesu X, změní se předposlední řádek obrazovky. Editor Emacs registruje změny v editovaném textu a jakmile nějaké nastanou, informuje o tom uživatele pomocí dvou hvězdiček před slovem Emacs. Pak bude uvedený řádek vypadat asi takto:

```
--**-Emacs: some_file.txt     (Fundamental)--Top--------------------
```

Hvězdičky zůstanou zobrazeny do té doby, než obsah bufferu uložíte. Obsah bufferu můžete v průběhu editace ukládat průběžně vícekrát – stačí stisknout klávesy C-X C-S (stiskněte klávesu Ctrl, držte ji stisknutou a pak stiskněte klávesu „X“ a „S“). Průběžné ukládání obsahu bufferu má význam především při vytváření nebo modifikaci velkých souborů.

Nyní si uvedeme seznam několika málo příkazů používaných v editoru Emacs spolu s těmi, které jsme již popsali. Je na vás, abyste si jejich používání procvičili. Než pokročíme dále, měli byste s těmito příkazy být důkladně seznámeni.

C-F	Posunutí kurzoru o jeden znak doprava.
C-B	Posunutí kurzoru o jeden znak doleva.
C-N	Posunutí kurzoru o jeden řádek dolů.
C-P	Posunutí kurzoru o jeden řádek nahoru.
C-A	Umístění kurzoru na začátek řádku.
C-E	Umístění kurzoru na konec řádku.
C-V	Zobrazení následující stránky.
C-L	Zobrazení stránky s aktuálním řádkem uprostřed.
C-D	Zrušení znaku, na kterém je umístěn kurzor.
C-K	Zrušení textu od pozice kurzoru do konce řádku.
C-X C-S	Uložení obsahu bufferu do odpovídajícího souboru.
Backspace	Zrušení znaku vlevo od kurzoru.

Používání editoru pod systémem X Window

Pokud chcete rychle editovat soubory pod systémem X Window, máte poněkud ulehčenou situaci. Editovaný text je zobrazen pod nabídkou funkcí editoru, proto je práce s editorem intuitivnější.

V textovém módu uvedená nabídka není dostupná.

Když poprvé spustíte editor Emacs pod systémem X Window, obsahuje nabídka čtyři hlavní položky: `Buffers`, `File`, `Edit` a `Help`. Potřebujete-li si některou z položek hlavní nabídky zpřístupnit, nastavte kurzor myši na tuto položku, stiskněte levé tlačítko myši (držte je stále stisknuté). Pak přesuňte kurzor na požadovanou funkci a tlačítko myši uvolněte. Jestliže žádnou funkci vybrat nechcete, přesuňte kurzor myši mimo nabídku a opět uvolněte tlačítko myši.

Nabídka `Buffers` obsahuje seznam různých souborů, které v této relaci s editorem Emacs editujete. Nabídka `File` obsahuje příkazy pro zavádění a ukládání souborů. Funkce této nabídky podrobněji popíšeme později. Nabídka `Edit` obsahuje některé nejdůležitější příkazy pro editování textového souboru a prostřednictvím nabídky `Help` si můžete interaktivně zobrazovat vybrané části dokumentace k editoru.

Jistě jste si všimli, že u každé funkce v nabídce je uvedena ekvivalentní kombinace kláves pro realizaci této funkce. Na jedné straně můžete funkce vyvolávat pomocí myši a menu, na druhé straně můžete k tomuto účelu používat ekvivalentní kombinace kláves, což je samozřejmě rychlejší.

Editování více souborů současně

V daném okamžiku je editor Emacs schopen pracovat s více soubory. Ve skutečnosti platí, že počet souborů, které mohou být současně uloženy v bufferech editoru, je omezen pouze velikostí paměti počítače. Po zadání příkazu Ⓒ-Ⓧ Ⓒ-Ⓕ se do bufferu editoru Emacs zavádí nový soubor. Když tento příkaz zadáte, v příkazovém řádku se objeví výzva k zadání jména souboru:

```
Find file: ~/
```

Syntaxe používaná k zadání jména souboru je stejná, jako v příkazovém řádku příkazového procesoru; lomítka se používají k oddělení adresářů a podadresářů, znak ~ je interpretován jako váš domovský adresář. I zde funguje možnost automatického **doplňování jména souboru** – zadáte-li dostatek znaků k tomu, aby mohlo být jméno souboru jednoznačně identifikováno, pak stačí stisknout klávesu Tab a zbytek jména souboru se doplní automaticky (nebo se zobrazí seznam jmen všech souborů, která zadané specifikaci vyhovují). Podobnou funkci jako klávesa Tab má klávesa mezerník. Necháme na vás, abyste zjistili, v čem se funkce vyvolané těmito klávesami liší. Máte-li zadáno jméno souboru, který chcete editovat, stiskněte klávesu Enter; editor zavede obsah souboru do své pracovní oblasti a zobrazí jej na obrazovce. V terminologii editoru Emacs se tento proces nazývá vyhledání souboru. Najděte si nějaký další soubor a popsaným způsobem jej zaveďte do editoru. Nyní máte v editoru nový buffer. Budeme předpokládat, že původní má název `some_file.txt` a nový má název `another_file.txt`. Zdá se, že se váš první buffer ztratil. Pravděpodobně se divíte, kam se poděl.

Neobávejte se, stále zůstává uvnitř editoru a k jeho opětovnému zobrazení stačí stisknout klávesu Ⓒ-Ⓧ Ⓑ. Pak se vás editor v příkazovém řádku zeptá, do kterého bufferu se má přepnout. Nabídne implicitní buffer, t.j. buffer, jehož obsah se zobrazí po stisknutí klávesy Enter (nemusíte jméno bufferu uvádět). Implicitní buffer je ten, který jste „opustili“ jako poslední. Jestliže tedy pracujete se dvěma soubory a často mezi nimi přepínáte, používejte klávesu Ⓒ-Ⓧ Ⓑ a nemusíte při každém přepnutí zadávat jméno souboru. Samozřejmě můžete jméno uvádět, ale bude vás to zdržovat.

I při přepínání mezi buffery lze použít funkci k doplňování jména – stejně jako v případě vyhledávání souboru. Po stisknutí klávesy Tab doplní editor automaticky jméno bufferu, je-li schopen jej z doposud zadaných znaků jednoznačně identifikovat. Kdykoliv jste v příkazovém řádku vyzváni k zadání jména, vyzkoušejte si, zda je editor schopen jméno automaticky doplnit. Automatické doplňování vám ušetří spoustu času a editor je schopen je realizovat pokaždé, když má vybrat nějakou položku z nějakého předdefinovaného seznamu.

Vše, co jste se naučili o příkazech pro editování v prvním bufferu platí i v druhém. Pokročme dále – v novém bufferu změňte nějaký text, ale neukládejte jej (pomocí kláves C-X C-S). Nyní předpokládejme, že nechcete provedené změny uložit a že chcete buffer zrušit. K tomu slouží příkaz C-X K. Nejdříve se vás editor zeptá, který buffer chcete zrušit. Stisknete-li klávesu Enter, zruší se implicitní buffer (což je asi nejčastější případ). Editor se vás znovu zeptá, zda chcete buffer opravdu zrušit, a když zadáte „yes“ a stisknete Enter, pak jej konečně zruší.

Nyní byste se měli důkladně procvičit v zavádění souborů, jejich modifikaci, ukládání, přepínání mezi buffery a rušení bufferů. Ujistěte se, že needitujete nějaké důležité soubory, které by mohly poškodit funkci vašeho systému.[3] Vytvořte si alespoň pět bufferů a vyzkoušejte si přepínání mezi nimi.

Ukončení práce s editorem

Když ukončíte práci s editorem Emacs, ujistěte se, že všechny buffery, které mají být uloženy jsou skutečně uloženy. Pak můžete editor ukončit pomocí kláves C-X C-C. Někdy se vás editor po zadání kláves C-X C-C v příkazovém řádku na něco zeptá. Nebuďte znepokojeni a odpovězte normálním způsobem. Jestliže se domníváte, že se budete do editoru chtít později vrátit, nepoužívejte klávesy C-X C-C, ale klávesu C-Z. Pak bude aktivita editoru pouze pozastavena. K opětovné aktivaci editoru můžete použít známý příkaz `fg`. Pozastavení editoru je mnohem efektivnější, než jej stále dokola ukončovat a znovu startovat, zejména když opakovaně editujete tytéž soubory.

V systému X Window má funkce C-Z jiný efekt – provede transformaci okna s editorem do ikony. Podrobně jsme tento mechanismus popsali v kapitole 5. Existují tedy dvě možnosti, jak transformovat okno obsahující editor do ikony – buď normálním způsobem pomocí správce oken, nebo pomocí klávesy C-Z. V systému X Window však k opětovné aktivaci editoru nelze použít příkaz fg – musíte použít správce oken.

Klíče Meta

Doposud jsme při práci s editorem Emacs používali kombinace kláves s klávesou Ctrl. Existují však další možné kombinace, a to s klávesou Meta. Těmto kombinacím se v terminologii editoru Emacs říká meta-klíče. Bohužel ne všechny klávesnice mají klávesu Meta umístěnu ve stejné poloze a některé ji nemají vůbec. V případě klávesnic u počítačů IBM PC je klávesa Meta totožná s klávesou Alt.

Klávesu Meta si můžete otestovat. Stiskněte tu klávesu, o které si myslíte, že by mohla být klávesou Meta, a zároveň stiskněte klávesu X. Pokud se v příkazovém řádku objeví malá nápověda (zpravidla M-X), pak jste kýženou klávesu nalezli. Stiskněte klávesu C-G a znovu se vrátíte do bufferu editoru Emacs.

3 Pokud nejste přihlášeni jako uživatel root, neměli byste být schopni vašemu systému ublížit, ale stejně buďte opatrní.

Jestliže se vám v příkazovém řádku nic neobjeví, stále existuje řešení. Místo klávesy Meta můžete použít klávesu Escape. Avšak v tomto případě ji nedržte stisknutou – stiskněte ji, uvolněte a zadejte další klávesu reprezentující danou funkci. Vyzkoušejte si naši oblíbenou kombinaci, tedy Escape a pak „X“. Nyní se vám v příkazovém řádku nápověda určitě objeví. Opět stiskněte klávesu C-G. Klávesa C-G je v editoru Emacs určena ke zrušení čehokoliv, co nehodláte realizovat. Editor zpravidla vyšle zvukový signál, aby vás upozornil, že danou funkci rušíte.[4]

Označení M-X je analogické označení C-X (kde x je klávesa reprezentující nějakou funkci). Pokud jste nalezli skutečný klíč Meta, pak jej používejte. Jinak budete muset používat popsanou sekvenci s klávesou Escape. Kdekoliv v následujícím textu uvidíte M-X, pak to znamená, že máte použít meta-klíč.

Práce s bloky textu

Editor Emacs, tak jako prakticky každý editor, umožňuje pracovat s bloky textu. Abyste tyto funkce mohli využívat, musíte mít prostředek pro definování **začátku bloku** a **konce bloku**. V editoru Emacs se definice bloku realizuje pomocí nastavení dvou pozic v bufferu, které se označují jako mark (značka) a point. Chcete-li v bufferu nastavit začátek bloku, nastavte kurzor na požadovanou pozici a stiskněte klávesu C-SPC (SPC je zkratka pro mezerník). V příkazovém řádku se objeví zpráva „Mark set“.[5] Nyní je začátek bloku nastaven. Editor nepoužívá žádné zvýrazňovací prostředky k tomu, aby byl začátek bloku viditelný.

A co konec bloku? Konec bloku je definován aktuální pozicí kurzoru. Proto se pro něj používá v terminologii editoru Emacs termín point. Point zrovna tak například znamená místo, od kterého se má vkládat text při kopírování nebo vkládání bloků. Nastavením začátku bloku a přesunutím kurzoru na kteroukoliv jinou pozici v textu je definován začátek a konec bloku. Tento blok se nazývá **region**. Region je tedy vždy část textu mezi značkou a aktuální pozicí kurzoru.

Pouhá definice regionu nezpřístupňuje blok ke kopírování. Nejdříve musíte blok „**zkopírovat**“ (copy), abyste jej pak mohli „**nalepit**“ (paste) někam jinam. Má-li se blok zkopírovat, stiskněte klávesu M-W. Nyní je blok uložen ve speciálním bufferu editoru Emacs. Jestliže nyní chcete blok nalepit někam jinam, nastavte si kurzor na požadovanou pozici a stiskněte C-Y.

Pokud chcete blok textu přesunout (a ne zkopírovat), pak místo příkazu M-W použijte příkaz C-W. Tímto způsobem se blok po přenesení do speciálního bufferu vymaže. Když zjistíte, že jste blok nechtěli vymazat, stiskněte klávesu C-Y a blok se obnoví. Místo, do kterého editor Emacs ukládá části textu, se nazývá **kill-ring**. U jiných editorů se tato oblast nazývá „clipboard“ nebo „paste buffer“.

Existuje ještě jeden způsob, jak vystřihovat a nalepovat text: kdykoliv stisknete klávesu C-K, dojde ke zrušení textu od pozice kurzoru do konce řádku a přitom se zrušený text také ukládá do oblasti kill-ring. Pokud takto zrušíte více než jeden řádek, uloží se všechny řádky do oblasti kill-ring a vy máte možnost je později nalepit najednou.

Někdy je tento způsob přesouvání a kopírování textu rychlejší, než používat systém s označováním začátku a konce bloku. Záleží jen na vás, kterému způsobu dáte přednost.

---

4 Příležitostně se může stát, že jedno stisknutí klávesy C-G nepřesvědčí editor Emacs, že chcete opravdu přerušit činnost, kterou právě děláte. Stačí však trvat na svém a pak se editor vrátí do předcházejícího módu.

5 Na některých počítačích příkaz C-SPC nebude fungovat. Místo toho použijte C-@.

Vyhledávání a náhrada řetězců

V editoru Emacs máte několik možností, jak vyhledávat text. Některé způsoby jsou komplikované a nemá smysl je zde popisovat. Nejjednodušší a nejčastěji používaná metoda je tzv. postupné vyhledávání označované jako „isearch“ (incremental search). Předpokládejme, že potřebujete vyhledat řetězec „gadfly“ v následujícím textu:

```
I was growing afraid that we would run out of gasoline, when my passenger ex-
claimed
''Gadzooks! There's gadfly in here!''.
```

Nastavte pozici kurzoru na začátek textu nebo na místo, o kterém víte, že předchází hledanému řetězci, a zadejte příkaz C-S. Tím nastavíte editor Emacs do módu vyhledávání. Nyní začněte zadávat řetězec, který chcete vyhledat. Jakmile však napíšete první znak, tedy „g“ „přeskočí“ kurzor na první výskyt písmene „g“ v textu. Jestliže máte v editoru text uvedený v našem příkladu, pak kurzor skočí na začátek slova „growing“. Nyní napište písmeno „a“ (druhé písmeno ve slově „gadfly”) a editor přesune kurzor na slovo „gasoline“, protože vyhledal první výskyt dvojice znaků „ga“. Když zadáte další písmeno „d“, skočí kurzor na slovo „gadzooks“ a nakonec po zadání písmene „f“ skočí na hledané slovo „gadfly“. Přitom jste nemuseli zadat kompletní hledaný řetězec.

Při postupném vyhledávání funguje editor Emacs tak, že po každém zadání dalšího znaku hledaného řetězce vyhledá první výskyt toho slova, jehož začátek je shodný s doposud zapsanými znaky. Jakmile zadáte tolik znaků, aby vyhledávání bylo jednoznačné, můžete funkci ukončit stisknutím klávesy Enter. Jestliže se domníváte, že hledaný řetězec je nad aktuální pozicí kurzoru, pak zadejte příkaz C-R, čímž inicializujete zpětné vyhledávání.

Pokud naleznete první výskyt zadaného řetězce, ale zajímá vás jeho další výskyt, pak znovu zadejte příkaz C-S. Editor Emacs vyhledá následující výskyt zadaného řetězce a tak můžete pokračovat dále. Když editor výskyt hledaného řetězce nenalezne, vypíše zprávu, že řetězec nenašel a začne znovu vyhledávat od začátku bufferu. Podobná pravidla platí pro příkaz C-R.

Nyní si popsané funkce vyzkoušejte. Najděte si soubor s anglickým textem a vyhledejte v něm výskyt slova „the“. Pak pomocí opakované funkce C-S najděte další výskyty. Všimněte si, že editor přitom vyhledá i jiné řetězce, například „them“, protože vyhovují zadané specifikaci. Jestliže chcete vyhledat pouze řetězec „the“, pak na konec hledaného řetězce budete muset přidat mezeru. Při editování hledaného řetězce můžete používat standardní editační klávesy Backspace a Delete. Chcete-li vyhledávání ukončit, vždy použijte klávesu Enter.

Editor Emacs také umožňuje nahradit výskyt jednoho řetězce jiným. Tento proces se v terminologii editoru Emacs označuje jako **query-replace**. Proces zahájíte tak, že stisknete M-X a napíšete `query-replace` a stisknete Enter.

I v případě zadávání příkazů funguje v editoru Emacs doplňování, a proto stačí, když například napíšete „query-re“ a stisknete klávesu Tab. Předpokládejme, že chcete řetězec „gadfly“ nahradit řetězcem „housefly“. V příkazovém řádku „`Query replace:`“ zadejte „gadfly“ a stiskněte Enter. Jak budete opět vyzváni k zadání řetězce, kterým se má původní řetězec nahradit – zadejte „housefly“. Editor Emacs bude nyní procházet text, zastavovat kurzor na každém výskytu řetězce „gadfly“ a bude se vás ptát, zda má nalezený řetězec nahradit. Pokud ano, stiskněte klávesu Y, pokud ne, stiskněte klávesu N. Jestliže je pro vás předcházející výklad příliš komplikovaný, popsané funkce si vyzkoušejte. Bude to mít pro vás větší význam, než kdybyste předcházející odstavce četli desetkrát.

Vnitřní funkce editoru Emacs

Všechny funkce editoru Emacs vyvolané stisknutím nějaké kombinace kláves mají nějaký název, kterému editor „rozumí“. Například klávesa C-P znamená pro editor Emacs provedení vnitřní funkce `previous-line`. Všechny vnitřní funkce mohou být volány prostřednictvím svého jména, a to po stisknutí klávesu Alt+X. Když například zapomenete, jaký klíč máte použít k nastavení kurzoru na předcházející řádek, stačí zadat: M-X `previous-line` a Enter. Vyzkoušejte si to a přesvědčte se, že C-P a M-X `previous-line` realizují tutéž funkci.

Autor editoru Emacs postupoval tak, že nejdříve definoval celou množinu vnitřních funkcí editoru a pak teprve navrhl klávesové zkratky pro realizaci nejčastěji používaných funkcí. Někdy je jednodušší použít explicitní volání funkce prostřednictvím M-X, než si pamatovat klávesovou zkratku svázanou s touto funkcí. Například funkce `query-replace` je u některých verzí editoru Emacs svázána s klávesou M-X. Kdo si ale má pamatovat takovou divnou kombinaci? Pokud budete náhradu řetězců provádět extrémně často, pak má samozřejmě smysl si klávesovou zkratku pamatovat.

Většina kláves, které stisknete, jsou písmena, číslice, případně další znaky, které se mají vkládat do textového bufferu. Každá z těchto kláves je **svázána** s funkcí, jež má jméno `self-insert-command`. Tato funkce nedělá nic jiného, než že dané písmeno nebo číslici vloží do bufferu. Kombinace kláves, například s klávesou Ctrl, jsou obecně svázány s jinými funkcemi – pohyb kurzoru a podobně. Například kombinace C-V je svázána s funkcí `scroll-up`, což znamená, že se editovaný text posune o jednu obrazovku dolů.

Jak ale postupovat, když do textu chcete vložit řídící znak? Konec konců, řídící znaky jsou také znaky ASCII, i když zřídkakdy používané. Může se stát, že budete chtít do textu takový znak vložit. Proto v editoru Emacs existuje prostředek, který zabrání editoru interpretovat kombinaci klávesy s klávesou Ctrl jako příkaz. Klávesa C-Q[6] je svázána se speciální funkcí, která má název `quoted-insert`. Jediné, co funkce `quoted-insert` dělá, je, že přečte následující klávesu a vloží ji do textového bufferu bez interpretace. Pokud chcete do textu vložit samotný znak C-Q, pak zadejte C-Q dvakrát.

V editoru Emacs existují některé funkce, jež nejsou svázány s žádnou kombinací kláves. Když například píšete dlouhý text, pak asi nebudete chtít ukončovat každý řádek klávesou Enter. Po editoru Emacs můžete chtít, aby to dělal za vás (po editoru Emacs můžete chtít cokoliv). Příkaz, který takovou funkci realizuje, má název auto-fill-mode. Není však implicitně svázán s žádnou kombinací kláves. Jestliže chcete tento příkaz inicializovat, zadejte `M-X auto-fill-mode`. Jak jsme si již řekli, „M-X“ je klíč umožňující vyvolat vnitřní funkci editoru jménem. Takto byste mohli vyvolat i funkci `next-line` (následující řádek) nebo `previous-line` (předcházející řádek), ale to by bylo velmi neefektivní, protože uvedené funkce jsou svázány s klávesami C-N a C-P.

Mimochodem, když se po vyvolání funkce `auto-fill-mode` podíváte na předposlední řádek, uvidíte zde na pravé straně slovo Fill. Dokud se zde toto slovo vyskytuje, bude editor Emacs ukončovat řádky za vás. Když funkci „`M-X auto-fill-mode`“ vyvoláte znovu, automatické ukončování řádků přestane fungovat – funkce je vytvořena jako přepínač.

Může se vám zdát, že zadávání funkcí prostřednictvím jejich dlouhých jmen není příliš pohodlné. Naštěstí i v tomto případě v editoru Emacs funguje doplňování jmen (stejně jako doplňování jmen souborů). Proto platí, že zřídkakdy budete muset vypisovat celé jméno funkce, písmeno po písmenu. Pokud si nejste jisti, zda bude editor schopen doplnit jméno, stiskněte klávesu Tab. V horším případě se vám objeví znak Tab, v lepším případě editor provede doplnění.

---

6 Pro kombinaci kláves C-Q se používá označení „klávesa“, protože představuje jediný znak z tabulky ASCII.

Nápověda v editoru Emacs

Editor Emacs má velmi rozsáhlou nápovědu. Tak rozsáhlou, že se o ní zmíníme jen velmi stručně. Nejvyšší úroveň nápovědy se vyvolá stisknutím klíče Ⓒ-Ⓗ a nějakého písmene. Například Ⓒ-Ⓗ Ⓚ vyvolá nápovědu vztahující se ke klíčům (budete vyzváni k zadání klíče a pak se objeví text s vysvětlením, jakou funkci klíč realizuje). Ⓒ-Ⓗ Ⓣ vyvolá výukový program editoru Emacs. K důležitým klíčům patří Ⓒ-Ⓗ Ⓒ-Ⓗ Ⓒ-Ⓗ, kdy se vám objeví „nápověda o nápovědě“. Zde se dozvíte vše o systému nápovědy. Když znáte jméno funkce editoru Emacs (například `save-buffer`), ale nemůžete si vzpomenout na klíč k jejímu vyvolání, použijte Ⓒ-Ⓗ Ⓦ („`where is`“) a zadejte jméno funkce. Zrovna tak máte možnost zjistit podrobné informace o dané funkci – zadejte Ⓒ-Ⓗ Ⓕ a pak jméno funkce.

Mějte na paměti, že editor Emacs je sice schopen doplňovat jména funkcí, avšak nemůžete na tuto vlastnost příliš spoléhat, pokud k nějaké funkci (jejíž název přesně neznáte) potřebujete nápovědu. Jestliže si myslíte, že můžete odhadnout slovo, kterým funkce začíná, zkuste je napsat a stiskněte Tab. Uvidíte, zda editor byl schopen funkci identifikovat. Pokud ne, zkuste něco jiného. Totéž platí pro jména souborů. I když si nemůžete vzpomenout, jak se jmenuje soubor, který jste editovali před mnoha měsíci, můžete název odhadnou a zkusit, co na to editor odpoví. Doplňování jmen je tedy nejen užitečné v tom, že šetří čas, ale pomáhá vám nalézt to, co jste již zapomněli, nebo to, co si přesně nepamatujete.

Existuje několik dalších znaků, které můžete zadat po příkazu Ⓒ-Ⓗ a tak získat nápovědu různým způsobem. Nejčastěji však budete používat Ⓒ-Ⓗ Ⓚ, Ⓒ-Ⓗ Ⓦ a Ⓒ-Ⓗ Ⓕ. Až budete blíže seznámeni s editorem Emacs, vyzkoušejte si například Ⓒ-Ⓗ Ⓐ. Pak se vás editor zeptá na řetězec a mezi všemi jmény funkcí nalezne to, které daný řetězec obsahuje. (Písmeno „a“ je zkratkou pro „apropos“ nebo „about“.)

Jiný zdroj informací o editoru Emacs nabízí systém pro čtení dokumentace v hypertextovém formátu **Info**. Ten můžete inicializovat přímo z editoru Emacs tak, že stisknete klíč Ⓒ-Ⓗ Ⓘ. Pak se vám objeví základní stránka systému Info, ve které najdete další pokyny, jak postupovat při vyhledávání informací.

Pracovní módy editoru Emacs

Každý buffer editoru Emacs je spjat s tzv. módem.[7] Módy byly zavedeny z toho důvodu, že například při psaní zprávy pro elektronickou poštu má uživatel jiné požadavky, než při psaní programu v jazyku C. Kdyby měl editor splňovat všechny požadavky najednou, bylo by jeho používání velmi komplikované.

Proto se autor editoru Emacs[8] rozhodl řešit tuto situaci pomocí módů. Chování editoru se mění podle toho, s jakým módem je konkrétní buffer spjat. Jednotlivé módy se od sebe liší vazbou mezi klíči a funkcemi, ale jsou zde i jiné rozdíly.

Nejzákladnějším módem je mód `fundamental`, který nemá žádné speciální funkce. Uvedeme zde, co o základním módu říká samotný editor Emacs:

```
Fundamental mode:
Major mode not specialized for anything in particular.
Other major modes are defined by comparison with this one.
```

7 Aby nebyla situace tak jednoduchá, existují zde hlavní módy (Major Modes) a vedlejší módy (Minor Modes). Zatím však pro nás nejsou podstatné.

8 Autor editoru Emacs je Richard Stallman.

Právě uvedenou informaci jsem získal takto: zadal jsem příkaz C+X B (což znamená vyvolání funkce `switch-to-buffer`) a zadal jsem „foo“ jako jméno bufferu, do kterého se chci přepnout. Protože buffer s takovým jménem nebyl doposud vytvořen, editor jej vytvořil a přepnul se do něj. Implicitně v něm nastavil základní mód (`fundamental-mode`). Všechny názvy módů mají tvar `<modename>-mode` a pomocí funkce „M-X“ lze pro každý buffer specifikovat mód explicitně právě pomocí jeho názvu. Abych získal více informací o základním módu, zadal jsem příkaz C-H M. Nakonec se zobrazila výše uvedená informace o základním módu.

Od základního módu je odvozen mírně užitečnější mód `text-mode` (textový mód), který má dva speciální příkazy: M-S pro funkci `center-paragraph` a M-S pro funkci center-line. Příkaz M-S znamená, že máte stisknout klávesu Shift, držet ji stisknutou, a pak stisknout klávesu Shift a „S“.

Nyní si můžete vyzkoušet vytvořit nový buffer a definovat v něm textový mód. Pak stiskněte klíč C-H M. Objeví se vám informace o textovém módu. Možná nebudete všemu rozumět, ale jistě zde najdete užitečné informace.

V dalších oddílech si probereme některé z nejpoužívanějších módů. Budete-li s nimi pracovat, nezapomeňte na klíč C-H M, kdykoliv budete chtít znát o aktuálním módu nějaké podrobnosti.

Programovací módy

Mód pro jazyk C

Jestliže použijete editor Emacs pro psaní programů v jazyku C, můžete po něm chtít, aby prováděl automatické odsazování. Soubory s příponou „.c“ nebo „.h“ zavádí editor Emacs automaticky v módu „c-mode“. To znamená, že jsou k dispozici některé speciální funkce vhodné pro psaní programů v jazyku C. Klávesa Tab je v módu `c-mode` svázána s funkcí `c-indent-command`. To znamená, že se po stisknutí klávesy Tab nevloží do textu znak Tab, ale dojde k automatickému odsazení řádku podle kontextu ve vytvářeném programu. Z toho vyplývá, že editor Emacs má jisté znalosti o syntaxi programů napsaných v jazyku C. V žádném případě však nepočítejte s tím, že vás bude editor upozorňovat na chyby v programu!

Navíc editor předpokládá, že jsou předcházející řádky odsazeny správně. Pokud v předcházejícím řádku chybí závorka, středník, složená závorka či cokoliv jiného, pak editor provede odsazení chybně. To je pro vás signál, že jste na něco zapomněli a můžete předcházející řádek opravit.

Uvedenou vlastnost editoru Emacs můžete využít ke kontrole oddělovačů ve zdrojovém programu. Nemusíte celý program číst od začátku a pracně hledat chybu, ale stačí zahájit odsazování řádků od začátku souboru pomocí klávesy Tab. Když dojde k nesprávnému odsazení, zkontrolujte předcházející řádek. Jinými slovy, v tomto směru za vás editor Emacs může udělat spoustu práce.

Mód pro jazyk Scheme

Tento mód pro vás bude užitečný jen tehdy, když používáte programovací jazyk Scheme. Tento jazyk není tak běžně používaným jazykem jako jazyk C nebo Pascal, ale v poslední době jeho popularita vzrůstá, a proto se mu budeme také věnovat. Většina toho, co platí pro jazyk Scheme, platí i pro jazyk Lisp.

Aby to nebylo tak jednoduché, v editoru Emacs jsou definovány dva módy pro programovací jazyk Scheme a každý uživatel se může rozhodnout, který mu bude lépe vyhovovat. Zaměříme se na popis módu s názvem `cmuscheme` a později, v oddíle o konfiguraci editoru Emacs, si vysvětlíme rozdíly mezi oběma módy. Nebuďte při čtení následujících řádků zneklidněni, když se váš edi-

tor Emacs bude chovat trochu jinak, než zde bude popisováno. V editoru lze konfigurovat prakticky vše a různé distribuce operačního systému Linux obsahují různě konfigurované editory Emacs.

V editoru Emacs můžete inicializovat interaktivní proces Scheme pomocí příkazu `M-X run-scheme`. Takto vytvoříte buffer nazvaný „`*scheme*`“, ve kterém se nachází obvyklý příkazový řádek jazyka Scheme. V tomto řádku můžete zadávat výrazy v jazyku Scheme ukončené klávesou Enter, jazyk Scheme je bude vyhodnocovat a bude zobrazovat příslušné odpovědi. Tímto způsobem můžete, máte-li interaktivně komunikovat s procesem Scheme, zadat definice všech svých funkcí a aplikací v příkazové řádku. Jiná situace nastane v případě, že máte zdrojový kód zapsaný v nějakém samostatném souboru. Pak by mohlo být jednodušší editovat tento soubor a definice odeslat prostřednictvím bufferu Scheme.

Jestliže do editoru Emacs zavedete soubor s příponou „`.ss`“ nebo „`.scm`“, pak se automaticky nastartuje mód **Scheme mode**. Pokud se z nějakých důvodů tento mód nenastartuje, můžete jej aktivovat prostřednictvím příkazu `M-X scheme-mode`. Tento mód ovšem není totožný s bufferem, ve kterém běží proces Scheme. V módu scheme-mode však máte k dispozici některé příkazy pro komunikaci s uvedeným bufferem.

Jestliže ve zdrojovém kódu jazyka Scheme máte vytvořenu definici nějaké funkce, pak ji stisknutím kláves C-C C-E můžete odeslat do bufferu, ve kterém běží proces Scheme. Pokud stisknete klávesy C-C M-E, pak se po odeslání definice funkce dostanete přímo do uvedeného bufferu a zde můžete zadávat interaktivní příkazy. Klávesy C-C C-L jsou určeny k zavedení souboru s kódem v jazyku Scheme (fungují pro buffer s procesem Scheme i pro buffer se zdrojovým kódem). Stejně jako u ostatních programovacích jazyků je klávesa Tab určena k odsazování řádků.

Pokud pracujete v bufferu, ve kterém běží proces Scheme, můžete používat klávesy M-P a M-N k zobrazení předcházejících nebo následujících příkazů (tzv. **input history**).

Předpokládejme, že ladíte nějakou funkci, řekněme '`rotate`', pro kterou jste použili argumenty, například:

```
> (rotate '(a b c d e))
```

pak můžete tento příkaz znovu zavést do příkazového řádku pomocí M-P. Nemusíte znovu v příkazovém řádku Scheme zadávat dlouhou sekvenci znaků a ušetříte tak spoustu času.

Editor Emacs má speciální módy jen pro několik málo programovacích jazyků. Patří mezi ně jazyk C, C++, Lisp a Scheme. (V současné době je dostupný mód snad pro každý programovací jazyk – např. Java, Python nebo JavaScript.)

Mód pro elektronickou poštu

Prostřednictvím editoru Emacs můžete také vytvářet a odesílat zprávy pro elektronickou poštu. K tomu je určen speciální buffer, tzv. „mail buffer“, který se inicializuje prostřednictvím klávesy C-X M. Nejdříve vyplníte položky „`To:`“ a „`Subject:`“ a pak se pomocí klíče C-N přenesete přes oddělovací čáru do pole, ve kterém můžete napsat zprávu. Oddělovací čáru nikdy needitujte ani nerušte, protože pak by editor Emacs nebyl schopen elektronickou zprávu odeslat. Oddělovací čáru používá k odlišení záhlaví od textu zprávy.

V poli pro text zprávy (pod oddělovací čarou) můžete uvést cokoliv. Po dokončení zprávy ji odešlete pomocí kláves C-C C-C. Editor Emacs obsah bufferu odešle.

Jak zvýšit efektivitu práce s editorem Emacs

Zkušení uživatelé editoru Emacs jsou až fanatičtí, pokud jde o zvyšování efektivity práce s editorem. Ve skutečnosti někdy stráví více času zvyšováním efektivity, než kolik pak ušetří. I když nechci, abyste se stali takovými fanatiky, existuje několik opatření, pomocí kterých se vám bude s editorem Emacs pracovat snadněji. Někteří zkušení uživatelé pohlížejí na začátečníky jako na hlupáky, protože neznají všechny „triky“ s editorem. Já osobně tento druh elitářství odsuzuji, protože jsem také kdysi byl začátečníkem. Nyní se však opět věnujme editoru.

Pro pohyb kurzoru existují některé další klávesy. Víme, že pro pohyb kurzoru o jeden znak doprava je určen klíč C-F. Chcete-li „poskočit“ s kurzorem o celé slovo, zadejte M-F. Vyzkoušejte si, co udělá příkaz M-B. To ale není vše. Pokud zapisujete věty tak, aby za poslední tečkou byly vždy dvě mezery, můžete pomocí klávesy M-E přeskakovat celé věty. Dvě mezery jsou podmínkou, protože jinak by editor nebyl schopen konec věty identifikovat. Opět si vyzkoušejte, jak funguje klíč M-A.

Možná si to ani neuvědomujete a používáte opakovaně klíč C-F k posunu kurzoru na konec řádku. Připomínáme, že k tomuto účelu je určen klíč C-E a k nastavení kurzoru na začátek řádku je určen klíč C-A. Možná, že často používáte klíč pro posun kurzoru na následující řádek, a přitom byste měli použít klíč C-V, kdy se vám zobrazí následující stránka. Zrovna tak využívejte klíč M-V.

Pokud jste někde u konce řádku a všimnete si, že jste někde na začátku udělali chybu, pak nepoužívejte klávesy Backspace nebo Delete, abyste se dostali k chybnému místu. Museli byste jinak dobře napsaný řádek psát znovu. Místo toho používejte klíčů M-B, C-B, případně C-F, opravte chybu a pak se pomocí C-E vraťte na konec řádku.

Pokud zadáváte jméno souboru, nezadávejte je nikdy celé. Zadejte jen nezbytný počet znaků, které soubor jednoznačně identifikují. Pak vám editor Emacs zbývající znaky po stisknutí klávesy Tab nebo mezerníku automaticky doplní. Šetřte sebe a ne procesor!

Když píšete nějaký dlouhý text, používejte funkci pro automatické ukončování řádků. Klíč M-Q vyvolá funkci `fill-paragraph` a lze jej použít ve všech textových módech. Také jej můžete použít k „zarovnání“ již napsaného odstavce. Stačí nastavit kurzor na nějakou pozici uvnitř odstavce a stisknout M-Q.

Někdy je užitečné použít klíč C-X U, kdy se editor pokusí vrátit zpět provedené změny. Editor Emacs v tomto případě odhadne, kolik změn má vrátit, a jeho odhad je obvykle velmi inteligentní. Klíč C-X U můžete použít opakovaně až do okamžiku, kdy editor vrátí poslední změnu, kterou si „pamatuje“.

Konfigurace editoru Emacs

Editor Emacs je tak velký a tak komplexní program, že má i svůj vlastní programovací jazyk. Vlastnosti editoru můžete modifikovat pomocí vlastních programů. Programovací jazyk integrovaný v editoru Emacs se jmenuje Emacs Lisp a je dialektem jazyka Lisp. Pokud máte s jazykem Lisp nějaké zkušenosti, pak se vám bude zdát opravdu přátelským. Pokud nemáte, ničeho se neobávejte. V tomto oddíle se nebudeme programováním v editoru Emacs zabývat příliš do hloubky a pokud se s jazykem Emacs Lisp budete chtít seznámit podrobněji, pak si prostudujte stránky dostupné v systému Info.

Většina funkcí editoru Emacs je definována pomocí kódu napsaného v jazyku Emacs Lisp.[9] Většina z těchto souborů je distribuována spolu s editorem a kolektivně se nazývají „Emacs Lisp library". Umístění této knihovny závisí na tom, jakým způsobem je editor Emacs instalován ve vašem systému. Nejčastěji jej můžete najít v adresáři: `/usr/lib/emacs/lisp`, `/usr/lib/emacs/19.19/lisp` a podobně. Číslo `19.19` značí verzi editoru Emacs a může se lišit od verze ve vašem systému.

Informace o uložení knihovny je uložena v interní proměnné **load-path** editoru Emacs, proto ji nemusíte pracně hledat v souborovém systému. Jestliže chcete zjistit hodnotu této proměnné, musíte ji **vyhodnotit**. To znamená, že musíte aktivovat interpretr Emacs Lisp. V editoru Emacs existuje speciální mód pro vyhodnocování výrazů jazyka Lisp zvaný **lisp-interaction-mode**. S tímto módem je obvykle spjat buffer „`*scratch*`". Pokud se vám jej nepodaří nalézt, vytvořte nový buffer s jakýmkoliv jménem a zadejte příkaz M-X lisp-interaction-mode.

Nyní se nacházíte v pracovním prostoru pro interaktivní komunikaci s interpretrem Emacs Lisp. Zadejte příkaz:

```
load-path
```

a pak stiskněte C-J. V módu interaktivní komunikace je klíč C-J svázán s funkcí `eval-print-last-sexp`. Slovo „sexp" znamená „s-expression" (**s-výraz**), což znamená vyváženou skupinu závorek – to je opravdu řečeno velmi zjednodušeně, ale brzy budete tušit, k čemu jsou takové výrazy užitečné při práci z jazykem Emacs Lisp. V každém případě po vyhodnocení proměnné `load-path` obdržíte zprávu, která bude vypadat přibližně takto:

```
load-path Ctrl+j
("/usr/lib/emacs/site-lisp/vm.5.35" "/home/kfogel/elithp"
"/usr/lib/emacs/site-lisp" "/usr/lib/emacs/19.19/lisp")
```

Toto hlášení nebude vypadat v každém systému stejně, protože závisí na způsobu instalace editoru Emacs. Uvedený příklad pochází z mého počítače s procesorem 386, na němž běží operační systém Linux. Jak vyplývá z předcházejícího výpisu, proměnná `load-path` je seznam řetězců. Každý řetězec uvádí adresář, který by mohl obsahovat soubory náležící do systému Emacs. Když potřebuje editor Emacs zavést soubory obsahující kód Lisp, hledá tyto soubory v uvedených adresářích v uvedeném pořadí. Pokud je v proměnné `load-path` uveden adresář, který v souborovém systému neexistuje, editor jej ignoruje.

Ve fázi startování se editor Emacs pokouší nalézt soubor .emacs ve vašem domovském adresáři. Proto platí, že pokud chcete mít svoji vlastní konfiguraci editoru Emacs, měli byste používat soubor `.emacs`. Nejvýznamnější konfigurační nastavení se týkají vazeb mezi klíči a funkcemi, proto se jim nyní budeme věnovat.

```
(global-set-key "\Ctrl+cl" 'goto-line)
```

Funkce `global-set-key` má dva argumenty: prvním z nich je klíč a druhým je funkce, která se má po stisknutí tohoto klíče realizovat. Slovo „`global`" znamená, že uvedená vazba mezi klíčem a funkcí bude platit ve všech hlavních módech. Existuje jiná funkce, `local-set-key`, jež nastavuje vazbu mezi klíčem a funkcí pro jediný buffer. V uvedeném příkladu jsme nastavili vazbu mezi klíčem C-C J a funkcí `goto-line`. Při definici klíče se musí použít řetězec, což znamená, že se klíč musí uzavřít do uvozovek. Speciální syntaxe „`\Ctrl+<char>`" znamená, že se má stisknout klávesa Ctrl, držet stisknutá, a pak se má stisknout klávesa `<char>`. Podobně platí, že „`\Alt+<char>`" indikuje kombinaci s klávesou Meta.

9 Někdy se neoficiálně nazývá „Elisp".

Konfigurace vazby mezi klíčem a funkcí vypadá jednoduše, ale kde získat informace o funkci „`go-to-line`“? Nebo naopak, předpokládejme, že chci klíč Ⓒ-Ⓒ Ⓛ svázat s funkcí, která umožní přeskočit na řádek specifikovaného pořadového čísla, ale jak mám zjistit jméno takové funkce?

V tomto okamžiku využijete vynikajících vlastností systému nápovědy, který je integrován v editoru Emacs. Jakmile se jednou rozhodnete, jaký druh funkce chcete použít, můžete použít editor Emacs k „vystopování“ jejího jména. Jedna sice rychlá, ale ne příliš přímočará metoda spočívá v tom, že se využije schopnosti editoru Emacs doplňovat jména funkcí. Měli byste si pamatovat, že klíč Ⓒ-Ⓗ Ⓕ vyvolá nápovědu popisující funkci (t.j. vyvolá funkci `describe-function`). Pak stiskněte Tab, aniž cokoliv zadáte. Tak „donutíte“ editor Emacs doplnit prázdný řetězec. Jinými slovy, editor vypíše jména všech funkcí, které jsou v něm interně definovány. Uvedená operace bude chvíli trvat, protože takových funkcí je v editoru definováno opravdu mnoho.

Nyní stiskněte klíč Ⓒ-Ⓖ, čímž funkci `describe-function` přerušíte. Nyní v editoru existuje buffer nazvaný „`*Completions*`“, který obsahuje požadovaný seznam všech interních funkcí editoru Emacs. Přepněte se do tohoto bufferu a pomocí postupného vyhledávání najděte funkci, kterou chcete použít. Můžete například využít předpokladu, že funkce pro přechod na řádek specifikovaného čísla bude ve svém názvu obsahovat slovo „`line`“. Proto zadejte vyhledávání slova „`line`“ a najdete tak všechny eventuality.

Vhodnější metoda spočívá v tom, že použijete klíče Ⓒ-Ⓗ Ⓐ (t.j. vyvoláte funkci `command-apropos`), kdy se vám zobrazí všechny funkce obsahující specifikovaný řetězec. Výstup z funkce `command-apropos` se poněkud hůře třídí, než výstup z funkce `describe-function`. Vyzkoušejte si obě metody a vyberte tu, která vám bude lépe vyhovovat.

Samozřejmě se může stát, že budete hledat funkci, která v editoru Emacs neexistuje. V takovém případě si ji budete muset napsat sami. Nebudeme zde popisovat, jak se v jazyku Emacs Lisp programuje. Doporučujeme prostudovat si příklady v knihovně Emacs Lisp a přečíst si stránky systému Info popisující jazyk Emacs Lisp. Pokud znáte někoho, kdo programování v jazyku Emacs Lisp ovládá, jistě vám pomůže.

Definice vlastních funkcí editoru Emacs není nic těžkého. Abych vás trochu navnadil, za poslední rok jsem jich bez problémů vytvořil 131. Vyžaduje to trochu zkušeností, ale programování v jazyku Emacs Lisp zvládnete poměrně rychle.

Další konfigurační možnosti spočívají v tom, že se v souboru `.emacs` nastaví hodnoty jistých proměnných. Přidejte například do vašeho souboru .emacs následující řádek a pak znovu nastartujte editor Emacs:

```
(setq inhibit-startup-message t)
```

Editor Emacs kontroluje ve fázi startování proměnnou inhibit-startup-message a podle její hodnoty se rozhodne, zda zobrazovat jisté informace (o verzi editoru, zárukách a podobně) nebo ne. Uvedený výraz jazyka Lisp používá příkaz `setq` k nastavení proměnné `inhibit-startup-message` na hodnotu „t“ což je speciální hodnota v jazyce Lisp pro „true“. Opakem je hodnota „nil“ což je speciální hodnota pro „false“. V mém konfiguračním souboru `.emacs` se nacházejí některá nastavení, jež možná shledáte užitečnými:

```
(setq case-fold-search nil) ; gives case insensitivity in searching
;; make C programs ident the way I like them to:
(setq c-indent-level 2)
```

První výraz nastavuje způsob vyhledávání na tzv. case-insensitive, což znamená, že se při vyhledávání nerozlišují malá a velká písmena (a to dokonce i tehdy, když hledaný řetězec obsahuje buď pouze malá, nebo pouze velká písmena). V druhém výrazu se nastavuje odsazování řádků při psaní programů v jazyku C na hodnotu 2. Je to poněkud méně, než je implicitní nastavení, ale to záleží na individuálním vkusu a na tom, jaká hodnota odsazení učiní váš program napsaný v jazyku C čitelnějším.

V jazyku Lisp se komentářový řádek označuje znakem „;“. Editor Emacs ignoruje vše, co se nachází za tímto znakem. Výjimku tvoří případ, kdy se znak „;“ nachází uvnitř řetězce. Například:

```
;; Tyto dva řádky jsou v jazyku Lisp ignorovány, ale
;; následující s-výraz bude plně vyhodnocen:
(setq some-literal-string "An awkward pause; for no purpose.")
```

Doporučuje se, abyste změny ve zdrojových souborech pro jazyk Lisp komentovali, protože jinak například po šesti měsících určitě zapomenete, jakou změnu jste dělali. Komentář na celý řádek uvádějte dvojicí znaků `;;`. Pak bude editor Emacs správně provádět odsazování řádků.

To, co jsme si řekli o vyhledávání interních funkcí editoru Emacs, platí i pro vyhledávání proměnných. Chcete-li vytvořit seznam všech proměnných, zadejte příkaz Ⓒ-Ⓗ v (describe-variable), nebo použijte Ⓒ-Ⓗ Ⓒ-Ⓐ (apropos). Použijete-li druhou možnost, pak musíte počítat s tím, že vám příkaz vyhledá funkce a proměnné dohromady.

Soubory se zdrojovým kódem v jazyku Emacs Lisp mají implicitní příponu „`.el`“, například „`c-mode.el`“. Aby však mohl kód vytvořený v jazyku Emacs Lisp běžet rychleji, umožňuje editor provést jakousi **předkompilaci** (soubory označované jako „**byte-compiled**“) a pak mají tyto soubory implicitní příponu „`.elc`“. Výjimku, pokud jde o implicitní příponu, tvoří soubor `.emacs`, jenž příponu „`.el`“ nepotřebuje, neboť jej editor hledá automaticky ve fázi startování.

Chcete-li zavést interaktivně soubor s kódem v jazyku Lisp, použijte příkaz `M+X load-file`. Editor vás vyzve k zadání jména souboru a pak specifikovaný soubor zavede do své pracovní oblasti. Jestliže chcete zavést soubor „zevnitř“ jiného souboru napsaného v jazyku Lisp, postupujte takto:

```
(load "c-mode") ; tento příkaz zavede buď soubor c-mode.el, nebo c.mode.elc
```

Editor Emacs nejdříve k uvedenému jménu přidá příponu `.elc` a pokusí se jej nalézt v některém adresáři specifikovaném v proměnné `load-path`. Pokud jej nenajde, přidá příponu `.el` a hledání zopakuje. Jestliže není úspěšný ani v tomto případě, pak použije přímo řetězec předaný funkci `load`. Překlad můžete realizovat prostřednictvím příkazu `Alt+X byte-compile-file`. Pokud však zdrojový soubor často aktualizujete, pak to zpravidla nemá smysl. Soubor .emacs nikdy nepřekládejte, ani mu nepřidávejte příponu `.el`.

Bezprostředně po zavedení souboru `.emacs` vyhledá editor Emacs soubor `default.el` a zavede jej. Tento soubor je obvykle umístěn v adresáři uvedeném v proměnné `load-path` s názvem `site-lisp` nebo `local-elisp` či v nějakém podobném adresáři (podívejte se na proměnnou `load-path`). Uživatelé, kteří provádějí údržbu editoru Emacs používají soubor `default.el` k nastavení globálních konfigurací, které pak platí pro každého uživatele v systému. Soubor `default.el` by také neměl být kompilován, protože se v něm často provádějí změny.

Jestliže váš osobní soubor `.emacs` obsahuje nějakou chybu, pak se editor Emacs nebude pokoušet načíst soubor `default.el`, ale zastaví svou činnost a vypíše hlášení: „`Error in init file`“. Uvidíte-li takové hlášení, pak jste pravděpodobně udělali nějakou chybu v souboru `.emacs`.

V souboru `.emacs` se ještě vyskytuje jeden druh výrazů. Knihovna Emacs Lisp někdy nabízí více sad programů, které realizují tytéž funkce různým způsobem. To znamená, že musíte specifikovat tu sadu, která se má používat (implicitní sada nemusí být pro vás vždy tou nejlepší). Tyto sady se například uplatňují v oblasti interaktivního rozhraní pro jazyk Scheme. S editorem Emacs se standardně distribuují dvě sady funkcí interaktivního rozhraní pro jazyk Scheme: `xscheme` a `cmuscheme`.

```
prompt> /usr/lib/emacs/19.19/lisp/*scheme*
/usr/lib/emacs/19.19/lisp/cmuscheme.el
/usr/lib/emacs/19.19/lisp/cmuscheme.elc
/usr/lib/emacs/19.19/lisp/scheme.el
/usr/lib/emacs/19.19/lisp/scheme.elc
/usr/lib/emacs/19.19/lisp/xscheme.el
/usr/lib/emacs/19.19/lisp/xscheme.elc
```

Já osobně dávám přednost sadě cmuscheme před sadou `xscheme`, avšak editor Emacs implicitně používá sadu `xscheme`. Jak „donutit" editor Emacs, aby byl nakonfigurován v souladu s mým přáním? Do souboru `.emacs` jsem vložil následující řádky:

```
;; notice how the expression can be broken across two lines. Lisp
;; ignores whitespaces, generally:
(autoload 'run-scheme "cmuscheme"
"Run an inferior Scheme, the way I like it". t)
```

Funkce `autoload` akceptuje jako argument jméno funkce (začínající apostrofem ') a „sdělí" editoru Emacs, že je tato funkce definována v jistém souboru. Jméno tohoto souboru se předává jako druhý argument, tedy řetězec (buď s příponou „`.el`", nebo „`.elc`"). Soubor pak bude vyhledán v adresářích definovaných v proměnné `load-path`.

Zbývající argumenty jsou volitelné, ale jsou nezbytné v tomto případě: Třetí argument je dokumentačním řetězcem pro specifikovanou funkci. Pokud použijete funkci `describe-function`, pak se jako popis k vyhledané funkci objeví právě tento řetězec.

Čtvrtý argument „sdělí" editoru Emacs, že specifikovaná funkce může být volána interaktivně, t.j. pomocí klíče Alt+X. V tomto případě je to velmi důležité, protože jedině tak lze spustit proces Scheme v prostředí editoru Emacs pomocí příkazu `Alt+x run-scheme`.

Nyní, když je funkce `run-scheme` definována jako automaticky zaveditelná, co se stane, když se zadá příkaz `Alt+x run-scheme`? Editor Emacs vyhledá funkci `run-scheme`, zjistí, zda je automaticky zaveditelná a zavede specifikovaný soubor (v našem případě „`cmuscheme`"). Protože kompilovaný soubor `cmuscheme.elc` existuje, editor jej zavede. Tento soubor musí definovat funkci `run-scheme`, jinak dojde při jeho zavádění k chybě. Naštěstí tuto funkci skutečně definuje, proto jde vše hladce a já mám k dispozici své oblíbené rozhraní pro jazyk Scheme.[10]

Automatické zavedení funkce zajišťuje, že bude editor Emacs schopen funkci najít, až ji budete potřebovat. Navíc můžete nad automaticky zaváděnými funkcemi získat jistou kontrolu. Dále platí, že automaticky zaveditelné funkce šetří paměť spotřebovanou editorem Emacs, protože se tyto funkce zavádějí až v okamžiku, kdy jsou potřebné. Řada příkazů není ve fázi startování editoru ve skutečnosti definována. Místo toho jsou nastaveny jako automaticky zaveditelné funkce. Pokud takový příkaz nezadáte, nikdy se odpovídající funkce nezavede. Uvedená vlastnost automatického zavádění funkcí je pro editor Emacs (a hlavně pro uživatele) „životně" důležitou. Kdyby editor ve

10 Jistě jste si všimli, že o rozhraní cmuscheme jsme hovořili již dříve. Proto byste si měli zavedení sady cmuscheme vyzkoušet.

fázi startování zaváděl všechny funkce, trvala by tato fáze asi dvacet minut a celá dostupná paměť vašeho počítače by byla obsazena. O automatické zavádění implicitních funkcí se nemusíte starat, editor Emacs je realizuje sám.

Kde získat další informace

V této kapitole jsme zdaleka neuvedli vše, co byste měli znát o editoru Emacs. Je zde zhruba jedno procento informací. Budete-li editor intenzivně využívat, pak budete potřebovat znát spoustu dalších „triků“ šetřících čas. Nejjednodušší bude, když vyčkáte, až budete muset řešit nějaký konkrétní problém. Pak vyhledáte funkci, která váš problém vyřeší.

Snad nejdůležitější je, abyste uměli plně využívat systém nápovědy integrovaný v editoru Emacs. Řekněme například, že budete chtít vložit do editovaného textu obsah jiného souboru. Snadno uhodnete, že asi existuje funkce `insert-file` (vložení souboru). Máte-li svou domněnku potvrdit, použijte příkaz Ⓒ-Ⓗ Ⓕ. V příkazovém řádku zadejte jméno funkce, kterou hledáte a o které si chcete přečíst nějaké informace. Protože víte, že editor Emacs je schopen doplňovat jména funkcí, můžete jako „počáteční“ odhad uvést řetězec „insert“. Pak stačí stisknout klávesu Tab. Objeví se vám seznam všech funkcí obsahujících řetězec „insert“ a funkce „`insert-file`“ je jedna z nich.

Nyní si můžete přečíst spoustu informací o funkci `insert-file` a také ji můžete použít – stačí zadat příkaz `Alt+x insert-file`. Chcete-li také zjistit, zda je tato funkce svázána s nějakým klíčem, zadejte příkaz `Ctrl+h w insert-file` a stiskněte Enter. Čím lépe budete znát vlastnosti systému nápovědy v editoru Emacs, tím snáze naleznete potřebné informace. Máte-li navíc dostatek erudice zkoumat nové věci a ochotu učit se, ušetříte mnoho času.

Jestliže si chcete objednat kopii manuálu k editoru Emacs a/nebo manuál „*Emacs Lisp Programming*“, pošlete objednávku na adresu:

Free Software Foundation
675 Mass Ave
Cambridge, MA 02139
USA

Oba tyto manuály jsou distribuovány v elektronické podobě spolu s editorem Emacs a jsou čitelné prostřednictvím systému Info (použijte klávesu Ⓒ-Ⓗ Ⓘ, pomocí které inicializujete rozhraní mezi editorem Emacs a systémem Info). Na druhé straně, cena za uvedené manuály je opravdu mírná a navíc se získané peníze budou věnovat na vývoj kvalitního volně šiřitelného programového vybavení. Také chceme poznamenat, že pomocí klíčů Ⓒ-Ⓗ Ⓒ-Ⓒ si můžete zobrazit informace o licenčních podmínkách platných pro editor Emacs. Jsou určitě zajímavější, než si teď myslíte, a pomohou vám vyjasnit si pojem „volné programové vybavení“. Pokud si myslíte, že pojem „volné programové vybavení“ znamená, že daný program nic nestojí, pak si licenční podmínky rychle přečtěte.

KAPITOLA 9

Konfigurace operačního systému Unix

Konfigurace příkazového interpretu bash

Filosofie operačního systému Unix se od filosofie jiných operačních systémů podstatně liší v jedné věci. Autoři Unixu se nepokoušeli předvídat všechny potřeby všech uživatelů. Místo toho se pokusili navrhnout operační systém tak, aby si každý uživatel pracovní prostředí svého operačního systému snadno upravil sám podle svých potřeb. Konfigurace jednotlivých programů operačního systému Unix se definuje prostřednictvím tzv. **konfiguračních souborů**. Někdy jsou označovány jako „ini-files“ nebo „rc files“ nebo dokonce jako „dot files“ (jedná se zpravidla o soubory, jejichž jména začínají tečkou). Pokud si vzpomínáte, tak jména souborů začínající znakem „.“ nejsou normálně příkazem `ls` zobrazována.

Nejdůležitější konfigurační soubory jsou ty, které používá příkazový interpret. Implicitním příkazovým procesorem v operačním systému Linux je `bash` a právě jemu bude věnována tato kapitola. Než začneme popisovat, jak příkazový interpret `bash` konfigurovat, podívejme se na soubory, které `bash` vyhledává.

Inicializace příkazového interpretu bash

Existuje několik různých způsobů, jak příkazový procesor `bash` spustit. Tzv. **login-shell** se automaticky spouští poté, co se přihlásíte do systému.

Další způsob spočívá ve spuštění **interaktivního příkazového procesoru** (interactive shell). To je jakýkoliv příkazový procesor, který se prezentuje příkazovým řádkem. Příkazový interpret, jenž se spustí bezprostředně po přihlášení se do systému, je také interaktivní. Jiným příkladem interaktivního příkazového interpretu je program xterm spuštěný z prostředí X Window.

Existují také **neinteraktivní příkazové interprety.** Tyto procesory se používají k vykonávání příkazů uvedených v souboru, jako jsou dávkové soubory s příponou .BAT v operačním systému MS-DOS. Analogií v operačním systému Unix jsou tzv. **skripty**. Skript příkazového procesoru je něco jako miniprogram. Jsou sice výrazně pomalejší než kompilovaný program, ale snadno se vytvářejí a modifikují.

Příkazové procesory v operačním systému Unix používají v závislosti na typu následující inicializační soubory:

Typ příkazového procesoru	inicializační soubor
Interaktivní příkazový interpreter spuštěný při přihlášení se do systému	`.bash_profile`
Interaktivní příkazový procesor	`.bashrc`
Neinteraktivní příkazový procesor	skript příkazového procesoru

Inicializační soubory

Protože většina uživatelů chce mít stejné uživatelské prostředí bez ohledu na to, jaký typ příkazového procesoru se spustí, začneme konfiguraci tím, že do konfiguračního souboru `.bash_profile` vložíme jednoduchý příkaz „source ~/.bashrc“. Příkaz source „sdělí“ příkazovému procesoru, že má jeho argument interpretovat jako skript. To znamená, že se při každém spuštění skriptu `.bash_profile` také spustí skript `.bashrc`.

Nyní budeme přidávat příkazy do našeho souboru `.bashrc`. Do souboru `.bash_profile` se zadávají pouze příkazy, které se mají spouštět při přihlášení se do systému.

Vytváření druhých jmen

Jakým způsobem lze měnit konfigurační nastavení? Hned uvedeme příklad, který si do svého konfiguračního souboru `.bashrc` zadává devadesát procent uživatelů operačního systému Unix:

```
alias ll="ls -l"
```

Příkaz definuje tzv. **alias** (druhé jméno) příkazu. V našem případě se jméno příkazu `ll` rozšiřuje na příkaz příkazového procesoru `"ls -l"`. Za předpokladu, že příkazový procesor bash přečetl uvedenou definici ve vašem konfiguračním souboru `.bashrc`, pak má příkaz `ll` stejný efekt jako příkaz `ls -l`. Přitom ušetříte polovinu stisknutých kláves. Když zadáte v příkazovém řádku `ll` a stisknete Enter, příkazový procesor zadaný příkaz rozvine podle definice a pak realizuje příkaz `ls -l`. Ve skutečnosti v systému příkaz `ll` neexistuje, ale příkazový procesor `bash` jej automaticky transformuje na platný program.

Některé příklady druhých jmen jsou uvedeny na této straně dole. Můžete si je vložit do vašeho souboru `.bashrc`. Zvláště zajímavý je první z nich. Je-li definován alias `ls="ls -F"`, pak po každém zadání příkazu `ls` bude automaticky aplikovat volbu `-F`. Platí, že se alias nikdy nepokusí rekurzivně rozšířit sám sebe. Uvedený příklad demonstruje nejčastěji používaný způsob automatického přidávání voleb do příkazů.

Všimněte si znaku „#“. Ve skriptech příkazového procesoru se tento znak používá k označení komentáře a příkazový procesor zbytek řádku za znakem # ignoruje.

Dále jste si mohli všimnout několika dalších věcí. Především se v některých definicích nevyskytují uvozovky, například v definici `pu`. Platí totiž, že pokud se na pravé straně znaku rovná se (=) vyskytuje jediné slovo, pak se uvozovky nemusejí uvádět.

Nic by se nestalo, kdyby zde uvozovky byly uvedeny. Určitě však uvozovky používejte v případě, když budete definovat nové jméno pro příkaz s volbami a/nebo argumenty. Například:

```
alias rf="refrobnicate -verbose -prolix -wordy -o foo.out"
alias ls="ls -F"              # give characters at end of listing
alias ll="ls -l"              # special ls
alias la="ls -a"
alias ro="rm  *~; rm.*~"      # removes backup files created by Emacs
alias rd="rmdir"              # saves typing!
alias md="mkdir"
alias pu=pushd         # pushd, popd, and dirs weren't covered
alias po=popd          # manual---you might want to look them up
alias ds=dirs          # in the bash manpage
# these all are just keyboard shortcuts
alias to="telnet cs.oberlin.edu"
alias ta="telnet altair.mcs.anl.gov"
alias tg="telnet wombat.gnu.ai.mit.edu"
alias tko="tpalk kold@cs.oberlin.edu"
alias tjo="talk jimb@cs.oberlin.edu"
alias mroe="more"             # spelling correction!
alias moer="more"
alias email="emacs -f rmail"  # my mail reader
alias ed2="emacs -d floss:0 -fg \"grey95\" -bg \"grey50\""
                                    # one way of invoking emacs
```

Asi se vám zdá, že poslední alias má divně umístěné uvozovky:

```
alias ed2="emacs -d floss:0 -fg \"grey95\" -bg \"grey50\""
```

Jak asi tušíte, chtěl jsem umístit uvozovky do samotných voleb. Proto jsem musel před každou uvozovku vložit obrácené lomítko. Příkazový procesor pak takovou uvozovku nebude interpretovat obvyklým způsobem.

Všimněte si také dvou definic pro opravu chybně zadaného příkazu `more`. Pokud omylem zadám „`mroe`“ nebo „`moer`“, bude příkazový procesor „vědět“, že jsem chtěl zadat „`more`“. Aliasy neinterferují s argumenty předávanými programům. To znamená, že například příkaz

```
/home/larry# mroe hurd.txt
```

bude fungovat dobře.

Určitě platí, že správné používání definice druhých jmen představuje polovinu konfiguračních možností, které jsou u příkazových procesorů k dispozici. Při práci v operačním systému Unix si všímejte, které příkazy často zadáváte, a pak si pro ně pořiďte druhá jména, tedy zkratky. Zjistíte, že se vám pak bude pracovat mnohem radostněji.

Systémové proměnné

V souboru `.bashrc` se definují další důležitá konfigurační nastavení prostřednictvím systémových proměnných (environment variables). Co jsou to systémové proměnné? Pojďme na to z druhé strany: předpokládejme, že čtete dokumentaci k programu `fruggle` a že narazíte na následující věty:

> *Fruggle normally looks for its configuration file, .frugglerc, in the user's home directory. However, if the environment variable FRUGGLEPATH is set to a different filename, it will look there instead.*

Každý program běží v nějakém **prostředí** a to je definováno příkazovým interpretem, jenž program volá.[1] Lze si představit, že prostředí existuje uvnitř příkazového interpretu. Programátoři mají k dispozici speciální funkci pro získání hodnoty systémové proměnné a program `fruggle` tuto funkci využívá. To znamená, že ověří hodnotu v systémové proměnné `FRUGGLEPATH`. Pokud není tato systémová proměnná definována, pak program použije soubor `.frugglerc` ve vašem domovském adresáři. Pokud však je definována, program `fruggle` použije její hodnotu místo implicitního souboru `.frugglerc`.

Nyní si uveďme ukázku, jak změnit prostředí v příkazovém procesoru `bash`:

```
/home/larry# export PGPPATH=/home/larry/secrets/pgp
```

Pod příkazem `export` si můžete představit následující větu: „Exportuj tuto proměnnou do prostředí, ze kterého budu spouštět program, tak, aby proměnná byla z tohoto programu viditelná.“ Později uvidíte, že jsou i jiné důvody pro používání příkazu `export`.

Uvedenou systémovou proměnnou používá program `pgp` pro šifrování, jehož autorem je Phil Zimmerman. Program `pgp` implicitně používá váš domovský adresář jako adresář, ve kterém hledá jisté soubory (šifrovací klíče). Také tento adresář využívá k vytvoření dočasných pracovních souborů. Nastavením systémové proměnné `PGPPATH` jsem změnil pracovní adresář na `/home/larry/secrets/pgp`. Abych zjistil přesné jméno této systémové proměnné, musel jsem si přečíst manuál k programu `pgp`. Systémové proměnné se zpravidla uvádějí velkými písmeny a končí na „`PATH`“.

Je užitečné vědět, jak hodnotu systémové proměnné zjistit:

```
/home/larry# echo $PGPPATH
/home/larry# .pgp
/home/larry#
```

Všimněte si, že jsme před jméno systémové proměnné uvedli znak `$`. Jedině tak lze zjistit hodnotu systémové proměnné. Pokud byste znak dolaru neuvedli, dostali byste následující výpis:

```
/home/larry# echo PGPPATH
PGPPATH
/home/larry#
```

Znak dolaru se používá k vyhodnocení systémové proměnné, avšak pouze v kontextu s příkazovým procesorem – přesněji s příkazovým procesorem, jenž realizuje interpretaci příkazu. V jakých případech realizuje příkazový procesor interpretaci?

Proměnná	Obsahuje	Příklad
`HOME`	**Váš domovský adresář**	`/home/larry`
`TERM`	**Typ vašeho terminálu**	`xterm, vt100, console`
`SHELL`	**Cesta k vašemu příkazovému procesoru**	`/bin/bash`
`USER`	**Jméno vašeho účtu**	`larry`
`PATH`	**Seznam adresářů, ve kterých se automaticky vyhledávají programy ke spuštění**	`/bin:/usr/local/bin:/usr/bin/X11`

Tabulka 9.1 – Některé důležité systémové proměnné

1 Nyní vidíte, proč jsou příkazové procesory tak důležité. Představte si, že byste museli definovat celé prostředí, kdykoliv budete spouštět nějaký program!

Je to v těch případech, kdy zadáváte příkaz z příkazového řádku nebo kdy příkazový procesor čte příkazy z nějakého souboru, například ze souboru `.bashrc`.

Existuje další důležitý příkaz, pomocí kterého lze získat informace o prostředí. Tímto příkazem je `env`. Po zadání příkazu `env` se vám zobrazí seznam všech systémových proměnných. Jste-li uživateli systému X Window, pak bude tento seznam velmi dlouhý, proto používejte příkaz `env | more`.

Některé systémové proměnné jsou opravdu důležité, proto jsou uvedeny v tabulce 9.1. Tyto systémové proměnné se automaticky definují po přihlášení se do systému. Není proto nutné je nastavovat v souboru `.bashrc` nebo `.bash_login`.

Nyní se blíže podívejme na systémovou proměnnou `TERM`. Abychom jí mohli porozumět, podívejme se zpět do historie operačního systému Unix. Operační systém musí znát jisté údaje o vaší konzole, aby mohl realizovat takové funkce, jako je zápis znaků na obrazovku, pohyb kurzoru v textovém řádku a podobně. V počátcích rozvoje výpočetní techniky výrobci terminálů průběžně rozšiřovali jejich vlastnosti: nejdříve inverzní video, později znaky pro evropské jazyky, dokonce i první funkce pro kreslení (připomínáme, že jde o dobu, kdyse ještě nikomu ani nesnilo o grafických oknech a myších). Každá nová vlastnost však přinášela programátorům problémy: jak měli vědět, co terminál podporuje a co ne? Jak měli podporovat nové vlastnosti a přitom „neodepsat" staré terminály?

V operačním systému Unix najdete odpověď na tyto otázky v souboru `/etc/termcap`. Soubor `/etc/termcap` obsahuje seznam všech terminálů, o kterých váš operační systém „ví," a dále informace, jakým způsobem se má řídit kurzor. Pokud systémový správce dostane nový terminál, pak pouze do souboru `/etc/termcap` přidá záznam vztahující se k novému terminálu a nemusí pracně konfigurovat celý operační systém Unix. Někdy je situace ještě jednodušší. Kdysi se stal terminál vt100 od firmy Digital Equipment Corporation jakýmsi pseudostandardem, který převážná většina moderních terminálů respektuje a umí emulovat.

V operačním systému Linux je někdy hodnota systémové proměnné `TERM` nastavena jako console, což znamená emulaci terminálu vt100 s některými speciálními funkcemi.

Další proměnná, `PATH`, je rovněž kritickou systémovou proměnnou z hlediska funkčnosti příkazového procesoru. Zde uvádím nastavení na mém počítači:

```
/home/larry# env | grep ^PATH
PATH=/home/larry/bin:/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/TeX/bin
/home/larry#
```

Systémová proměnná `PATH` obsahuje seznam adresářů oddělených dvojtečkou, ve kterých systém automaticky vyhledává spustitelné programy. Když například zadám příkaz `ls` a stisknu klávesu Enter, bude příkazový procesor `bash` hledat program `ls` nejdříve v adresáři `/home/larry/bin`, který jsem vytvořil pro ukládání mých vlastních programů.

Program `ls` jsem však nenapsal já (ten byl pravděpodobně napsán ještě před tím, než jsem se narodil), proto zde příkazový procesor tento příkaz nenašel. Jako další prohledává příkazový procesor adresář /bin. A zde program `ls` našel. Protože soubor `ls` je spustitelný program, přestane příkazový procesor dále hledat a spustí jej. Může se stát, že v jiném adresáři bude také uložen spustitelný program `ls` (například v adresáři `/usr/bin`), ale příkazový procesor jej nespustí, pokud mu to výslovně nepřikážete:

```
/home/larry# /usr/bin/ls
```

Systémová proměnná PATH existuje hlavně proto, abychom nemuseli při každém spouštění nějakého programu zadávat celou cestu k tomuto programu. Zadáte-li tedy jakýkoliv příkaz, prohledá příkazový procesor všechny adresáře uvedené v systémové proměnné PATH. Když jej najde, spustí jej, a pokud ne, vypíše následující zprávu:

```
/home/larry# clubly
clubly: command not found
```

Všimněte si, že moje systémová proměnná PATH neobsahuje aktuální adresář, tedy „.“. Pokud by obsahovala, pak by vypadala takto:

```
/home/larry# echo $PATH
.:/home/larry/bin:/bin:/usr/bin:/usr/local/bin:/usr/bin/X11:/usr/TeX/bin
/home/larry#
```

Zda zadávat nebo nezadávat aktuální adresář do systémové proměnné PATH je předmětem diskuse v kuloárech okolo operačního systému Unix. Problém tkví v tom, že aktuální adresář v systémové proměnné PATH by mohl představovat „díru“ v bezpečnosti systému. Předpokládejme, že se přepnete do adresáře, ve kterém někdo nechal virový program „Trójský kůň“ a nazval jej ls. Vy nic zlého netušíte a zadáte příkaz ls, což je přirozené, chcete-li se seznámit s novým adresářem. Protože je aktuální adresář uveden v systémové proměnné PATH jako první, spustí příkazový procesor právě tuto verzi příkazu ls. I když jste nechtěli udělat nic špatného, rozpoutali jste ve vašem systému virovou nákazu. Proti takovým haváriím neexistuje dost účinná ochrana. Virový program může spustit osoba, která ani nemá privilegia uživatele root. Stačí, aby měla právo zapisovat do adresáře, ve kterém se virový program nachází. Dokonce to může být její domovský adresář.

Pokud jde o váš systém, je pravděpodobné, že jeden uživatel neklade druhému různé nástrahy ve formě „Trojských koňů“ a že kolektiv uživatelů je založen na přátelských a kolegiálních vztazích. Avšak ve velkých systémech s mnoha uživateli (jako jsou například universitní počítače), mohou být desítky programátorů, které byste nejraději ani nepotkali. Z toho vyplývá, že zařazení aktuálního adresáře do systémové proměnné PATH závisí na konkrétní situaci.[2]

Skutečný způsob, kterým je nastavena systémová proměnná PATH v mém počítači, je dostatečně ilustrativní. Zde je uvedeno nastavení v souboru .bashrc:

```
export PATH=${PATH}:.:${HOME}/bin:/bin:/usr/bin:/usr/local/bin:/usr
  /bin/X1:/usr/TeX/bin
```

Zde jsem využil skutečnosti, že proměnná HOME je nastavena před tím, než příkazový procesor bash přečte můj soubor .bashrc. Systémová proměnná PATH je tedy nastavena prostřednictvím systémové proměnné HOME. Složené závorky („{...}“) představují další úroveň uvozovek. Oddělují to, co se vyhodnotí po znaku $, proto bude příkazový procesor moci jednoznačně identifikovat následující část příkazu (tedy „/bin“ v tomto případě). Zde uvádíme další příklad efektu, který vyvolají složené závorky:

```
/home/larry# echo ${HOME}foo
/home/larryfoo
/home/larry#
```

2 Pamatujte si, že program z aktuálního adresáře můžete vždy spustit explicitně, tedy například „./foo“.

Bez uvedení složených závorek se nevypíše nic, protože neexistuje proměnná prostředí `HOMEfoo`:

```
/home/larry# echo $HOMEfoo
/home/larry#
```

Nyní se podívejme na další zvláštní konstrukci v uvedeném příkazu. Jaký je význam řetězce „$PATH"? Tento řetězec zařadí do proměnné prostředí `PATH` hodnotu proměnné prostředí `PATH` dříve definovanou. Kde byla nastavena původní hodnota? Soubor `/etc/profile` slouží jako jistý typ globálního souboru `.bash_profile`, kde se nacházejí nastavení společná pro všechny uživatele systému. Jeden soubor s globálním nastavením má jistou výhodu. Má-li například systémový správce přidat do proměnné prostředí `PATH` důležitou cestu, stačí, když to udělá v souboru s globální platností a nemusí opravovat inicializační soubory všech uživatelů. Proto je proměnná prostředí `PATH` již nastavena a vy ji můžete použít.

Prostřednictvím jisté proměnné prostředí můžete také specifikovat, jak má vypadat váš příkazový řádek. Tato proměnná prostředí má označení jako `PS1`. Předpokládejme, že dáváte přednost tomu, aby se stále zobrazovala cesta do aktuálního adresáře. Pak nastavte proměnnou prostředí `PS1` takto:

```
export PS1='$PWD#'
```

Jak asi tušíte, jsou zde ve skutečnosti použity dvě proměnné prostředí. První, která se nastavuje, je `PS1` a druhou je `PWD`. Proměnná prostředí `PWD` může znamenat buď „Print Working Directory", nebo „`PATH` to Working Directory". Vyhodnocení proměnné prostředí však probíhá uvnitř jednoduchých uvozovek. Jednoduché uvozovky slouží k vyhodnocení vnitřního výrazu. Výsledkem tohoto vyhodnocení je proměnná prostředí `PWD`. Kdybyste použili výraz `export PS1=$PWD`, pak by se vám stále zobrazoval ten adresář, který byl aktuální v době vyhodnocení tohoto výrazu. Trochu jsme pohled na vyhodnocování proměnných prostředí zkomplikovali, ale to není tak důležité. Pouze si pamatujte, že budete-li chtít zobrazovat v příkazovém řádku aktuální adresář, budete muset použít jednoduché uvozovky.

Možná, že budete chtít podobný příkazový řádek jako v operačním systému MS-DOS. Pak zadejte `export PS1='$PWD>'`. Chcete-li mít v příkazovém řádku jméno vašeho systému, zadejte `PS1='hostname''>'`.

V posledním příkladu jsme použili nový typ uvozovek, tzv. zpětné uvozovky. Tyto uvozovky ve skutečnosti nic nechrání. Výraz uzavřený ve zpětných uvozovkách se vyhodnotí jako příkaz a výstup se objeví na místě zpětných uvozovek.

Vyzkoušejte si příkazy `echo` `` `ls` `` nebo `wc` `` `ls` ``. Čím více budete seznámeni s příkazovým procesorem, tím užitečnější vám budou uvedené techniky.

V souboru `.bashrc` je mnoho dalších možností, jak konfigurovat váš příkazový procesor, ale zde není dostatek prostoru všechny prodiskutovat. Další podrobnosti si nastudujte v manuálových stránkách k příkazovému procesoru `bash` nebo se zeptejte zkušených uživatelů. Dále uvádíme kompletní soubor `.bashrc`, abyste si jej mohli nastudovat. Představuje typický standard, i když je proměnná prostředí `PATH` poněkud dlouhá.

```
# some random stuff:
ulimit -c unlimited
export history_control=ignoredups
export PS1='$PWD>'
umask 022
# application-specific PATHs:
```

```
export MANPATH=/usr/local/man:/usr/man
export INFOPATH=/usr/local/info
export PGPPATH=${HOME}/.pgp
# make the main PATH:
homepath=${HOME}:~/bin
stdpath=/bin:/usr/bin:/usr/local/bin:/usr/ucb:/etc:/usr/etc:/urs
  /games
pubpath=/usr/public/bin:/usr/gnusoft/bin:/usr/local/contribs/bin
softpath=/usr/bin/X11:/usr/local/bin/X11:/usr/TeX/bin
export PATH=.:${homepath}:${stdpath}:${pubpath}:${softpath}
# Technically, the curly braces were not necessary, because the
  colons
# were valid delimiters; nevertheless, the curly braces are a good
# habit to get into, and they can't hurt.
# aliases
alias ls="ls -CF"
alias fg1="fg %1"
alias fg2="fg %2"
alias tba="talk sussman@tern.mcs.anl.gov"
alias tko="take kold@cs.oberlin.edu"
alias tji="talk jimb@totoro.bio.indiana.edu"
alias mroe="more"
alias moer="more"
alias ll="ls -l"
alias la="ls -a"
alias ro="rm *~; rm.*~"
alias rd="rmdir"
alias pu=pushd
alias po=popd
alias ds=dirs
alias to="telnet cs.oberlin.edu"
alias ta="telnet altair.mcs.anl.gov"
alias tg="telnet wobat.gnu.ai.mit.edu"
alias email="emacs -f rmail"
alias ed2="emacs -d floss:0 -fg \"grey95\" -bg \"grey50\""
function gco
{
gcc -o $1 $1.c -g
}
```

Inicializační soubory systému X Window

Většina lidí dává při své práci přednost grafickému uživatelskému prostředí, kterým je v opeačním systému Unix systém X Window. Jestliže jste seznámeni s operačním systémem Macintosh nebo Microsoft Windows, pak vám systém X Window bude připadat velmi známý. Méně známé vám však budou připadat konfigurační možnosti, které systém X Window nabízí.

V případě operačního systému Macintosh nebo Microsoft Windows se konfigurace realizuje přímo v grafickém uživatelském prostředí. Když chcete například změnit barvu pozadí, klepnete myší na novou barvu nějakého speciálního inicializačního grafického programu. V systému X Window se implicitní hodnoty řídí textovými soubory, které můžete přímo editovat – jinými slovy, chcete-li například zadat novou barvu pozadí, musíte její jméno uvést v jistém inicializačním souboru.

Nikdo nepopírá, že inicializační metody v systému X Window jsou poněkud těžkopádnější než u komerčních programů. Domnívám se, že tendence zachovávat textově orientované inicializační metody dokonce i v grafickém uživatelském prostředí tkví v tom, že například systém X Window vytvořila poměrně nesourodá skupina programátorů, kteří až příliš lpí na tradicích dodržovaných v operačních systémech typu Unix. Tyto tendence se mohou v příštích verzích systému X Window změnit (alespoň doufám, že se změní), ale nyní se budeme zabývat inicializací prostřednictvím textových souborů. Tak alespoň máte k dispozici velmi flexibilní a přesnou kontrolu nad konfigurací.

Nejdůležitější konfigurační soubory pro systém X Window jsou tyto:

`.xinitrc` Skript, který systém X Window spouští ve fázi startování.

`.twmrc` Soubor, který čte správce oken `twm`.

`.fvwmrc` Soubor, který čte správce oken `fvwm`.

Všechny uvedené soubory by měly být uloženy ve vašem domovském adresáři.

Soubor `.xinitrc` je jednoduchý skript příkazového procesoru, jenž se automaticky spouští ve fázi startování systému X Window. Může dělat vše, co mohou dělat ostatní skripty, avšak nejvíce ze všeho má smysl jej použít k nastartování různých programů systému X Window a k nastavení parametrů. Posledním příkazem v souboru .xinitrc je obvykle příkaz pro spuštění správce oken, například `/usr/bin/X11/twm`.

Jaký druh konfiguračních nastavení má smysl v souboru `.xinitrc` uvádět? Snad nějaká volání programu `xsetroot`, čímž si můžete nastavit pozadí okna a vlastnosti kurzoru. Dále volání programu `xmodmap`, jenž předá serveru[3] informace o tom, jak má interpretovat signály z vaší klávesnice. Všechny ostatní programy, které se mají spustit pokaždé spolu se systémem X Window (například `xclock`).

Zde uvádím některé řádky z mého souboru `.xinitrc`. Váš konfigurační soubor bude jistě vypadat jinak, proto je považujte za pouhý příklad.

```
#!/bin/sh
# The first line tells the operating system which shell to use in
# interpreting this script. The script itself ought to be marked as
# executable; you can make it so with "chmod +x ~/.xinitrc".
# xmodmap is a program for telling the X server how to interpret your
# keyboard's signals. It is *definitely* worth learning out. You
# can do "man xmodmap", "xmodmap -help", "xmodmap -grammar", and more.
# I don't guarantee that the expressions below will mean anything on
# your system (I don't even guarantee that they mean anything on
# mine):
xmodmap -e 'clear Lock'
xmodmap -e 'keycode 176 = Control_R'
xmodmap -e 'add control = Control_R'
xmodmap -e 'clear Mod2'
xmodmap -e 'add Mod1 = Alt_L Alt_R'
# xset is a program for setting other parameters of the X server:
xset m 3 2 & # mouse parameters
xset s 600 5 & # screen saver prefs
xset s noblank # ditto
xset fp+ /home/larry/x/fonts # for cxterm
```

3 Server představuje hlavní proces systému X Window, tedy program, se kterým musejí všechny ostatní programy běžící pod systémem X Window komunikovat, pokud chtějí využívat grafické prostředí. Tyto ostatní programy se nazývají klienti a systém jako celek bývá označován termínem systém „klient-server“.

```
# To find out more, do "xset -help"
# Tell the X server to superimpose fish.cursor over fish.mask, and use
# the resulting pattern as my mouse cursor:
xsetroot -cursor /home/lab/larry/x/fish.cursor /home/lab/larry/x/fish.mask &
# a pleasing background pattern and color:
xsetroot -bitmap /home/lab/larry/x/pyramid.xbm -bg tan
# todo: xrdb here? What about .Xdefaults file?
# You should do "man xsetroot", or "xsetroot -help" for
# more information on the program above.
# A client program, the imposing circular color-clock by Jim Blandy:
/usr/local/bin/circles
# Maybe you'd like to have a clock on your screen at all time:
/usr/bin/X11/xclock -digital &
# Allow client program running at occs.cs.oberlin.edu to display
# themselves here, do the same thing for juju.mcs.anl.gov:
xhost occs.cs.oberlin.edu
xhost juju.mcs.anl.gov
# You can simply tell the X server to allow clients running on any
# other host (a host being a remote machine) to display here, but this
# is a security hole -- those clients can be run by someone else,
# and watch your keystrokes as you type your password or something!
# However, if you wanted to do it anyway, you could use a "+" to stand
# for all possible hostnames, instead of a specific hostname, like
# this:
# xhost +
# And finally, run the window namager:
/usr/bin/X11/twm
# Some people prefer other window manager. I use twm, but fvwm is
# often distributed with Linux too:
# /usr/bin/X11/fvwm
```

Všimněte si, že některé programy běží na pozadí. Jsou to ty programy, jejichž zadání je ukončeno znakem `&`. Jiné programy se na pozadí nespouštějí. Rozdíl spočívá v tom, že se startují současně se systémem X Window a běží na pozadí, dokud systém X Window neukončíte. Jiné se vykonají bezprostředně a ihned skončí – jedním z nich je `xsetroot`, který pouze nastaví základní okno a chování kurzoru a pak skončí.

Jakmile se nastartuje správce oken, načte svůj vlastní inicializační soubor. Tento inicializační soubor řídí takové věci, jako je nastavení nabídek, pozice oken, řízení ikon a další. Pokud používáte jako správce oken program `twm`, pak tímto inicializačním souborem je soubor `.twmrc`, který se nachází ve vašem domovském adresáři. Jestliže však používáte `fvwm`, pak je inicializačním souborem soubor `.fvwmrc`. V následujících oddílech se budeme zabývat pouze těmito dvěma, protože se běžně distribuují s operačním systémem Linux.

Konfigurace programu twm

Soubor `.twmrc` není skript příkazového procesoru – je napsán v jazyku speciálně vyvinutém pro program `twm`.[4] Při konfiguraci prostřednictvím souboru `.twmrc` si uživatelé nejraději hrají s nastavením oken (barvy a podobně) a nabídek. Zde uvádíme příklad konfiguračního souboru `.twmrc`:

4 Toto je jedna z odpuzujících vlastností inicializačních souborů: některé z nich mají svůj vlastní příkazový jazyk. To znamená, že uživatel musí být velmi zdatný a rychle se naučit další jazyk. Předpokládám, že takové vymyšlenosti mohly být zajímavé v ranných dobách operačního systému Unix, kdy programátoři stále toužili po něčem novém. Dnes je ale vyčerpávající učit se stále dokola nové a nové syntaxe jazyků, které konec konců slouží jedinému programu.

```
# Set colors for various parts of windows. This has a great
# impact no the "feel" of your environment.
Color
{
BorderColor "OrangeRed"
BorderTitleForeground "Black"
BorderTitleBackground "Black"
TitleForeground "black"
TitleBackground "gold"
MenuForeground "black"
MenuBackground "LightGrey"
MenuTitleForeground "LightGrey"
MenuTitleBackground "LightSlateGrey"
MenuShadowColor "black"
IconForeground "DimGray"
IconBackground "Gold"
IconBorderColor "OrangeRed"
IconManagerForeground "black"
IconManagerBacground "honeydew"
}
# I hope you don't have a monochrome system, but if you do...
Monochrome
{
BorderColor "black"
BorderTitleForeground "black"
BorderTitleBackground "white"
TitleForeground "black"
TitleBackground "white"
}
# I created beifang.bmp with the program "bitmap". Here I tell twm to
# use it as the default highlight pattern on windows' title bars:
Pixmaps
{
TitleHighlight "/home/larry/x/beifang.bmp"
}
# Don't worry about this stuff, it's only for power users :-)
BorderWidth 2
TitleFont "-adobe-new century schoolbook-bold-r-normal--14-140-75-75
  -p-87-iso8859-1"
MenuFont "6x13"
IconFont "lucidasans-italic-14"
ResizeFont "fixed"
Zoom 50
RandomPlacement
# These programs will not get a window titlebar by default:
NoTitle
{
"stamp"
"xload"
"xclock"
"xlogo"
"xbiff"
"xeyes"
```

```
"oclock"
"xoid"
}
# "AutoRaise" means that a window is brought to the front whenever the
# mouse pointer enters it. I find this annoying, so I have turned
# off. As you can see, I inherited my .twmrc from people who also
# did not like autoraise.
AutoRaise
{
"nothing" # I don't like auto-raise # Me either # nor I
}
# Here is where the mouse button functions are defined. Notice the
# pattern: a mouse button pressed on the root window, with no modifier
# key being pressed, always brings up a menu. Other locations usually
# result in window manipulation of some kind, and modifier keys are
# used in conjuction with mouse buttons to get at the more
# sophisticated window manipulations.
#
# You don't have to follow this pattern in your own .twmrc -- it's
# entirely up to you how you arrange your environment.
# Button = KEYS : CONTEXT : FUNCTION
# ----------------------------------
Button1 = : root : f.menu "main"
Button1 = : title : f.raise
Button1 = : frame : f.raise
Button1 = : icon : f.iconify
Button1 = m : window : f.iconify
Button2 = : root : f.menu "stuff"
Button2 = : icon : f.move
Button2 = m : window : f.move
Button2 = : title : f.move
Button2 = : frame : f.move
Button2 = s : frame : f.zoom
Button2 = s : window : f.zoom
Button3 = : root : f.menu "z"
Button3 = : title : f.lower
Button3 = : frame : f.lower
Button3 = : icon : f.raiselower
# You can write your own functions; this one gets used in the menu
# "windowops" near the end of this file:
Function "raise-n-focus"
{
f.raise
f.focus
}
# Okay, below are the actual menus reffered to in the mouse button
# section. Note that many of these menu entries themselves call
# sub-menus. You can have as many levels of menus as you want, but be
# aware that recursive menus don't work. I tried it.
menu "main"
{
"Vanilla" f.title
"Emacs" f.menu "emacs"
```

```
"Logins" f.menu "logins"
"Xlock" f.menu "xlock"
"Misc" f.menu "misc"
}
# This allows me to invoke emacs on several different machines. See
# the section on .rhosts files for more information about how this
# works:
{
"Emacs" f.title
"here" !"/usr/bin/emacs &"
"" f.nop
"phylo" !"rsh phylo \"emacs -d floss:0\" &"
"geta" !"rsh geta \"emacs -d floss:0\" &"
"darwin" !"rsh darwin \"emacs -d floss:0\" &"
"ninja" !"rsh ninja \"emacs -d floss:0\" &"
"indy" !"rsh indy \"emacs -d floss:0\" &"
"oberlin" !"rsh cs.oberlin.edu \"emacs -d floss.life.uiuc.edu:0\" &"
"gnu" !"rsh gate-1.gnu.ai.mit.edu \"emacs -d floss.life.uiuc.edu:0\" &"
}
# This allows me to invoke xterms on several different machines. See
# the section on .rhosts files for more information about how this
# work:
menu "logins"
{
"Logins" f.title
"here" !"/usr/bin/X11/xterm -ls -T `hostname` -n `hostname` &"
"phylo" !"rsh phylo \"xterm -ls -display floss:0 -T phylo\" &"
"geta" !"rsh geta \"xterm -ls -display floss:0 -T geta\" &"
"darwin" !"rsh darwin \"xterm -ls -display floss:0 -T darwin\" &"
"ninja" !"rsh ninja \"xterm -ls -display floss:0 -T ninja\" &"
"indy" !"rsh indy \"xterm -ls -display floss:0 -T indy\" &"
}
# The xlock screensaver, called with various options (each of which
# gives a different pretty picture):
menu "xlock"
{
"Help" !"xlock -mode hop &"
"Qix" !"xlock -mode qix &"
"Flame" !"xlock -mode flame &"
"Worm" !"xlock -mode worm &"
"Swarm" !"xlock -mode swarm &"
"Hop NL" !"xlock -mode hop -nolock &"
"Qix NL" !"xlock -mode qix -nolock &"
"Flame NL" !"xlock -mode flame -nolock &"
"Worm NL" !"xlock -mode worm -nolock &"
"Swarm NL" !"xlock -mode swarm -nolock &"
}
# Miscellaneous programs I run occassionally:
menu "misc"
{
"Xload" !"/usr/bin/X11/xload &"
"XV" !"/usr/bin/X11/xv &"
"Bitmap" !"/usr/bin/X11/bitmap &"
```

```
"Tetris" !"/usr/bin/X11/xtetris &"
"Hextris" !"/usr/bin/X11/xhextris &"
"XRoach" !"/usr/bin/X11/xroach &"
"Analog Clock" !"/usr/bin/X11/xclock -analog &"
"Digital Clock" !"/usr/bin/X11/xclock -digital &"
}
# This is the one I bound to the middle mouse button:
menu "stuff"
{
"Chores" f.title
"Sync" !"/bin/sync"
"Who" !"who | xmessage -file - -columns 80 -lines 24 &"
"Xhost +" !"/usr/bin/X11/xhost + &"
"Rootclear" !"/home/larry/bin/rootclear &"
}
# X functions that are sometimes convenient
menu "x"
{
"X Stuff" f.title
"Xhost +" !"xhost + &"
"Refresh" f.refresh
"Source .twmrc" f.twmrc
"(De)Iconify" f.iconify
"Move Window" f.move
"Resize Window" f.resize
"Destroy Window" f.destroy
"Window Ops" f.menu "windowops"
"" f.nop
"Kill twm" f.quit
}
# This is submenu from above:
menu "windowops"
{
"Window Ops" f.title
"Show Icon Mgr" f.showiconmgr
"Hide Icon Mgr" f.hideiconmgr
"Refresh" f.refresh
"Refresh Window" f.winrefresh
"twm version" f.version
"Focus on Root" f.unfocus
"Source .twmrc" f.twmrc
"Cut File" f.cutfile
"(De)Iconify" f.iconify
"DeIconify" f.deiconify
"Move Window" f.move
"ForceMove Window" f.forcemove
"Resize Window" f.resize
"Raise Window" f.raise
"Lower Window" f.lower
"Raise or Lower" f.raiselower
"Focus on Window" f.focus
"Raise-n-Focus" f.function "raise-n-focus"
"Destroy Window" f.destroy
"Kill twm" f.quit
}
```

Věřte mi, že to není zdaleka nejobsáhlejší soubor `.twmrc`, jaký jsem kdy viděl. Je vysoce pravděpodobné, že nějaký příklad souboru .twmrc bude obsažen ve vaší distribuci systému X Window. Prohledejte adresář `/usr/lib/X11/twm/` nebo `/usr/X11/lib/X11/twm`. Zde byste měli uvedený soubor nalézt.

Často dělají uživatelé chybu a zapomínají uvádět znak `&` na konec příkazů. Pokud systém X Window „zatuhne“ právě když spustíte nějaký příkaz, pak pravděpodobně tkví příčina v tom, že jste zapomněli uvést znak `&`. V takovém případě ukončete systém X Window kombinací kláves Ctrl-Alt-Backspace opravte soubor `.twmrc` a spusťte systém X Window znovu.

Konfigurace programu fvwm

Pokud používáte jako správce oken program `fvwm`, prohledejte tento adresář:

```
/usr/lib/X11/fvwm/ nebo /usr/X11/lib/X11/fvwm.
```

Zde najdete nějaké příklady konfiguračních souborů.

Poznámka: O programu `fvwm` nic nevím. Asi bych byl schopen něco vyčíst z příkladů konfiguračních souborů, ale zůstal bych pouze čtenářem a těžko bych dokázal něco vysvětlovat. Zájemcům o program fvwm a jeho konfiguraci doporučuji přečíst si příslušné manuálové stránky a prostudovat příklady konfiguračních souborů nacházejících se ve výše zmíněných adresářích.

Ostatní inicializační soubory

Za zmínku stojí následující inicializační soubory:

`.emacs`	Inicializační soubor editoru Emacs. Editor jej čte ve fázi startování.
`.netrc`	Soubor obsahující implicitní jména a hesla pro `ftp`.
`.rhosts`	Zpřístupňuje váš účet vzdáleným systémům.
`.forward`	Inicializační soubor pro automatické přesměrování elektronické pošty.

Inicializační soubor pro editor Emacs

Pokud používáte editor Emacs jako primární editor, pak má pro vás soubor `.emacs` mimořádný význam. Podrobně jsme jej popsali v kapitole 8.

Implicitní nastavení pro FTP

V souboru `.netrc` můžete mít uložena některá implicitní nastavení pro `ftp`. Následující řádky obsahují příklad takového souboru:

```
machine floss.life.uiuc.edu login larry password fishSticks
machine darwin.life.uiuc.edu login larry password fishSticks
machine geta.life.uiuc.edu login larry password fishSticks
machine phylo.life.uiuc.edu login larry password fishSticks
machine ninja.life.uiuc.edu login larry password fishSticks
machine indy.life.uiuc.edu login larry password fishSticks


machine clone.mcs.anl.gov login fogel password doorm@
machine osprey.mcs.anl.gov login fogel password doorm@
machine tern.mcs.anl.gov login fogel password doorm@
machine altair.mcs.anl.gov login fogel password doorm@
machine dalek.mcs.anl.gov login fogel password doorm@
```

```
machine juju.mcs.anl.gov login fogel password doorm@
machine sunsite.unc.edu login anonymous password larry@cs.oberlin.edu
```

Každý řádek souboru `.netrc` specifikuje jméno počítače, přihlašovací jméno, které se má použít jako implicitní pro tento počítač, a heslo. Uvedená nastavení vám ušetří velké množství času, protože jinak byste při přihlašování se k jednotlivým serverům `ftp` museli pokaždé zadávat dlouhé identifikační řetězce. Pokud se budete přihlašovat k serveru ftp, jehož jméno je uvedeno v souboru `.netrc`, pokusí se program `ftp` aplikovat uživatelské jméno a heslo z tohoto souboru.

Při spouštění programu `ftp` můžete pomocí parametru -n specifikovat, že nechcete použít implicitní nastavení ze souboru .netrc. Příkaz pak bude mít tvar „`ftp -n`“.

Musíte se ujistit, že soubor `.netrc` jste schopni číst pouze vy. K nastavení příslušných přístupových práv použijte program chmod. Kdyby soubor .netrc mohli číst jiní uživatelé, pak by mohli odhalit vaše hesla platná pro servery ftp uvedené v tomto souboru.

Takový případ by představoval značnou „bezpečnostní díru“. Naštěstí program `ftp` a ostatní programy, které čtou soubor `.netrc`, odmítnou pokračovat ve své činnosti, pokud zjistí, že přístupová práva k tomuto souboru nejsou v pořádku.

Další informace týkající se souboru .netrc si najděte v manuálových stránkách prostřednictvím příkazu „`man .netrc`“ nebo „`man ftp`“.

Povolení snadného vzdáleného přístupu k vašemu účtu*

Jestliže se ve vašem domovském adresáři nachází soubor `.rhosts`, pak lze ze vzdálených počítačů spouštět aplikace na vašem počítači. Uveďme si příklad. Předpokládejme, že jste přihlášeni na počítači cs.oberlin.edu. Na počítači `floss.life.uiuc.edu` je správně konfigurován soubor `.rhosts`.

Pak ze svého počítače můžete na počítači `floss.life.uiuc.edu` spustit aplikaci, jejíž výstup bude přesměrován na vaši obrazovku, a dokonce se před tím nebudete muset přihlašovat a zadávat heslo.

Soubor `.rhosts` může vypadat takto:

```
frobnozz.cs.knowledge.edu jsmith
aphrodite.classics.havhaahd.edu wphilps
frobbo.hoola.com trixie
```

Formát souboru je velmi jednoduchý: jméno počítače následované jménem uživatele. Předpokládejme nyní, že uvedený příklad je mým skutečným souborem .rhosts na vzdáleném počítači `floss.life.uiuc.edu`. To by znamenalo, že bych mohl spustit program na počítači floss a výstup by mohl být přesměrován na kterýkoliv počítač uvedený v tomto souboru, pokud bych byl přihlášen jako odpovídající uživatel.

Přesný mechanismus, prostřednictvím kterého uživatel uskutečňuje spouštění vzdáleného programu, je realizován programem rsh, jehož název je zkratkou pro „remote shell“, tedy „vzdálený příkazový procesor“. Ten nastartuje příkazový procesor na vzdáleném počítači a spustí specifikovaný program. Uveďme si příklad:

```
frobbo$ whoami
trixie
frobbo$ rsh floss.life.uiuc.edu "ls ~"
```

* Poznámka korektora: Používání programu `rsh` a `rlogin` je v současné době nahrazeno podobným programem ssh, který je bezpečnější, ale nemůže být standardní součástí distribucí Linuxu.

```
foo.txt mbox url.ps snax.txt
frobbo$ rsh floss.life.uiuc.edu "more ~/snax.txt"
[nyní se zobrazí stránky souboru snax.txt]
```

Uživatel trixie na počítači floss.life.uiuc.edu, který má soubor `.rhosts` uvedený v předcházejícím příkladě, umožňuje uživateli trixie na počítači `frobbo.hoola.com` spouštět programy z počítače floss.

Soubor `.rhosts` bude pracovat správně, i když nemáte na všech počítačích stejné uživatelské jméno. Chcete-li „sdělit“ vzdálenému počítači, jaké jméno uživatele chcete použít k přihlášení se, použijte při zadání příkazu `rsh` volbu „`-l`“. Pokud takový uživatel na vzdáleném počítači existuje a pokud zde existuje soubor `.rhosts` obsahující jméno vašeho (t.j. lokálního) počítače a jméno uživatele, pak se příkaz `rsh` provede úspěšně.

```
frobbo$ whoami
trixie
frobbo$ rsh -l larry floss.life.uiuc.edu "ls ~"
[zde se objeví seznam souborů mého adresáře na počítači floss]
```

Uvedený příklad bude fungovat, pokud má uživatel `larry` na počítači `floss.life.uiuc.edu` takový soubor `.rhosts`, jenž umožňuje uživateli `trixie` z počítače `frobbo.hoola.com` spouštět programy. Není relevantní, zda se jedná nebo nejedná o tutéž osobu. Důležitá jsou pouze jména uživatelů, jména počítačů a záznamy v souboru `.rhosts`.

V souboru .rhosts mohou být uvedeny i jiné kombinace – například jméno uživatele následující za jménem vzdáleného počítače se může vynechat a tak umožnit kterémukoliv uživateli na vzdáleném počítači spouštět programy na vašem počítači. To je však spojeno s jistými riziky. Nepovolaná osoba by mohla například zrušit vaše soubory. Pokud se rozhodnete pro takto organizovaný soubor `.rhosts`, pak byste se měli ujistit, že právo číst soubor .rhosts máte pouze vy.

Přesměrování elektronické pošty

Kromě jiných souborů můžete také mít soubor `.forward`, který nelze přímo nazvat „inicializačním“ souborem. Pokud tento soubor obsahuje adresu elektronické pošty, pak veškeré zprávy elektronické pošty budou odesílány na tuto adresu. Soubor je užitečný v případě, kdy máte účet na několika různých systémech, ale elektronickou poštu chcete číst pouze na jednom.

Na vašem počítači se mohou nacházet i jiné inicializační soubory. Jejich počet se liší podle operačního systému, jenž používáte, a také podle programů, které máte nainstalovány. Jeden ze způsobů, jak získat více informací o inicializačních souborech, spočívá v tom, že si prostudujete soubory ve vašem domovském adresáři začínající tečkou. Není sice garantováno, že všechny takové soubory jsou inicializační, ale převážná většina z nich určitě bude.

Kde si můžete prohlédnout některé příklady

Jestliže na svém počítači máte instalován operační systém Linux a jestliže máte přístup do sítě Internet, pak se můžete přihlásit prostřednictvím služby telnet do systému `floss.life.uiuc.edu`. Přihlašte se jako „guest“ a heslo uveďte jako „explorer“. Připravili jsme pro vás spoustu příkladů, z nichž většina je uložena v adresáři `/home/kfogel`. Tyto příklady si můžete zkopírovat a prostudovat. Buďte ale opatrní. Počítač floss nepředstavuje zcela zabezpečený systém a když se budete dostatečně snažit, podaří se vám získat přístupová práva uživatele root. Dali jsme přednost důvěře před neustálou ostražitostí a doufáme, že toho nikdo nezneužije.*

* Poznámka korektora: Autor knihy to zřejmě myslel jako vtip.

KAPITOLA 10

Komunikace s ostatními systémy

Moderní operační systémy typu Unix jsou dobře přizpůsobeny ke komunikaci s ostatními počítači, což znamená, že jsou vybaveny prostředky pro práci v síti. Dva různé počítače s operačním systémem Unix si mohou vyměňovat informace mnoha způsoby. Tato kapitola je věnována metodám, pomocí kterých budete moci komunikovat prostřednictvím sítě s ostatními počítači.

Budeme se zabývat elektronickou poštou, zájmovými skupinami a některými základními programy pro komunikaci.

Elektronická pošta

Mezi nejpopulárnější vlastnosti operačního systému Unix patří možnost odesílat a přijímat zprávy elektronické pošty. Máte-li k dispozici elektronickou poštu, nepotřebujete papír, inkoust a pero, obálky, známky a nesrovnatelně pomalejší poštovní službu.

Odesílání elektronické pošty

Potřebujete-li odeslat zprávu elektronickou poštou, stačí zadat příkaz `mail username` a pak zapsat vaši zprávu.

Předpokládejme například, že chcete odeslat zprávu elektronickou poštou uživateli jménem sam:

```
/home/larry# mail sam
Subject: The user documentation
Just testin out the mail system.
EOT
/home/larry#
```

Program `mail` je velmi jednoduchý. Podobně jako program cat čte text ze standardního vstupu řádek po řádku, dokud nenarazí na znak „konec textu“ jenž je reprezentován klávesou Ⓒ-Ⓓ. To znamená, že po dokončení textu zprávy stisknete Enter a pak Ⓒ-Ⓓ.

Příkaz `mail` představuje nejrychlejší způsob, jak odeslat zprávu elektronické pošty, a je užitečný zejména ve spojení s prostředky pro přesměrování vstupu/výstupu či rourami. Jestliže chcete například odeslat soubor `report1` uživateli jménem „Sam“, můžete použít příkaz `mail sam < report1` nebo dokonce „`sort report1 | mail sam`“.

Používání příkazu `mail` je však spojeno s velkou nevýhodou. Jestliže uděláte v nějakém řádku chybu, pak ji již nemůžete opravit. Proto vám doporučuji (pokud nepoužijete možnost s přesměrováním vstupu/výstupu) k odesílání elektronické pošty používat editor Emacs. Postup jsme popsali v oddílu Pracovní módy editoru Emacs.

Čtení zpráv elektronické pošty

`mail [`*`user`*`]`

Program `mail` poskytuje poněkud těžkopádnou možnost číst zprávy elektronické pošty. Zadáte-li příkaz `mail` bez jakýchkoliv parametrů, objeví se vám následující hlášení:

```
/home/larry# mail
No mail for larry
/home/larry#
```

Předpokládejme, že chcete odeslat zprávu elektronickou poštou sami sobě, abyste si mohli vyzkoušet způsoby jejího čtení:

```
/home/larry# mail larry
Subject: Frogs!
and toads!
EOT
/home/larry# echo "snakes" | mail larry
/home/larry# mail
Mail version 5.5. 6/1/90 Type ? for help.
"/usr/spool/mail/larry" 2 messages new
>N 1 larry Tue Aug 30 18:11 10/211 "Frogs!"
N 2 larry Tue Aug 30 18:12 10/211
&
```

Příkazový řádek uvnitř programu pro čtení elektronické pošty je uvozen znakem ampersand („&"). Umožňuje specifikovat spoustu jednoduchých příkazů a po zadání znaku ? a enter zobrazuje stručnou nápovědu.

Základními příkazy pro práci s programem mail jsou:

`t` *`message-list`*	Na obrazovce se zobrazí zprávy uvedené v seznamu *`message-list`*.
`d` *`message-list`*	Zprávy uvedené v seznamu *`message-list`* se zruší.
`s` *`message-list file`*	Zprávy uvedené v seznamu *`message-list`* se uloží do souboru *`file`*.
`r` *`message-list`*	Odpověď na zprávy – program `mail` zahájí proces sestavování nových zpráv, které budou odeslány každému, kdo vám odeslal zprávu uvedenou v seznamu *`message-list`*.
`q`	Program `mail` se ukončí a uloží každou zprávu, která nebyla zrušena příkazem `d` do souboru mbox ve vašem domovském adresáři.

Jak vypadá seznam *`message-list`*? Skládá se ze seznamu čísel oddělených mezerami, nebo může být vyjádřen ve formě intervalu, tedy například 2-4 (což znamená „2 3 4"). Také můžete uvést uživatelské jméno odesílatele. Například `t sam` zobrazí všechny zprávy odeslané uživatelem sam. Jestliže se seznam *`message-list`* neuvede, zobrazí se vždy poslední zpráva.

Se čtením zpráv elektronické pošty je spojeno několik problémů. Především platí, že pokud má zpráva více řádků, než se vejde na obrazovku, program mail se nezastaví! Takou zprávu budete muset uložit do souboru a pak si ji prohlédnout prostřednictvím příkazu `more`. Dále program mail nemá dobré prostředky pro čtení starých zpráv, tedy zpráv uložených do souborů.

Editor Emacs má prostředek pro čtení zpráv elektronické pošty, jenž se jmenuje `rmail`. V této knize se však programem rmail nebudeme zabývat. V operačním systému Linux jsou navíc k dispozici další programy pro čtení elektronické pošty, jako je `elm` nebo .*

Jak vyhledat uživatele sítě

Příkaz finger

Příkaz `finger` vám umožní získat informace o ostatních uživatelích vašeho systému nebo o uživatelích sítě Internet. Jméno příkazu nepochybně vzniklo jako zkratka s reklamním podtextem ve firmě AT&T.

`finger [-slpm] [`*`user`*`][`*`@machine`*`]`

Volitelné parametry v příkazu `finger` mohou být mírně matoucí. Pomocí příkazu `finger` můžete získat informace o lokálním uživateli (například „sam“), o jiném počítači (například „@lionsden“), informace o uživateli vzdáleného počítače (například „sam@liondsen“) nebo informace o lokálním počítači (neuvede se žádný parametr).

Příkaz finger má další zajímavou vlastnost. Pokud se pokusíte získat informace o uživateli, jehož jméno přesně neznáte, pokusí se příkaz `finger` najít jméno sám (zkouší různé kombinace založené na původně zadaném jménu). To znamená, že když například zadám příkaz `finger Greenfield`, obdržím zprávu, že účet `sam` existuje pro jméno `Sam Greenfield`.

```
/home/larry# finger sam
Login: sam Name: Sam Greenfield
Directory: /home/sam Shell: /bin/tcsh
Last login Sun Dec 25 14:47 (EST) on tty2
No Plan.
/home/larry# finger greenfie@gauss.rutgers.edu
[gauss.rudgers.edu]
Login name: greenfie In real life: Greenfie
Directory: /gauss/u1/greefie Shell: /bin/tcsh
On since Dec 25 15:19:41 on ttyp0 from tiptop-slip-6439
13 minutes Idle Time
No unread mail
Project: You must be joking!
No Plan.
/home/larry# finger
Login Name Tty Idle Login Time Office Office Phone
larry Larry Greenfield 1 3:51 Dec 25 12:50
larry Larry Greenfield p0 Dec 25 12:51
/home/larry#
```

* Poznámka korektora: Samozřejmě můžete používat i jiné poštovní klienty, např. i Netscape Communicator.

Použijete-li u příkazu `finger` volbu `-s`, pak se vám vždy zobrazí stručný výpis (stejný, který obdržíte, když program `finger` použijete k získání informací o počítači) a pokud použijete volbu `-l`, zobrazí se vám vždy kompletní výpis (i tehdy, když program `finger` použijete k získání informací o počítači). Po zadání volby `-p` se nezobrazí informace ze souborů `.forward`, `.plan` a `.project`. Zadáte-li volbu `-m` a žádáte-li pouze informace o uživateli, pak se vám zobrazí pouze přihlašovací jméno.

Soubory .plan a .project

Nyní bychom měli vysvětlit, jaký je význam souborů `.plan` a `.project`. Jedná se o soubory, které jsou uloženy v domovském adresáři uživatele a jejichž obsah se zobrazí pokaždé, když je na daného uživatele aplikován program `finger`. Soubory `.plan` a `.project` si můžete vytvořit sami – jediné omezení spočívá v tom, že ze souboru .project se zobrazuje pouze první řádek.

Dále platí, že každý, kdo chce aplikovat program `finger` musí mít možnost procházet vaším domovským adresářem (`chmod a+x ~/`) a každý musí být schopen číst soubory `.plan` a `.project` (`chmod a+r ~/.plan ~/.project`).

Používání systémů vzdálenými počítači

`telnet` *`vzdálený systém`*

Hlavní prostředek pro využívání vzdálených systémů založených na operačním systému Unix představuje program Telnet. (V současné době se spíše používá program `ssh`, který na rozdíl od příkazu `telnet` umožňuje šifrovanou komunikaci se vzdáleným systémem.) Používání programu telnet je velmi jednoduché:

```
/home/larry# telnet lionsden
Trying 128.2.36.41...
Connected to lionsden
Escape character is '^]'.
lionsden login:
```

Jak vidíte z následujícího příkladu, po zadání příkazu `telnet` budete vyzváni k přihlášení se ke vzdálenému systému. Uživatelské jméno lze zadat jakékoliv (ovšem heslo musíte znát správné) a pak je vám vzdálený systém k dispozici téměř stejně jako váš lokální systém.

Normální způsob ukončení programu `Telnet` spočívá v odhlášení se, ale je také možnost zadat znak Escape, kterým je zpravidla Ⓒ-Ⓘ. Tak obdržíte nový příkazový řádek uvozený řetězcem `telnet>`. Nyní stačí napsat `quit` a pak stisknout klávesu Enter – spojení se přeruší a program `Telnet` se ukončí. Pokud si to rozmyslíte a nechcete relaci ukončit, stiskněte pouze klávesu Enter.

Pokud jste uživateli systému X Window, pak si pro komunikaci se vzdáleným uživatelem otevřete nové terminálové okno – například prostřednictvím příkazu „`xterm -title "lionsden" -e telnet lionsden &`“. Uvedený příkaz otevře nové terminálové okno, ve kterém automaticky poběží program `Telnet`. Jestliže takový příkaz budete používat častěji, pak doporučujeme, abyste si pro něj vytvořili alias.

Přenášení souborů

`ftp` *`vzdálený systém`*

Normální způsob přenášení souborů zprostředkovává v operačním systému Unix program `ftp`, což je zkratka pro „**file transfer protocol**“. Po zadání příkazu `ftp` budete vyzváni, abyste se přihlásili ke vzdálenému systému téměř stejným způsobem, jako v případě programu `telnet`. Pak se vám zobrazí speciální příkazový řádek.

Nyní máte k dispozici několik příkazů běžných v operačním systému Unix, jež můžete aplikovat v příkazovém řádku programu `ftp`. Například příkaz `cd` i zde slouží k přepínání se mezi adresáři a příkaz `ls` slouží k zobrazení seznamu souborů uložených v aktuálním adresáři.

Navíc máte k dispozici dva důležité příkazy: `get` a `put`. Příkaz `get` je určen k přenosu souborů ze vzdáleného počítače do vašeho lokálního počítače a příkaz `put` slouží k opačnému úkolu. Oba příkazy jako implicitní používají váš domovský adresář a lokální adresář na vzdáleném počítači (který můžete měnit prostřednictvím příkazu `cd`).

S používáním programu `ftp` je spojen jeden problém. Spočívá v rozlišení mezi znakovými a binárními soubory. Protokol pro přenos dat programem `ftp` je velmi starý a implicitně předpokládá, že přenášené soubory jsou textové. Aplikuje-li se tento implicitní přenos na binární soubory, pak budou s největší pravděpodobností po přenosu poškozeny. Před přenosem binárního souboru proto používejte příkaz `binary`.

Program `ftp` ukončíte příkazem `bye`.

Putování po stránkách WWW

WWW, neboli World Wide Web (doslova „celosvětová pavučina“) zřejmě představuje nejpopulárnější způsob využívání Internetu. Skládá se ze stránek, z nichž každá je spojena s tzv. lokátorem URL (**uniform resource locator**). Lokátory URL jsou komické řetězce následujícího tvaru: `http://www.rutgers.edu/`. Stránky jsou vytvořeny v jazyku HTML (**hypertext markup language**).

Jazyk HTML umožňuje autorům stránek WWW začlenit do dokumentů odkazy na kterékoliv jiné stránky WWW (nebo obrázky) lokalizované kdekoliv jinde v celosvětovém systému WWW. Když uživatel čte nějaký dokument, pak se pouhým klepnutím na odkaz (prezentovaný klíčovým slovem nebo tlačítkem) přenese na jinou stránku WWW, která může být uložena v počítači na druhém konci světa.

`netscape [`*`url`*`]`

Nejpopulárnějším programem pro prohlížení stránek WWW je v operačním systému Linux program Netscape od firmy Netscape Corporation. Program Netscape může běžet pouze pod systémem X Window.

Program Netscape je velmi jednoduchý, pokud jde o ovládání. Je založen na knihovně Motif a připomíná program napsaný pro Microsoft Windows (samozřejmě, že pro Microsoft Windows také existuje verze programu Netscape). Odkazy na další stránky WWW jsou v programu Netscape zobrazeny modře. Stačí na odkaz klepnout levým tlačítkem myši a vzápětí se vám zobrazí nová stránka WWW.

Operační systém Linux podporuje i jiné programy pro prohlížení stránek WWW, například program lynx, což je textově orientovaný program, proto není schopen zobrazovat obrázky a jiné grafické prvky dokumentů WWW. Je ovšem schopen pracovat pod systémem X Window.

`lynx [url]`

Naučit se pracovat s programem `lynx` je poněkud obtížnější, než naučit se pracovat s programem `Netscape`. Při práci si budete muset zvyknout na používání kurzorových kláves. Klávesy „šipka nahoru“ a „šipka dolů“ jsou určeny k přepínání mezi odkazy na dané stránce WWW, klávesa „šipka doprava“ zobrazí novou stránku (na kterou odkazuje zvýrazněný odkaz) a klávesa „šipka doleva“ je určena k zobrazení předcházející stránky. K ukončení programu `lynx` použijte klávesu Q. Program `lynx` má mnohem více příkazů, jejichž kompletní popis najdete v manuálových stránkách.

KAPITOLA 11

Zábavné příkazy

Většina lidí, která má co do činění s operačním systémem Unix, asi nebude souhlasit s titulem této kapitoly. Někteří se dokonce rozčílí, protože si u každého příkazu musejí pamatovat až desítky parametrů a najednou se jim někdo snaží namluvit, že by používání takových příkazů mohlo být zábavné. V nadpisu této kapitoly se spíš odráží až sarkastický humor autorů operačního systému Unix. Dále uvedené příkazy totiž nemají ekvivalentní příkazy v operačním systému MS-DOS. Přitom jsou velmi výkonné, a když je zatvrzelí příznivci operačního systému MS-DOS chtějí používat, musejí si je koupit (speciální verze těchto příkazů pro MS-DOS byly dodatečně vytvořeny). Protože operační systém Unix odjakživa soupeří s operačními systémy od firmy Microsoft, jsou touto skutečností příznivci operačního systému Unix pobaveni.

V této kapitole se budeme zabývat příkazy `find` (příkaz pro vyhledávání skupin souborů v adresářové struktuře), `tar` (příkaz pro archivaci souborů nebo celých adresářů), `dd` (příkaz pro kopírování) a `sort` (příkaz pro třídění souborů). Předem je nutno upozornit na skutečnost, že tyto příkazy nejsou standardizovány. Zaměříme se na popis verzí náležejících do projektu GNU, protože právě tyto verze jsou implementovány v operačním systému Linux a právě tyto verze pravděpodobně budou (tak jako ostatní programy vytvořené v rámci projektu GNU) v blízké budoucnosti představovat standard. Používáte-li jiný operační systém typu Unix, pak nezapomeňte konfrontovat příslušné manuálové stránky.

Příkaz find

Všeobecné poznámky

Mezi doposud probranými příkazy jsou takové, které umožňují rekurzivní procházení stromovou strukturou adresářů. Příkladem jsou příkazy `ls -R` nebo `rm -R`. Příkaz `find` je také rekurzivní. Kdykoliv byste měli něco dělat s jistým typem souborů v adresáři a ve všech jeho podadresářích, vzpomeňte si na příkaz `find`. V jistém smyslu platí, že vyhledávání souborů příkazem `find` představuje spíše vedlejší efekt.

Základní struktura příkazu `find` je následující:

`find` *`cesta`* `[...]` *`výraz`* `[...]`

Uvedená syntaxe platí pouze pro verzi GNU. Ostatní verze neumožňují specifikovat více než jednu cestu (path), což z praktického hlediska není tak důležité. Hrubé vysvětlení funkce příkazu je následující: prostřednictvím prvního parametru zadáte, odkud má prohledávání začít (parametr path; u verze GNU nemusíte tento parametr vůbec uvádět a prohledávání pak začne od aktuálního adresáře), a druhý parametr (expression) určuje typ vyhledávání.

Standardní chování příkazu `find` je poněkud lstivé a stojí za to tomuto faktu věnovat trochu pozornosti. Předpokládejme, že vaším domovským adresářem je adresář garbage a že obsahuje soubor foobar. Řekněme, že náhodou napíšete příkaz `find .-name foobar`. Tento příkaz by měl podle všech předpokladů vyhledat soubory nazvané `foobar` – avšak nic se nestane. Potíž je v tom, že program `find` je tzv. tichý (silent) příkaz. Pouze vrátí 0 (ať už nějaký soubor najde nebo ne), nebo vrátí záporné číslo, pokud nastal nějaký problém. Uvedený případ nenastane, pokud používáte operační systém Linux, ale je dobré jej mít na paměti.

Výrazy

Výraz neboli parametr expression může být rozdělen do čtyř různých skupin klíčových slov: *volby*, *testy*, *akce* a *operátory*. Každé klíčové slovo může vracet hodnotu true nebo false a přitom může produkovat nějaký vedlejší efekt. Rozdíl mezi skupinami klíčových slov popisuje následující seznam:

volby celkově ovlivňují operaci vyhledávání a netýkají se zpracování souboru. Příkladem volby je `-follow`, která „sdělí“ příkazu `find`, aby procházel symbolické odkazy. Volby vždy vracejí hodnotu true.

testy jsou skutečnými testy. Například test `-empty` ověřuje, zda je soubor prázdný. Testy mohou vracet hodnotu true nebo false.

akce produkují jako vedlejší efekt jméno programu. Také mohou vracet hodnotu true nebo false.

operátory ve skutečnosti nevracejí žádnou hodnotu (dle konvence vracejí hodnotu true) a používají se ke skládání výrazů. Příkladem je operátor `-or`, jenž jako výsledek produkuje logickou operaci OR nad operandy, kterými jsou dva výrazy. Jsou-li vedle sebe uvedeny dva výrazy bez operátoru, pak se implicitně aplikuje operátor `-and`.

Poznamenejme, že program `find` spoléhá na syntaktickou analýzu příkazu, kterou realizuje příkazový procesor. To znamená, že všechna klíčová slova musejí být oddělena nezobrazitelnými znaky a zejména platí, že spousta znaků musí být oddělena znaky Escape – jinak by je příkazový procesor nemohl správně interpretovat. Jako znaky Escape mohou být použita obrácená lomítka, jednoduché nebo dvojité uvozovky. V později uvedených příkladech se jednoznaková klíčová slova uvozují obráceným lomítkem, protože je to nejjednodušší.

Volby

V následujícím seznamu uvádíme přehled všech voleb, které je schopen příkaz `find` akceptovat (připomínáme, že se jedná o verzi GNU). Nezapomeňte, že volby vždy vracejí hodnotu true.

- `-daystart`

  Volba `-daystart` měří dobu, která uplyne od poslední půlnoci. Počítačový nadšenec asi nemůže pochopit, proč má nějaký program takovou volbu, ale programátor, který pracuje od osmi do pěti, ji ocení.

- `-depth`

  Tato volba zajistí, že příkaz `find` bude zpracovávat obsah každého podadresáře před zpracováním adresáře samotného. Abych řekl pravdu, neumím si představit užitečné uplatnění této volby, kromě případu, kdy se emuluje příkaz `rm -F` (podadresáře nemůžete zrušit, dokud obsahují nějaké soubory).

- `-noleaf`

  Volba `-noleaf` vypíná optimalizaci, která indikuje, že adresář obsahuje o dva podadresáře méně, než by měl obsahovat. Kdyby byl svět dokonalý, pak by bylo možné odkazovat se na všechny adresáře z prostředí každého jejich podadresáře (pomocí volby `..`), pomocí odkazu `.` uvnitř samotného adresáře a prostřednictvím jeho skutečného jména z jeho nadřazeného adresáře.

  To znamená, že na každý adresář musejí existovat alespoň dva odkazy (jeden z adresáře samotného a jeden z adresáře nadřazeného) a případně další odkazy ze všech podadresářů. V praxi se však může stát, že symbolické odkazy a distribuované souborové systémy mohou toto pravidlo porušit.[1]

- `-maxdepth` *`levels`*, `-mindepth` *`levels`*

  Parametry `levels` jsou nezáporná čísla, jež udávají maximální a minimální úroveň podadresářů, které má příkaz `find` prohledávat. Uveďme příklady: `-maxdepth 0` indikuje, že příkaz `find` má být realizován pouze pro argumenty uvedené v příkazovém řádku a že se žádné podadresáře prohledávat nemají. `-mindepth 1` zakazuje zpracování příkazu pro argumenty uvedené v příkazovém řádku, zatímco ostatní soubory v podadresářích budou zpracovány.

- `-version`

  Po zadání parametru `-version` se pouze vypíší informace o verzi programu find.

- `-xdev`

  Parametr `-xdev` má poněkud zavádějící označení. Pro program `find` předává informaci o tom, že se dané zařízení (tedy souborový systém) nemá prohledávat. Volba `-xdev` je velmi užitečná v případě, kdy se má vyhledávat něco v hlavním souborovém systému. Hlavní souborový systém většinou obsazuje malou diskovou oblast. Kdyby se použil příkaz `find /` bez parametru `-xdev`, pak by se prohledávala celá adresářová struktura!

Testy

První dva testy jsou velmi jednoduché: `-false` vždy vrací hodnotu false a `-true` vždy vrací hodnotu true. K dalším testům, které nevyžadují specifikaci hodnoty, patří test `-empty` (vrací hodnotu true, pokud je daný soubor prázdný) a dále dvojice `-nouser / -nogroup`, kdy test vrací hodnotu true, právě když se v souboru `/etc/passwd` nebo `/etc/group` nevyskytuje záznam identifikující vlastníka souboru jako uživatele nebo skupinu. Jedná se o postižení situace, kdy je v systému zrušen účet uživatele, ale soubory jím vlastněné jsou stále uloženy někde v souborovém systému. Podle Murphyho zákonů jsou takové soubory neobyčejně velké.

Samozřejmě je možné vyhledávat soubory vlastněné jistým uživatelem nebo jistou skupinou. Odpovídající testy jsou `-uid nn` a `-gid nn`. Naneštěstí nelze přímo zadat uživatelské jméno, ale musí se vždy uvést jeho identifikační číslo `nn`.

Je povoleno použít zápis identifikačního čísla ve tvaru `+nn`, což znamená „ostře větší než“ nebo `-nn`, což znamená „ostře menší než“. Ve spojení s identifikačními čísly uživatelů se tato možnost jeví jako hloupá, ale v souvislosti s ostatními testy může být užitečná.

Další užitečnou volbou je volba `-type c`, která vrací hodnotu true, pokud je soubor typu `c`. Mnemotechnické zkratky jsou stejné, jako v případě příkazu `ls`: **b** pro binární soubor, **c** pro znakový soubor, **d** pro adresáře, **p** pro pojmenované roury, **l** pro symbolické odkazy a **s** pro sokety (sockets). Obyčejné soubory jsou specifikovány prostřednictvím zkratky **f**. K testu `-type` se vztahuje

1 Distribuované souborové systémy umožňují, aby se soubory lokalizované někde jinde jevily jako lokální.

test `-xtype`, který je podobný, ale chová se jinak v případě symbolických odkazů – pokud není zadána volba `-follow`, ověřuje se místo vlastního symbolického odkazu soubor, na který odkaz odkazuje.

Testy `-inum nn` a `-links nn` ověřují, zda je číslo i-uzlu příslušné k danému souboru nn nebo zda má soubor nn symbolických odkazů. Test `-size nn` má hodnotu true, jestliže je pro soubor alokováno nn bloků po 512 bajtech. Dnes již však neplatí, že se velikost souboru vždy měří (například po zadání příkazu `ls -s`) v blocích po 512 bajtech – Linux například používá bloky po 1024 bajtech. Proto je možné číslo `nn` doplnit znakem `b` (pak se měří velikost v bajtech) nebo k (pak se měří velikost v kilobajtech).

Bity specifikující přístupová práva se testují pomocí testu `-perm` *`mode`*. Pokud argumentu *`mode`* nepředchází žádné znaménko, pak bity specifikující přístupová práva musejí přesně souhlasit. Pokud předchází znaménko `-`, pak musejí být nastavena příslušná přístupová práva, ale o ostatních se nic nepředpokládá. Pokud předchází znak +, pak je podmínka splněna, právě když je kterýkoliv z bitů nastaven. Důležité je, že hodnota mode musí být uvedena buď symbolicky, nebo jako oktalové číslo, tak jako v případě příkazu `chmod`.

Následující skupina testů se vztahuje k času, kdy byl soubor naposledy použit. Takové testy jsou zejména užitečné tehdy, když dojde k zaplnění diskového prostoru a uživatel potřebuje vyhledat staré soubory, které lze zrušit. Staré a zapomenuté soubory se těžko hledají a příkaz `find` vám dává naději, že se vám je podaří nalézt. Test `-atime nn` vrací hodnotu true, pokud byl daný soubor zpřístupněn před `nn` dny, `-ctime nn` vrací hodnotu true, když byly atributy přístupových práv daného souboru změněny před `nn` dny (například příkazem `chmod`). Test `-mtime nn` vrací hodnotu true, když byl soubor naposledy modifikován před `nn` dny. Užitečným bude zřejmě test `-newer file`, který vrací hodnotu true, když byl uvažovaný soubor modifikován později než soubor uvedený jako parametr `file`. GNU-verze příkazu `find` také akceptuje testy `-anewer` a `-cnewer`, které se chovají podobně jako test `-newer`, a dále testy `-amin`, `-cmin` a `-mmin`, které pracují s časem uvedeným v minutách.

V neposlední řadě se musíme zmínit o testu `-name pattern`. Tento test vrací hodnotu true, pokud jméno souboru vyhovuje šabloně uvedené prostřednictvím parametru `pattern`. Pro šablonu platí prakticky stejná pravidla jako pro používání pseudoznaků ve jménech souborů, například při používání příkazu `ls`. Jistě si pamatujete, že příkazový procesor je schopen zvláštním způsobem interpretovat tzv. pseudoznaky (hvězdičku a otazník). Avšak test `-name foo*` nevrátí tu hodnotu, kterou byste očekávali. Místo toho budete muset použít test `-name foo` nebo `-name "foo*"`. Dobře si tuto situaci zapamatujte, většina uživatelů používá nesprávný test. Je zde ještě jeden rozdíl oproti příkazu `ls` – tečky na začátku nejsou interpretovány. Pokud se potřebujete vyrovnat s tímto problémem, použijte test `-path pattern`, který se o tečky a zpětná lomítka při porovnávání cest nestará.

Akce

Jak jsme si již řekli, argumenty charakterizující akce slouží k inicializaci nějaké operace. Přesto mezi tyto parametry patří například akce `-prune`, která nic neaktivuje, pouze realizuje krok dolů ve stromové struktuře adresářů. Většinou je tato akce spjata s parametrem `-fstype`, prostřednictvím kterého se realizuje výběr mezi různými souborovými systémy. Ostatní akce mohou být rozděleny do dvou širokých kategorií.

- Akce, které něco tisknou. Je zřejmé, že implicitní akcí bude akce `-print`. Ta v průběhu činnosti programu find tiskne jména souborů, které se právě zpracovávají (ovšem za předpokladu, že ostatní podmínky uvedené v příkazovém řádku vracejí hodnotu true). Akce `-print` má několik variant. První z nich, `-fprint file`, používá soubor uvedený jako pa-

rametr `file` místo standardního výstupu. Další, `-ls`, zobrazuje jméno aktuálního souboru jako příkaz `ls -dils`. Akce `-printf format` se chová podobně jako funkce `printf()` v jazyku C (v této variantě lze specifikovat formát výstupu). Totéž realizuje akce `-fprintf file`, ale do souboru uvedeném prostřednictvím parametru `file`. Tato akce rovněž vždy vrací hodnotu true.

- Akce, které aktivují nějaký příkaz. Jejich syntaxe je poněkud zvláštní, a proto se na ně podívejme podrobněji.

  ```
  -exec command \;
  ```

  Akce `-exec` spustí příkaz zadaný prostřednictvím parametru `command` a vrací hodnotu true, pokud má příkaz status ukončení 0. Za příkazem je uvedena dvojice znaků „`\;`“, protože jinak by příkaz `find` nebyl schopen rozeznat, kde příkaz `command` končí (trik s uvedením akce `-exec` na konec příkazu `find` není aplikovatelný). Proto se příkaz `command` ukončuje znakem „`;`“, což je zároveň znak k oddělování příkazů v jazyku příkazového procesoru. Bez znaku „`\`“ by středník byl příkazovým procesorem na základě syntaktické analýzy odstraněn a do příkazu `find` by se již nedostal. Proto se zde znak „`\`“ musí uvést. Další věc, kterou si musíte zapamatovat, spočívá ve způsobu specifikace jména aktuálního souboru uvnitř příkazu `command`. Jméno souboru musí být uvedeno pomocí dvojice složených závorek `{}`. Některé starší verze programu `find` vyžadují, aby bylo jméno odděleno netisknutelnými znaky (white spaces), například mezerami. To však není příliš šikovné, proto je ve verzi GNU zavedena konstrukce se složenými závorkami umožňující generovat řetězce. Asi vás napadá otázka, zda musejí být složené závorky uzavřeny do uvozovek. Podle mých zkušeností nemusejí – ani v případě, že použijete příkazový procesor `bash`, ani v případě, že použijete příkazový procesor `tcsh`. Proto zůstanou v příkazu `find` řetězce uvedené ve složených závorkách příkazovým procesorem nedotčeny.

  ```
  -ok command \;
  ```

  Akce `-ok` se chová podobně jako akce `-exec` s tím rozdílem, že pro každý vybraný soubor je uživatel vyzván k potvrzení příkazu (tj. zda se má provést nebo ne). Pokud odpověď začíná písmenem „y“ nebo „Y“, příkaz se provede a akce vrátí hodnotu true. Jinak se akce neprovede a vrácená hodnota je false.

Operátory

V příkazu `find` lze použít spoustu operátorů. Následující seznam je uvádí v pořadí s klesající prioritou.

```
\( expr \)
```

Kulaté závorky mění prioritu operandu. Závorkám musí předcházet znak „`\`“, protože v příkazovém procesoru mají speciální význam.

```
! expr
-not expr
```

Operátory `!` a `-not` mění pravdivostní hodnotu výrazu `expr`. Je-li hodnota `expr` true, změní se na false a naopak. Znak „`!`“ nemusí být oddělen obráceným lomítkem nebo uvozovkami, protože je následován netisknutelným znakem (mezerou).

```
expr1 expr2
expr1 -a expr2
expr1 -and expr2
```

Všechny tři varianty odpovídají logické operaci AND, přičemž se nejčastěji používá první varianta. Jestliže je první výraz `expr1` vyhodnocen jako false, pak se již druhý výraz nevyhodnocuje.

```
expr1 -o expr2
expr1 -or expr2
```

Obě varianty odpovídají logické operaci OR. Je-li první výraz `expr1` vyhodnocen jako true, pak se již druhý výraz `expr2` nevyhodnocuje.

```
expr1 , expr2
```

Uvedený operátor má poněkud speciální význam. Oba výrazy `expr1` i `expr2` se vyhodnotí (samozřejmě se všemi vedlejšími účinky) a hodnota konečného výrazu je nastavena na hodnotu výrazu `expr2`.

Příklady

Jak vyplývá z předcházejících seznamů, příkaz `find` má spoustu parametrů. Existuje však několik často používaných konstrukcí s příkazem `find`, které stojí za to si zapamatovat. Některé z nich si uvedeme jako příklady.

```
% find .-name foo\* -print
```

První příklad vyhledá všechny soubory, jejichž jména začínají řetězcem `foo`. Pokud se mají hledat soubory, jejichž jména mají řetězec `foo` někde uvnitř, pak je vhodné místo `foo` použít „`*foo*`“.

```
% find /usr/include/ -xtype f -exec grep foobar \
        /dev/null {}\;
```

V druhém příkladu se rekurzivně provádí příkaz `grep`, a to od adresáře `/usr/include`. V tomto případě se zajímáme o obyčejné soubory a symbolické odkazy, které na obyčejné soubory odkazují. Proto jsme použili test `-xtype`. V mnoha případech je jednodušší tento test nepoužít, zejména tehdy, kdy jsme si jisti, že žádný binární soubor neobsahuje požadovaný řetězec. A proč jsme použili příkaz `/dev/null`? Jedná se o trik, který příkaz `grep` „přinutí“ vypsat jméno souboru, pro který nebylo splněno výběrové kritérium. Příkaz `grep` je aplikován na každý soubor jinak, a proto „nepovažuje“ za nezbytné vypisovat jméno souboru. Ale nyní jsou zde dva soubory – aktuální soubor a soubor `/dev/null`. Jiná možnost spočívá v použití roury a příkazu `xargs`. Když jsem ji zkusil, zničil jsem si celý souborový systém.

```
%find / -atime +1 -fstype ext2 -name core \
        -exec rm {}\;
```

Tento příklad představuje klasickou úlohu pro crontab. Zruší ze souborového systému `ext2` všechny soubory s názvem `core`, které nebyly posledních 24 hodin zpřístupněny. Je možné, že někdo tyto soubory používá v souvislosti s programem `gdb` k ladění, ale je málo pravděpodobné, že je bude používat po 24 hodinách.

```
% find /home -xdev -size +500k -ls > piggies
```

Další příklad umožňuje zjistit, kdo vlastní soubory větší než 500 kilobajtů a kdo tedy zatěžuje souborový systém. Poznamenejme, že pokud se zajímáme pouze o jeden souborový systém, pak argument `-xdev` není nutný.

Slovo na závěr

Mějte na paměti, že příkaz `find` spotřebuje velmi mnoho času, pokud má provádět operace s každým souborem v celém souborovém systému. Proto je nutné počet operací vhodně optimalizovat, zejména když se na vašem systému pravidelně realizují úlohy pro údržbu prostřednictvím crontab. Jako poučný příklad vezměme následující: Předpokládejme, že chceme zrušit soubory končící na `.BAK`, přitom chceme změnit přístupová práva všech adresářů na 771 a přístupová práva souborů končících na `.sh` změnit na 755. Přitom chceme ještě připojit souborový systém NFS na telefonní linku, a v tomto systému nechceme žádné soubory kontrolovat. Proč bychom měli psát tři různé příkazy? Nejefektivněji uvedenou úlohu splníme takto:

```
% find .\( -fstype nfs -prune \) -o \
       \( -type d -a -exec chmod 771 {}\; \) -o \
       \( -name "*.BAK" -a -exec /bin/rm {}\; \) -o \
       \( -name "*.sh" -a -exec chmod 755 {}\; \)
```

Asi se vám bude takový příkaz zdát nesrozumitelný, ale když si jej pozorně prohlédnete, zjistíte, že je logický. Pamatujte si, že příkazy tohoto typu neprovádějí nic jiného, než že vyhodnocují výrazy, jejichž hodnota je buď false, nebo true. Samozřejmě mají vedlejší účinky – ty se realizují tehdy, když příkaz `find` musí vyhodnotit část obsahující výraz s akcí `-exec`, což nastane právě tehdy, když je levá strana výrazu vyhodnocena jako pravdivá (true). Když je například právě zpracovávaným souborem adresář, pak se bude realizovat první akce `-exec` a přístupová práva adresáře budou změněna na 771. Ostatní části příkazu budou vynechány. Budete-li takové příkazy aplikovat prakticky, přestanou se vám zdát nesrozumitelné a budete je používat zcela přirozeně.

Archivační program tar

Úvod

Tar je obecně použitelný program pro archivaci souborů, který je schopen sloučit (spakovat) velké množství souborů do jediného archivního souboru, přičemž zachovává veškeré informace o souborech, jako jsou uživatelská práva. Jméno `tar` je zkratkou „tape archive“, protože tento nástroj byl původně používán pro archivaci na magnetické pásky. Jak však uvidíme, používání příkazu `tar` není zdaleka omezeno pouze na záložní soubory pro magnetické pásky.

Hlavní funkce

Formát příkazu `tar` je následující:

`tar` *`functionoptions files...`*

kde *`function`* je jednoznaková indikace operace, která se má provést; *`options`* je seznam (jednoznakových) voleb pro tuto operaci a *`files`* je seznam souborů, jež se mají spakovat nebo rozpakovat. (Všimněte si, že argument function není oddělen od options mezerou.) Parametr `function` může být:

- `c` pro vytvoření nového archivního souboru
- `x` pro extrakci souborů z archivního souboru
- `t` pro výpis obsahu archivního souboru
- `r` pro přidání souborů na konec archivního souboru
- `u` pro aktualizaci souborů, které jsou novější než soubory v archivním souboru
- `d` pro porovnání souborů v archivním souboru se soubory v souborovém systému

Nejčastěji budete používat parametr `c`, `x` a `t`; ostatní budete používat jen zřídka.

Volby

Nejčastěji používané volby `options` jsou tyto:

- `v` pro tisk podrobné informace v průběhu pakování a rozpakování
- `k` pro zachování existujících souborů při rozpakování, tj. žádný existující soubor nebude přepsán souborem z archivního souboru
- `f` *`filename`* pro specifikaci jména archivního souboru

Další volby popíšeme v následujícím oddílu později.

Příklady

Ačkoliv se syntaxe příkazu `tar` zdá být na první pohled složitá, je jeho praktické použití velmi jednoduché. Předpokládejme, že máme adresář `mt` obsahující následující soubory:

```
rutabaga% ls -l mt
-rw-r--r-- 1 root root 24 Sep 21 1993 Makefile
-rw-r--r-- 1 root root 847 Sep 21 1993 README
-rw-r--r-- 1 root root 9220 Nov 16 19:03 mt
-rwxr-xr-x 1 root root 2775 Aug 7 1993 mt.1
-rw-r--r-- 1 root root 6421 Aug 7 1993 mt.c
-rw-r--r-- 1 root root 3948 Nov 16 19:02 mt.o
-rw-r--r-- 1 root root 11204 Sep 5 1993 st_info.txt
```

Nyní chceme spakovat pomocí příkazu `tar` obsah tohoto adresáře do jediného archivního souboru. Použijeme tedy příkaz:

```
tar cf mt.tar mt
```

Prvním argumentem v příkazu `tar` je operace (zde `c` pro vytvoření) následovaná volbou `options`, kde jsme použili `f mt.tar` a specifikovali tak jméno archivního souboru `mt.tar`. Jako poslední je uveden seznam souborů – pokud se místo seznamu uvede název adresáře, `tar` spakuje všechny soubory v tomto adresáři.

Poznamenejme, že prvním argumentem musí být písmeno, označující operaci, následované seznamem voleb „options“. Proto není důvod dávat před první argument pomlčku, jak to vyžaduje většina systémů Unix. Příkaz `tar` v systému Linux však tuto pomlčku připouští, proto by mohl mít posledně uvedený příklad tvar:

```
tar -cf mt.tar mt
```

V některých verzích musí být typ operace (jako je `c`, `t`, nebo `x`) na prvním místě, v jiných verzích na pořadí písmen nezáleží. Jistý přehled o průběhu pakování (nebo rozpakování) získáte prostřednictvím volby `v` – pak se bude každý soubor ukládaný do archivního souboru vypisovat na obrazovce. Například:

```
rutabaga% tar cvf mt.tar mt
mt/
mt/st_info.txt
mt/README
mt/mt.1
mt/Makefile
mt/mt.c
mt/mt.o
mt/mt
```

Použijete-li volbu `v` vícekrát, zobrazovaná informace bude podrobnější:

```
rutabaga% tar cvvf mt.tar mt
drwxr-xr-x root/root 0 Nov 16 19:03 1994 mt/
-rw-r--r-- root/root 11204 Sep 5 13:10 1993 mt/st_info.txt
-rw-r--r-- root/root 847 Sep 21 16:37 1993 mt/README
-rwxr-xr-x root/root 2775 Aug 7 05:50 1993 mt/mt.1
-rw-r--r-- root/root 24 Sep 21 16:03 1993 mt/Makefile
-rw-r--r-- root/root 6421 Aug 7 09:50 1993 mt/mt.c
-rw-r--r-- root/root 3948 Nov 16 19:02 1994 mt/mt.o
-rw-r--r-- root/root 9220 Nov 16 19:03 1994 mt/mt
```

Tyto podrobné informace jsou důležité zejména tehdy, když chcete zkontrolovat, zda `tar` realizuje pakování dle vašich představ. U některých verzí programu `tar` musí být volba `f` uvedena jako poslední. Je to z toho důvodu, že se za volbou `f` předpokládá jméno souboru. Pokud neuvedete volbu `f`, předpokládá `tar` (z historických důvodů), že má použít zařízení `/dev/rmt0`, což je první magnetopásková jednotka.

Nyní můžeme soubor `mt.tar` předat někomu jinému a ten si jej může rozpakovat na svém počítači. K rozpakování by měl použít následující příkaz:

```
tar xvf mt.tar
```

Tento příkaz vytvoří adresář `mt` a umístí do něj všechny soubory, které jsme dříve uvedli – s týmiž přístupovými právy jako v původním systému. Nové soubory budou vlastněny uživatelem, který provedl příkaz `tar xvf` – pokud ovšem nepoužijete tento příkaz jako root (pak je původní vlastník zachován). Parametr `x` zajistí rozpakování souborů a volba `v` opět zobrazí soubory, které se ukládají na disk:

```
courgette% tar xvf mmt.tar
mt/
mt/st_info.txt
mt/README
mt/mt.1
mt/Makefile
mt/mt.c
mt/mt.o
mt/mt
```

Všimněte si, že tar zachoval cestu každého souboru relativní k poloze v původní struktuře adresářů. To znamená, že když jsme vytvářeli archivní soubor pomocí příkazu `tar cf mt.tar mt`, jediný soubor, který jsme specifikovali místo seznamu souborů, byl adresář `mt` obsahující soubory. Proto tar uložil do archivního souboru samotný adresář a soubory, které se v něm nacházely. Při rozpakování se napřed vytvořil adresář `mt` a pak do něj byly uloženy soubory – tedy přesně opačný proces, než jaký probíhal při vytváření archivního souboru. tar implicitně rozpakuje všechny soubory relativně k pracovnímu adresáři, v němž jej spustíte. Pokud se například pokusíte spakovat obsah adresáře `/bin` příkazem:

```
tar cvf bin.tar /bin
```

dostanete varovné hlášení:

```
tar: Removing leading / from absolute path names in the archive.
```

To znamená, že soubory jsou ukládány v archivním souboru uvnitř adresáře `/bin`. Když tento soubor rozpakujete, adresář `/bin` bude vytvořen jako podadresář vašeho pracovního adresáře, v němž spouštíte `tar` – ne jako absolutní adresář `/bin`. Toto je velmi důležitý mechanismus, který byl vymyšlen za účelem ochrany před fatálními chybami při rozpakování pomocí příkazu `tar`. Jinak byste, podle předchozího příkladu, mohli přepsat soubory v adresáři `/bin`, a tak zničit systém.

Pokud byste však opravdu chtěli rozpakovat takový archivní soubor do adresáře `/bin`, museli byste mít jako pracovní adresář nastaven `/`. Výše uvedený mechanismus můžete potlačit pomocí volby `p`, ale nedoporučuje se to. Jiný způsob, jak vytvořit soubor `mt.tar`, spočívá v tom, že se přepnete do adresáře `mt` pomocí příkazu `cd` a zadáte příkaz:

```
tar cvf mt.tar *
```

Potom ovšem nebude podadresář `mt` ukládán do archivního souboru a při rozpakování budou soubory ukládány přímo do vašeho pracovního adresáře. Proto doporučujeme vždy pakovat soubory tak, aby archivní soubor obsahoval podadresář, jak jsme ukázali pomocí příkazu `tar cvf mt.tar mt`. Pak se vždy před rozpakováním vytvoří adresář pro uložení souborů a nemůže se stát, že přepíšete soubory ve svém pracovním adresáři. Navíc zbavíte osobu, která provádí rozpakování, starostí s vytvářením adresáře pro ukládání souborů a ušetříte jí dost času. Samozřejmě existuje mnoho situací, při nichž nebude vhodné doporučený postup dodržovat, ale všeobecně se takový postup považuje za jakousi etiketu při práci s programem `tar`.

Při vytváření archivních souborů můžete uvést seznam souborů nebo adresářů, které se mají do archivního souboru uložit. V prvním příkladu jsme použili příkaz `tar` pro jediný adresář a ukázali jsme použití hvězdičky, kterou příkazový procesor rozšíří na seznam všech souborů v pracovním adresáři. Před rozpakováním archivního souboru pomocí příkazu `tar` je vhodné se podívat, co je jeho obsahem. Tak se například dozvíte, zda budete muset vytvořit adresář pro uložení souborů vlastními silami, nebo bude vytvořen automaticky. K prohlížení obsahu archivního souboru použijte příkaz:

```
tar tvf tarfile
```

Ten zobrazí obsah archivního souboru `tarfile`. Poznamenejme, že pokud použijete funkci t, stačí uvést jednu volbu `v` a obdržíte podrobnou informaci o obsahu archivního souboru:

```
courgette% tar tvf mt.tar mt
drwxr-xr-x root/root 0 Nov 16 19:03 1994 mt/
```

```
-rw-r--r-- root/root 11204 Sep 5 13:10 1993 mt/st_info.txt
-rw-r--r-- root/root 847 Sep 21 16:37 1993 mt/README
-rwxr-xr-x root/root 2775 Aug 7 05:50 1993 mt/mt.1
-rw-r--r-- root/root 24 Sep 21 16:03 1993 mt/Makefile
-rw-r--r-- root/root 6421 Aug 7 09:50 1993 mt/mt.c
-rw-r--r-- root/root 3948 Nov 16 19:02 1994 mt/mt.o
-rw-r--r-- root/root 9220 Nov 16 19:03 1994 mt/mt
```

Žádné rozpakování v tomto případě neproběhne, ale pouze se zobrazí informace o obsahu archivního souboru. Zde vidíme, že jména souborů jsou doplněna jménem adresáře `mt`, a proto při rozpakování bude tento adresář nejdříve vytvořen a pak teprve do něj budou ukládány soubory. Příkaz `tar` můžete rovněž použít k extrakci jednotlivých souborů z archivního souboru. Potom použijte příkaz ve tvaru:

```
tar xvf tarfile files
```

kde `files` je seznam souborů, které se mají rozpakovat. Jak jsme si již ukázali, pokud neuvedeme žádný seznam, `tar` rozpakuje všechny soubory z archivního souboru.

Jestliže specifikujete soubory, které se mají rozpakovat, musíte uvést jejich plná jména včetně cesty tak, jak jsou uvedena v archivním souboru. Chceme-li například rozpakovat soubor `mt.c` z výše uvedeného archivního souboru `mt.tar`, použijeme následující příkaz:

```
tar xvf mt.tar mt/mt.c
```

který nejdříve vytvoří podadresář `mt` a do něj pak umístí soubor `mt.c`. Program `tar` má mnohem více možností, než které jsme zde uvedli. Ty, o nichž jsme se zde zmínili, budete zřejmě používat nejčastěji. Verze GNU implementovaná operačnímu systému Linux má řadu přípon a představuje ideální nástroj pro pořizování záložních kopií. Více informací naleznete v manuálové stránce k příkazu `tar`.

Jak používat program tar spolu s programem gzip

Program tar neprovádí při ukládání dat do archivního souboru žádnou komprimaci. Jestliže vytvoříte soubor `tar` ze tří souborů o velikosti 200 KB, pak výsledný soubor bude mít velikost 600 KB. Proto patří k běžné praxi komprimovat soubory `tar` pomocí programu `gzip` (nebo starším programem `compress`). Komprimovaný soubor tar můžete vytvořit pomocí následujících příkazů:

```
tar cvf tarfile files...
gzip -9 tarfile
```

Provedení takových příkazů je však těžkopádné a vyžaduje, abyste měli na disku dostatek místa pro nekomprimovaný soubor `tar`, než spustíte `gzip`.

Mnohem efektivnější způsob spočívá v tom, že se využije možnost programu tar zapisovat archivní soubor do standardního výstupu. Pokud použijete místo jména archivního souboru pomlčku (`-`), pak bude `tar` číst data ze standardního vstupu nebo zapisovat do standardního výstupu. K vytvoření komprimovaného souboru tar tedy můžeme použít příkaz:

```
tar cvf - files... | gzip -9 > tarfile.tar.gz
```

Zde vytváří `tar` archivní soubor ze souborů uvedených v seznamu `files`, zapisuje jej do standardního výstupu; `gzip` čte data ze standardního vstupu, komprimuje je a zapisuje do svého standardního výstupu. Nakonec jsme přesměrovali komprimovaný soubor `tar` do `tarfile.tar.gz`. Podobně bychom mohli k rozpakování takového souboru použít příkaz:

```
gunzip -9c tarfile.tar.gz | tar xvf -
```

Zde `gunzip` provede dekomprimaci uvedeného archivního souboru a zapisuje výsledek do standardního výstupu sloužícího jako standardní vstup pro program `tar`, jenž soubory rozpakuje a ukládá. Není práce v systému Unix zábavná? Samozřejmě, že jsou oba výše uvedené příkazy poněkud těžkopádné. Naštěstí GNU-verze programu tar má možnost uvést volbu `z`, která zajistí automatické vytváření nebo rozpakování komprimovaných archivních souborů `tar`. (Diskusi o použití volby `z` jsme úmyslně zařadili až na toto místo, abyste si uvědomili její výhody.) K vytvoření a rozpakování archivních souborů máte tedy možnost používat následující příkazy:

```
tar cvzf tarfile.tar.gz files...
```

a

```
tar xvzf tarfile.tar.gz
```

Poznamenejme, že byste měli soubory vytvořené tímto způsobem označovat pomocí přípony `.tar.gz`, aby byl zřejmý jejich formát. Volba funguje i ve spojení s dalšími parametry, jako je například `t`. Možnost používat volbu `z` podporuje pouze GNU-verze programu `tar`. Jestliže používáte `tar` na jiných systémech Unix, musíte pro realizaci stejného úkolu zadávat delší příkazy, jak jsme uvedli dříve. Téměř všechny verze operačního systému Linux používají verzi GNU.

Nyní využijeme svých znalostí a napíšeme krátké skripty pro příkazový procesor, které vám mohou sloužit jako návod pro vytváření a rozpakování souborů `tar`. V příkazovém procesoru bash doplňte následující řádky do souboru `.bashrc`:

```
tarc () {tar czvf $1.tar.gz $1 }
tarx () {tar xzvf $1 }
tart () {tar tzvf $1 }
```

Budete-li nyní chtít vytvořit komprimovaný archivní soubor obsahující soubory z jednoho adresáře, stačí zadat příkaz:

```
tarc directory
```

Výsledný archivní soubor bude mít název `directory.tar.gz`. (Přesvědčte se, že jste neuvedli před jménem adresáře lomítka (`/`) – jinak bude archivní soubor vytvořen jako `.tar.gz` uvnitř daného adresáře.) K zobrazení obsahu archivního souboru stačí zadat příkaz

```
tart file.tar.gz
```

a k jeho rozpakování příkaz

```
tarx file.tar.gz
```

Triky při používání programu tar

Protože tar ukládá do archivního souboru informace o vlastnictví souborů, přístupových právech, adresářové struktuře i symbolických odkazech, velmi dobře se hodí ke kopírování nebo přesouvání celých adresářových stromů z jednoho místa na druhé v rámci jednoho systému (a dokonce i mezi systémy, jak uvidíte). Použijete-li syntaxi s pomlčkou (`-`), budete zapisovat výsledný soubor do standardního výstupu, který může být čten jako standardní vstup a rozpakován kdekoliv.

Předpokládejme například, že máme adresář obsahující dva podadresáře: `from-stuff` a `to-stuff`. Adresář `from-stuff` obsahuje celý strom dalších podadresářů s mnoha soubory, symbolickými odkazy atd. a bylo by velmi obtížné zkopírovat jej pomocí příkazu `cp`. Ke zkopírování celého adresáře `from-stuff` do adresáře `to-stuff` však můžeme použít následující příkazy:

```
cd from-stuff
tar cf - .| (cd ../to-stuff; tar xvf -)
```

Velice jednoduché a elegantní! Začínáme v adresáři `from-stuff`, kde vytvoříme soubor tar obsahující celý adresář, a zapíšeme jej do standardního výstupu. Tento výstup pak čte podřízený příkazový procesor (příkazy uvedené v závorkách), který se nejdříve přepne do adresáře `../to-stuff` a pak spustí příkaz `tar xvf`, jenž čte ze standardního vstupu. Přitom se žádný archivní soubor `tar` neukládá na disk – data jsou přímo posílána z jednoho procesu tar do druhého. Druhý proces tar používá volbu `v` pro zobrazení informací o každém souboru, který se ukládá, a vy můžete kontrolovat, zda kopírování obsahu adresářů probíhá správně.

Pomocí tohoto „triku“ můžete ve skutečnosti přenášet celé adresáře mezi jednotlivými systémy – stačí vložit vhodný příkaz `rsh` mezi příkazy uzavřené v závorkách. Pak vzdálený příkazový procesor spustí `tar`, který bude číst archivní soubor jako standardní vstup. (Poznamenejme, že verze GNU příkazu `tar` má možnost automaticky číst nebo zapisovat soubory `tar` z/do jiných počítačů v síti; informace najdete v příslušné manuálové stránce.)

Program dd

Existuje jakási legenda, podle které v ranných dobách vývoje operačního systému Unix potřebovali programátoři program pro binární kopírování dat mezi jednotlivými zařízeními. Protože velmi spěchali, „vypůjčili“ si syntaxi příkazu používanou v operačním systému na počítačích IBAlt+360 a později vyvinuli rozhraní konzistentní s ostatními příkazy operačního systému Unix. Nevím, zda je tato legenda pravdivá, ale pěkně se vypráví.

Volby

Přes svůj legendou opředený původ není program `dd` úplně jiný než ostatní příkazy operačního systému Unix. Ve skutečnosti představuje filtr, který implicitně čte data ze standardního vstupu a zapisuje je na standardní výstup. Pokud zadáte na terminálu pouze příkaz `dd`, pak bude program `dd` tiše očekávat vstup z klávesnice.

Syntaxe příkazu `dd` je následující:

```
dd [if=file] [of=file] [ibs=bytes] [obs=bytes]
   [bs=bytes] [cbs=bytes] [skip=blocks] [seek=blocks]
   [count=blocks] [conv={ascii, ebcdic, ibm, block,
   unblock, lcase, ucase, swab, noerror, notrunc, sync}]
```

Všechny volby mají tvar *option=value*. Před a za znakem „=“ se nesmí objevit mezera, což je nepříjemné, protože v takové situaci neprovede příkazový procesor doplnění jména souboru, pokud se uvedou pseudoznaky. Verze příkazového procesoru bash v operačním systému Linux je však dostatečně inteligentní, proto nemusíte mít obavy. Všechny výše uvedené číselné hodnoty (`bytes` a `blocks`) mohou být následovány multiplikativní konstantou. Pro tyto konstanty lze použít následující zkratky: **b** pro bloky (multiplikativní konstanta 512), **k** pro kilobajty (multiplikativní konstanta 1024), **w** pro slova (multiplikativní konstanta 2) a prostřednictvím **xm** se číselná hodnota násobí konstantou **m**.

Význam voleb příkazu `dd` je popsán v následujícím seznamu.

- `if=`*`filein`* a `of=`*`fileout`* jsou volby specifikující jméno vstupního a výstupního souboru. Výstupní soubor je zkrácen na velikost danou volbou `seek`. Pokud není klíčové slovo `seek` uvedeno, pak je hodnota `seek` rovna nule (soubor je před provedením operace zrušen). Volba `notrunc` (viz dále) však může toto implicitní chování příkazu `dd` změnit.
- `ibs=`*`nn`* a `obs=`*`nn`* jsou volby specifikující počet bajtů, které se mají najednou přečíst a najednou zapsat. Domnívám se, že implicitní hodnoty jsou jeden blok (t.j. 512 bajtů), ale nejsem si tím tak docela jist. Uvedené parametry jsou velmi důležité v případě, kdy se jako vstup nebo výstup používají speciální zařízení – například při čtení ze sítě je hodnota `ibs` nastavena na 10 kilobajtů, zatímco disketa 3,5 palce má přirozenou délku bloku 18 kilobajtů. Nevhodné nastavení těchto hodnot může nejen značně prodloužit realizaci programu, ale také může vést k neodstranitelným chybám. Proto buďte opatrní.
- `bs=`*`nn`* je volba, která předefinuje nastavení `ibs` a `obs` – obě hodnoty se nastaví na stejnou konstantu *nn*.
- Volba `cbs=`*`nn`* nastavuje vyrovnávací paměti pro konverzi na *nn* bajtů. Vyrovnávací paměť se používá při konverzi z formátu ASCII na EBCDIC nebo při konverzi mezi textovými a binárními soubory a podobně. Například soubory vytvořené pod operačním systémem VMS mají zpravidla velikost bloku 512 bajtů, proto byste měli například při čtení dat zapsaných na magnetickou pásku v operačním systému VMS nastavit hodnotu cbs na 1 bajt.
- `skip=`*`nbl`* a `seek=`*`nbl`* jsou volby, které „sdělí“ programu `dd`, kolik bloků má „přeskočit“ při čtení vstupu a při zápisu na výstup. Samozřejmě platí, že druhá volba má smysl jedině tehdy, když je specifikována konverze notrunc. Velikost bloků je dána volbami `ibs` a `obs`. Uvědomte si, že pokud nezadáte hodnotu ibs a zadáte hodnotu `skip=1b`, pak ve skutečnosti dojde k „přeskočení“ 512x512 bajtů, což je 256 kilobajtů.
- Volbou `count=`*`nbl`* se nastavuje, kolik bloků se má ze vstupu kopírovat. Velikost bloku je přitom dána hodnotou `ibs`. Uvedená volba spolu s předcházející volbou je užitečná v případě, kdy chcete obnovit co nejvíce bajtů z porušeného souboru – „přeskočíte“ nečitelnou část souboru a zbytek zkopírujete do nového souboru.
- Volba `conv=`*`conversion`*`,[`*`conversion`*`,...]` je určena ke konverzi podle specifikace. Možné konverze jsou následující: `ascii` – konverze formátu EBCDIC na formát ASCII; `ebcdic` nebo `ibm` – konverze z formátu ASCII na EBCDIC (jednoznačná konverze z formátu EBCDIC na formát ASCII neexistuje; první představuje standardní konverzi a druhá funguje lépe, pokud se má soubor tisknout na tiskárně IBM); `block` – vyplňuje mezerami záznamy ukončené znakem newline na délku uvedenou prostřednictvím volby `cbs`; `unblock` – opačná konverze ke konverzi `block`; `lcase` – konverze z velkých písmen na malá; `ucase` – konverze z malých písmen na velká; `swab` – konverze, při které se každý pár vstupních bajtů zamění (chcete-li například použít na počítači s procesorem Intel soubor obsahující celá čísla, který byl vytvořen na počítači s procesorem 680x0, budete takovou

konverzi potřebovat); `noerror` – program `dd` bude pokračovat v činnosti i poté, co nastane nějaká chyba; `sync` – každý vstupní blok se doplní znaky NUL tak, aby měl délku definovanou prostřednictvím volby `ibs`.

Příklady

Dříve či později se setkáte s případem, kdy budete chtít pod operačním systémem Linux vytvořit svoji první disketu. Jak zapsat data na disketu bez souborového systému MS-DOS? Řešení je jednoduché:

```
% dd if=disk.img of=/dev/fd0 obs=18k count=80
```

V uvedeném příkladu jsem se rozhodl nepoužít volbu `ibs`, protože nevím jaká je optimální volba pro velikost bloku na pevném disku. Nemůže však uškodit, když se místo volby `obs` použije volba `bs` – pak je proces kopírování dokonce rychlejší. Všimněte si nastavení počtu sektorů pro zápis (18 kilobajtů je velikost sektoru, proto je hodnota `count` nastavena na `80`). Pro disketovou jednotku se v operačním systému Linux používá jméno zařízení `/dev/fd0`.

Další užitečné použití příkazu `dd` je v oblasti zálohování, zejména při zapojení počítače do sítě. Předpokládejme, že máte počítač alfa a že na počítači beta je magnetopásková jednotka `/dev/rst0` obsahující soubor, který chcete zkopírovat. Dále předpokládejme, že máte stejná přístupová práva na obou počítačích a že na pevném disku počítače beta není dostatek místa pro kopii uvažovaného souboru. Pak můžete použít příkaz:

```
% rsh beta 'dd if=/dev/rst0 ibs=8k obs=20k' | tar xvBf -
```

Celou operaci takto realizujete „na jeden průchod“. Zajisté jste si všimli, že se v příkazu využila schopnost příkazového procesoru číst data z magnetopáskové jednotky. Velikosti vstupních a výstupních bloků jsou nastaveny na implicitní hodnoty, tedy 8 kilobajtů pro čtení a 20 kilobajtů pro zápis do sítě ethernet.

Poznámka na závěr: zapomněl jsem říci, že označení příkazu `dd` je zkratkou pro výraz „data duplicator“.

KAPITOLA 12

Chyby, skryté závady a další nepříjemnosti

Jak předcházet chybám

Řada lidí na nejrůznějších fórech dává najevo zklamání z operačního systému Unix. Jejich zklamání zpravidla vyplývá z toho, že jej neumějí efektivně využívat – obvykle se jedná o uživatele dříve zvyklé na operační systémy s pohodlnou obsluhou, jako je Microsoft Windows, Macintosh Operating System a podobně. Naopak lidé, kteří zvládli filosofii operačního systému Unix a jsou schopni v něm efektivně pracovat, na něj nedají dopustit. Cení si zejména toho, že jen málokterý příkaz vyžaduje v průběhu své realizace nějakou komunikaci s uživatelem, a proto v operačním systému vše běží hladce, rychle a bez problémů.

Právě skutečnost, že například příkazy `rm` a `mv` nikdy nevyžadují potvrzení, zda mají skutečně zrušit specifikované soubory, vede často k problémům. Proto si nyní projděme malý seznam, v němž uvádíme zásady, jak se takovým a podobným problémům vyhnout.

- Pořizujte si záložní kopie. Tato výzva platí zejména pro uživatele, kteří používají systém sami. Každý systémový administrátor by měl pravidelně pořizovat záložní kopie – dostatečný interval je asi jeden týden. Podrobnosti najdete v *„Příručce správce operačního systému Linux“*.
- Každý uživatel by si měl pořizovat své vlastní záložní kopie. Pokud používáte více jak jeden systém, pak si uchovávejte aktualizované kopie všech svých souborů na každém systému. Jestliže máte přístup k disketové jednotce, pak si na ni ukládejte kopie důležitých souborů. Přinejhorším si kopie důležitých souborů uchovávejte alespoň v oddělených adresářích.
- Důkladně si promyslete, zda jste správně zadali „destruktivní“ příkazy, jako je `mv`, `rm` a `cp`. Zrovna tak si dejte pozor na přesměrování (>) souborů – i to může být nebezpečné a vyžaduje zvláštní pozornost. Nevinně vyhlížející opomenutí může způsobit katastrofu:

  ```
  /home/larry/report# cp report-1992 report-1993 backups
  ```

  Stačí vynechat poslední parametr a neštěstí je hotovo (místo zálohy máte zrušen jeden soubor):

  ```
  /home/larry/report# cp report-1992 report-1993
  ```

- Autor rovněž doporučuje na základě vlastních zkušeností neprovádět zálohování ani další úlohy údržby pozdě v noci, kdy jste unaveni a snadno uděláte chybu. Jestli o půl druhé v noci zjistíte, že máte příliš plný disk, pak jej nechte plným a nezačínejte hned rušit soubory – takovou práci si nechte až na ráno, kdy budete vyspaní a odpočatí.
- Používejte takovou výzvu příkazového řádku, která vás bude informovat o aktuálním adresáři. Pokud takovou výzvu nepoužíváte, rovněž hrozí nebezpečí. Následující příklad nám zaslal člen diskusní skupiny `comp.unix.admin`[1], který si nevšiml, že není v adresáři `/tmp`:

```
mousehouse> pwd
/etc
mousehouse> ls /tmp
passwd
mousehouse> rm passwd
```

  Série právě uvedených příkazů by vás mohla učinit velmi nešťastnými při pohledu na to, jak si rušíte soubor `passwd`, bez kterého se nemůže nikdo do operačního systému Unix přihlásit.

Chyba není ve vás

Naneštěstí pro programátory na celém světě platí, že ne všechny problémy pocházejí z chyb uživatelů. Operační systémy Unix a Linux jsou velmi komplikované a všechny známé verze mají chyby či skryté závady. Některé z těchto závad lze jen velmi těžko odstranit, protože se projevují jen za velmi speciálních okolností.

V souvislosti s chybami programového vybavení se používá termín „bug“ (doslova štěnice). Asi nejvýstižnější překlad bude „skrytá chyba“, protože se jedná o chybu, o které autoři programového vybavení neví. Chyba tohoto druhu se odhalí až při testování daného programu, někdy až po velmi dlouhé době.

Co dělat, když objevíte skrytou chybu

Když vám počítač dá chybnou odpověď (nejdříve důkladně prověřte, zda je odpověď opravdu chybná) nebo nějaký program zhavaruje, pak lze uvažovat o skryté chybě.

Skrytou chybu může obsahovat program, který stále běží, přesto že by měl skončit – i v tomto případě byste však měli nejdříve zkontrolovat, zda neprovádí příliš složité a zdlouhavé operace. Než použijete nový příkaz, důkladně si prostudujte příslušnou dokumentaci (nejlépe manuálové stránky).

Některá hlášení vám budou připadat jako skryté chyby, i když ve skutečnosti skrytými chybami nebudou. Prostudujte si oddíl Hlášení jádra systému a příslušnou dokumentaci, než učiníte nějaké závěry o skrytých chybách.

Například taková hlášení, jako „`disk full`“ nebo „`lp0 in fire`“ neznamenají, že je program vadný, ale že je něco v nepořádku s vaším technickým vybavením – nedostatek diskového prostoru nebo špatně připojená tiskárna.

Pokud nemůžete najít něco v dokumentaci, pak je to chybou dokumentace a měli byste autora programového vybavení kontaktovat a o nedostatku informovat. Zrovna tak je chybou dokumentace, když popisuje jiné vlastnosti programového vybavení, než jaké ve skutečnosti jsou.[2] Je-li něco nekompletního nebo nejasného v dokumentaci, pak se jedná o chybu dokumentace.

---

1 Jedná se o mezinárodní diskusní skupinu, která řeší otázky kolem správy operačního systému Unix.

Naopak, jestliže nejste schopni porazit šachový program `gnuchess`, pak je to tím, že algoritmus tohoto programu je opravdu dobrý a neznamená to nutně, že je ve vašem mozku skrytá chyba.

Jak ohlásit chybu

Jakmile jste si jisti, že jste odhalili skrytou chybu, je nutné zajistit, aby se hlášení o ní dostalo na správné místo. Pokuste se zjistit, co chybu způsobilo a pokuste se rekonstruovat všechny okolnosti, za jakých chyba nastala. Jestli se vám chybu nepodaří znovu „vyvolat" pročtěte si zprávy z diskusní skupiny `comp.os.linux.help` nebo `comp.unix.misc`. Také si důkladně pročtěte manuálové stránky k danému programu.

Při odesílání zpráv o skryté chybě se dává přednost elektronické poště. Pokud nemáte možnost odesílat zprávy prostřednictvím elektronické pošty, kontaktujte osobu, od které máte operační systém Linux nebo zkuste kontaktovat někoho, kdo přístup k elektronické poště má. Také se můžete obrátit na komerčního dodavatele operačního systému Linux. Ten má určitě zájem na tom, aby se skryté chyby rychle odstraňovaly. Mějte na paměti, že nikdo není povinen odstraňovat chyby, dokud s ním nemáte podepsanou příslušnou smlouvu.

Když odesíláte hlášení o chybě, pak uveďte všechny informace a okolnosti, za kterých se chyba projevila. Dodržujte dále uvedené zásady:

- Popište, co si myslíte, že má program dělat a co ve skutečnosti dělá. Například: „Program vrací hodnotu 5, když mu byl zadán výraz 2+2". Nebo „Program ohlásil `segment violation -- core dumped`". Je velmi důležité popsat, co se stalo, aby mohl autor chybu odstranit.
- Uveďte všechna nastavení proměnných prostředí.
- Uveďte verzi jádra operačního systému (najdete ji v souboru `/proc/version`) a verzi systémových knihoven (podívejte se do adresáře `/lib`, nebo pošlete výpis obsahu tohoto adresáře).
- Popište, jak jste program spustili, a popište, co jste dělali, když k chybě došlo.
- Uveďte všechny okolnosti, za kterých k chybě došlo. Řekněme například, že příkaz `w` některému uživateli nezobrazí informace o aktuálním procesu. Nestačí napsat: „Příkaz `w` nefunguje pro jistého uživatele". Chyba může nastat proto, že jméno uživatele má osm znaků nebo proto, že se přihlásil prostřednictvím sítě. Správná informace bude tedy znít takto: „Příkaz `w` nezobrazuje informace o aktuálním procesu uživateli greenfie, když se přihlásí prostřednictvím sítě.".
- Buďte zdvořilí. Většina lidí pracuje obětavě na volném programovém vybavení z vlastní iniciativy, vlastní dobré vůle a především proto, aby vám předložili program, který potřebujete. Nebuďte na lidi, kteří pracují od rána do noci, hrubí – právě hrubé osočování dokázalo odradit nejednoho nadějného programátora od práce na operačním systému Linux.

2 To bude asi také platit o tomto manuálu.

PŘÍLOHA A

Technické informace

Editor vi (vyslovuje se [ví áj]) je jediným editorem, který se nachází v každé instalaci operačního systému Unix. Původně byl vytvořen na univerzitě California v Berkeley, jeho různé verze se nacházejí ve všech distribucích operačního systému Unix a je také součástí distribuce operačního systému Linux. Editor vi se poměrně těžko učí, ale má mnoho velmi výkonných funkcí. Obecně se doporučuje začátečníkům editor Emacs, protože se mnohem snáze používá. Avšak uživatelé, kteří používají více počítačových platforem, raději pracují s editorem vi.

Abychom pochopili, proč klávesa K znamená posun kurzoru o jeden řádek nahoru a proč editor pracuje ve třech odlišných módech, musíme se podívat do historie. Pokud máte pokušení naučit se pracovat s editorem vi, pak můžete považovat tento dodatek za učebnici, která vás provede veškerými základy. Také zde předkládáme přehled příkazů, který můžete považovat za referenční příručku.

Dokonce i v případě, že editor vi nebude editorem pro vaši běžnou práci, není čas věnovaný jeho základům úplně zbytečný. Je téměř jisté, že v operačním systému typu Unix, který používáte, je editor vi instalován. Někdy je nezbytné použít editor vi při instalaci jiných programových produktů, například editoru Emacs. Spousta nástrojů operačního systému Unix, aplikací a her používá jistou podmnožinu příkazů editoru vi.

Stručná historie editoru vi

První textové editory byly orientovány na zpracování textových souborů řádek po řádku a používaly se na neinteligentních terminálech. Typickým editorem, který takto funguje, je editor **Ed**. Editor Ed je velmi výkonný a využívá velmi málo zdrojů počítače. V porovnání s editorem Ed nabízí editor vi uživateli vizuální alternativu s mnohem širší množinou příkazů.

Editor vi se startuje stejně jako řádkový editor ex. Platí, že editor ex je vlastně editor vi spuštěný ve speciálním módu. Vizuální složka editoru ex může být inicializována z příkazového řádku pomocí příkazu editoru vi nebo přímo z editoru ex.

Editor ex/vi byl vyvinut na univerzitě California v Berkeley a jeho autorem je William Joy. Původně byl dodáván jako nepodporovaný obslužný program. Oficiálně byl zařazen až v operačním systému System 5 Unix od firmy AT&T. Postupně se stal velmi populárním a dodnes konkuruje moderním celoobrazovkovým editorům.

V důsledku své popularity se editor vi objevil v mnoha verzích a dnes existují verze pro většinu operačních systémů (i jiných, než Unix). Cílem této kapitoly není popsat všechny dostupné příkazy editoru vi a jejich modifikace. Právě v důsledku toho, že vzniklo mnoho verzí editoru vi, není množina jeho příkazů standardizována. Řada klonů editoru vi dokonce nepodporuje některé původní příkazy.

Jestliže máte nějaké zkušenosti s editorem ed, pak se editor vi naučíte mnohem snáze. I když editor vi nehodláte používat, budou se vám základní znalosti hodit zejména v nouzových situacích.

Stručný výklad příkazů editoru Ed

Cílem tohoto oddílu je naučit vás pracovat s editorem ed. Byl navržen tak, aby se jej každý snadno naučil. Než jej budete moci používat, budete muset chvíli trénovat. Při probírání jednotlivých příkazů si hned vyzkoušejte uvedené příklady – tak se naučíte s editorem pracovat velmi rychle.

Vytvoření souboru

Editor ed je schopen editovat v daném okamžiku pouze jeden soubor. Vyzkoušejte si následující příklad a vytvořte si svůj první soubor prostřednictvím editoru ed.

```
/home/larry# ed

a

This is my first text file using Ed.
This is really fun.
.
w firstone.txt
q
/home/larry#
```

Nyní si můžete obsah souboru ověřit pomocí příkazu cat nebo more:

```
/home/larry# cat firstone.txt
```

Předcházející příklad ilustruje spoustu důležitých aspektů. Po spuštění editoru se objeví prázdný řádek. Klíč a je určen k přidání textu do souboru. Pokud chcete ukončit zadávání textu, zapište do prvního sloupce na novém řádku tečku. Chcete-li text uložit, pak zadejte příkaz w a jméno souboru. Samotný klíč q činnost editoru ukončí.

Nejdůležitější je poznatek, že editor pracuje ve dvou módech – v **textovém módu** a v **příkazovém módu**. Svoji činnost editor zahajuje v příkazovém módu, kdy lze na prázdných řádcích zadávat příkazy. Ty jsou definovány prostřednictvím jisté množiny klíčů.

Jak editovat existující soubor

Pokud chcete do existujícího souboru přidat řádek, postupujte dle následujícího příkladu:

```
/home/larry# ed firstone.txt

a

This is a new line of text.

q
```

Když zkontrolujete soubor firstone.txt pomocí příkazu cat, zjistíte, že nový řádek byl vložen mezi původní první a druhý řádek. Jak editor ed pozná, kam má vložit nový textový řádek?

Po načtení souboru si editor `ed` uchovává informaci o aktuálním řádku. Příkaz a vkládá nový text za aktuální řádek. V editoru `ed` také můžete vkládat nový řádek před aktuální řádek a to prostřednictvím příkazu `i`.

Nyní je patrné, že editor `ed` pracuje s textem řádek po řádku. Všechny příkazy mohou být aplikovány na specifikovaný řádek.

Dále uveďme příklad, jak přidat textový řádek na konec souboru:

```
/home/larry# ed firstone.txt
$a
The last line of text.
.
w
q
```

Modifikátor příkazu `$` „sdělí“ editoru `ed`, že má přidat řádek za poslední řádek stávajícího textu. Pokud byste chtěli přidat textový řádek za první řádek, použijte modifikátor `1`. Nyní máte k dispozici příkazy, pomocí kterých lze vkládat nový text před nebo za řádek specifikovaného čísla.

Jak se dozvíme, který řádek je aktuální? Stačí zadat příkaz `p` a zobrazí se obsah aktuálního řádku. Pokud chcete změnit aktuální řádek na hodnotu 2, pak postupujte dle následujícího příkladu:

```
/home/larry# ed firstone.txt
2p
q
```

Podrobnosti o číslování řádků

Ukázali jsme si, jak prostřednictvím příkazu `p` zobrazit obsah aktuálního řádku. Také víme, že pro příkazy existují modifikátory ve formě pořadových čísel řádků. Chceme-li vypsat obsah druhého řádku, stačí zadat příkaz:

2p

Existují také jiné speciální modifikátory, prostřednictvím kterých lze měnit nastavení aktuálního řádku. Znak dolaru `$` se používá pro poslední řádek textu. Chcete-li vypsat obsah posledního řádku, zadejte:

```
$p
```

Pro aktuální číslo řádku se používá speciální modifikátor tečka. Obsah aktuálního řádku prostřednictvím tohoto modifikátoru lze vypsat takto:

```
.p
```

Možná se vám takový příkaz zdá zbytečný. Když však často měníte obsah aktuálního řádku, pak zjistíte, jak je příkaz užitečný.

Pokud chcete zobrazit text v rozsahu řádků od 1 do 2, pak musíte specifikovat rozsah:

```
1,2p
```

První číslo odkazuje na počáteční řádek, jenž se má zobrazit, a druhé číslo odkazuje na poslední řádek, jenž se má zobrazit. Aktuální řádek se po provedení tohoto příkazu nastaví na druhou hodnotu.

Další příklad ukazuje, jak zobrazit obsah souboru od prvního řádku po aktuální řádek:

```
1,.p
```

Také můžete zobrazit obsah souboru od aktuálního řádku po řádek poslední:

```
.,$p
```

Nyní budete jistě umět zadat příkaz, který zobrazí obsah celého souboru.

Dále si ukážeme, jak zrušit první dva řádky souboru:

```
1,2d
```

Příkaz `d` ruší text řádek po řádku. Chcete-li zrušit celý text, stačí zadat:

```
1,$d
```

Pokud provedete v editovaném textu velké množství změn a nechcete obsah souboru uložit, pak je nejlepší editor ukončit bez zápisu.

Většina uživatelů nepoužívá editor `ed` jako svůj hlavní editor. Moderní editory jsou celoobrazovkové a nabízejí mnohem pružnější množinu příkazů. Editor `ed` představuje dobrý úvod k editoru `vi` a pomůže vám pochopit, odkud příkazy editoru `vi` pocházejí.

Stručný výklad příkazů editoru vi

Cílem tohoto oddílu je naučit vás pracovat s editorem `vi`. Předpokládáme, že nemáte s editorem `vi` žádné zkušenosti a probereme si deset nejzákladnějších příkazů. Tyto příkazy vám budou stačit k realizaci nejnutnějších kroků při editování souboru a další příkazy se pak snadno naučíte podle svých potřeb. Při probírání jednotlivých příkazů si hned vyzkoušejte uvedené příklady – tak se naučíte s editorem pracovat velmi rychle.

Jak spustit editor vi

Chcete-li spustit editor `vi`, zadejte jméno programu a jako parametr uveďte jméno editovaného textového souboru. Objeví se vám obrazovka, na jejíž levé straně bude zobrazen sloupec znaků tilda (~). Editor `vi` se nyní nachází v příkazovém módu – cokoliv napíšete, bude považováno za příkaz a nikoliv za vstupní text. Jestliže chcete zadávat text, musíte nejdříve zadat příkazy. Pro vkládání textu existují tyto dva základní příkazy:

- `i` vložení textu vlevo od kurzoru
- `a` přidání textu vpravo od kurzoru.

Protože se v tomto okamžiku nacházíte na začátku prázdného souboru, nezáleží na tom, který z uvedených příkazů použijete. Zadejte tedy jeden z nich a zapište následující text. (Jedná se o báseň, jejímž autorem je Augustus DeMorgan. Je uvedena v manuálu *„The Unix Programming Environment“,* který napsali pan B.W. Kernighan a R. Pike.):

```
Great fleas have little fleas<Enter>
  upon their backs to bite 'em,<Enter>
And little fleas have lesser fleas<Enter>
  and so and infinitum.<Enter>
And great fleas themselves, in turn,<Enter>
  have greater fleas to go on;<Enter>
While these again have greater still,<Enter>
  and greater still, and so on.<Enter>
<Esc>
```

Všimněte si, že k ukončení vkládaného textu se používá klíč Esc. Pak editor opět přejde do příkazového módu.

Příkazy pro pohyb kurzoru

- `h` posunutí kurzoru o jeden znak doleva
- `j` posunutí kurzoru o jeden řádek dolů
- `k` posunutí kurzoru o jeden řádek nahoru
- `l` posunutí kurzoru o jeden znak doprava

Uvedené příkazy mohou být opakovány tak, že danou klávesu budete držet stisknutou. Nyní si vyzkoušejte pohyb kurzoru všemi směry. Pokud se pokusíte posunout kurzor na neexistující pozici (například když stisknete klávesu K a přitom je kurzor na prvním řádku souboru), pak obrazovka blikne nebo se ozve zvukový signál. Ničeho se neobávejte, váš soubor nebude v takovém případě nijak poškozen.

Jak rušit text

- `x` zrušení znaku na pozici kurzoru
- `dd` zrušení řádku

Posuňte kurzor na druhý řádek a nastavte jeho pozici pod apostrof ve slově `'em`. Stiskněte klávesu X a apostrof zmizí. Nyní stiskněte klávesu I (přepnete tak editor `vi` do módu, ve kterém se vkládá text) a zapište „`th`“. Nakonec stiskněte klávesu Esc.

Uložení souboru

- `:w` uložení souboru (na disk)
- `:q` ukončení editoru `vi`

Nejdříve se přesvědčte, že jste v příkazovém módu – stiskněte klávesu Esc. Nyní zadejte `:w`. V tomto okamžiku se vámi vytvořený text uloží do diskového souboru.

Příkaz pro ukončeni editoru `vi` je reprezentován klíčem `:q`. Jestliže chcete zkombinovat uložení souboru a ukončení editoru, pak použijte příkaz `:wq`. Pro příkaz `:wq` existuje ekvivalentní zkratka `ZZ`. Zkratka `ZZ` se bude hodit zejména programátorům. Podívejme se na typickou posloupnost akcí, které provádí programátor při ladění programu: spuštění programu; zjištění problému; editování souboru obsahujícího zdrojový kód programu; provedení změn v souboru a jeho uložení na disk; přeložení programu; spuštění programu; ...a tak stále dokola. Pak bude programátor příkaz `ZZ` zřejmě používat velmi často. Ve skutečnosti není příkaz `ZZ` přesným ekvivalentem příkazu `:wq`. Příkaz `ZZ` totiž neprovede uložení souboru, pokud v textu neprovedete žádné změny, zatímco příkaz `:wq` soubor před ukončením editoru soubor vždy uloží.

Jestliže jste udělali nějaké změny, ale pak nechcete soubor ukládat, použijte příkaz `:q!` (nezapomeňte předtím stisknout klávesu Esc). Pokud byste zapomněli znak „!“ editor `vi` vám nepovolí ukončit práci bez uložení souboru.

Co bude následovat

Deset příkazů, které jsme v předcházejících odstavcích probrali, stačí k tomu, abyste byli schopni pracovat s editorem vi. Tyto příkazy však představují pouze základ práce s editorem. Existují další příkazy, například pro kopírování textu z jednoho místa na druhé, pro přesouvání textu z jednoho souboru do druhého, pro konfiguraci editoru a podobně. V editoru vi lze aplikovat asi 150 příkazů.

Pokročilejší techniky práce s editorem vi

Velká výhoda editoru `vi` spočívá v tom, že jej můžete používat, i když znáte jen několik málo základních příkazů. Většina uživatelů je nadšena, že stačí znát jen pár příkazů, ale jakmile s editorem pracují déle, potřebují k realizaci některých úloh další příkazy.

V následujícím oddílu se předpokládá, že se uživatel seznámil se základními příkazy uvedenými v předcházejícím oddílu. Nyní si probereme další příkazy – od kopírování textů až po definici maker.

Také uvedeme oddíl o konfiguraci editoru `vi`, kde se dozvíte, jak nastavit některé vlastnosti editoru, aby byla vaše práce co nejefektivnější. Následující odstavce jsou zaměřeny spíše na popis příkazů a nejsou již tolik zaměřeny na jejich procvičování. Proto doporučujeme, abyste si probírané příkazy průběžně procvičovali sami.

Nebudeme uvádět zcela vyčerpávající seznamy příkazů editoru `vi`, ale zaměříme se na příkazy nejčastěji používané zejména z praktického hlediska. I když si nakonec zvolíte pro svou běžnou práci jiný editor, budou se vám nabyté znalosti vždy hodit.

Příkazy pro změnu polohy kurzoru

K nejzákladnějším funkcím editoru patří příkazy pro změnu polohy kurzoru. Zde uvádíme jejich přehled.

h	posunutí kurzoru o jeden znak doleva
j	posunutí kurzoru o jeden řádek dolů
k	posunutí kurzoru o jeden řádek nahoru
l	posunutí kurzoru o jeden znak doprava

Některé implementace editoru vi také umožňují používat kurzorové klávesy.

w	posunutí kurzoru na začátek následujícího slova
e	posunutí kurzoru na konec následujícího slova
E	posunutí kurzoru na konec následujícího slova před mezeru
b	posunutí kurzoru na začátek předcházejícího slova
0	posunutí kurzoru na začátek řádku
^	posunutí kurzoru na první slovo v aktuálním řádku
$	posunutí kurzoru na konec řádku
<CR>	posunutí kurzoru na začátek následujícího řádku
-	posunutí kurzoru na začátek předcházejícího řádku
G	posunutí kurzoru na konec souboru
1G	posunutí kurzoru na začátek souboru
nG	posunutí kurzoru na řádek s pořadovým číslem n
Ctrl-Shift-G	zobrazení čísla aktuálního řádku
%	posunutí kurzoru na odpovídající hranatou závorku
H	posunutí kurzoru na první řádek obrazovky
M	posunutí kurzoru na prostřední řádek obrazovky
L	posunutí kurzoru na spodní řádek obrazovky
nl	posunutí kurzoru na sloupec n

Jestliže kurzor dospěje na spodní nebo horní řádek obrazovky, pak se zobrazený text vždy příslušným směrem posune. Nyní uvedeme alternativní příkazy, které umožňují posouvání zobrazovaného textu na obrazovce (tzv. rolování).

Ctrl-f	posunutí textu o stránku nahoru
Ctrl-b	posunutí textu o stránku dolů
Ctrl-d	posunutí textu o půl stránky dolů
Ctrl-u	posunutí textu o půl stránky nahoru

Právě uvedené příkazy jsou určeny k pohybu kurzoru. Některé z nich mohou být modifikovány pomocí čísla, které se uvede před příkaz. Předřazené číslo n znamená, že se příkaz bude opakovat n-krát.

Uveďme si tvar příkazu pro posunutí kurzoru o n pozic doleva:

nl	posunutí kurzoru o n pozic (znaků) doprava

Jestliže chcete například zařadit před text větší počet mezer, můžete použít podobnou modifikaci příkazu i pro vkládání textu. Zadejte počet mezer, pak příkaz i následovaný mezerou a nakonec stiskněte klávesu ESC.

ni opakované vložení textu n-krát

Příkazy odkazující na řádky mohou jako modifikátor použít číslo řádku. Ukázkovým příkladem je příkaz G:

1G posunutí kurzoru na první řádek textového souboru

Editor `vi` má velké množství příkazů, které lze použít ke změně pozice kurzoru – od jednoduchých příkazů, kde se kurzor posune o jednu pozici, až po složitější příkazy, kdy se kurzor nastavuje na předem specifikovanou pozici. Také lze zadat nastavení kurzoru na specifikovaný řádek při spouštění editoru, například:

```
vi +10 myfile.tex
```

Uvedený příkaz otevře soubor myfile.tex a umístí kurzor na desátý řádek od začátku souboru.

Vyzkoušejte si příkazy uvedené v tomto oddíle a procvičte se v jejich používání. Většina uživatelů používá pouze jistou podmnožinu uvedených příkazů. V dalším oddílu si probereme příkazy umožňující text měnit.

Modifikace textu

Nyní je naším cílem měnit obsah textového souboru a editor vi za tímto účelem nabízí spoustu příkazů.

V tomto oddíle budeme probírat příkazy určené k editování textu, rušení textu a podobně. Až oddíl dočtete, budete mít dostatek znalostí potřebných k vytvoření jakéhokoliv textového souboru. Následující oddíly jsou pak zaměřeny na další užitečné příkazy.

Při vkládání textu lze vložit více textových řádků prostřednictvím klávesy Enter. Předpokládejme, že jste udělali chybu a že jste stále na řádku, ve kterém se chyba vyskytuje.

Pak můžete použít klávesu Backspace a vrátit se k místu, kde se nachází chyba. Pokud jde o klávesu Backspace, různé implementace editoru vi se chovají různě. Některé z nich pouze posunují kurzor zpět a zapsaný text nechávají nedotčený. Jiné text postupně zprava ruší. Některé implementace dokonce umožňují používat v módu zadávání textu kurzorové klávesy. To vše však nepatří k normálnímu chování editoru `vi`. Pokud je text viditelný a použijete klávesu Esc (přitom jste na řádku, ve kterém jste použili klávesu Backspace), pak budou znaky za kurzorem zrušeny. Funkci klávesy Backspace si budete muset vyzkoušet, abyste zjistili, jak se tato klávesa ve vaší konkrétní implementaci editoru vi chová.

a	Přidání textu od aktuální pozice kurzoru
A	Přidání textu na konec řádku
i	Vložení textu vlevo od pozice kurzoru
I	Vložení textu vlevo od prvního výskytu zobrazitelného znaku (non-white character) v aktuálním řádku
o	Otevření nového řádku a vložení textu za aktuální řádek
O	Otevření nového řádku a vložení textu před aktuální řádek

Nyní máme několik příkazů pro vkládání textu a dále si uvedeme příkazy pro zrušení textu. Editor `vi` má několik příkazů pro zrušení textu, které mohou být spojeny s modifikátorem.

x	zrušení znaku, na jehož pozici je umístěn kurzor
dw	zrušení textu od aktuální pozice kurzoru do konce slova
dd	zrušení aktuálního řádku
D	zrušení textu od aktuální pozice kurzoru do konce řádku

Prostřednictvím modifikátorů se síla příkazů zvýší. Následující příklady jsou pouze podmnožinou všech možností:

nx	n-krát opakované zrušení znaku, na jehož pozici je umístěn kurzor
ndd	zrušení n řádků
dnw	zrušení n slov (stejnou funkci má příkaz ndw)
dG	zrušení textu od aktuální pozice kurzoru do konce souboru
d1G	zrušení textu od aktuální pozice kurzoru do začátku souboru
d$	zrušení textu od aktuální pozice kurzoru do konce řádku
dn$	zrušení textu od aktuální pozice kurzoru do konce n-tého řádku

Uvedené příklady ukazují, že operace zrušení textu mohou být velmi efektivní. Je to zejména evidentní tehdy, když tyto operace použijete ve spojení s příkazy pro změnu polohy kurzoru. Poznamenejme, že příkaz `D` tvoří výjimku, protože ignoruje jakékoli modifikátory.

Příležitostně nastanou situace, kdy budete chtít provedené změny vrátit zpět. Následující příkazy jsou určeny pro obnovení textu:

u	zrušení posledně provedeného příkazu
U	zrušení všech změn provedených v tomto řádku
:e!	obnova stavu souboru od okamžiku, kdy byl naposledy uložen

Editor `vi` umožňuje nejen vracet zpět provedené změny, ale také vracet zpět vrácené změny. Například pomocí příkazu `5dd` zrušíte pět řádků a pak je obnovíte pomocí příkazu `u`. Když použijete příkaz `u` znovu, budou řádky opět zrušeny.

Nové verze editoru `vi` nabízejí příkazy, které umožňují editovat text v tzv. přepisovacím módu. Znamená to, že chcete-li měnit nějaký text, pak jej nemusíte nejdříve rušit.

r c	přepsání znaku na pozici kurzoru znakem
R	přepsání textu novým textem
c w	změna textu v aktuálním slově
c $	změna textu od aktuální pozice kurzoru do konce řádku
c n w	změna následujících n slov (totéž jako ncw)
c n $	změna do konce n-tého řádku
c	změna do konce řádku (totéž jako c$)
c c	změna aktuálního řádku
S	v textu, který právě píšete, nahradí aktuální znak
n s	v textu, který právě píšete, nahradí n aktuálních znaků

Série příkazů realizujících změny vám umožní zadat řetězec znaků, který musí být ukončen klávesou Esc.

Příkaz c w platí od aktuální pozice kurzoru ve slově do konce slova. Když použijete příkaz realizující změnu a přitom zadáte „vzdálenost“, editor `vi` zobrazí znak `$` na pozici, do které bude změna provedena. Nový text může být delší nebo kratší než text původní.

Kopírování a přesouvání částí textu

Pro přesouvání textu existuje řada příkazů, jejichž kombinací lze dosáhnout kýženého efektu. V tomto oddíle popíšeme pojmenované a nepojmenované buffery spolu s příkazy umožňujícími „vystřihnout“ a „nalepit“ text.

Základní kopírování textu probíhá ve třech krocích:

1. Kopírování textu do bufferu („**vystřižení**“ textu).
2. **Nastavení kurzoru** na cílovou pozici.
3. Kopírování textu z bufferu na novou pozici („**nalepení**“ textu).

K vystřižení textu do nepojmenovaného bufferu použijte některý z následujících příkazů:

y y	Přesunutí kopie aktuálního řádku do nepojmenovaného bufferu.
Y	Přesunutí kopie aktuálního řádku do nepojmenovaného bufferu.
c y y	Přesunutí kopie následujících n řádků do nepojmenovaného bufferu.
c Y	Přesunutí kopie následujících n řádků do nepojmenovaného bufferu.
y w	Přesunutí slova do nepojmenovaného bufferu.
y n w	Přesunutí n slov do nepojmenovaného bufferu.
n y w	Přesunutí n slov do nepojmenovaného bufferu.
y $	Přesunutí textu od aktuální pozice do konce řádku.

Nepojmenovaný buffer je dočasná oblast paměti, jejíž obsah může být zrušen prostřednictvím jiných často používaných příkazů. Někdy se stane, že jistou část textu budete potřebovat po dlouhou dobu. V takovém případě byste měli použít pojmenovaný buffer. Editor `vi` nabízí 26 pojmenovaných bufferů. Jako identifikační jméno bufferu se používá jednoznakové písmeno. Pro rozlišení pojmenovaného bufferu editor `vi` používá znak `"`. Při odkazu na pojmenovaný buffer se používají malá písmena (pokud má být obsah bufferu zcela nahrazen) nebo velká písmena (pokud má být obsah bufferu doplněn). Uveďme si příklady:

" a y y	Přesunutí aktuálního řádku do pojmenovaného bufferu a.
" a Y	Přesunutí aktuálního řádku do pojmenovaného bufferu a.
" b y w	Přesunutí aktuálního slova do pojmenovaného bufferu b.
" B y w	Přidání aktuálního slova k obsahu pojmenovaného bufferu b.
" b y 3 w	Přesunutí následujících tří slov do pojmenovaného bufferu b.

Ke zkopírování („nalepení“) obsahu bufferu použijte příkaz `p`:

p	Zkopírování obsahu nepojmenovaného bufferu VPRAVO od pozice kurzoru.
P	Zkopírování obsahu nepojmenovaného bufferu VLEVO od pozice kurzoru.
n P	Zkopírování obsahu nepojmenovaného bufferu n-krát VLEVO od pozice kurzoru.
" a p	Zkopírování obsahu pojmenovaného bufferu a VPRAVO od pozice kurzoru.
" b 3 P	Zkopírování obsahu pojmenovaného bufferu b třikrát VLEVO od pozice kurzoru.

Jestliže používáte editor `vi` v terminálovém okně systému X Window, pak máte k dispozici ještě jeden prostředek pro kopírování textů. Oblast textu, kterou chcete kopírovat, vyznačte kurzorem myši (levé tlačítko myši přitom držte stisknuté). Vyznačená oblast textu bude zobrazena inverzně a přitom se automaticky přenese do speciálního bufferu vyhrazeného pro systém X Window. Po-

kud chcete text „nalepit“, stačí stisknout prostřední tlačítko myši. Nezapomeňte, že předtím musíte editor vi přepnout do vkládacího módu, protože jinak by mohl být vstup interpretován jako příkaz a výsledek by byl nepředvídatelný. Prostřednictvím stejné techniky lze kopírovat jednotlivá slova. Slovo se přenese do bufferu dvojnásobným klepnutím levým tlačítkem myši. Obsah bufferu se změní pouze tehdy, když se vyznačí nová oblast textu.

Přesouvání textu se rovněž odehrává ve třech krocích:

1. **Zrušení textu** a jeho současné zkopírování do pojmenovaného nebo nepojmenovaného bufferu.
2. **Nastavení kurzoru** na cílovou pozici.
3. **Kopírování textu** z pojmenovaného nebo nepojmenovaného bufferu na novou pozici („nalepení“ textu).

Proces je stejný jako kopírování textu, avšak v prvním kroku se text zruší. Když se provede příkaz `dd`, přesune se zrušený řádek do nepojmenovaného bufferu. Pak můžete obsah bufferu „nalepit“ stejným způsobem, jako při kopírování textu.

" a d d	zrušení řádku a přesunutí do pojmenovaného bufferu a
" a A d d	zrušení čtyř řádků a přesunutí do pojmenovaného bufferu a
d w	zrušení slova a přesunutí do nepojmenovaného bufferuDalší příklady na zrušení textu jsme uvedli v oddíle o modifikaci textu

V případě, že systém zhavaruje, obsahy pojmenovaného i nepojmenovaných bufferů se ztratí. Obsah editovaného bufferu však může být obnoven (viz oddíl „Užitečné příkazy“).

Vyhledávání a nahrazování textů

Editor `vi` má spoustu příkazů pro vyhledávání řetězců. Můžete vyhledávat jednotlivé znaky, ale také můžete při vyhledávání použít regulární výrazy.

Hlavní vyhledávací příkazy jsou `f` a `t`:

f c	vyhledání následujícího znaku c vpravo od kurzoru
F c	vyhledání následujícího znaku c vlevo od kurzoru
t c	přesunutí kurzoru na pozici vlevo od následujícího znaku c
T c	Přesunutí kurzoru na pozici vpravo od předcházejícího znaku c (V některých verzích editoru vi je tento příkaz shodný s příkazem Fc.)
;	opakování posledního příkazu f, F, t nebo T
,	stejný příkaz jako ;, ale mění směr vyhledávání původního příkazu

Jestliže hledaný znak neexistuje, upozorní vás editor `vi` zvukovým nebo jiným signálem.

Editor `vi` rovněž umožňuje v editovaném textu hledat řetězec:

/`str`	vyhledání specifikovaného řetězce od aktuální pozice kurzoru
?`str`	vyhledání specifikovaného řetězce před aktuální pozicí kurzoru
n	opakování předcházejícího příkazu / nebo ?
N	opakování předcházejícího příkazu / nebo ?, přičemž se změní směr vyhledávání

Když použijete příkaz / nebo ?, „vyčistí“ se spodní řádek obrazovky. Pak v tomto řádku zadáte hledaný řetězec a stisknete klávesu Enter.

Řetězec uvedený za příkazy / a ? může být regulárním výrazem. V regulárním výrazu se prostřednictvím pseudoznaků popisují jisté množiny znaků. Pseudoznaky používané v editoru `vi` jsou následující: `.`, `*`, `[`, `]`, `^` a `$`. Následující seznam specifikuje význam pseudoznaků:

`.`	vyhovuje jakýkoliv znak kromě znaku „newline“
`\`	uvozuje jakýkoliv speciální znak
`*`	vyhovuje výskyt předcházejících znaků
`[]`	vyhovuje právě jeden znak uvedený v hranatých závorkách
`^`	vyhovuje právě jeden znak, jen pokud se nachází na začátku řádku
`$`	vyhovuje znak předcházející konci řádku
`[^]`	vyhovují znaky, které nejsou uvedeny v hranatých závorkách
`[-]`	Vyhovují znaky z uvedeného rozsahu

Nejjednodušší způsob, jak se naučíte používat regulární výrazy, spočívá v tom, že je budete používat. Vyzkoušejte si následující příklady:

`c.pe`	vyhovují slova jako cope, cape, caper a podobně
`c\.pe`	vyhovují slova jako c.pe, c.per a podobně
`sto*p`	vyhovují slova jako stp, stop, stoop a podobně
`cat.*n`	vyhovují slova jako carton, cartoon a podobně
`xyz.*`	vyhovují xyz do konce řádku
`^The`	vyhovují všechny řádky začínající na The

`atime$`	vyhovují všechny řádky končící na atime
`^Only$`	vyhovují všechny řádky, ve kterých se vyskytuje pouze slovo Only
`b[aou]rn`	vyhovují slova barn, born nebo burn
`Ver[D-F]`	vyhovují slova VerD, VerE nebo VerF
`Ver[^1-9]`	vyhovují slova začínající na Ver, po kterých nenásleduje žádná číslice
`the[ir][re]`	vyhovují slova their, therr, there a theie
`[A-Za-z][A-Za-z]*`	vyhovuje jakékoliv slovo

Editor vi používá příkazový mód `ex` k realizaci operací vyhledávání a náhrady řetězců. Všechny příkazy začínající dvojtečkou představují požadavek v módu `ex`.

Příkazy pro vyhledání a náhradu řetězce umožňují, aby byly realizovány v jistém řádkovém rozsahu. Uživatel může vyžadovat, aby byl před náhradou řetězce vyzván k potvrzení, zda se má konkrétní výskyt řetězce nahrazovat nebo ne. Uveďme si obecný tvar příkazu pro náhradu řetězce a několik příkladů:

...syntaxe obecného příkazu :<start>,<finisf>s/<find>/<replace>/g

:1,$s/the/The/g	vyhledá všechny výskyty slova the a nahradí je slovem The
:%s/the/The/g	znak % znamená „celý soubor“. Tento příkaz provede stejnou funkci jako předcházející příkaz
:.,5s/^.*//g	zruší obsah editovaného souboru od aktuálního řádku po pátý řádek
:%s/the/The/gc	nahradí všechny výskyty slova the slovem The a před každou náhradou si vyžádá potvrzení
:%s/^..../g	zruší první čtyři znaky každého řádku

Jak je z předcházejících příkladů patrné, jsou příkazy pro vyhledávání a náhradu řetězců velmi výkonné, zejména když se kombinují s regulárními výrazy. Pokud se direktiva `g` v příkazu neuvede, provede se náhrada pouze u prvního výskytu hledaného řetězce.

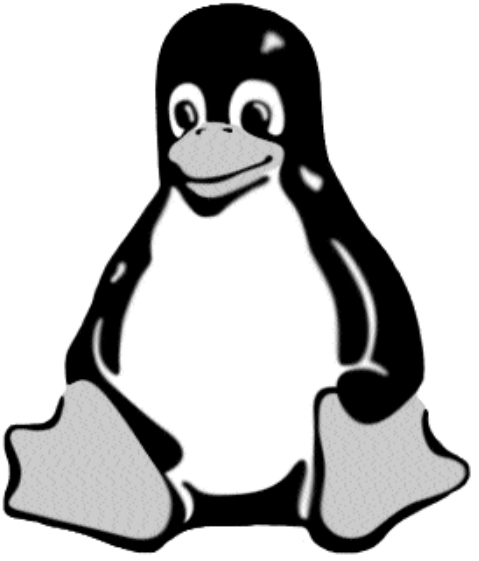
Někdy se může stát, že původní vyhledávaný řetězec se má použít v nahrazujícím řetězci. Za tímto účelem nabízí editor `vi` některé speciální znaky:

:1,5s/help/&ing/g	v prvních pěti řádcích nahradí slovo help slovem helping
:%s/ */&&/g	v celém souboru zdvojnásobí počet mezer mezi slovy

Používání kompletního řetězce má v editoru `vi` jisté omezení, protože k stanovení rozsahu náhrady editor používá kulaté závorky `( a )`. V takovém případě je nutno použít znak `\`:

:s/^\(.*\):.*/\1/g	zruší vše, co následuje po dvojtečce včetně dvojtečky samé
:s/\(.*\):\(.*\)/\2:\1/g	vymění slova na obou stranách dvojtečky

Posledně uvedené příklady asi budete muset číst velmi pozorně. Editor `vi` nabízí velmi výkonné příkazy, které moderní editory zpravidla nemají. Za to se ovšem platí jistá cena – naučit se všechny příkazy editoru vi není vůbec snadné. Pokud jsme vás neodradili, pokuste se znovu si projít uvedené příklady a uvidíte, že se používání editoru `vi` pro vás stane zcela přirozenou záležitostí.



ČÁST II

Příručka správce operačního systému

Zdrojová podoba a předformátované verze

Zdrojový kód této knihy a další počítačově čitelné formáty můžete získat na Internetu na stránkách Linux Documentation Project na adrese *http://linuxdoc.org/LDP*, nebo na domovské stránce této knihy na adrese *http://www.linuxdoc.org/LDP/nag2*. K dispozici jsou minimálně verze PostScript a DVI.

Českou verzi této části naleznete na adrese *http://www.cpress.cz/knihy/ldp2/ldp-castII.pdf*

KAPITOLA 1

Úvod

Na počátku byl soubor nesličný a pustý, a prázdno se vznášelo nad povrchem bitů. A Ruka Autorova dosedla na povrch klávesnice i řekl Autor: Budiž slova! I byla slova.

Příručka správce operačního systému Linux popisuje ty aspekty používání operačního systému, jež se vztahují k jeho správě neboli administraci. Je určený lidem, kteří o správě operačního systému neví zhola nic (ptají se teď, co to je), avšak zvládají přinejmenším základy jeho běžného užívání. Nenaleznete zde návod, jak Linux instalovat. Instalace systému je podrobně popsaná v dokumentu „Průvodce instalací a začátky". Bližší informace o sadě manuálů systému Linux jsou uvedeny níže.

Administrací (správou systému) rozumíme všechny činnosti, které je nutno pravidelně vykonávat, aby počítačový systém zůstal v provozuschopném stavu. Zahrnuje například zálohování souborů (a v případě potřeby jejich obnovování), instalaci nových programů, vytváření uživatelských účtů (a jejich mazání v případě, že jsou nepotřebné), kontroly a opravy případných poškození systému souborů a další. Když si počítač představíte jako dům, pak by správou systému byla jeho údržba. Ta by zahrnovala například úklid, zasklívání rozbitých oken a další podobné věci. Místo prostého pojmu „údržba systému" se používá termín „administrace", aby to nevypadalo na první pohled zas až tak jednoduše[1].

Příručka je strukturovaná tak, že většinu kapitol můžete číst nezávisle na sobě. Když například hledáte nějaké informace o zálohování, stačí, když si přečtete příslušnou kapitolu. Doufáme, že díky tomu bude možné používat tuto knihu i jako referenční příručku a v případě potřeby místo „louskání" celého textu číst pouze jeho menší části. Přesto manuál zůstává především a převážně učebnicí a jakýmsi průvodcem. To, že poslouží i jako referenční příručka je pouze šťastným řízením osudu.

Nelze předpokládat, že by tato knížka pokryla celou problematiku administrace systému. Správce systému bude potřebovat řadu další dokumentace operačního systému Linux. Koneckonců, administrátor je v podstatě jenom uživatel, který má zvláštní práva a povinnosti. Velmi významným pramenem jsou manuálové stránky, po kterých by správce měl sáhnout pokaždé, když si není funkcí některého příkazu zcela jist.

Manuál je zaměřen především na operační systém Linux, ale v obecných principech může být užitečný i pro správce jiných unixových systémů. Bohužel je mezi různými verzemi Unixu tolik rozdílů (o správě systému to platí dvojnásob), že není prakticky možné postihnout všechny známé varianty. Je totiž obtížné – vezmeme-li v potaz způsob, jakým se Linux vyvíjí – pokrýt i všechny možnosti jen tohoto operačního systému.

Neexistuje jediná oficiální distribuce Linuxu od jediného výrobce. Různí lidé používají různá nastavení a konfigurace. Navíc si řada uživatelů vytváří své vlastní. Proto tato kniha není zaměřená

1 Jsou někteří, kteří termín „údržba" používají, ale to jenom proto, že nečetli tuto příručku. Chudáci.

na některou z konkrétních distribucí (i když autor osobně dává téměř výlučně přednost systému Debian GNU/Linux). V rámci možností se v příručce snažíme upozornit na některé odlišnosti a objasnit i jiné možné alternativy.

Než abychom podali strohý seznam „pěti jednoduchých kroků“ pro řešení každého úkolu, dáváme přednost popisu základních principů, tedy objasnění toho, jak věci doopravdy fungují. V knize proto najdete hodně informací, které nejsou nezbytné pro každého. Takovéto části jsou v textu označeny a v případě, že používáte systém s předem nastavenou konfigurací, můžete je klidně přeskočit. Pochopitelně, přečtete-li si knihu celou, proniknete do systému hlouběji, a pak by pro vás mohly být o něco příjemnější i jeho používání a jeho správa.

Tak jako vše ostatní spojené s vývojem Linuxu, byla i tato práce založena na principu dobrovolnosti. Pustili jsme se do ní, protože jsme si mysleli, že by to mohla být zábava. Dalším důvodem byl pocit, že je potřeba tuto práci udělat. Přesto – jako konečně u každé dobrovolné práce – jsou určité hranice nasazení a úsilí, které můžete vynaložit. Navíc vás omezuje také to, kolik vědomostí a zkušeností máte. Přirozeně, manuál není tak dobrý, jak by mohl být v případě, že by přišel někdo s kouzelnou hůlkou a dobře zaplatil za jeho napsání. Pak by bylo možné strávit i několik dalších let jeho zdokonalováním. Samozřejmě si myslíme, že je celkem povedený, nicméně berte to jako varování.

Je jeden konkrétní bod, ve kterém jsme manuál dost „ořezali“ – není v něm vyčerpávajícím způsobem popsaná řada věcí, které již jsou podrobně zdokumentované v jiných volně dostupných příručkách. Vztahuje se to zvlášť na dokumentaci k jednotlivým programům. Neuvádíme například všechny podrobnosti použití programu **mkfs**. Popisujeme jenom funkci programu a pouze tolik z jeho dalších možností, kolik je potřeba pro dosažení účelu této knihy. Laskavého čtenáře, jenž hledá podrobnější informace, odkazujeme na onu další dokumentaci. Převážná většina dokumentů, na které se odvoláváme v odkazech, je součástí úplné sady dokumentace k operačnímu systému Linux.

Lars se snažil o napsání co nejlepší příručky a já jako současný správce bych rád v jeho snaze pokračoval.

Budeme vděčni za všechny vaše nápady jak ji vylepšit. Gramatické a věcné chyby, nápady týkající se nových oblastí, o které by bylo možno knihu rozšířit, opakující se části, informace o rozdílech mezi různými verzemi Unixu – to všechno jsou připomínky, které se zájmem očekáváme. Kontaktní informace najdete prostřednictvím služby World Wide Web na adrese *http://www.iki.fi/viu/* .

Při práci na této knize nám přímo či nepřímo pomáhalo mnoho lidí. Rádi bychom zvlášť poděkovali Mattu Welshovi za inspiraci a vedení projektu LDP; Andy Oramovi za to, že nás znovu a znovu zaměstnával řadou velmi podnětných připomínek; Olafu Kirschovi za to, že nám dokázal, že vše lze zvládnout; Adamu Richterovi z Yggdrasil a dalším za to, že nám ukázali, že tato práce může být zajímavá i pro jiné lidi.

Stephen Tweedie, H. Peter Anvin, Rémy Card a Theodore Ts'o odvedli kus práce, kterou jsme si formou odkazů a referencí „zapůjčili“ (tím pádem je naše kniha na pohled tenčí a o to víc působivá) – sem patří porovnání souborových systémů xia a ext2, seznam zařízení nebo popis souborového systému ext2. Tyto části jsme z knihy vyřadili. Za toto jsme vděční vůbec nejvíc a zároveň se velmi omlouváme za předchozí verze manuálu, které občas v některých oblastech postrádaly odpovídající úroveň.

Kromě toho patří náš dík Marku Komarinskemu za jeho materiály z roku 1993 i mnoho dalších sloupků v Linux Journalu, jež se týkaly problematiky správy systému. Jsou velmi informativní a inspirující.

Dostali jsme množství užitečných připomínek od velkého počtu dalších lidí. Díky malé černé díře v našem archivu nelze dohledat všechna jména, takže alespoň některá z nich (v abecedním pořadí): Paul Caprioli, Ales Cepek, Marie-France Declerfayt, Dave Dobson, Olaf Flebbe, Helmut Geyer, Larry Greenfield a jeho otec, Stephen Harris, Jyrki Havia, Jim Haynes, York Lam, Timothy Andrew Lister, Jim Lynch, Michael J. Micek, Jacob Navia, Dan Poirier, Daniel Quinlan, Jouni K Seppänen, Philippe Steindl, G. B. Stotte. Omlouváme se všem, na které jsme zapomněli.

Linux – dokumentační projekt

Projekt tvorby dokumentace pro operační systém Linux (anglicky The Linux Documentation Project, zkráceně LDP) je volné sdružení autorů, korektorů a editorů, kteří spolupracují na úplné dokumentaci systému. Hlavním koordinátorem projektu je Greg Hankins.

Tento manuál je jedním ze sady dokumentů, které jsou v rámci LDP šířeny. Tato sada obsahuje Příručku uživatele systému, Příručku správce systému, Příručku správce sítě nebo Průvodce jádrem systému. Všechny příručky jsou k dispozici ve zdrojovém formátu, ve formátu .dvi a v jazyce PostScript, a to na anonymním serveru FTP na adrese *ftp://sunsite.unc.edu*, v adresáři `/pub/Linux/docs/LDP`.

Rádi bychom v závěru dodali odvahy všem čtenářům se sklonem k tvůrčímu psaní či redigování, aby se připojili k naší snaze o zdokonalení dokumentace operačního systému Linux. Máte-li zájem a přístup k účtu pro elektronickou poštu, kontaktujte prostřednictvím Internetu Grega Hankinse na adrese *gregh@sunsite.unc.edu*.

Přehled operačního systému Linux

A viděl Bůh vše, což učinil, a aj, bylo velmi dobré.
Genesis 1:31[2]

Tato kapitola podává zevrubný přehled o operačním systému Linux. V první části jsou popsány nejdůležitější ze služeb, jež systém nabízí. Další části se bez přílišných podrobností zabývají programy, které popsané služby realizují. Cílem kapitoly je podat výklad principů systému jako celku s tím, že každá část bude podrobněji probrána později, na jiném místě knihy.

Různé části operačního systému

Operační systém typu Unix se skládá z *jádra* a *systémových programů*. Kromě toho pro různou běžnou práci existují *aplikační programy*. Jádro je srdcem operačního systému[3]. Udržuje záznamy o souborech na disku, spouští programy, řídí jejich současný běh, přiděluje paměť a další technické prostředky různým procesům, přijímá a odesílá pakety z a do počítačové sítě a tak dál. Jádro systému samotné toho dělá velmi málo, ale poskytuje základní služby různým nástrojům, pomocí kterých mohou být realizovány všechny ostatní služby. Jádro rovněž hlídá, aby nikdo nemohl přistupovat k hardwarovým zařízením přímo. Když chtějí uživatelé a procesy používat technické prostředky, musí používat nástroje, které nabízí jádro systému. Tímto způsobem je zabezpečená i vzájemná ochrana uživatelů. Nástroje jádra systému, o nichž byla řeč, lze využívat prostřednictvím *systémových volání*. Podrobnější informace o systémových voláních uvádí sekce 2 manuálových stránek.

Systémové programy realizují služby, které se vyžadují od operačního systému. Využívají při tom nástroje, které nabízí jádro systému. Systémové i všechny ostatní programy běží jakoby „na povrchu“ jádra. Říká se tomu *uživatelský režim*. Rozdíl mezi systémovými a aplikačními programy je v jejich určení. Pomocí aplikačních programů mohou uživatelé dělat některé užitečné věci (popřípadě se bavit – je-li aplikace, kterou si zrovna spustili, počítačová hra). Systémové programy jsou potřebné k tomu, aby systém vůbec fungoval. Textový editor je aplikace, **telnet** je systémový program. Hranice mezi aplikačními a systémovými programy je často dost neostrá, vlastně celé tako-

2 Pozn. překladatele: Podle českého vydání Bible, vydaného Českou biblickou společností v roce 1991.

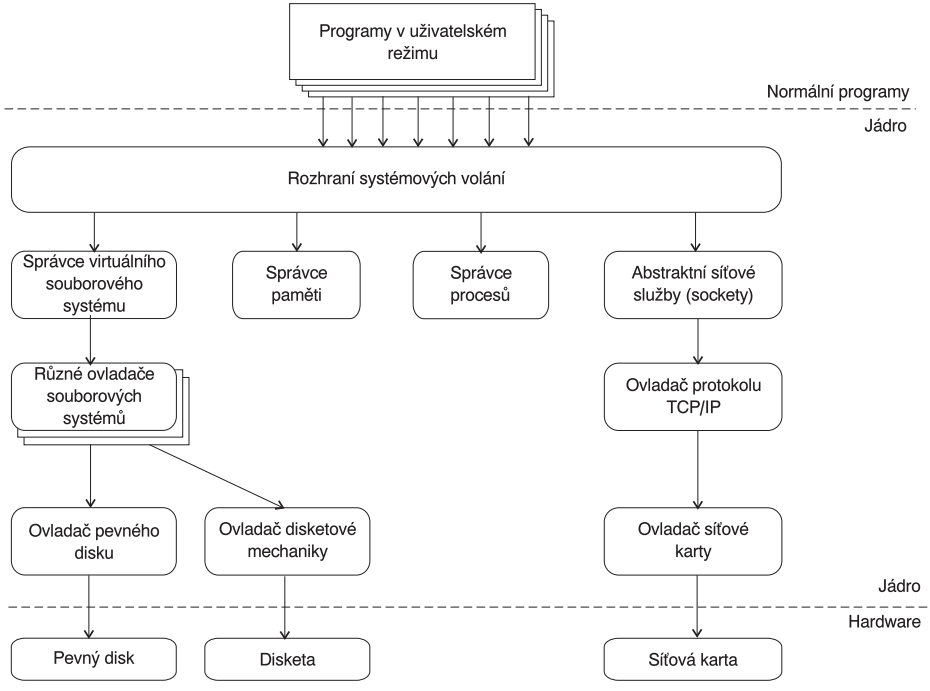
3 Vlastně bývá často mylně považováno za samotný operační systém, jenže to není pravda. Operační systém poskytuje mnohem více služeb než jen prosté jádro.

této rozdělení je poněkud samoúčelné – důležité čistě pro vymezení samotných klasifikačních kritérií.

Součástí operačního systému mohou být i překladače programovacích jazyků a jejich knihovny (v případě Linuxu překladač GCC a knihovna jazyka C), nicméně tomu tak nemusí být. Další částí systému je dokumentace, někdy dokonce i některé hry. Tradičně se za operační systém pokládá obsah jeho instalační pásky či instalačních disků. Pokud jde o systém Linux, není uvedená definice zcela jasná, protože na mnoha serverech FTP po celém světě existuje množství různých instalací systému.

Důležité části jádra systému

Jádro Linuxu sestává z několika důležitých subsystémů. Jsou to části řízení procesů, správy paměti, ovladačů technických prostředků, ovladačů souborových systémů, správy sítě a různé další kusy a kousky. Některé z nich jsou zobrazeny na obrázku 2.1.



Obrázek 2.1 – Důležité části jádra Linuxu

Snad nejdůležitějšími subsystémy (bez nichž nic jiného nefunguje) jsou správce paměti a správce procesů. Subsystém správy paměti zajišťuje přidělování paměťových oblastí a odkládacího prostoru jednotlivým procesům, částem jádra a vyrovnávací paměti. Subsystém správy procesů vytváří procesy a přepínáním mezi aktivními procesy, které využívají procesor, zabezpečuje multitasking.

Jádro systému na nejnižší úrovni obsahuje ovladače pro všechny druhy technických zařízení, které operační systém podporuje. Vzhledem k tomu, že na světě existuje celá řada různých typů hardwaru, je počet ovladačů zařízení velký. Je ale mnoho jinak podobných zařízení, která se často liší pouze v tom, jak spolupracují s programy. Takovéto podobnosti umožňují definovat obecné třídy ovladačů, jež podporují podobné operace. Každý člen takovéto třídy má stejné rozhraní k ostatním částem jádra. Liší se v tom, jak tyto operace implementuje. Například všechny ovladače disků vypadají pro zbytek jádra podobně. To znamená, že všechny znají operace jako „inicializuj diskovou jednotku“, „čti sektor N“ a „zapiš sektor N“.

Některé softwarové služby, jež poskytuje jádro samotné, mají rovněž podobné vlastnosti. Proto mohou být také rozdělené do tříd. Kupříkladu různé síťové protokoly byly vyčleněny do jednoho programového rozhraní – knihovny „BSD socket library“. Dalším příkladem je vrstva *virtuálního souborového systému* (VFS). Ta odděluje operace souborového systému od jejich implementace. Každý typ souborového systému obstarává implementaci určité množiny operací, společné všem systémům souborů. Když se některý z prvků systému pokouší využít určitý souborový systém, žádost jde přes VFS. Ten ji směřuje k požadovanému ovladači konkrétního systému souborů.

Nejdůležitější služby v unixovém systému

Tato část popisuje některé významnější služby systému Unix, avšak opět bez větších podrobností. Všechny služby budou později podrobně vysvětleny v dalších kapitolách.

init

Nejdůležitější služby v systému Unix poskytuje proces **init**. Spuštění procesu **init** jako prvního z procesů je v každém unixovém systému posledním krokem, který provede jádro systému při zavádění. Po spuštění proces **init** pokračuje v proceduře zavádění systému. Vykonává různé úkoly, které se při spouštění systému obvykle provádí (kontroluje a připojuje souborové systémy, spouští démony a tak dále).

Přesný seznam úloh, které **init** při zavádění dělá, závisí na verzi tohoto programu i operačního systému. Proces **init** často obstarává takzvaný *jednouživatelský režim*. V jednouživatelském režimu se do systému nemůže nikdo přihlásit a příkazový interpret může z konzoly používat pouze root. Běžným režimem práce je *víceuživatelský režim*. Tyto režimy práce některé systémy Unix zobecňují do takzvaných *úrovní běhu*. Jednouživatelský a víceuživatelský režim tak představují dvě různé úrovně, na kterých může systém běžet. Kromě nich mohou existovat i další, například úroveň, při které se na konzole spustí grafické rozhraní X Window a podobně.

V běžné situaci program **init** kontroluje, zda fungují procesy **getty** (umožňující uživatelům připojit se do systému) a adoptuje procesy – sirotky. Sirotci jsou procesy, jejichž rodičovské procesy byly z různých důvodů ukončeny – říká se, že umřeli. V systému Unix *musí* být *všechny* procesy součástí jediné hierarchické stromové struktury. Proto musí proces **init** sirotky adoptovat.

Když se systém vypíná, proces **init** zodpovídá za ukončení všech ostatních procesů, odpojení všech souborových systémů, zastavení procesoru a za vše ostatní, co má podle dané konfigurace udělat.

Přihlášení z terminálů

Přihlášení uživatelů prostřednictvím terminálů (připojených na sériové linky) a konzoly (v případě, že neběží X Window) obstarává program **getty**. Proces **init** spouští zvláštní instanci **getty** pro každý terminál, ze kterého se bude možno do systému přihlásit. Program **getty** dále čte zadáva-

né uživatelské jméno a spouští program **login**, jenž čte přístupové heslo. Jestli jsou uživatelské jméno a heslo správné, spustí program **login** příkazový interpret neboli *shell*. Když je příkazový interpret ukončen – jakmile se uživatel odhlásí ze systému, nebo když je program **login** ukončen proto, že nesouhlasí uživatelské jméno a heslo – proces **init** to zjistí a spustí pro daný terminál novou instanci programu **getty**. Samotné jádro systému nemá vůbec přehled o přihlašování uživatelů do systému. Všechno kolem toho obstarávají systémové programy.

Syslog

Jádro systému i mnoho systémových programů hlásí různé chyby, vypisuje varování a jiná hlášení. Velmi často je důležité, aby bylo možno tyto zprávy prohlížet později, dokonce i s velkým časovým odstupem. Je tedy vhodné je zapisovat do nějakých souborů. Program, který to má na starost, se jmenuje **syslog**. Lze jej nastavit tak, aby třídil zprávy a hlášení do různých souborů, a to podle původce, případně stupně významnosti. Hlášení jádra systému jsou obvykle směřována do jiného souboru než hlášení jiných procesů a programů. Jsou většinou významnější a je potřeba číst je pravidelně, aby bylo možné rozeznat případné problémy v zárodku.

Periodické vykonávání příkazů: cron a at

Uživatelé i správci systému často potřebují spouštět některé programy pravidelně. Například administrátor systému, který musí sledovat zaplněnost disku, by mohl chtít pravidelně spouštět příkaz, jenž by „vyčistil“ adresáře dočasných souborů (`/tmp` a `/var/tmp`). Program by odstranil starší dočasné soubory, které po sobě programy z různých důvodů korektně nesmazaly.

Takovéto služby nabízí program **cron**. Každý uživatel má vlastní soubor `crontab`, jenž obsahuje seznam příkazů, které chce vlastník spustit, a časy, kdy se mají tyto příkazy provést. Démon **cron** má na starost spouštění těchto příkazů v požadovaném čase.

Služba **at** je podobná službě **cron**, provede se ale jenom jednou. Příkaz je vykonán v určeném čase, ale jeho spouštění se neopakuje.

Grafické uživatelské rozhraní

Unix a Linux nezačleňují uživatelská rozhraní do jádra systému. Místo toho je implementují pomocí programů uživatelské úrovně. To se týká jak textového módu, tak grafického uživatelského prostředí.

Díky takovémuto řešení je samotný systém flexibilnější. Má to ale nevýhodu v tom, že je na druhou stranu velmi jednoduché implementovat pro každý program různá uživatelská rozhraní. Důsledkem je, že se takovýto systém uživatelé pomaleji učí.

Grafické prostředí, které Linux používá primárně, se nazývá „X Window System“ (zkráceně X). Ale ani X přímo neimplementují uživatelské rozhraní. X Window pouze zavádí systém oken, tedy sadu nástrojů, pomocí kterých může být grafické uživatelské rozhraní implementované.

Tři z nejpopulárnějších stylů uživatelských rozhraní postavených na X jsou Athena, Motif a Open Look.

Komunikace prostřednictvím počítačové sítě

Komunikace pomocí počítačové sítě je propojení dvou nebo více počítačů tak, že mohou komunikovat navzájem každý s každým. V současnosti používané metody propojování a komunikace jsou docela komplikované, ale výsledný efekt stojí za to.

Operační systémy Unix mají řadu síťových funkcí. Většinu základních služeb – služby souborových systémů, tisky, zálohování a podobně, lze využívat i prostřednictvím sítě. To ulehčuje správu systému a umožňuje centralizovanou administraci. Zachovávají se výhody mikropočítačové technologie i přínos distribuovaných systémů (nižší náklady a lepší odolnost vůči poruchám).

Tato kniha se komunikací prostřednictvím počítačové sítě zabývá jenom zběžně. Podrobnosti o této problematice, včetně základního popisu principů počítačových sítí, přináší *Příručka správce sítě*.

Přihlášení do systému ze sítě

Přihlášení do systému ze sítě funguje trochu odlišně než běžné přihlášení přes terminál. Pro každý terminál, prostřednictvím kterého je možné se přihlásit, je vyhrazená samostatná fyzická sériová linka. Pro každého uživatele, který se přihlašuje prostřednictvím sítě, existuje jedno samostatné virtuální síťové spojení, nicméně těchto spojení může být velký počet[4]. Proto není možné, aby běžely samostatné procesy **getty** pro všechna možná virtuální spojení. Kromě toho existuje několik různých způsobů přihlášení prostřednictvím sítě. Dva nejdůležitější způsoby v sítích TCP/IP jsou **telnet** a **rlogin**.

Síťová přihlášení mají místo řady procesů **getty** jednoho démona pro každý ze způsobů připojení (**telnet** a **rlogin** mají každý vlastního démona). Tento démon vyřizuje všechny přicházející žádosti o přihlášení. Dostane-li takovouto žádost, spustí svou novou instanci. Nová instance pak obsluhuje tuto jedinou žádost a původní instance nadále sleduje další příchozí žádosti o přihlášení. Nová instance pracuje podobně jako program **getty**.

Síťové souborové systémy

Jednou z nejužitečnějších věcí, kterou lze využít díky síťovým službám, je sdílení souborů pomocí *síťového souborového systému*. Jeden z nejběžněji používaných typů se nazývá Network File System, zkráceně NFS, a byl vyvinut společností Sun.

V síťovém souborovém systému jsou všechny operace se soubory, které dělá program na jednom počítači, odesílány prostřednictvím počítačové sítě na jiný počítač. Pro program, který běží na lokálním počítači vzniká iluze, že soubory, které se nachází na vzdáleném počítači, jsou ve skutečnosti umístěny na počítači, na němž tento program běží. Takovýmto způsobem je velmi jednoduché sdílet informace, navíc lze používat již existující programy bez toho, že by je bylo potřeba měnit.

Pošta

Elektronická pošta je obvykle tím nejdůležitějším způsobem počítačové komunikace. Elektronický dopis je uložený v textovém souboru se zvláštním formátem. K jeho odeslání nebo přečtení se používají speciální programy.

Každý uživatel systému má vlastní *schránku na příchozí poštu*. Je to soubor určitého formátu, ve kterém jsou uloženy všechny nově příchozí zprávy. Když někdo odesílá poštu, program zjistí adresu poštovní schránky příjemce a připojí dopis k jeho souboru s příchozí poštou. Jestli je schránka příjemce na jiném počítači, je dopis odeslán na tento stroj a ten se bude snažit doručit jej do schránky příjemce.

Systém elektronické pošty se skládá z několika typů programů. Doručení pošty do místních nebo vzdálených poštovních schránek má na starost první z nich – *agent pro přenos pošty* (MTA), například **sendmail** nebo **smail**. Uživatelé používají ke čtení pošty množství různých programů,

[4] Rozhodně jich může být hodně. Přenosová kapacita sítě představuje stále jisté omezení, takže je praktické nastavit nějaký limit počtu současných síťových připojení.

takzvaných *uživatelských poštovních agentů* (MUA), například **pine** nebo **elm**. Poštovní schránky uživatelů jsou obvykle uloženy v adresáři `/var/spool/mail`.

Tisk

Tiskárnu může současně využívat pouze jeden uživatel. Nesdílet tiskárny mezi uživateli je ale dost neekonomické. Tiskárnu proto řídí program, jenž realizuje takzvanou *tiskovou frontu*. Všechny tiskové úlohy všech uživatelů systému jsou zařazeny do fronty. Hned, jak tiskárna ukončí jednu úlohu, automaticky se jí odesílá další v pořadí. Uživatelé si nemusí zabezpečovat frontu požadavků na tisk organizačně a odpadá i nutnost soupeřit a handrkovat se o přístup k tiskárně[5].

Program pro obsluhu tiskové fronty navíc *ukládá* (takzvaný *spool*) všechny tiskové výstupy na disk, takže pokud je tisková úloha ve frontě, je text uložen v nějakém souboru. Tento mechanismus aplikačním programům umožňuje rychle odeslat tiskové úlohy programu, jenž tiskovou frontu obsluhuje. Aplikace sama tak může pokračovat ve své práci. Nemusí čekat, než se úloha, která se právě tiskne, ukončí. To je v mnoha případech skutečně výhodné. Umožní vám to například zahájit tisk jedné verze dokumentu, přičemž nemusíte čekat, než se tisk ukončí, a můžete mezitím pracovat na nové, zcela pozměněné verzi.

Organizace systému souborů

Souborový systém je rozdělen na mnoho částí. Obvykle jsou hierarchicky uspořádané a nejvýše stojí kořenový souborový systém root. Souborový systém root obsahuje adresáře `/bin`, `/lib`, `/etc`, `/dev` a několik dalších; souborový systém `/usr`, který obsahuje programy a data, která se nemění. Souborový systém `/var` obsahuje data, jež se naopak často mění (například logovací soubory). Posledním je souborový systém `/home`. Svá data a soubory si v něm ukládají uživatelé systému. Rozdělení souborového systému na jednotlivé svazky může být jiné. Závisí zejména na hardwarové konfiguraci systému a rozhodnutích jeho správce. Všechna data a soubory mohou být nakonec uloženy i v jediném systému souborů.

Některé další detaily týkající se uspořádání systému souborů popisuje kapitola 3. Ještě více podrobností o tomto tématu přináší dokument Linux Filesystem Standard.

5 Místo toho si uživatelé vytvoří vlastní frontu *u tiskárny,* kde čekají, až jim jejich dokument vyjede, protože ještě nikdo nedokázal naprogramovat obsluhu síťových front tak, aby se vědělo, kdy se co vytiskne. Je to velký příspěvek k mezilidským vztahům na pracovišti.

KAPITOLA 3

Přehled adresářové struktury

Dva dny nato seděl Pú na své větvi, pohupoval nohama a tam,
vedle něj, stály čtyři hrníčky medu. . .
A. A. Milne

Kapitola popisuje důležité části standardní struktury adresářů operačního systému Linux, která je založená na systému souborů standardu FSSTND. Načrtneme v ní také běžný způsob rozdělení struktury adresářů do samostatných souborových systémů (svazků) s odlišnými účely a tento způsob rozdělení zdůvodníme. Naznačíme i některé alternativní způsoby rozdělení.

Základy

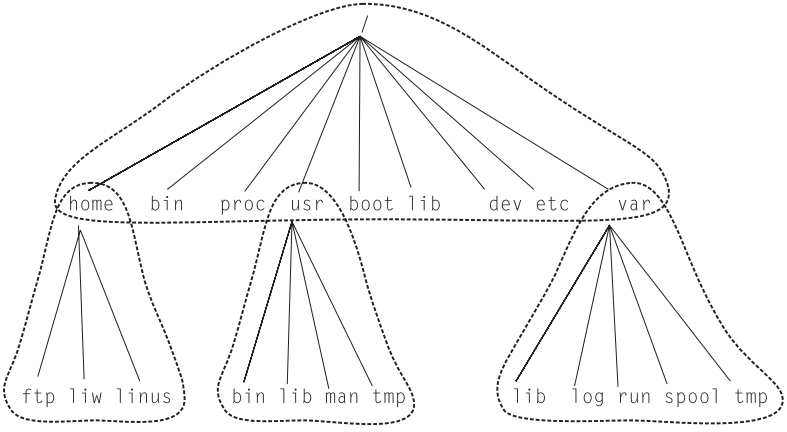
Tato kapitola volně vychází z normy „Standard systému souborů operačního systému Linux FSSTND, verze 1.2“ (viz bibliografie), který je pokusem zavést jisté konvence do organizace adresářového stromu operačního systému Linux. Výhodou přijetí takovéto normy je, že když bude vše na svém obvyklém místě, bude jednodušší psát programy a přenášet na Linux software z jiných platforem. Zároveň to ulehčí správu počítačů, na kterých běží operační systém Linux. I když neexistuje autorita, která by vývojáře, programátory a distributory donutila přizpůsobit se této normě, je její podpora v současnosti součástí většiny (ne-li všech) distribucí Linuxu. Není vhodné se neřídit standardem FSSTND, nejsou-li pro to velmi závažné důvody. Norma FSSTND se snaží sledovat tradice Unixu i současné trendy jeho vývoje. Motivací je snaha o usnadnění přechodu na Linux pro uživatele, kteří mají zkušenosti s jinými systémy Unix a naopak.

Tato kapitola není tak detailní jako FSSTND. Správce systému – chce-li plně proniknout do problematiky systémů souborů – by si měl přečíst i normu FSSTND.

Kapitola rovněž nepopisuje podrobnosti týkající se všech typů souborových systémů. Nebylo také cílem popsat všechny adresáře a soubory, ale nabídnout čtenáři přehled o celém systému z perspektivy systému souborů. Podrobnější informace o popisovaných souborech jsou k dispozici na jiných místech této knihy, případně na manuálových stránkách.

Celou stromovou strukturu adresářů je možné rozdělit na menší části, každá z těchto částí může být umístěna na vlastním disku nebo samostatné diskové oblasti. Tak se lze jednoduše přizpůsobit omezením velikosti disků a zároveň usnadnit zálohování i ostatní úkoly spojené se správou systému. Nejdůležitější z těchto částí jsou souborový systém root a dále souborové systémy `/usr`, `/var` a `/home` (viz obrázek 3.1). Každý systém souborů má jiné určení. Adresářová stromo-

vá struktura byla navržena tak, aby fungovala i v síti počítačů s operačním systémem Linux. Uživatelé a programy tak mohou pomocí sítě sdílet některé části systémů souborů, a to buď prostřednictvím zařízení určených pouze pro čtení (například CD-ROM), nebo pomocí sítě se systémem NFS.



Obrázek 3.1 – Části adresářové struktury Unixu. Přerušované čáry označují hranice diskových oblastí

Smysl jednotlivých částí adresářové struktury je popsán v dalším textu.

- Souborový systém root je specifický pro každý počítač. Obecně je uložen na lokálním disku (avšak může to být i virtuální disk v paměti RAM – takzvaný „ramdisk“, nebo síťová disková jednotka). Kořenový svazek obsahuje soubory nutné pro zavedení systému a jeho uvedení do stavu, ve kterém mohou být připojeny ostatní souborové systémy. Obsah kořenového souborového systému postačuje pro práci v jednouživatelském režimu. Na tomto svazku jsou rovněž uloženy nástroje pro opravy poškozeného souborového systému a pro obnovení ztracených souborů ze záloh.
- Souborový systém `/usr` obsahuje všechny příkazy, knihovny, manuálové stránky a jiné soubory, jejichž obsah se nemění a které uživatel potřebuje při běžném provozu. Žádný ze souborů na tomto svazku by neměl být specifický pro daný počítač. Rovněž by se neměl při normálním provozu měnit. Tyto podmínky zaručují, že soubory uložené v souborovém systému `/usr` bude možné efektivně sdílet v síti. Sdílení tohoto svazku je výhodné jak z hlediska nákladů – šetří se tím místo na disku (v souborovém systému `/usr` mohou být uloženy stovky megabajtů dat), tak z hlediska usnadnění správy systému (například při instalaci novější verze aplikace se pak mění pouze systém `/usr` na hostitelském počítači a ne na každé stanici zvlášť). Je-li souborový systém `/usr` na lokálním disku, může být připojen pouze pro čtení. To snižuje pravděpodobnost poškození systému souborů při havárii systému.
- Souborový systém `/var` obsahuje soubory, které se v čase mění. Tedy především sdílené adresáře pro elektronickou poštu, systém news, tiskárny, logovací soubory, formátované manuálové stránky a dočasné soubory. Historicky bývaly všechny soubory, které jsou nyní uloženy v systému `/var`, v souborovém systému `/usr`. To však znemožňovalo připojit svazek `/usr` pouze pro čtení.
- Souborový systém `/home` obsahuje domovské adresáře uživatelů, tedy všechna „reálná data“. Vyčlenění uživatelských domovských adresářů do vlastní adresářové stromové struktury nebo samostatného souborového systému ulehčuje zálohování. Ostatní části adresářo-

vého stromu totiž buď nevyžadují zálohování vůbec, nebo – vzhledem k tomu, že se nemění tak často – se zálohují jenom zřídka. Velký souborový systém `/home` je dobré rozdělit na několik menších částí hierarchicky nižší úrovně a rozlišit je jménem, například `/home/students` a `/home/staff`.

Různé části, na které je hierarchická adresářová struktura rozčleněna, byly v našem přehledu označeny jako souborové systémy. Není ale žádný zvláštní důvod k tomu, aby ve skutečnosti ležely na samostatných oddělených svazcích. Všechny by mohly být nakonec i v jediném souborovém systému. Takové řešení má význam hlavně pro malé jednouživatelské systémy, kdy je prioritou jednoduchost. Celá stromová adresářová struktura může být rozdělena na souborové systémy i jinak. Způsob jejího rozčlenění závisí na tom, jak velký je disk a jak velký diskový prostor bude vyhrazen pro různé účely. Jedinou věcí, na kterou je potřeba dbát, jsou standardní unixová *jména*. Ta je potřeba zachovat. I když bude adresář `/var` a `/usr` ve stejné diskové oblasti, musí být zachována standardní jména, jako např. `/usr/lib/libc.a` nebo `/var/log/messages`. Totéž platí i v případě, že se například přesune adresář `/var` do adresáře `/usr/var` a na původním místě přesunutého adresáře bude symbolický odkaz na adresář `/usr/var`.

Struktura souborového systému Unixu sdružuje soubory podle jejich účelu, tedy všechny příkazy na jednom místě, data na jiném, dokumentace na dalším a tak dál. Alternativou by bylo sdružovat soubory podle toho, ke kterému programu patří. Pak by mohly být například všechny soubory pro program Emacs v jednom adresáři, všechny soubory pro TEX v jiném a podobně. Problém druhého přístupu je v tom, že je velmi obtížné sdílet soubory (adresář určitého programu často obsahuje jak statické soubory, které lze sdílet, tak soubory, jejichž obsah se mění, a ty sdílet nelze). Rovněž by bylo velmi složité dohledávat v rámci celého systému soubory určitého typu, například manuálové stránky aplikací uložené na mnoha různých místech. Programátory by jistě strašila noční můra – jak v takovémto případě vytvořit programy, které by byly schopné v případě potřeby manuálové stránky všech aplikací nalézt.

Souborový systém `root`

Kořenový svazek root by obecně měl být malý, protože obsahuje velmi kritické soubory. U malého souborového systému, který se mění jenom zřídka, je menší pravděpodobnost poškození. Poškození souborového systému root většinou znamená, že operační systém nebude možné zavést. Tento problém lze řešit pouze pomocí speciálních opatření (například zavedením systému z diskety), a to by chtěl riskovat asi málokdo.

Obecně by kořenový adresář neměl obsahovat žádné soubory, snad kromě standardního obrazu systému. Ten se obvykle jmenuje `/vmlinuz`. Všechny ostatní soubory by měly být uloženy v podadresářích kořenového adresáře, obvykle tímto způsobem:

`/bin`	Příkazy potřebné při zavádění systému, které mohou použít i normální uživatelé.
`/sbin`	Stejně jako `/bin`, příkazy ale nejsou určeny pro normální uživatele, i když i ti je mohou použít, je-li to nutné a je-li to povoleno.
`/etc`	Konfigurační soubory specifické pro daný počítač.
`/root`	Domovský adresář superuživatele.
`/lib`	Sdílené knihovny pro programy v kořenovém souborovém systému.
`/lib/modules`	Zaveditelné moduly jádra systému – zvláště ty, které jsou potřeba pro zavedení systému při zotavení po havárii (například síťové ovladače a ovladače pro souborový systém).

`/dev`	Soubory zařízení.
`/tmp`	Dočasné soubory. Programy, které se spouští až po zavedení systému, by měly používat místo adresáře `/tmp` adresář `/var/tmp`, protože je velmi pravděpodobné, že leží na větším disku.
`/boot`	Soubory, jež používá zavaděč operačního systému, například LILO. Často se zde ukládají obrazy jádra (místo v kořenovém adresáři). V případě, že jich máte víc, může obsah adresáře `/boot` značně narůst a pak bude lepší mít jej v samostatném souborovém systému. Tím se také zajistí, že obrazy jádra budou uloženy na prvních 1024 cylindrech disku IDE.
`/mnt`	Přípojné místo pro dočasná připojení dalších systémů souborů správcem systému. Nepředpokládá se, že by tento adresář využívaly pro automatická připojení souborových systémů programy. Adresář `/mnt` může být rozdělen na podadresáře (například `/mnt/dosa` pro disketovou mechaniku používanou v souborovém systému MS-DOS a `/mnt/exta` pro tutéž mechaniku využívanou se souborovým systémem ext2 a podobně).
`/proc`, `/usr`, `/var`, `/home`	Přípojná místa pro další souborové systémy.

Adresář `/etc`

Adresář `/etc` obsahuje mnoho souborů. Některé z nich jsou popsány v dalším textu. Pokud jde o ostatní, měli byste nejdřív zjistit, ke kterému programu patří, a pak si přečíst manuálové stránky k tomuto programu. V adresáři `/etc` je uloženo také hodně síťových konfiguračních souborů, které jsou popsány v *Příručce správce sítě*.

`/etc/rc` **nebo** `/etc/rc.d` **nebo** `/etc/rc?.d`

Skripty nebo adresáře skriptů, které se spouští při startu, nebo v případě, že se mění úroveň běhu systému. Podrobnější informace najdete v kapitole o procesu **init**.

`/etc/passwd`

Databáze uživatelů systému s položkami, v nichž je uloženo uživatelské jméno i skutečné jméno uživatele, domovský adresář, šifrované heslo a některé další informace. Formát je popsán v manuálové stránce programu **passwd**.

`/etc/fdprm`

Tabulka parametrů disketové jednotky. Popisuje jak vypadají různé formáty disket. Používá ji program **setfdprm**. Víc informací uvádí manuálová stránka programu **setfdprm**.

`/etc/fstab`

Seznamy souborových systémů připojovaných automaticky při startu příkazem **mount -a** (ve skript `/etc/rc`, nebo nějakém ekvivalentu). V systému Linux obsahuje rovněž informace o odkládacích oblastech, které používá příkaz **swapon -a**. Podrobnější informace viz část *Připojování a odpojování* v kapitole 4 a manuálové stránky příkazu **mount**.

`/etc/group`

Soubor podobný souboru `/etc/passwd`, ale místo uživatelů popisuje pracovní skupiny. Podrobnější informace viz manuálová stránka **group**.

`/etc/inittab`

Konfigurační soubor procesu **init**.

`/etc/issue`

Soubor obsahuje výstup programu **getty**, který se zobrazí před výzvou pro přihlášení uživatele. Obvykle obsahuje stručný popis systému nebo uvítací hlášení. Obsah určuje správce systému.

`/etc/magic`

Konfigurační soubor programu **file**. Obsahuje popisy různých formátů souborů, podle kterých pak program **file** tyto typy rozpoznává. Více informací najdete v manuálových stránkách pro **magic** a **file**.

`/etc/motd`

Takzvaná *zpráva pro tento den* – automatický výstup na terminál uživatele po úspěšném přihlášení do systému. Obsah volí správce systému. Často se využívá pro předávání informací (například upozornění na plánovaná zastavení systému apod.) všem uživatelům systému.

`/etc/mtab`

Seznam aktuálně připojených souborových systémů. Jeho obsah po zavedení systému a připojení určených souborových systémů prvotně nastavují inicializační skripty, v běžném provozu pak automaticky příkaz **mount**. Používá se v případech, kdy je potřeba zjistit, které souborové systémy jsou připojené, například při zadání příkazu **df**.

`/etc/shadow`

Soubor stínových hesel uživatelů v systémech, které mají nainstalovanou podporu stínových hesel. Kódovaná stínová hesla jsou z bezpečnostních důvodů přenesena ze souboru `/etc/passwd` do souboru `/etc/shadow`, který může číst pouze superuživatel. Snižuje se tak pravděpodobnost odhalení některého z přístupových hesel.

`/etc/login.defs`

Konfigurační soubor příkazu **login**.

`/etc/printcap`

Podobně jako u souboru `/etc/termcap`, až na to, že soubor je určený pro tiskárny. Odlišná je i jeho syntaxe.

`/etc/profile`, `/etc/csh.login`, `/etc/csh.cshrc`

Soubory spouštěné při přihlášení uživatele nebo při startu systému interprety příkazů Bourne shell nebo C shell. Umožňují správci systému stanovit globální nastavení stejná pro všechny uživatele. Viz manuálové stránky k příslušným interpretům příkazů.

`/etc/securetty`

Soubor identifikuje zabezpečené terminály, tedy terminály, ze kterých se může přihlašovat superuživatel. Typicky jsou v seznamu uvedeny pouze virtuální konzoly, takže je nemožné (nebo přinejmenším těžší) získat oprávnění superuživatele přihlášením se po modemu nebo ze sítě.

`/etc/shells`

Soubor, jenž uvádí seznam důvěryhodných interpretů příkazů. Příkaz **chsh** umožňuje uživatelům změnit příkazový interpret spuštěný při přihlášení, a to pouze na některý z interpretů uvedený v tomto souboru. Proces **ftpd**, jenž běží na hostitelském počítači a poskytuje služby FTP pro klientské počítače, rovněž kontroluje, zda je uživatelův příkazový interpret uveden v tomto souboru a nedovolí připojit se klientům, jejichž příkazový interpret v tomto seznamu uveden není.

`/etc/termcap`

Databáze vlastností terminálu. Popisuje, kterými escape sekvencemi se řídí různé typy terminálů. Každý program je napsaný tak, že místo přímého výstupu escape sekvence, jež by fungovala pouze s konkrétním typem terminálu, hledá v tabulce `/etc/termcap` sekvenci, která odpovídá tomu, co chce program na terminálu zobrazit. Pak může většina programů správně obsluhovat většinu typů terminálů. Více informací uvádí manuálové stránky pro `termcap`, `curs-termcap` a `terminfo`.

Adresář `/dev`

Adresář `/dev` obsahuje speciální soubory všech zařízení. Speciální soubory jsou pojmenované podle určitých konvencí. Ty jsou podrobně popsány v části *Seznam zařízení*. Speciální soubory se vytváří v průběhu instalace operačního systému, v běžném provozu pak skriptem `/dev/MAKEDEV`. Podobný je skript `/dev/MAKEDEV.local`. Ten upravuje a používá správce systému, když vytváří čistě lokální speciální soubory a odkazy. Lokální speciální soubory jsou ty, které nejsou vytvořeny standardním postupem pomocí skriptu `MAKEDEV`, typicky například speciální soubory pro některé nestandardní ovladače zařízení.

Souborový systém `/usr`

Souborový systém `/usr` je často dost velký, protože jsou v něm instalované všechny programy. Všechny soubory v systému `/usr` jsou obvykle instalované přímo z distribuce systému Linux. Všechny další lokálně instalované programy se ukládají do adresáře `/usr/local`. To umožňuje správci systému instalovat vyšší verze Linuxu z nové verze distribuce nebo i úplně jiné distribuce operačního systému bez toho, že by bylo potřeba současně instalovat všechny programy znovu. Některé z podadresářů adresáře `/usr` jsou popsány níže, ty méně významné neuvádíme. Více informací přináší popis standardu FSSTND.

`/usr/X11R6`

Všechny soubory systému X Window. Soubory pro X nejsou integrální součástí operačního systému z důvodů zjednodušení vývoje a instalace X. Adresářová struktura `/usr/X11R6` je podobná stromu, který je vytvořen pod adresářem `/usr` samotným.

`/usr/X386`

Podobně jako u `/usr/X11R6`, ale pro systém X11 Release 5.

`/usr/bin`

Zde se nachází téměř všechny uživatelské příkazy. Některé další příkazy jsou uloženy v adresáři `/bin` nebo `/usr/local/bin`.

`/usr/sbin`

Obsahuje ty příkazy pro správu systému, které nejsou potřeba přímo v kořenovém souborovém systému (například převážná většina serverových programů).

`/usr/man`, `/usr/info`, `/usr/doc`

Manuálové stránky, informační dokumenty GNU, případně různé jiné soubory s dokumentací. Novější distribuce používají adresář /usr/share/.

`/usr/include`

Hlavičkové soubory pro programovací jazyk C. Z důvodů zachování konzistence by měly být spíš v adresáři `/usr/lib`, ale z historických důvodů jsou umístěny ve zvláštním adresáři.

`/usr/lib`

Datové soubory pro programy a subsystémy, které se nemění. Jsou zde rovněž uloženy některé globální konfigurační soubory. Jméno `lib` je odvozeno od anglického slova „library" (knihovna). Původně totiž byly v adresáři `/usr/lib` uloženy knihovny podprogramů.

`/usr/local`

Místo pro lokálně instalovaný software a další soubory.

Souborový systém `/var`

Systém `/var` obsahuje data, která se při běžném provozu systému mění. Soubory jsou specifické pro každý systém, a proto se data mezi jinými počítači v síti nesdílí.

`/var/catman`

Vyrovnávací paměť pro manuálové stránky, které jsou formátovány na požádání. Zdrojové texty manuálových stránek jsou obvykle uloženy v adresáři `/usr/man/man*`. Některé stránky se dodávají v předem formátované verzi a jsou pak uloženy v adresáři `/usr/man/cat*`. Jiné stránky je třeba při prvním prohlížení naformátovat. Formátované verze jsou pak uloženy právě v adresáři `/var/catman`. Další uživatel, který si chce stejné stránky prohlížet, tak nemusí čekat na jejich opakované formátování. (Soubory v adresáři `/var/catman` se obvykle mažou stejným mechanismem jako dočasné soubory.)

`/var/lib`

Soubory, které se při normálním provozu systému mění.

`/var/local`

Měnící se data pro programy instalované v adresáři `/usr/local` (tedy programy instalované správcem systému). Upozorňujeme, že lokálně instalované programy by měly podle potřeby používat i ostatní podadresáře nadřazeného adresáře `/var`, například `/var/lock`.

`/var/lock`

Soubory zámků. Většina programů dodržuje určitou konvenci a vytváří v adresáři `/var/lock` zámky. Tím dávají ostatním programům najevo, že dočasně využívají některé zařízení nebo soubor. Jiné programy, které by chtěly stejné zařízení či soubor ve stejném okamžiku používat, se o to nebudou pokoušet.

`/var/log`

Adresář obsahuje logovací soubory různých programů, zejména programu **login** (do souboru `/var/log/wtmp` se zaznamenávají všechna přihlášení a odhlášení uživatelů systému) a **syslog** (do souboru `/var/log/messages` ukládá všechna hlášení jádra systému a systémových programů). Velikost souborů v adresáři `/var/log` dost často nekontrolovaně roste, proto se musí v pravidelných intervalech mazat.

`/var/run`

Adresář, do něhož se ukládají soubory obsahující informace o systému, jež platí až do jeho dalšího zavedení. Tak například soubor `/var/run/utmp` obsahuje informace o momentálně přihlášených uživatelích systému.

`/var/spool`

Adresáře pro elektronickou poštu, systém news, tiskové fronty a další subsystémy, které využívají metodu spoolingu a princip řazení úloh do fronty. Každý z těchto subsystémů má v tomto adresáři svůj vlastní podadresář, například poštovní schránky uživatelů jsou uloženy v podadresáři `/var/spool/mail`.

`/var/tmp`

Do adresáře `/var/tmp` se ukládají velké dočasné soubory a dočasné soubory, které budou existovat déle než ty, které se ukládají do adresáře `/tmp`. (Avšak správce systému by měl dbát na to, aby stejně jako v adresáři `/tmp`, ani v adresáři `/var/tmp` nebyly uloženy velmi staré dočasné soubory.)

Souborový systém `/proc`

Systém souborů `/proc` je vlastně imaginárním souborovým systémem. Ve skutečnosti na disku neexistuje. Místo toho jej v paměti vytváří jádro systému. Ze systému souborů `/proc` lze získávat různé aktuální informace o systému (původně o procesech – z toho je odvozeno jeho jméno). Některé z významnějších souborů a adresářů popisujeme níže. Samotný souborový systém `/proc` je podrobněji popsán na manuálové stránce *proc*.

`/proc/1`

Adresář s informacemi o procesu číslo 1. Každý z procesů má v adresáři `/proc` vlastní podadresář, jehož jméno je stejné jako identifikační číslo procesu.

`/proc/cpuinfo`

Různé informace o procesoru. Například typ, výrobce, model, výkon a podobně.

`/proc/devices`

Seznam ovladačů zařízení konfigurovaných pro aktuálně běžící jádro systému.

`/proc/dma`

Informuje o tom, které kanály DMA jsou právě využívány.

`/proc/filesystems`

Souborové systémy konfigurované v jádru systému.

`/proc/interrupts`

Informuje o tom, která přerušení jsou využívána a kolikrát nastala.

`/proc/ioports`

Informuje o tom, které ze vstupně-výstupních portů se momentálně využívají.

`/proc/kcore`

Obraz fyzické paměti systému. Má velikost odpovídající velikosti fyzické paměti systému. Ve skutečnosti ale samozřejmě nezabírá takovéto množství paměti, protože jde o soubor generovaný „na požádání“, tedy pokaždé jenom v okamžiku, kdy k němu různé programy přistupují. Uvědomte si, že soubory souborového systému `/proc` nezabírají ve skutečnosti (než je zkopírujete na nějaké jiné místo na disku) vůbec žádný diskový prostor.

`/proc/kmsg`

Výstupní hlášení jádra systému. Zde uložená hlášení dostává i program **syslog**.

`/proc/ksyms`

Tabulka symbolů jádra systému.

`/proc/loadavg`

Statistika zatížení systému – tři celkem nic neříkající indikátory toho, kolik práce systém momentálně má.

`/proc/meminfo`

Informace o využití paměti, jak fyzické, tak virtuální.

`/proc/modules`

Informuje o tom, které moduly jádra jsou právě zavedeny v paměti.

`/proc/net`

Informace o stavu síťových protokolů.

`/proc/self`

Symbolický odkaz do adresáře procesů toho programu, který zrovna přistupuje k souborovému systému `/proc`. Když k systému souborů `/proc` současně přistupují dva různé procesy, budou mít přidělené dva různé odkazy. Tímto způsobem se mohou programy pohodlně a jednoduše dostat k vlastnímu adresáři.

`/proc/stat`

Různé statistiky týkající se systému. Například počet výpadků stránek od zavedení systému a podobné.

`/proc/uptime`

Informuje o tom, jak dlouho systém běží.

`/proc/version`

Verze jádra systému.

Většina výše popsaných souborů jsou jednoduše čitelné textové soubory, nicméně občas mohou být formátovány složitěji. Proto existuje mnoho příkazů, které – kromě toho, že čtou informace obsažené v souborech systému `/proc`, upravují jejich formát do srozumitelnější podoby. Tak například program **free** čte data ze souboru `/proc/meminfo`, převádí velikost v bajtech na kilobajty a přidá něco málo dalších informací o využití paměti.

KAPITOLA 4

Disky a jiná média

Na prázdném disku lze hledat věčně

Při instalaci a aktualizaci systému vás u disků čeká hodně práce. Je potřeba vytvořit souborové systémy, do kterých se budou ukládat soubory, a vyhradit na discích prostor pro různé části systému.

Tato kapitola popisuje všechny tyto úvodní činnosti. Když tuto práci jednou podstoupíte a systém nastavíte, obvykle to už nebudete muset dělat znovu. Výjimkou je používání disket. K této kapitole se také budete vracet pokaždé, když budete přidávat nový pevný disk nebo pokud budete chtít optimálně vyladit diskový subsystém.

Mezi základní úkoly při správě disků patří:

- Formátování pevného disku. Formátování disku je posloupností několika různých dílčích činností (jako je například kontrola výskytu vadných sektorů), které tento disk připravují na další použití. (V současnosti je většina nových pevných disků formátovaná výrobcem a jejich formátování není nutné.)
- Rozdělení pevného disku na oblasti. Když chcete disk využívat pro několik činností, o kterých se nepředpokládá, že by se vzájemně ovlivňovaly, můžete jej rozdělit na samostatné diskové oblasti. Jedním z důvodů pro rozdělení disku na oblasti je instalace a provozování různých operačních systémů na jednom disku. Jinou výhodou rozdělení pevného disku na oblasti je oddělení uživatelských souborů od souborů systémových. Tím se zjednoduší zálohování a sníží se pravděpodobnost poškození systémových souborů.
- Vytvoření souborového systému (vhodného typu) na každém disku nebo diskové oblasti. Z pohledu operačního systému nestačí disk pouze zapojit, Linux jej může používat až poté, co na něm vytvoříte nějaký souborový systém. Pak lze na disk ukládat soubory a přistupovat k nim.
- Vytvoření jediné stromové struktury připojením různých souborových systémů. Systémy souborů se připojují buď automaticky, nebo manuálně – podle potřeby. (Ručně připojené souborové systémy se obvykle musí také ručně odpojit.)

Kapitola 5 obsahuje i informace o virtuální paměti a diskové vyrovnávací paměti. Pracujete-li s disky, měli byste být s jejich principy obeznámeni.

Dva druhy zařízení

Unix, a tedy i Linux, zná dva různé typy zařízení. Jednak bloková zařízení s náhodným přístupem (například disky) a jednak znaková zařízení (například pásky a sériové linky), která mohou být buď sériová, nebo s náhodným přístupem. Každému z podporovaných zařízení odpovídá v systé-

mu souborů jeden soubor zařízení. Když se čtou či zapisují data z anebo do souboru zařízení, přenáší se ve skutečnosti na zařízení, které tento soubor reprezentuje. Takže pro přístup k zařízením nejsou nutné žádné zvláštní programy a žádné zvláštní programovací techniky (jako například obsluha přerušení nebo cyklické dotazování sériového portu). Když například chcete vytisknout nějaký soubor na tiskárně, stačí zadat

```
$ cat jméno_souboru > /dev/lp1
$
```

a obsah souboru se vytiskne (soubor musí mít přirozeně nějakou formu, kterou tiskárna zná). Avšak vzhledem k tomu, že není příliš rozumné, aby několik uživatelů současně kopírovalo soubory na jedinou tiskárnu, pro tiskové úlohy se běžně používá zvláštní program (nejčastěji **lpr**). Tento program zajistí, že se v určitém okamžiku bude tisknout pouze jeden soubor. Ihned po ukončení jedné tiskové úlohy automaticky pošle na tiskárnu další úlohu. Podobný mechanismus přístupu vyžaduje většina zařízení. Takže ve skutečnosti se běžný uživatel o soubory zařízení skoro vůbec nemusí zajímat.

Protože se zařízení chovají jako soubory souborového systému (uložené v adresáři `/dev`), lze velmi lehce zjistit, které soubory zařízení existují. Lze použít například příkaz **ls** nebo jiný vhodný program. V prvním sloupci výstupu příkazu **ls -l** je uveden typ souboru a jeho přístupová práva. Chcete-li si prohlédnout sériové zařízení systému, zadáte

```
$ ls -l /dev/cua0
crw-rw-rw- 1 root uucp 5, 64 Nov 30 1993 /dev/cua0
$
```

Podle prvního písmene prvního sloupce, tedy písmene „c“ v řetězci „`crw-rw-rw-`“, informovaný uživatel pozná o jaký typ souboru jde. V tomto případě se jedná o znakové zařízení. U běžných souborů je prvním písmenem „`-`“, u adresářů je to „`d`“ a pro bloková zařízení se používá písmeno „b“. Podrobnější informace uvádí manuálová stránka k příkazu **ls**.

Všimněte si, že všechny speciální soubory obvykle existují, i když zařízení samotné není nainstalované. Takže například pouhý fakt, že v systému souborů je soubor `/dev/sda`, neznamená, že skutečně máte pevný disk SCSI. Díky tomu, že všechny speciální soubory po instalaci operačního systému existují, lze zjednodušit instalační programy. Méně složité je tím pádem i přidávání nových hardwarových komponent (pro nově přidávané součásti již není třeba hledat správné parametry a vytvářet speciální soubory).

Pevné disky

Tato část zavádí terminologii, která se v oblasti pevných disků používá. Znáte-li tyto pojmy a principy, můžete ji přeskočit.

Na obrázku 4.1 jsou schematicky znázorněny důležité části pevného disku. Disk se skládá z jedné nebo více kruhových *desek*[6], jejichž *povrchy* jsou pokryty magnetickou vrstvou, na kterou se zaznamenávají data. Každý povrch má svou *čtecí a zapisovací hlavu*, která čte nebo zaznamenává údaje. Desky rotují na společné hřídeli, typická rychlost otáčení je 5400 otáček za minutu. (Pevné disky s vysokým výkonem používají i vyšší rychlosti.) Hlavy se pohybují podél roviny povrchu, spojením tohoto pohybu s rotací povrchu může hlava přistupovat ke všem částem magnetických povrchů.

6 Desky jsou vyrobeny z nějakého pevného materiálu, například hliníku – odtud pochází název *pevný* disk.

Procesor a disk spolu komunikují prostřednictvím *řadiče disku*. Díky řadiči se zbytek systému nemusí zajímat o to, jak pracovat s diskem, protože lze použít pro různé typy disků řadiče, jež mají pro ostatní součásti počítače stejné rozhraní. Takže počítač může disku (místo zadávání sérií dlouhých a složitých elektrických signálů, podle nichž se hlava nejdřív přesune na odpovídající místo disku, pak čeká, až se pod ni dostane žádaná pozice a dělá další nepříjemnosti, které jsou pro danou operaci potřeba) jednoduše vzkázat „hele, milý disku, dej mně to, co potřebuji“. (Ve skutečnosti je sice rozhraní řadiče stejně dost složité, ale rozhodně ne tak složité, jako by bylo v případě, kdyby ostatní prvky systému přistupovaly k disku přímo.) Řadič navíc může dělat některé další operace, obsluhuje například vyrovnávací paměť, automaticky nahrazuje vadné sektory a podobně.

Výše uvedené obvykle každému postačí k pochopení toho, jak hardware pevného disku funguje. Existuje pochopitelně ještě řada dalších částí, například motory, které otáčí diskem a přemísťují hlavy, elektronika, jež řídí operace dalších mechanických částí a tak dále, které ale většinou nejsou pro pochopení principu práce pevného disku tak důležité.

Magnetické vrstvy disku jsou obvykle rozděleny do soustředných kružnic, kterým se říká *stopy*. Stopy se dělí na *sektory*. Toto rozdělení se používá pro určování místa na disku a přidělování diskového prostoru souborům. Když například chcete na disku najít určité místo, zadáte „souřadnice“: vrstva 3, stopa 5, sektor 7. Počet sektorů je většinou pro všechny stopy stejný, ale některé pevné disky mají na vnějších stopách víc sektorů (všechny sektory mají stejnou fyzickou velikost, takže většina z nich je umístěna na delších, vnějších stopách). Typicky bude v jednom sektoru uloženo 512 bajtů dat. Disk samotný pak neumí pracovat s menším množstvím dat, než je jeden sektor.

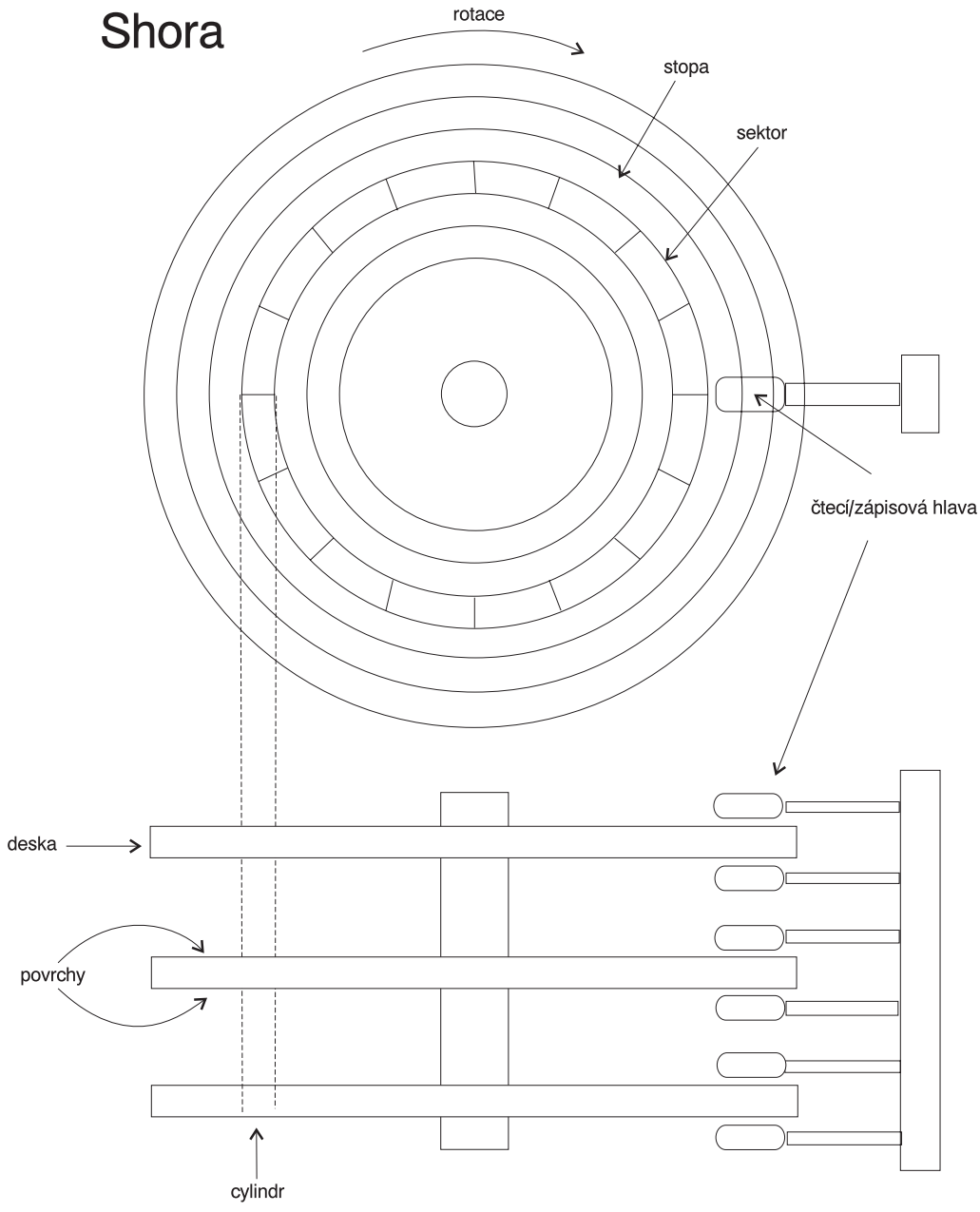
Každý povrch je rozdělen na stopy a sektory stejným způsobem. Takže když je hlava jednoho povrchu nad určitou stopou, hlavy ostatních povrchů se také nachází nad odpovídajícími stopami. Všem těmto stopám dohromady se říká *cylindr*. Přemístění hlavy z jedné stopy (cylindru) na jinou trvá jistou dobu. Takže ukládáním dat, ke kterým se často přistupuje najednou (například data jednoho souboru) na stejný cylindr, se zamezí zbytečnému přesouvání hlav při jejich pozdějším čtení. Snižuje se přístupová doba a zvyšuje výkon. Není ale vždycky možné uložit data na disk tímto způsobem. Souborům, které jsou na disku uloženy na několika místech, se říká *fragmentované*.

Počet povrchů (respektive hlav, což je to samé), cylindrů a sektorů se dost liší. Specifikace jejich počtu se nazývá *geometrií* pevného disku. Geometrie je obvykle uložena ve zvláštní bateriemi zálohované oblasti paměti, které se říká *CMOS RAM*, odkud si ji operační systém načítá vždy během zavádění nebo inicializace ovladače disku.

Naneštěstí má BIOS[7] omezení, které neumožňuje zadat v paměti CMOS počet stop větší než 1024, což je pro pevné disky velké kapacity příliš málo. Uvedené omezení lze obejít tak, že řadič pevného disku bude lhát ohledně skutečné geometrie disku a bude *překládat adresy*, požadované systémem na adresy, které odpovídají realitě. Představme si například pevný disk, jenž má 8 hlav, 2048 stop a 35 sektorů na stopu[8]. Řadič tohoto disku bude zbytku systému lhát a tvrdit, že má 16 hlav, 1024 stop a 35 sektorů na stopu, což nepřekračuje omezení v počtu stop. Pak bude při každém požadavku systému na přístup k disku překládat adresu, kterou dostane tak, že počet hlav vydělí dvěma a počet stop dvěma vynásobí. Matematické úpravy budou v praxi složitější, protože skutečná čísla nejsou tak „pěkná“, jako v uvedeném příkladě. Ale nezbývá než zopakovat, že detaily nejsou tak důležité pro pochopení principu. Překládání adres zkresluje pohled operačního

7 BIOS je vestavěný software počítače, uložený v paměti ROM. Stará se mimo jiné o zahájení procesu zavádění operačního systému.

8 Čísla jsou zhola vymyšlená.



Obrázek 4.1 – Schematické znázornění disku

systému na to, jak je disk ve skutečnosti organizovaný. To je nepraktické, protože nelze pro snížení přístupové doby a zvýšení výkonu použít „trik“ s ukládáním všech souvisejících dat na jeden cylindr.

Překlady adres jsou výlučně problémem disků typu IDE. Disky typu SCSI používají sekvenční čísla sektorů. To znamená, že řadič disku SCSI překládá každé sekvenční číslo sektoru na trojici [hlava, cylindr, sektor]. Navíc používá úplně jinou metodu komunikace s CPU, a proto jsou disky SCSI těchto problémů ušetřeny. Uvědomte si ale, že ani u disků SCSI počítač nezná jejich skutečnou geometrii.

Vzhledem k tomu, že operační systém Linux obvykle nezná skutečnou geometrii disku, nebudou se ani jeho souborové systémy pokoušet ukládat soubory na stejné cylindry. Místo toho se pokusí souborům přidělit sekvenčně řazené sektory. Tato metoda téměř vždy zaručí podobný výkon, jakého lze dosáhnout při ukládání souvisejících dat na jeden cylindr. Celá problematika je o něco složitější, řadiče například využívají vlastní vyrovnávací paměti, nebo mechanismus automatického, řadičem řízeného „přednačítání“ sekvenčně řazených sektorů.

Každý pevný disk je v systému reprezentován samostatným speciálním souborem. Typicky mohou být v systému maximálně dva nebo čtyři pevné disky IDE. Zastupují je pak speciální soubory `/dev/hda`, `/dev/hdb`, `/dev/hdc` a `/dev/hdd`. Pevné disky SCSI reprezentují speciální soubory `/dev/sda`, `/dev/sdb` a tak dále. Podobné konvence týkající se názvů speciálních souborů platí i pro pevné disky jiných typů. Podrobnější informace viz *Seznam zařízení*. Pamatujte na to, že speciální soubory zastupující pevné disky umožňují přístup k disku jako celku, bez ohledu na diskové oblasti (o kterých budeme hovořit za chvíli). Není proto těžké při práci s disky pochybit. Neopatrnost může v tomto případě vést ke ztrátě dat. Speciální soubory disků se obvykle používají pouze pro přístup k hlavnímu zaváděcímu sektoru disku, o kterém se také zmíníme později.

Diskety

Disketa sestává z pružné membrány pokryté z jedné nebo obou stran podobnou magnetickou substancí, jako pevný disk. Pružný disk samotný nemá čtecí a zápisovou hlavu, ta je součástí disketové mechaniky. Disketa vlastně odpovídá jedné desce pevného disku, ale na rozdíl od pevného disku je vyměnitelná. Jednu disketovou mechaniku lze využít pro přístup k různým disketám, kdežto pevný disk je jedinou nedílnou jednotkou.

Podobně jako pevný disk se i disketa dělí na stopy a sektory. Dvě korespondující si stopy každé strany diskety tvoří cylindr. Počet stop a sektorů je ale pochopitelně o hodně nižší než u pevného disku.

Disketová jednotka umí obvykle pracovat s několika různými typy disket. Například 3,5palcová disketová mechanika může pracovat jak s disketami o velikosti 720 kB, tak 1,44 MB. Vzhledem k tomu, že disketová jednotka musí zacházet s každým typem diskety trochu jinak, musí i operační systém vědět, který typ diskety je zrovna zasunutý v mechanice. Proto existuje pro jednotky pružných disků množství speciálních souborů, a to vždy jeden pro každou kombinaci typu disketové jednotky a typu diskety. Takže soubor `/dev/fd0H1440` pak reprezentuje první disketovou jednotku (`fd0`). Musí to být 3,5palcová mechanika pracující s 3,5palcovými disketami vysoké hustoty záznamu (proto `H` v názvu zařízení) s kapacitou 1 440 kB (proto `1440`). To jsou běžné 3,5palcové diskety HD. Podrobnější informace o konvencích pro pojmenovávání speciálních souborů reprezentujících disketové mechaniky viz *Seznam zařízení*.

Konstrukce tvorby názvů speciálních souborů zastupujících disketové jednotky je poměrně složitá. Proto má Linux pro diskety i zvláštní typy zařízení. Tato zařízení automaticky detekují typ diskety zasunuté v mechanice. Fungují tak, že se při požadavku na přístup pokouší přečíst první sek-

tor vložené diskety, přičemž postupně zkouší jejich různé typy, až se jim podaří najít ten správný. Samozřejmě, podmínkou je, aby vložená disketa byla nejdříve naformátovaná. Automatická zařízení zastupují speciální soubory `/dev/fd0`, `/dev/fd1` a tak dále.

Parametry, které tato automatická zařízení používají pro přístup k disketám, lze nastavit také programem **setfdprm**. To se může hodit jednak v případě, že používáte diskety, jež nemají běžnou kapacitu, to znamená, že mají neobvyklý počet sektorů, dále v případě, že autodetekce z neznámých důvodů selže a nebo když vlastní speciální soubor chybí.

Kromě toho, že Linux zná všechny standardní formáty disket, umí pracovat i s množstvím nestandardních formátů. Některé z nich ale vyžadují speciální formátovací programy. Touto problematikou se teď nebudeme zabývat a doporučíme projít si soubor `/etc/fdprm`. Ten blíže specifikuje nastavení, která program **setfdprm** podporuje.

Operační systém musí vědět o tom, že byla disketa v mechanice vyměněna. Jinak by totiž mohl například použít data z dříve vložené diskety, uložená ve vyrovnávací paměti. Bohužel vodič, jenž se používá k signalizaci výměny diskety, bývá někdy poškozený. Při používání disketové jednotky v systému MS-DOS nebude mechanika vůbec schopná indikovat systému výměnu média, což je ještě horší situace. Jestli jste se někdy setkali s podivnými, zdánlivě nevysvětlitelnými problémy při práci s disketami, jejich příčinou mohla být právě nefunkční indikace výměny média. Jediným způsobem, jak lze tyto problémy odstranit, je nechat disketovou mechaniku opravit.

Jednotky CD-ROM

Jednotky CD-ROM čtou opticky data z plastických disků. Informace jsou zaznamenány na povrchu těchto disků[9] jako miniaturní „dolíčky“ seřazené v husté spirále, jež začíná uprostřed disku a končí na jeho okraji. Jednotka vysílá laserový paprsek, který čte data z disku tak, že sleduje tuto spirálu. Od hladkého povrchu disku se odráží jinak, než když narazí na dolík. Tímto způsobem lze jednoduše kódovat binární informace. Ostatní je prostě pouhá mechanika.

Ve srovnání s pevnými disky jsou jednotky CD-ROM pomalé. Pevné disky mají průměrnou přístupovou dobu typicky menší než 15 milisekund, kdežto rychlá jednotka CD-ROM bude data vyhledávat s přístupovou dobou řádově v desetinách sekundy. Skutečná rychlost přenosu dat je ale celkem vysoká, zhruba stovky kilobajtů za sekundu. To, že je jednotka CD-ROM „pomalá“, znamená, že ji nelze pohodlně využít jako alternativu pevných disků, i když to samozřejmě možné je. Některé distribuce Linuxu totiž nabízejí takzvané „live“ souborové systémy na CD-ROM. Ty fungují tak, že část souborů se při instalaci nakopíruje na pevný disk a v případě potřeby se čtou přímo z kompaktního disku. Instalace je pak jednodušší, méně časově náročná a ušetří se také značná část diskového prostoru. Jednotky CD-ROM lze s výhodou využít při instalaci nového programového vybavení, protože při instalování není vysoká rychlost určujícím faktorem.

Je několik způsobů, jak se data na disky CD-ROM ukládají. Nejrozšířenější z nich upravuje mezinárodní standard ISO 9660. Tato norma definuje minimální souborový systém, který je ještě o něco primitivnější než systém souborů používaný systémem MS-DOS. Na druhou stranu je souborový systém ISO 9660 tak jednoduchý, že by jej měly bez potíží dokázat mapovat na svůj vlastní souborový systém všechny operační systémy.

Pro běžnou práci v Unixu je ale souborový systém ISO 9660 prakticky nepoužitelný. Proto se zavedlo rozšíření tohoto standardu, kterému se říká „Rock Ridge“. Rock Ridge například umožňuje používat dlouhá jména souborů, symbolické odkazy a mnoho dalších vymožeností, díky kterým se disk CD-ROM tváří víceméně jako unixový souborový systém. Navíc, rozšíření Rock Ridge za-

9 Respektive na povrchu *uvnitř* disku – tenké kovové vrstvě zatavené v plastickém obalu.

chovává kompatibilitu se souborovým systémem ISO 9660, takže je použitelné i v jiných než unixových systémech. Operační systém Linux podporuje jak ISO 9660, tak Rock Ridge. Rozšíření rozezná a dále používá zcela automaticky.

Souborový systém je ale pouze jednou stranou mince. Většina disků CD-ROM často obsahuje data, ke kterým lze přistupovat výhradně pomocí zvláštních programů. Bohužel většina těchto programů není určena pro Linux (možná s výjimkou některých programů, jež běží pod **dosemu**, linuxovým emulátorem systému MS-DOS).

K jednotce CD-ROM se také přistupuje prostřednictvím odpovídajícího speciálního souboru. Je několik způsobů jak připojit jednotku CD-ROM k počítači: buď rozhraním SCSI, nebo pomocí zvukové karty, nebo rozhraním EIDE. Popis dalších podrobností technického řešení jednotlivých způsobů připojení jednotky CD-ROM je nad rámec této knihy. Pro nás je podstatné, že podobně jako u jednotek pružných disků určuje způsob připojení vždy jiný speciální soubor. Další podrobnosti, které by mohly více objasnit tuto problematiku, uvádí *Seznam zařízení*.

Pásky

Magnetopáskové jednotky používají pásky podobné[10] těm, které se používají pro záznam zvuku na magnetofonových kazetách. Páska je již svou povahou sériovým zařízením – aby bylo možné dostat se k některé její části, je nejdřív nutné projít všechny předchozí záznamy. K údajům na disku lze přistupovat náhodně, takže je možné „skočit“ přímo na kterékoliv místo na něm. Sériový přístup k datům na páskách je příčinou toho, že jsou pomalé.

Na druhou stranu jsou pásky relativně levné, což kompenzuje nedostatky v rychlosti. Bez problému je lze vyrobit dost dlouhé, takže je na ně možno uložit velké množství dat. To vše předurčuje pásková zařízení k archivaci a zálohování, u nichž se nevyžaduje vysoká rychlost, ale naopak, využívá se nízkých nákladů a velké kapacity.

Formátování

Formátování je procedura zápisu značek, které se používají na označení stop a sektorů na magnetickém médiu. Předtím, než je disk naformátován, jsou magnetické signály na jeho povrchu neuspořádané, chaotické. Formátování do tohoto chaosu vnáší určitý řád. Načrtnou se – obrazně řečeno – linie, ve kterých vedou stopy a ty se pak rozdělí na sektory. Skutečné podrobnosti jsou trochu jiné, ale to není pro tuto chvíli podstatné. Důležité je to, že disk nelze používat bez toho, že by byl naformátován.

Pokud jde o formátování, je terminologie mírně zavádějící. V systému MS-DOS se pod pojmem „formátování“ rozumí i proces vytvoření souborového systému (o němž bude řeč později). Takže se obě tyto procedury označují jediným pojmem – obzvlášť u disket. Když je nutné je rozlišit, používá se pro formátování v pravém slova smyslu termín *nízkoúrovňové formátování* a pro vytvoření souborového systému označení *vysokoúrovňové formátování*. Terminologie používaná v unixovém světě označuje obě tyto činnosti tradičními pojmy „formátování“ a „vytvoření souborového systému“. Nejinak tomu bude i v této knize.

Disky IDE a některé disky SCSI jsou formátovány ve výrobě a není třeba je po připojení formátovat znovu, takže většina lidí se o formátování disků nestará. Vlastní naformátování disku dokonce může disku uškodit, protože některé disky vyžadují zvláštní způsob formátování, který pak umožňuje například automatické přemísťování vadných sektorů.

10 Nicméně úplně jiné.

Disky, které je potřeba formátovat (a které lze formátovat), vyžadují obvykle speciální formátovací programy, protože rozhraní formátovací logiky zabudované v jednotce se liší od jednoho typu disku k druhému. Takovéto formátovací programy jsou často buď součástí řadiče BIOS, nebo běží pouze pod systémem MS-DOS. Ani jeden z těchto prostředků tedy nelze bez problémů použít v systému Linux.

V průběhu formátování můžete narazit na chybná místa na disku, kterým se říká *vadné bloky* nebo *vadné sektory*. S vadnými bloky si někdy poradí samotný řadič pevného disku, ale v případě, že se jich objeví víc, je potřeba nějakým opatřením zamezit možnému použití těchto vadných částí disku. Mechanismus, který problém vadných bloků řeší, je součástí souborového systému. Způsob, jakým systém souborů pracuje s informacemi o vadných sektorech, je popsán v dalších částech této kapitoly. Jinou alternativou je vytvoření malé diskové oblasti, která by obsahovala pouze vadné bloky disku. Takovýto alternativní postup je vhodný zejména v případě, že je rozsah vadných sektorů velmi velký. Souborové systémy totiž mohou mít s rozsáhlými oblastmi vadných bloků problémy.

Diskety se formátují programem **fdformat**. Speciální soubor, který se má formátovat, se zadává jako jeho parametr. Následujícím příkazem bychom například zformátovali 3,5palcovou disketu s vysokou hustotou záznamu, vloženou do první disketové jednotky:

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
$
```

Pamatujte si, že když chcete použít zařízení s autodetekcí (například `/dev/fd0`), *musíte* nejdříve nastavit parametry zařízení pomocí programu **setfdprm**. Stejný výsledek jako v prvním příkladě mají příkazy:

```
$ setfdprm /dev/fd0 1440/1440
$ fdformat /dev/fd0
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... done
$
```

Je obvykle výhodnější vybrat správný speciální soubor, jenž odpovídá typu diskety. Zapamatujte si, že není rozumné formátovat disketu na vyšší kapacitu než je ta, pro kterou je určena.

Program **fdformat** zároveň prověří disketu – zkontroluje, zda neobsahuje vadné bloky. Pokud narazí na vadný blok, opakovaně se pokusí vadný blok použít (obvykle to uslyšíte – zvuky, které jednotka při formátování vydává, se dramaticky změní). Je-li na disketě pouze „dočasná“ chyba (špatná úroveň signálu po zápisu znečištěnou zápisovou hlavou, planý poplach a podobně), program **fdformat** nic neřekne. Naopak skutečná chyba – fyzicky poškozený sektor, přeruší proces kontroly povrchu diskety a jejího formátování. Jádro systému zapíše hlášení do logovacího souboru po každé vstupně-výstupní chybě, na kterou při formátování narazí. Tato hlášení se zároveň objeví na konzole. Když běží program **syslog**, zapisují se tato hlášení také do souboru `/var/log/messages`. Samotným programem **fdformat** uživatel nezjistí, kde přesně se vadný blok nachází (obvykle to ani nikoho nezajímá – diskety jsou tak levné, že se ty vadné automaticky vyhazují).

```
$ fdformat /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
Verifying ... read: Unknown error
$
```

Vadné bloky na disku nebo diskové oblasti (i disketě) lze vyhledat pomocí příkazu **badblocks**. Ten ale neumí disk zformátovat, takže jej lze použít jenom ke kontrole existujících souborových systémů. Níže uvedený příklad hledá chyby na 3,5palcové disketě se dvěma vadnými bloky:

```
$ badblocks /dev/fd0H1440 1440
718
719
$
```

Výstupem příkazu **badblocks** jsou čísla vadných sektorů. Většina souborových systémů umí takovéto vadné bloky označit jako nepoužitelné. Systémy souborů udržují seznam, tabulku vadných sektorů, která se zakládá, když se systém souborů vytváří. Tento seznam pak lze kdykoliv měnit. První kontrola vadných bloků se dělá příkazem **mkfs**, jímž se vytváří souborový systém. Další kontroly se dělají programem **badblocks** a nové chybné bloky se přidávají do seznamu vadných bloků příkazem **fsck**. Příkazy **mkfs** a **fsck** popíšeme později.

Řada moderních disků umí automaticky zjistit výskyt vadných bloků a pokouší se „opravit“ je tak, že místo nich použije k těmto účelům zvlášť vyhrazené správné sektory. Mechanismus náhrady chybného bloku správným je pro operační systém transparentní. Pokud se zajímáte o podrobnosti, měly by být popsány v návodu k disku. Avšak i disky tohoto typu by teoreticky mohly selhat, kdyby počet vadných bloků výrazně vzrostl, nicméně s největší pravděpodobností disk „umře“ z jiných důvodů daleko dříve.

Diskové oblasti

Pevný disk může být rozdělen na několik *diskových oblastí*. Každá disková oblast se chová tak, jako by byla samostatným diskem. Diskové oblasti mají smysl v případě, že máte jeden pevný disk a chcete na něm používat například dva operační systémy – pak disk rozdělíte na dvě diskové oblasti a každý operační systém bude používat vlastní diskovou oblast a nebude zasahovat do druhé. Takto mohou oba operační systémy v klidu a míru koexistovat na jediném disku. Kdybyste nepoužili rozdělení disku na samostatné diskové oblasti, museli byste zakoupit pevný disk pro každý z operačních systémů.

Diskety se na diskové oblasti nerozdělují. Z technického hlediska to možné je, ale vzhledem k tomu, že mají malou kapacitu, by oblasti byly prakticky využitelné jenom v ojedinělých případech. Ani disky CD-ROM se obvykle nedělí na oblasti, protože je praktičtější je používat jako jeden velký disk a skutečně málokdy je potřeba mít na jednom disku CD-ROM uloženo několik operačních systémů.

Hlavní zaváděcí sektor, zaváděcí sektory a partiční tabulka

Informace o tom, jak je pevný disk rozdělen na diskové oblasti, je uložena v prvním sektoru (to jest, prvním sektoru první stopy první vrstvy disku). Tento sektor obsahuje takzvaný *hlavní zaváděcí záznam* (master boot record, MBR). Je to sektor, který se načítá a spouští systémem BIOS pokaždé, když se počítač spustí. Zaváděcí sektor disku obsahuje krátký program, jenž načte tabulku diskových oblastí a zjistí, která oblast disku je aktivní (tedy označená jako zaváděcí). Pak přečte první sektor této oblasti, takzvaný *zaváděcí sektor*. (MBR je také zaváděcí sektor, ale má zvlášt-

ní postavení, a proto i zvláštní označení.) Zaváděcí sektor na diskové oblasti obsahuje další krátký program, který načítá první část operačního systému, jenž je na této diskové oblasti uložený (samozřejmě za předpokladu, že je disková oblast bootovatelná), a spouští jej.

Metoda dělení disku na diskové oblasti není hardwarově implementovaná a není ani součástí systému BIOS. Jde čistě o konvenci podporovanou většinou operačních systémů. Ale ne všechny operační systémy se chovají podle těchto konvencí, jsou i výjimky. Některé operační systémy sice podporují dělení disku na diskové oblasti, ale v rámci své diskové oblasti používají svou vlastní interní metodu dělení. Tyto typy operačních systémů mohou bez jakýchkoliv zvláštních prostředků spolupracovat s jinými systémy (včetně operačního systému Linux). Naopak, takový operační systém, jenž rozdělení disku na diskové oblasti nepodporuje, nemůže koexistovat na stejném pevném disku s jiným operačním systémem.

Je dobré si na kousek papíru vypsat tabulku rozdělení disku na oblasti. Kdyby pak náhodou v budoucnu došlo k poškození některé oblasti, nemusíte díky tomuto jednoduchému bezpečnostnímu opatření přijít o všechna data. (Poškozenou tabulku oblastí lze opravit programem **fdisk**.) Důležité informace získáte příkazem **fdisk -l**:

```
$ fdisk -l /dev/hda

Disk /dev/hda: 15 heads, 57 sectors, 790 cylinders
Units = cylinders of 855 * 512 bytes

        Device Boot    Begin   Start   End     Blocks  Id      system
/dev/hda1              1       1       24      10231+  82      Linux swap
/dev/hda2              25      25      48      10260   83      Linux native
/dev/hda3              49      49      408     153900  83      Linux native
/dev/hda4              409     409     790     163305  5       Extended
/dev/hda5              409     409     744     143611+ 83      Linux native
/dev/hda6              745     745     790     19636+  83      Linux native

$
```

Rozšířené a logické diskové oblasti

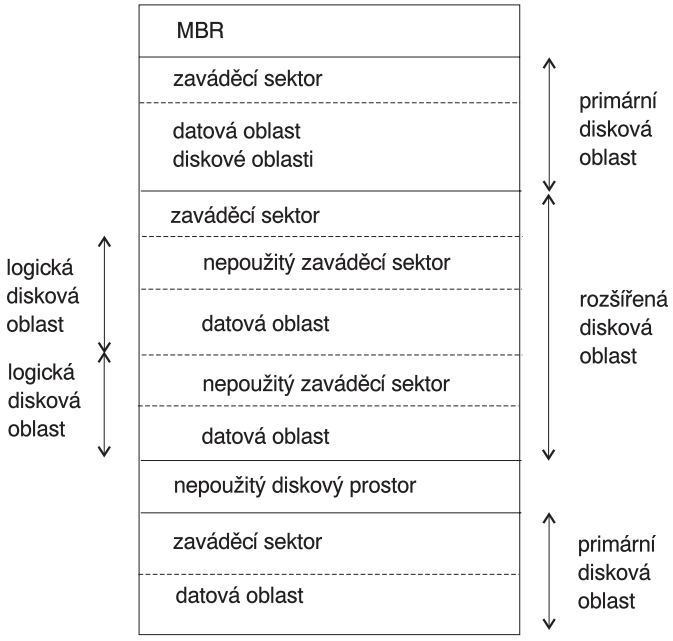
Původní schéma dělení disků počítačů PC na diskové oblasti umožňuje vytvořit pouze čtyři oblasti. To se záhy v praxi ukázalo jako nedostatečné. Z části například proto, že někteří uživatelé chtěli mít na svém počítači i víc než čtyři operační systémy (Linux, MS-DOS, OS/2, Minix, FreeBSD, NetBSD nebo Windows/NT, a to jsme vyjmenovali jenom některé), ale především proto, že je výhodné mít několik diskových oblastí i pro jeden operační systém. Například z důvodů zlepšení odezvy operačního systému je lepší nevyužívat pro odkládací prostor systému Linux hlavní diskovou oblast operačního systému, ale mít odkládací prostor na samostatné diskové oblasti (viz dále).

Aby bylo možné omezení počtu diskových oblastí obejít, byly zavedeny takzvané *rozšířené diskové oblasti*. Tento trik umožňuje rozdělit *primární diskové oblasti* na podoblasti. Takto rozdělená primární oblast je onou rozšířenou oblastí a její části (podoblasti) jsou takzvané *logické diskové oblasti*. Chovají se stejně jako primární[11], ale jsou vytvořeny jiným způsobem. Není mezi nimi ale žádný rozdíl v rychlosti.

Struktura oblastí pevného disku by mohla vypadat například tak, jak je uvedeno na obrázku 4.2. Disk je rozdělen na tři primární diskové oblasti. Druhá z nich je rozdělena na dvě logické. Část

11 Nelogické?

disku nepatří vůbec žádné diskové oblasti. Disk jako celek a každá primární oblast má svůj zaváděcí sektor.



Obrázek 4.2 – Příklad rozdělení pevného disku na diskové oblasti

Typy diskových oblastí

Tabulky rozdělení disku (jedna v MBR a další na rozšířených diskových oblastech) mají vyhrazený jeden bajt pro každou diskovou oblast a ten identifikuje její typ. Snaží se popsat operační systém, který diskovou oblast používá, nebo její účel. Informační bajt by měl zamezit tomu, aby dva operační systémy pracovaly s jednou diskovou oblastí. Avšak ve skutečnosti většina operačních systémů označení typu diskové oblasti ignoruje. I Linux se například vůbec nezajímá o to, jak je určitá disková oblast označena. Dokonce – což je ještě horší, některé operační systémy tento bajt používají nesprávně. Například přinejmenším některé verze systému DR-DOS ignorují nejvyšší bit tohoto bajtu.

Žádný z úřadů pro normalizaci nespecifikoval, co která hodnota bajtu znamená. Některé obecně přijaté hodnoty a odpovídající typy uvádí tabulka 4.1. Obsáhlejší seznam vypíše linuxový program **fdisk**.

Tabulka 4.1 – Typy diskových oblastí (podle programu fdisk)

0	prázdný	40	Venix 80286	94	Amoeba BBT
1	DOS 12-bit FAT	51	Novell?	a5	BSD/386
2	XENIX root	52	Microport	b7	BSDI fs
3	XENIX usr	63	GNU HURD	b8	BSDI swap
4	DOS 16-bit FAT (<32 M)	64	Novell	c7	Syrinx
5	Extended	75	PC/IX	db	CP/M
6	DOS 16-bit FAT (>= 32 M)	80	Old MINIX	e1	DOS access
7	HPFS/NTFS	81	Linux/MINIX	e3	DOS R/O
8	AIX	82	Linux swap	f2	DOS secondary
9	AIX bootable	83	Linux native	ff	BBT
a	OS/2 Boot Manag	93	Amoeba		

Dělení pevného disku na diskové oblasti

Je mnoho programů, které umí vytvářet a mazat diskové oblasti. Součástí většiny operačních systémů je nějaký takový a bývá rozumné používat program, jenž je součástí operačního systému, v jehož prostředí s diskovými oblastmi pracujete, hlavně proto, kdyby dělal něco speciálního, co jiné nedělají. Většinou se tyto programy jmenují **fdisk** (včetně toho, který je součástí distribuce systému Linux) nebo nějak podobně. Detaily týkající se možností linuxového programu **fdisk** podrobně popisuje jeho manuálová stránka. Podobný je příkaz **cfdisk**, který má hezčí, celoobrazovkové uživatelské rozhraní.

V případě, že používáte disky IDE, musí být celá zaváděcí disková oblast (tedy disková oblast, na které jsou uloženy soubory obrazů jádra) na prvních 1024 cylindrech. Tento problém je již vyřešen v posledních verzích LILO. To proto, že při zavádění operačního systému (předtím, než systém přechází do chráněného režimu) se k disku přistupuje přes BIOS a ten neumí pracovat s více než 1024 cylindry. V některých případech je možné používat zaváděcí diskovou oblast, která leží na prvních 1024 cylindrech jenom částečně. To je možné jenom když na prvních 1024 cylindrech budou uloženy všechny soubory, které při inicializaci čte systém BIOS. Vzhledem k tomu, že takovéhoto uspořádání je velmi obtížné dosáhnout, *je lepší je vůbec nepoužívat* – nikdy si totiž nemůžete být jistí, zda změna parametrů jádra systému nebo defragmentace disku nezpůsobí, že systém nebude vůbec možné zavést. Proto se raději vždy ujistěte, že je zaváděcí disková oblast vašeho systému celá na prvních 1024 cylindrech.

Některé nové verze systémů BIOS a disků IDE již umí pracovat i s disky, které mají více než 1024 cylindrů. Máte-li takovýto systém, můžete na tento problém zapomenout. Jestli si v tom nejste zcela jisti, umístěte raději podle doporučení zaváděcí diskovou oblast na prvních 1024 cylindrů.

Každá disková oblast by měla mít sudý počet sektorů, protože souborové systémy Linuxu používají diskové bloky o velikosti 1 kB, tedy dva sektory. Lichý počet sektorů diskové oblasti způsobí, že poslední sektor bude nevyužitý. Nemělo by to způsobit žádné zvláštní problémy, ale není to příliš elegantní. Některé verze programu **fdisk** vás budou na tento stav upozorňovat.

Při změně velikosti diskové oblasti je nejlepší vytvořit zálohu všeho, co chcete na diskovou oblast zachránit (a ještě lepší je zálohovat úplně všechno), pak diskovou oblast smazat, vytvořit novou

a obnovit na ní soubory ze zálohy. Chcete-li zvětšit velikost nějaké diskové oblasti, budete muset pravděpodobně upravit i velikosti (tedy vytvořit zálohy a obnovit soubory) sousedních diskových oblastí.

Vzhledem k tomu, že změny velikostí diskových oblastí jsou dost pracné, je lepší nastavit je správně hned napoprvé. Jinak budete potřebovat efektivní a jednoduchý systém zálohování. Instalujete-li poprvé systém z médií, jež nevyžadují časté zásahy obsluhy (například z CD-ROM – protikladem jsou diskety), je často jednodušší si nejdřív pohrát s různými konfiguracemi. Jelikož zatím na disku nemáte žádná data, která by bylo potřeba zálohovat, není natolik bolestné několikrát změnit velikosti diskových oblastí.

Existuje program pro systém MS-DOS, který se jmenuje fips, a ten umí změnit velikosti diskových oblastí systému MS-DOS bez toho, že by bylo potřeba zálohovat, mazat a obnovovat soubory z těchto diskových oblastí. Pro jiné souborové systémy je to zatím nadále nevyhnutné.

Speciální soubory a diskové oblasti

Každá disková oblast a rozšířená disková oblast má svůj vlastní soubor zařízení. Podle konvence pro konstrukci názvů souborů zařízení se číslo diskové oblasti připojí za jméno celého disku s tím, že 1–4 budou primární disková oblast (podle toho, kolik primárních diskových oblastí bylo vytvořeno) a 5–8 jsou logické diskové oblasti (bez ohledu na to, na které z primárních diskových oblastí jsou vytvořeny). Například `/dev/hda1` je první primární disková oblast prvního pevného disku IDE a `/dev/sdb7` je třetí rozšířená disková oblast na druhém pevném disku SCSI. Více informací najdete v *Seznamu zařízení*.

Souborové systémy

Co jsou to souborové systémy?

Souborový systém tvoří metody a struktury dat, pomocí kterých operační systém udržuje záznamy o souborech na discích a diskových oblastech. Jde tedy o způsob, jakým jsou soubory na disku organizované. Tento termín se ale používá i k označení diskové oblasti nebo disku, na kterém se ukládají soubory, nebo ve významu typu souborového systému. Takže když někdo řekne „mám dva souborové systémy“, může mít na mysli to, že má disk rozdělen na dvě diskové oblasti, nebo to může znamenat, že používá „rozšířený souborový systém“, tedy určitý typ souborového systému.

Rozdíl mezi diskem či diskovou oblastí a souborovým systémem, který je na nich vytvořený, je dost podstatný. Jen některé programy (mezi nimi – zcela logicky – programy, pomocí kterých se souborové systémy vytváří) pracují přímo se sektory disku nebo diskové oblasti. Jestli na nich byl předtím vytvořený systém souborů, bude po spuštění takovýchto programů zničen nebo vážně poškozen. Většina aplikací ale pracuje se souborovým systémem. Proto je nelze použít na diskové oblasti, na které není vytvořen žádný systém souborů (nebo na které je vytvořen souborový systém nesprávného typu).

Předtím, než bude možné diskovou oblast nebo disk použít, je potřeba je inicializovat – musí se na ně zapsat určité datové struktury. Tento proces se označuje jako *vytvoření souborového systému*.

Většina unixových souborových systémů má podobnou obecnou strukturu, v dalších podrobnostech se ale celkem dost liší. Mezi ústřední pojmy patří *superblok*, *inode*, *datový blok*, *adresářový blok* a *nepřímý blok*. Superblok obsahuje informace o souborovém systému jako celku, například jeho velikost (zrovna u této položky závisí přesná hodnota na konkrétním souborovém systému). Inode obsahuje všechny informace o souboru kromě jeho jména. Jméno souboru je uloženo v ad-

resáři společně s odpovídajícím číslem inode. Adresářová položka obsahuje jména souborů a čísla inodů, které tyto soubory reprezentují. Inode dále obsahuje čísla datových bloků, v nichž jsou uložená data souboru, který daný inode zastupuje. V inode je ale místo jenom pro několik čísel datových bloků. Když je jich potřeba víc, je dynamicky alokováno víc místa pro další ukazatele na datové bloky. Tyto dynamicky alokované bloky jsou uloženy v nepřímém bloku. Jak jejich název naznačuje, v případě, že je potřeba najít datový blok, musí se nejdřív najít jeho číslo v nepřímém bloku.

Unixové souborové systémy obvykle umožňují vytvořit v souboru *díru*, a to pomocí programu `lseek` (podrobnosti uvádí příslušná manuálová stránka). Vypadá to tak, že souborový systém předstírá, že je na konkrétním místě souboru uložen blok nulových bajtů, nicméně pro něj není vyhrazen žádný sektor pevného disku (to znamená, že takovýto soubor zabírá o něco méně diskového prostoru). S tím se můžete setkat zvlášť často u malých binárních souborů, sdílených knihoven Linuxu, některých databází a v několika dalších speciálních případech. (Díry jsou implementovány prostřednictvím speciální adresy datového bloku v nepřímém bloku nebo inode. Tato zvláštní adresa znamená, že pro určitou část souboru není alokován žádný datový blok, tedy že je v tomto souboru díra.)

Díry jsou částečně užitečné. Na autorově systému bylo jednoduchými prostředky dokázáno, že tímto způsobem lze potencionálně ušetřit asi 4 MB z celkových 200 MB využitého diskového prostoru. A to je na tomto systému poměrně málo programů a vůbec žádné databázové soubory.

Galerie souborových systémů

Linux podporuje několik typů souborových systémů. Mezi nejdůležitější patří:

minix

Je nejstarší a je považován za nejspolehlivější. Má ale několik omezení – chybí časová razítka, jména souborů mohou být nejvíce 30 znaků dlouhá, souborový systém může mít maximálně 64 MB a další.

xia

Modifikovaná verze souborového systému minix. Nemá omezení v délce jmen souborů a velikosti souborového systému, jinak nepřináší žádné nové rysy. Mezi uživateli není příliš oblíbený, ale jinak má pověst velmi spolehlivého systému.

ext2

Souborový systém, který má nejvíce různých možností ze všech zde uvedených původních souborových systémů pro Linux. V současnosti je také nejpopulárnější. Byl navržen tak, aby byl zpětně kompatibilní, takže nové verze kódu souborového systému nevyžadují nové, opakované vytváření již existujících souborových systémů.

ext

Starší verze ext2, která nebyla zpětně kompatibilní. Lze ji jenom stěží používat v nových instalacích Linuxu – většina uživatelů již přešla na systém souborů ext2.

Kromě toho je podporováno několik souborových systémů jiných operačních systémů, což umožňuje přenášet soubory mezi různými operačními systémy. Tyto cizí, nepůvodní souborové systémy jinak fungují jako původní systémy souborů pro Linux, ale obvykle postrádají některé rysy typické pro Unix, případně mají některá neobvyklá omezení nebo jiné zvláštnosti.

msdos

Souborový systém kompatibilní se souborovým systémem FAT operačního systému MS-DOS (také OS/2 a Windows NT).

umsdos

Rozšiřuje možnosti ovladače souborového systému msdos pro systém Linux. Umí pracovat s dlouhými názvy souborů, zná vlastníky souborů, přístupová práva, odkazy a speciální soubory. To umožňuje používat běžný souborový systém msdos jako by byl originálním linuxovým souborovým systémem. Rovněž není potřeba mít oddělené diskové oblasti pro systémy Linux a MS-DOS.

iso9660

Standardní souborový systém disků CD-ROM. Oblíbené rozšíření „Rock Ridge“ tohoto standardu automaticky zavádí delší jména souborů a další možnosti.

nfs

Síťový souborový systém, který umožňuje sdílení souborových systémů mezi větším počtem počítačů a jednoduchý přístup k souborům každého z nich.

hpfs

Souborový systém operačního systému OS/2.

sysv

Souborové systémy operačních systémů SystemV/386, Coherent a Xenix.

Výběr konkrétního souborového systému závisí na dané situaci. V případě, že jsou kompatibilita nebo jiná omezení nutnou podmínkou výběru některého souborového systému jiného operačního systému, nezbývá než použít tento nepůvodní systém souborů. Pokud nejste ve výběru omezení, bude pravděpodobně nejrozumnější používat ext2, protože má ze všech uvedených systémů souborů nejvíce možností a nemá nedostatky z hlediska výkonu.

Dalším ze souborových systémů je systém souborů `proc`, nejčastěji dostupný prostřednictvím adresáře /proc. Ve skutečnosti ale není souborovým systémem v pravém slova smyslu, i když tak na první pohled vypadá. Systém souborů proc umožňuje přístup k určitým datovým strukturám jádra systému, například k seznamu procesů (odtud jeho jméno). Přizpůsobuje tyto datové struktury tak, že se navenek chovají jako soubory souborového systému. K systému souborů proc pak lze přistupovat všemi běžnými nástroji, které se soubory běžně pracují. Takže kdybyste chtěli znát například seznam všech procesů, zadali byste příkaz

```
$ ls -l /proc
total 0
dr-xr-xr-x     4 root        root        0 Jan 31 20:37 1
dr-xr-xr-x     4 liw         users       0 Jan 31 20:37 63
dr-xr-xr-x     4 liw         users       0 Jan 31 20:37 94
dr-xr-xr-x     4 liw         users       0 Jan 31 20:37 95
dr-xr-xr-x     4 root        users       0 Jan 31 20:37 98
dr-xr-xr-x     4 liw         users       0 Jan 31 20:37 99
-r--r--r--     1 root        root        0 Jan 31 20:37 devices
-r--r--r--     1 root        root        0 Jan 31 20:37 dma
-r--r--r--     1 root        root        0 Jan 31 20:37 filesystems
```

```
-r--r--r--      1 root     root        0 Jan 31 20:37 interrupts
-r--------      1 root     root  8654848 Jan 31 20:37 kcore
-r--r--r--      1 root     root        0 Jan 31 11:50 kmsg
-r--r--r--      1 root     root        0 Jan 31 20:37 ksyms
-r--r--r--      1 root     root        0 Jan 31 11:51 loadavg
-r--r--r--      1 root     root        0 Jan 31 20:37 meminfo
-r--r--r--      1 root     root        0 Jan 31 20:37 modules
dr-xr-xr-x      2 root     root        0 Jan 31 20:37 net
dr-xr-xr-x      4 root     root        0 Jan 31 20:37 self
-r--r--r--      1 root     root        0 Jan 31 20:37 stat
-r--r--r--      1 root     root        0 Jan 31 20:37 uptime
-r--r--r--      1 root     root        0 Jan 31 20:37 version
$
```

(v seznamu bude vždy několik souborů, které neodpovídají žádným procesům. Výstup ve výše uvedeném příkladu byl zkrácen.)

Je důležité si uvědomit, že i když se pro systém proc používá označení „souborový systém“, žádná z jeho částí neleží na žádném z disků. Existuje jenom „v představách“ jádra systému. Kdykoliv k některé části souborového systému proc přistupuje uživatel systému nebo některý z procesů, jádro „předstírá“, že je tato část uložená na disku, ale ve skutečnosti tomu tak není. Takže i když je v souborovém systému proc uložený několikamegabajtový soubor `/proc/kcore`, ve skutečnosti nezabírá na disku žádné místo.

Který souborový systém použít?

Obvykle není moc důvodů používat několik různých souborových systémů. V současné době je nejoblíbenějším souborový systém ext2fs a to je současně pravděpodobně ta nejrozumnější volba. Ve vztahu k zmiňovaným účetním strukturám, rychlosti, (pochopitelně) spolehlivosti, kompatibilitě a vzhledem k různým jiným důvodům by mohlo být vhodné zvolit i jiný systém souborů. Takovéto požadavky je třeba posuzovat případ od případu.

Vytváření souborového systému

Souborové systémy se vytváří a inicializují příkazem mkfs. Je to vlastně vždy jiný program pro každý typ souborového systému. Program **mkfs** je jenom koncové rozhraní, program, který spouští některé další programy, podle typu požadovaného souborového systému. Typ souborového systému se volí přepínačem `-t fstype`.

Programy volané příkazem **mkfs** mají nepatrně odlišné rozhraní příkazové řádky. Běžně používané a nejdůležitější volby jsou uvedeny níže. Více informací najdete v manuálových stránkách.

`-t fstype`	Typ souborového systému.
`-c`	Vyhledat a ošetřit chybné bloky.
`-l filename`	Přečíst seznam vadných bloků ze souboru.

Kdybyste chtěli vytvořit souborový systém ext2 na disketě, zadali byste následující příkazy:

```
$ fdformat -n /dev/fd0H1440
Double-sided, 80 tracks, 18 sec/track. Total capacity 1440 kB.
Formatting ... done
$ badblocks /dev/fd0H1440 1440 / bad-blocks
$ mkfs -t ext2 -l bad-blocks /dev/fd0H1440
```

```
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
$
```

V prvním kroku se disketa formátuje (volba `-n` zakáže její validaci, tedy kontrolu vadných sektorů). Vadné bloky pak vyhledává program **badblocks**, a to s výstupem přesměrovaným do souboru `bad-blocks`. Nakonec se vytvoří souborový systém a mezi účetní struktury se uloží seznam vadných bloků, ve kterém budou všechny vadné sektory, jež našel program **badblocks**.

Místo příkazu **badblocks** je možno použít parametr `-c` programu **mkfs** tak, jak to uvádí následující příklad:

```
$ mkfs -t ext2 -c /dev/fd0H1440
mke2fs 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
360 inodes, 1440 blocks
72 blocks (5.00%) reserved for the super user
First data block=1
Block size=1024 (log=0)
Fragment size=1024 (log=0)
1 block group
8192 blocks per group, 8192 fragments per group
360 inodes per group
Checking for bad blocks (read-only test): done
Writing inode tables: done
Writing superblocks and filesystem accounting information: done
$
```

Volba `-c` programu **mkfs** je sice výhodnější než použití programu **badblocks**, ale příkaz **badblocks** pro kontrolu vadných bloků je vždy nutné použít po vytvoření souborového systému.

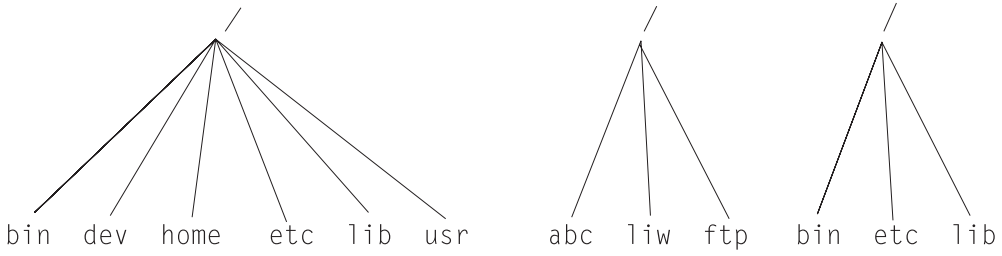
Postup, kterým se vytváří souborové systémy na pevných discích a diskových oblastech, je stejný jako naznačený postup inicializace souborového systému na disketě, s tím rozdílem, že není nutné je formátovat.

Připojení a odpojení

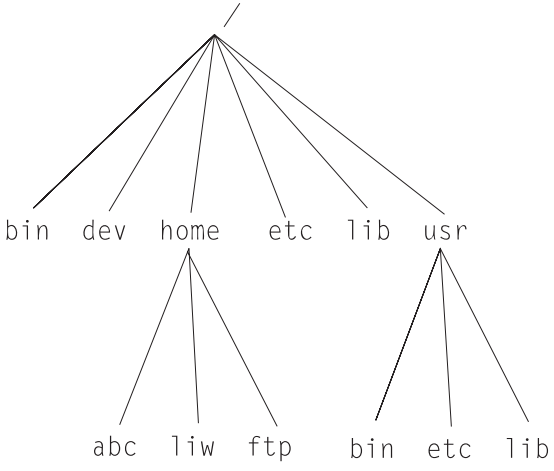
Souborový systém se musí před použitím *připojit*. Po připojení systému souborů dělá operační systém některé účetní operace, kterými se ověřuje funkčnost připojení. Protože všechny soubory v Unixu jsou součástí jediného hierarchického adresářového stromu, operace připojení souborového systému začlení obsah připojovaného souborového systému do některého z adresářů dříve připojeného systému souborů.

Na obrázku 4.3 jsou například zobrazeny tři samostatné souborové systémy, každý se svým vlastním kořenovým adresářem. Když budou poslední dva souborové systémy připojeny pod adresá-

ře `/home` a `/usr` prvního systému souborů, dostaneme jeden adresářový strom, který je vyobrazen na obrázku 4.4.



Obrázek 4.3 – Tři samostatné souborové systémy



Obrázek 4.4 – Připojené souborové systémy `/home` a `/usr`

Souborové systémy lze připojit například zadáním následujících příkazů:

```
$ mount /dev/hda2 /home
$ mount /dev/hda3 /usr
$
```

Příkaz **mount** má dva parametry. Prvním je soubor zařízení odpovídající disku nebo diskové oblasti, na které leží připojovaný souborový systém. Druhým parametrem je adresář, pod nímž bude souborový systém připojen. Po provedení těchto příkazů vypadá obsah obou připojovaných systémů souborů tak, jako by byl součástí hierarchického stromu adresářů `/home` a `/usr`. Říkáme, že „`/dev/hda2` je připojený do adresáře `/home`“ a podobně to platí i pro adresář `/usr`. Když si pak chcete prohlédnout některý ze souborových systémů, procházíte strukturou adresářů, do nichž jsou připojené, jako by to byly jakékoliv jiné adresáře. Je důležité si uvědomit rozdíl mezi souborem zařízení `/dev/hda2` a adresářem `/home`, ke kterému je systém souborů připojen. Speciální soubor zařízení umožňuje přístup k „syrovému“ obsahu pevného disku, kdežto prostřednic-

tvím adresáře `/home` se přistupuje k souborům, které jsou na tomto disku uloženy. Adresář, ke kterému je souborový systém připojený, se nazývá *bod připojení*.

Operační systém Linux podporuje mnoho typů souborových systémů. Příkaz **mount** se vždy pokusí rozeznat typ připojovaného systému souborů. Lze také použít přepínač `-t typ_fs`, který přímo specifikuje typ připojovaného souborového systému. Někdy je to potřeba, protože heuristika programu **mount** nemusí vždy pracovat správně. Chcete-li například připojit disketu systému MS-DOS, zadáte příkaz:

```
$ mount -t msdos /dev/fd0 /floppy
$
```

Adresář, ke kterému se souborový systém připojuje, nemusí být prázdný, ale musí existovat. Nicméně v něm uložené soubory budou po dobu připojení nového souborového systému nedostupné prostřednictvím svých názvů. (Soubory, které byly v době připojení otevřené, budou nadále přístupné. Soubory, na něž směřují pevné odkazy z jiných adresářů, budou přístupné prostřednictvím těchto odkazů.) Nehrozí žádné nebezpečí poškození těchto souborů a někdy to navíc může být i užitečné. Někteří uživatelé například rádi používají synonyma `/tmp` a `/var/tmp`. Proto si dělají symbolický odkaz adresáře `/tmp` do adresáře `/var/tmp`. Předtím, než se při zavádění systému připojí souborový systém `/var`, je adresář `/var/tmp` v kořenovém souborovém systému. Po připojení systému `/var`, bude adresář `/var/tmp` v kořenovém souborovém systému nepřístupný. V případě, že by adresář `/var/tmp` v souborovém systému root neexistoval, bylo by použití dočasných souborů před připojením systému souborů `/var` nemožné.

Jestli nemáte v úmyslu v připojovaném souborovém systému cokoliv zapisovat, zadejte programu mount přepínač `-r`. Tím se vytvoří připojení pouze pro čtení. Jádro systému pak zabrání každému pokusu o zápis do tohoto souborového systému a nebude ani aktualizovat časy posledního přístupu v inodech souborů. Připojení systému souborů pouze pro čtení je podmínkou u médií, na která nelze zapisovat, například u disků CD-ROM.

Pozorný čtenář si již uvědomil drobný logický problém. Jakým způsobem se připojuje první souborový systém, tedy kořenový souborový systém (obsahující kořenový adresář celé stromové struktury), když zjevně nemůže být připojený na jiný souborový systém? Odpověď je jednoduchá – je to kouzlo[12]. Kořenový souborový systém se magicky připojuje při zavádění systému a lze se spolehnout na to, že bude připojený vždy. Kdyby z nějakých důvodů souborový systém root nebylo možné připojit, systém se vůbec nezavede. Jméno souborového systému, jenž se připojuje jako kořenový, je buď zkompilované přímo v jádře systému, nebo se nastavuje zavaděčem LILO, případně programem **rdev**.

Kořenový svazek se obvykle nejdříve připojí pouze pro čtení. Pak inicializační skripty spustí program **fsck**, který jej zkontroluje. Když se neobjeví žádný problém, kořenový svazek se připojí znovu, a to tak, že na něj bude možno i zapisovat. Program **fsck** se nesmí spouštět na připojeném souborovém systému s možností zápisu, protože jakékoliv změny v souborovém systému, ke kterým by došlo při kontrole (a opravách) chyb v souborovém systému) způsobí *vážné* potíže. Když je kořenový souborový systém při kontrole připojený pouze pro čtení, program **fsck** může bez obav opravovat všechny chyby, protože operace opětovného připojení souborového systému vyprázdní všechna metadata, která má souborový systém uložená v paměti.

Na řadě systémů se při startu automaticky připojují i jiné souborové systémy. Jsou specifikované v souboru `/etc/fstab`. Podrobnosti týkající se formátu tohoto souboru najdete v manuálové

12 Podrobnosti viz průvodce jádrem.

stránce k souboru `fstab`. Přesné detaily procesu připojování dalších souborových systémů závisí na množství faktorů a je-li potřeba, může je správce systému nastavit, viz kapitola 6.

Když už není třeba mít souborový systém připojený, je možno jej odpojit příkazem **umount**[13]. Program **umount** má jediný parametr, buďto soubor zařízení, nebo bod připojení. Například odpojení adresářů připojených v předchozím příkladu lze provést příkazy

```
$ umount /dev/hda2
$ umount /usr
$
```

Podrobnější instrukce jak používat příkaz **umount** najdete na manuálové stránce k tomuto programu. Je nutné vždycky odpojit disketovou mechaniku! *Nestačí jenom vytáhnout disketu z mechaniky!* Vzhledem k použití vyrovnávacích pamětí nemusí být data fyzicky zapsána, dokud není disketa odpojena. Předčasné vytažení diskety z mechaniky by mohlo způsobit poškození jejího obsahu. Když z diskety pouze čtete, není její poškození moc pravděpodobné, ale když zapisujete – byť třeba jen omylem – důsledky mohou být katastrofální.

Připojování a odpojování souborových systémů vyžaduje oprávnění superuživatele, takže ho může dělat pouze uživatel root. Je tomu tak například proto, že kdyby měl kterýkoliv z uživatelů právo připojit si jednotku pružných disků na libovolný adresář, nebylo by velmi těžké připojit disketu s virem typu trojského koně „přestrojeného“ za program `/bin/sh`, nebo jiný často používaný příkaz. Avšak dost často je potřeba, aby uživatelé mohli s disketami pracovat. Je několik způsobů, jak jim to umožnit:

- Sdělit uživatelům heslo superuživatele. To je z hlediska bezpečnosti zjevně špatné, avšak to nejjednodušší řešení. Funguje dobře v případě, že vůbec není potřeba zabývat se zabezpečením systému. To se týká řady osobních systémů – tedy systémů, které nejsou připojeny do počítačové sítě.
- Používat programy jako je **sudo**, jenž umožní uživatelům používat příkaz **mount**. Toto je z hlediska bezpečnosti rovněž špatné řešení, avšak tímto způsobem přímo nepředáváte privilegia superuživatele každému[14].
- Umožnit uživatelům používat balík programů **mtools**, jenž umožňuje manipulovat se souborovými systémy MS-DOS bez toho, že by je bylo potřeba připojovat. Řešení funguje dobře pouze v případě, že jsou diskety systému MS-DOS vším, co budou uživatelé systému potřebovat. V ostatních případech je nevyhovující.
- Zapsat všechny disketové jednotky a pro ně přípustné přípojné body spolu s dalšími vhodnými volbami do seznamu připojovaných systémů v souboru `/etc/fstab`.

Posledně uvedenou alternativu lze realizovat přidáním níže uvedeného řádku do souboru `/etc/fstab`:

```
/dev/fd0 /floppy msdos user,noauto 0 0
```

Význam jednotlivých položek (zleva doprava): soubor zařízení, které se má připojit; adresář, kam se má toto zařízení připojit; typ souborového systému; parametry; četnost zálohování (používá program **dump**); posledním parametrem je pořadí kontroly programem `fsck` (určuje pořadí, ve kterém by měly být souborové systémy prověřovány při startu systému; 0 znamená nekontrolovat).

13 Samozřejmě to mělo být **unmount**, ale někdy v 70. letech se *n* záhadně ztratilo a od té doby je nikdo nenašel. Pokud na ně narazíte, pošlete je laskavě na Bell Labs, NJ.

14 Uživatel se nejprve bude muset na několik sekund hluboce zamyslet.

Přepínač `noauto` zakáže automatické připojení při startu systému (tedy nepovolí příkazu **mount –a** toto zařízení připojit). Parametr `user` umožní kterémukoliv uživateli připojit si souborový systém. Z důvodů bezpečnosti zamezí možnosti spouštět programy (jak běžné, tak programy s příznakem setuid) a interpretaci souborů zařízení z připojeného souborového systému. Pak si každý uživatel může připojit disketovou jednotku se souborovým systémem msdos tímto příkazem:

```
$ mount /floppy
$
```

Disketu lze odpojit (a musí být odpojena) odpovídajícím příkazem **umount**.

Chcete-li umožnit přístup k několika různým typům disket, musíte zadat několik přípojných bodů. Nastavení pro každý přípojný bod mohou být různá. Například přístup k disketám s oběma typy souborových systémů – MS-DOS i ext2 – byste umožnili přidáním těchto řádků do souboru `/etc/fstab`:

```
/dev/fd0 /dosfloppy msdos user,noauto 0 0
/dev/fd0 /ext2floppy ext2 user,noauto 0 0
```

U souborových systémů MS-DOS (nejenom na disketách) budete pravděpodobně vyžadovat omezený přístup s využitím systémových parametrů `uid`, `gid` a `umask`. Ty jsou podrobně popsané v manuálové stránce příkazu **mount**. Následkem neopatrnosti při připojování souborového systému MS-DOS může být totiž to, že kterýkoliv uživatel získá oprávnění číst v připojeném systému souborů kterékoliv soubory, což není moc dobré.

Kontrola integrity souborového systému programem fsck

Souborové systémy jsou poměrně složité struktury a tím také mají sklony k chybovosti. Správnost a platnost souborového systému se kontroluje příkazem **fsck**. Lze jej nastavit tak, aby při kontrole automaticky opravoval méně závažné chyby a upozorňoval uživatele na výskyt chyb, které nelze odstranit. Naštěstí je kód implementující souborové systémy odladěný velmi dobře, takže problémy vznikají jen zřídka a jsou obvykle zapříčiněny výpadkem napájecího napětí, selháním technického vybavení nebo chybou obsluhy, například nesprávným vypnutím systému. Většina systémů je nastavena tak, že spouští program **fsck** automaticky při zavádění systému, takže jakékoliv chyby jsou detekovány (a většinou i odstraněny) předtím, než systém přechází do běžného pracovního režimu. Používáním poškozeného systému souborů se totiž jeho stav obvykle ještě více zhoršuje. Jsou-li v nepořádku datové struktury souborového systému, práce se systémem je pravděpodobně poškodí ještě víc, důsledkem čeho mohou být ztráty dat většího rozsahu. Kontrola velkých souborových systémů programem **fsck** chvíli trvá. Když se ale systém vypíná správně, chyby souborových systémů se vyskytují jenom velmi zřídka. Lze pak využít několika triků, díky kterým se lze kontrole (velkých) souborových systémů vyhnout, není-li to nutné. První trik spočívá v tom, že existuje-li soubor `/etc/fastboot`, neprovádí se žádná kontrola. Druhý: souborový systém ext2 má ve svém superbloku speciální příznak, jehož hodnota je nastavena podle toho, jestli byl souborový systém po předchozím připojení odpojen správně, či nikoliv. Podle tohoto příznaku pozná pak program **e2fsck** (verze příkazu **fsck** pro souborový systém ext2), jestli je nutné provádět kontrolu tohoto systému souborů, či nikoliv. V případě, že podle hodnoty příznaku byl daný systém souborů odpojen korektně, kontrola se neprovádí. Předpokládá se, že řádné odpojení systému zároveň znamená, že v souborovém systému nevznikly žádné defekty. To, zda se při proceduře zavádění systému vynechá proces validace systému souborů v případě, že soubor `/etc/fastboot` existuje, záleží na nastavení spouštěcích skriptů systému. Trik s příznakem souborového systému ext2 ale funguje vždy, když systém souborů kontrolujete programem **e2fsck**. Kdybyste totiž chtěli programu **e2fsck** přiká-

zat, aby prověřil i korektně odpojený systém souborů, museli byste tento požadavek explicitně zadat odpovídajícím přepínačem. (Podrobnosti o tom jak, uvádí manuálová stránka programu **e2fsck**.)

Automatické prověřování správnosti souborových systémů se provádí jenom u systémů souborů, které se připojují automaticky při startu operačního systému. Manuálně lze program **fsck** použít pro kontrolu jiných souborových systémů, například na disketách.

Najde-li program **fsck** neodstranitelné chyby, připravte se na to, že budete potřebovat buď hluboké znalosti obecných principů fungování souborových systémů a konkrétního typu poškozeného souborového systému zvlášť, nebo dobré zálohy. Druhou možnost lze jednoduše (ačkoli někdy dost pracně) zajistit. Nemáte-li potřebné „know-how" sami, mohou první alternativu v některých případech zajistit vaši známí, dobrodinci z diskusních skupin o Linuxu na Internetu, popřípadě jiný zdroj technické podpory. Rádi bychom vám poskytli víc informací týkajících se této problematiky, bohužel nám v tom brání nedostatek podrobných znalostí a zkušeností. Jinak užitečný by pro vás mohl být program **debugfs**, jehož autorem je Theodore T'so.

Program **fsck** může běžet pouze na odpojených souborových systémech, v žádném případě ne na připojených (s výjimkou kořenového souborového systému připojeného při zavádění systému pouze pro čtení). To proto, že program přistupuje k „syrovému" disku, a tak může modifikovat systém souborů bez toho, že by využíval operační systém. Když se vám podaří operační systém tímto způsobem „zmást", téměř určitě můžete očekávat problémy.

Kontrola chyb na disku programem badblocks

Je vhodné pravidelně kontrolovat výskyt vadných bloků na disku. Dělá se to příkazem **badblocks**. Jeho výstupem je seznam čísel všech vadných bloků, na které program narazil. Tímto seznamem pak můžete „nakrmit" program **fsck**, který podle něj provede záznamy do datových struktur souborového systému. Podle informací těchto účetních struktur se řídí operační systém a když pak ukládá na disk data, nepokouší se využívat v seznamu uvedené vadné bloky. V následujícím příkladu je naznačen celý postup.

```
$ badblocks /dev/fd0H1440 1440 > bad-blocks
$ fsck -t ext2 -l bad-blocks /dev/fd0H1440
Parallelizing fsck version 0.5a (5-Apr-94)
e2fsck 0.5a, 5-Apr-94 for EXT2 FS 0.5, 94/03/10
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Check reference counts.
Pass 5: Checking group summary information.

/dev/fd0H1440: ***** FILE system WAS MODIFIED *****
/dev/fd0H1440: 11/360 files, 63/1440 blocks
$
```

Je-li v seznamu vadných bloků uveden blok, který je již některým ze souborů využíván, program **e2fsck** se pokusí tento sektor přemístit na jiné místo disku. Když je blok skutečně vadný a nejde jenom o logickou chybu vzniklou při zápisu, bude obsah souboru s největší pravděpodobností poškozený.

Boj s fragmentací

Když se soubor ukládá na disk, nemůže být vždy zapsán do po sobě jdoucích bloků. Soubor, jenž není uložen do sekvenční řady za sebou jdoucích bloků je *fragmentovaný*. Načtení takového souboru pak trvá déle, protože čtecí hlava disku se musí při čtení víc pohybovat. Proto je žádoucí zamezit fragmentaci souborů, i když pro systémy s velkou vyrovnávací pamětí a dopředným čtením nepředstavuje až tak závažnou komplikaci.

Souborový systém ext2 se snaží udržet fragmentaci souborů na minimu. I v případě, že všechny bloky jednotlivých souborů nelze uložit do sektorů, jež jdou po sobě, ukládá je tak, aby byly co nejvíce pohromadě. Systém souborů ext2 navíc vždy efektivně alokuje volné sektory, které jsou nejblíže ke zbylým blokům ukládaného souboru. Používáte-li proto systémy souborů ext2, nemusíte se o fragmentaci příliš starat. Přesto existuje program, jenž umí defragmentovat tento souborový systém – viz v bibliografii uvedený odkaz na program **ext2-defrag**.

Pro systém MS-DOS existuje celá řada defragmentačních programů. Ty přesouvají bloky v souborovém systému tak, aby fragmentaci odstranily. V ostatních souborových systémech se musí defragmentace dělat tak, že se vytvoří záloha souborového systému, který se znovu vytvoří a ze zálohy se obnoví soubory. Doporučení zálohovat souborový systém před jeho defragmentací se týká všech souborových systémů, protože v průběhu procesu defragmentace může dojít k poškození systému souborů i dalším chybám.

Další nástroje pro všechny souborové systémy

Existují i některé další nástroje užitečné pro správu souborových systémů. Program **df** ukazuje volný diskový prostor v jednom či několika souborových systémech. Program **du** ukazuje, kolik diskového prostoru zabírá adresář a všechny soubory v něm uložené. Lze jej s výhodou využít při „honu“ na uživatele, kteří zabírají svými (mnohdy zbytečnými) soubory nejvíc místa na disku.

Program **sync** zapíše všechny neuložené bloky z vyrovnávací paměti (viz část *Vyrovnávací paměť* v kapitole 5) na disk. Jen zřídkakdy je potřeba zadávat jej ručně, automaticky to totiž dělá démon **update**. Má to význam především v případě nečekaných událostí, například když je proces **update** nebo jeho pomocný proces **bdflush** nečekaně ukončen, nebo v případě, že musíte *ihned* vypnout napájení a nemůžete čekat, než se opět spustí program **update**.

Další nástroje pro souborový systém ext2

Kromě programu, kterým se tento systém souborů vytváří (**mke2fs**), a programu pro kontrolu jeho integrity (**e2fsck**), jež jsou přístupné přímo z příkazové řádky, případně přes koncové programy nezávislé na typu souborového systému, existují pro souborový systém ext2 některé další nástroje, jež mohou být při správě systému užitečné.

Program **tune2fs** umí upravit parametry souborového systému. Uvedeme některé z těch významnějších parametrů:

- Maximální počet připojení, po němž program **e2fsck** provede kontrolu souborového systému bez ohledu na to, že je nastaven příznak korektního vypnutí. U systémů, které jsou určeny k vývoji nebo testování, je rozumné tento limit snížit.
- Maximální čas mezi kontrolami integrity. Příkaz **e2fsck** hlídá maximální periodu mezi dvěma kontrolami, a provede opět kontrolu i v případě, že je nastaven příznak korektního vypnutí a souborový systém nebyl připojen vícekrát, než je povoleno. Opakované kontroly lze zakázat.

- Počet bloků vyhrazených pro superuživatele. Souborový systém ext2 rezervuje některé bloky pro superuživatele. Když se pak souborový systém jako celek zaplní, není potřeba nic mazat a systém lze v omezené míře spravovat. Rezervovaný počet bloků je implicitně nastaven na 5 %, což u většiny disků stačí k tomu, aby se zamezilo jejich přeplnění. V případě disket nemá tato rezervace smysl.

Více informací najdete v manuálové stránce programu **tune2fs**.

Program **dumpe2fs** vypisuje informace o souborovém systému typu ext2, většinou z jeho superbloku. Na obrázku 4.5 je příklad výstupu tohoto programu. Některé z těchto informací jsou ryze technické a vyžadují hlubší pochopení problematiky fungování tohoto souborového systému. Většina údajů je ale snadno pochopitelná i pro správce – laiky.

```
dumpe2fs 0.5b, 11-Mar-95 for EXT2 FS 0.5a, 94/10/23
Filesystem magic number: 0xEF53
Filesystem state: clean
Errors behavior: Continue
Inode count: 360
Block count: 1440
Reserved block count: 72
Free blocks: 1133
Free inodes: 326
First block: 1
Block size: 1024
Fragment size: 1024
Blocks per group: 8192
Fragments per group: 8192
Inodes per group: 360
Last mount time: Tue Aug 8 01:52:52 1995
Last write time: Tue Aug 8 01:53:28 1995
Mount count: 3
Maximum mount count: 20
Last checked: Tue Aug 8 01:06:31 1995
Check interval: 0
Reserved blocks uid: 0 (user root)
Reserved blocks gid: 0 (group root)
Group 0:
Block bitmap at 3, Inode bitmap at 4, Inode table at 5
1133 free blocks, 326 free inodes, 2 directories
Free blocks: 307-1439
Free inodes: 35-360
```

Obrázek 4.5 – Příklad výstupu programu dumpe2fs

Program **debugfs** je nástroj pro ladění souborového systému. Umožňuje přímý přístup k strukturám dat souborového systému uloženým na disku, a lze jej proto použít při opravách disků poškozených natolik, že to nezvládne program **fsck**. Lze jej též využít k obnově smazaných souborů. Když ale chcete použít program **debugfs**, je velmi důležité, abyste skutečně věděli, co děláte. V případě, že zcela neporozumíte některé jeho funkci, se totiž může stát, že všechna svá data zničíte.

Programy **dump** a **restore** lze použít při zálohování souborového systému ext2. Jsou to specifické verze tradičních nástrojů pro zálohování používaných v systému Unix, určené speciálně pro systém souborů ext2. Víc informací o zálohování najdete v kapitole 10.

Disky bez souborových systémů

Ne na všech discích nebo diskových oblastech se vytváří souborové systémy. Například disková odkládací oblast nebude používat žádný souborový systém. Dalším příkladem jsou diskety, jejichž mechaniky se mnohdy používají pro emulaci páskové jednotky. Program **tar** nebo jiný archivační nástroj pak zapisuje přímo na „syrový“ disk bez souborového systému. Zaváděcí diskety systému Linux rovněž nemají souborový systém, pouze holé jádro systému.

To, že se na disku (disketě) nevytvoří souborový systém, má výhodu v tom, že lze využít větší část kapacity disku, protože každý souborový systém má vždy nějakou režii. Navíc lze takto dosáhnout větší kompatibility disků s jinými operačními systémy, například soubor s formátem, jenž používá program **tar**, má stejnou strukturu ve všech operačních systémech, kdežto souborové systémy samotné jsou ve většině operačních systémů různé. Další výhodou je, že disky bez souborových systémů lze v případě potřeby použít velmi rychle (odpadá operace vytváření a validace souborového systému). Zaváděcí diskety Linuxu také nutně nemusí obsahovat souborový systém, ačkoliv to možné je.

Dalším z důvodů proč používat „syrová“ zařízení je možnost vytvořit přesný zrcadlový obraz – kopii disku. Když například disk obsahuje částečně poškozený souborový systém, je vhodné předtím, než se pokusíte chybu opravit, udělat přesnou kopii poškozeného disku. Pak není problém začít znova v případě, že při neúspěšném pokusu opravit chybu poškodíte systém souborů ještě víc. Jedním ze způsobů, jak zrcadlovou kopii disku udělat, je použít program **dd**:

```
$ dd if=/dev/fd0H1440 of=floppy-image
2880+0 records in
2880+0 records out
$ dd if=floppy-image of=/dev/fd0H1440
2880+0 records in
2880+0 records out
$
```

První příkaz **dd** uloží přesný obraz diskety do souboru `floppy-image`, druhý zapíše tento obraz na další disketu. (Samozřejmě se předpokládá, že uživatel před zadáním druhého příkazu diskety v mechanice vyměnil. Jinak by byly tyto příkazy k ničemu.)

Přidělování diskového prostoru

Způsoby rozdělování disku na diskové oblasti

Rozdělit disk na diskové oblasti tím nejlepším možným způsobem není vůbec jednoduché. A co je nejhorší – neexistuje univerzálně správný způsob, jak toho dosáhnout. Celý problém totiž komplikuje příliš mnoho různých faktorů.

„Tradiční“ variantou je mít (relativně) malý kořenový souborový systém, jenž obsahuje adresáře `/bin`, `/etc`, `/dev`, `/lib`, `/tmp` a další programy a konfigurační soubory, které jsou potřeba k tomu, aby se systém spustil a běžel. Vše, co je třeba k tomu, aby mohl být systém uveden do chodu, je právě kořenový souborový systém (na vlastní diskovou oblast nebo na samostatném disku). Důvodem tohoto „tradičního“ schématu je fakt, že když je kořenový svazek malý a méně často používaný, je menší pravděpodobnost, že se v případě havárie systému poškodí. S takovýmto uspořádáním je také jednodušší odstranit případné problémy způsobené havárií systému. V dalším kroku se pak vytvoří samostatné diskové oblasti (nebo použijí další disky) pro stromovou strukturu svazku `/usr`, domovské adresáře uživatelů (nejčastěji pod adresářem `/home`) a pro od-

kládací prostor. Tím, že se vyčlení vlastní disková oblast (disk) domovským adresářům, v nichž jsou uloženy soubory jednotlivých uživatelů, se zjednoduší zálohování, protože programy, které jsou uloženy v adresáři `/usr`, obvykle není potřeba zálohovat. V síťovém prostředí je navíc možné adresář `/usr` sdílet mezi několika počítači (například využitím NFS) a tím snížit celkovou potřebu diskového prostoru. Ta by jinak dosahovala několika desítek či stovek megabajtů, násobeno počtem stanic v síti.

Problém několika samostatných diskových oblastí je v tom, že rozdělují celkové množství volného diskového prostoru na mnoho malých částí. V současnosti, kdy jsou disky a (snad) i operační systémy spolehlivější, stále více uživatelů preferuje možnost mít jenom jednu oblast, ve které jsou uloženy všechny soubory. Na druhou stranu zálohování (a obnovování) malých diskových oblastí je ale méně pracné.

U malých pevných disků (předpokládá se, že neděláte zrovna něco jako vývoj jádra systému) je obvykle nejlepší mít jenom jednu diskovou oblast. U velkých pevných disků je pro změnu výhodnější rozdělit je na několik větších oblastí, pouze pro případ, že by se stalo něco, s čím jste při instalaci systému nepočítali. (Uvědomte si ale, že pojmy „malý“ a „velký“ se zde používají v relativním smyslu, jedině vaše konkrétní potřeby diskového prostoru rozhodnou o tom, kde bude ležet jejich hranice.)

Máte-li k dispozici několik disků, je vhodné umístit kořenový souborový systém (včetně adresáře `/usr`) na jeden a domovské adresáře uživatelů na druhý disk.

Je dobré připravit se na malé „experimentování“ s různými způsoby rozdělení disku na oblasti – kdykoliv, ne pouze při první instalaci systému. Je s tím sice dost práce, protože nezbytnou podmínkou je opakovaná instalace systému poté, co se některý pokus nezdaří. Nicméně je to jediný způsob, jak si ověřit správnost rozdělení disku.

Nároky na diskový prostor

Distribuce Linuxu, kterou budete instalovat, vám obvykle nějakým způsobem sdělí, kolik diskového prostoru je potřeba pro různé konfigurace operačního systému. Programy, které se instalují dodatečně, se budou většinou chovat stejně. Díky tomu si můžete udělat představu o nárocích na diskový prostor a naplánovat si jeho rozdělení. Měli byste se ale připravit i na budoucnost a vyhradit si nějaké místo navíc pro věci, na které si vzpomenete později.

Prostor, který byste měli vyčlenit pro soubory uživatelů systému, závisí na tom, co budou dělat. Většina lidí totiž obvykle spotřebuje tolik diskového prostoru, kolik je jenom možné. Nicméně množství, které jim skutečně stačí, je velmi různé. Někteří uživatelé vystačí s psaním textů a spokojeně přežijí i s několika megabajty. Jiní dělají složitou editaci grafických souborů a budou potřebovat gigabajty volného prostoru.

Mimochodem, když budete při odhadech nároků na diskový prostor srovnávat velikosti souborů v kilobajtech nebo megabajtech a diskový prostor v megabajtech, uvědomte si, že tyto dvě jednotky mohou být různé. Někteří výrobci disků rádi tvrdí, že kilobajt je 1000 bajtů a megabajt je 1000 kilobajtů, kdežto zbytek počítačového světa používá pro oba koeficienty číslo 1024. Proto váš 345MB pevný disk bude mít ve skutečnosti pouze 330 MB[15].

O přidělování odkládacího prostoru pojednává odstavec *Alokace odkládacího prostoru* v kapitole 5.

15 Sic transit discus mundi.

Příklady rozvržení diskového prostoru

Autor manuálu dlouho používal 109MB pevný disk. V současné době má k dispozici pevný disk o velikosti 330 MB. V dalším textu bude vysvětleno, jak a proč byly tyto disky rozděleny na jednotlivé diskové oblasti.

Disk o velikosti 109 MB byl velmi často „přerozdělován" mnoha různými způsoby podle toho, jak se měnily konkrétní potřeby a používané operační systémy. Popíšeme dva typické scénáře. Při prvním z nich byly na jednom disku instalovány operační systémy MS-DOS a Linux. První diskový oblast o velikosti asi 20 MB byla vyhrazena pro systém MS-DOS, některý z kompilátorů jazyka C, textový editor, několik dalších utilit a rozpracovaný program. Několik megabajtů volného diskového prostoru zbylo proto, aby nevznikaly pocity klaustrofobie. Odkládacímu prostoru systému Linux bylo vyhrazeno 10 MB na samostatné diskové oblasti a zbytek, tedy 79 MB, na další diskové oblasti byl vyčleněn pro soubory operačního systému Linux. Rozdělovat takovýto prostor na samostatné oblasti pro souborové systémy root, `/usr` a domovské adresáře `/home` nemá praktický význam.

Když pak nebylo třeba systému MS-DOS, změnil jsem rozdělení disku tak, že odkládací prostor používal 12 MB a zbytek byl opět jeden souborový systém.

Disk o velikosti 330 MB lze rozdělit na několik diskových oblastí následujícím způsobem:

5 MB	kořenový souborový systém
10 MB	odkládací oblast
180 MB	systém `/usr`
120 MB	systém `/home`
15 MB	nevyužitá oblast

Neobsazenou diskovou oblast lze využívat na různé experimenty, při nichž je potřeba mít samostatný diskový segment, například při testování různých distribucí Linuxu, nebo srovnávání rychlosti souborových systémů a podobně. Jinak lze neobsazenou diskovou oblast rovněž využít jako odkládací prostor (hlavně v případě, že máte rádi hodně otevřených oken).

Zvětšování diskového prostoru pro Linux

Zvětšení diskového prostoru vyhrazeného pro Linux je jednoduché. Především v případě, že se instalují nové pevné disky (popis instalace disků jde ale nad rámec této knihy). Je-li to nutné, je potřeba disky zformátovat. Pak se podle výše uvedeného postupu vytvoří diskové oblasti a souborový systém a přidají se odpovídající řádky do souboru `/etc/fstab` kvůli tomu, aby se nové disky připojovaly automaticky.

Tipy jak ušetřit místo na disku

Nejlepším způsobem jak ušetřit diskový prostor je vyvarovat se instalování nepotřebných programů. Většina distribucí Linuxu při instalaci nabízí možnost výběru balíků programů, které se mají instalovat. Podle této nabídky byste měli být schopni analyzovat své potřeby a pak pravděpodobně zjistíte, že většinu z nich nebudete potřebovat. Tím se ušetří hodně místa na disku, protože některé z programů a aplikačních balíků jsou dost objemné. I když zjistíte, že budete některou aplikaci nebo i celý balík programů potřebovat, určitě nebudete muset instalovat všechny jejich součásti. Obvykle není potřeba instalovat například on-line dokumentaci, stejně jako některé ze souborů Elisp pro GNU verzi programu Emacs, některé z fontů pro X11 či některé knihovny programovacích jazyků.

V případě, že nemůžete některé balíky programů trvale odinstalovat, můžete využít kompresi. Komprimační programy jako **gzip** nebo **zip** zabalí (a rozbalí) jednotlivé soubory, případně celé skupiny souborů. Program **gzexe** transparentně komprimuje a dekomprimuje programy tak, že to uživatel v běžném provozu nepostřehne (nepoužívané programy se komprimují a rozbalí se, až když jsou potřeba). V současné době se testuje systém DouBle, který transparentně komprimuje všechny soubory v souborovém systému. (Znáte-li program Stacker pro MS-DOS, princip je podobný.)

Správa paměti

Minnet, jag har tappat mitt minne,
är jag svensk eller finne, kommer inte ihÍg[16]
(Bosse Österberg)

V této kapitole jsou popsány hlavní rysy systému správy paměti operačního systému Linux, tedy subsystémy virtuální paměti a diskové vyrovnávací paměti. Kapitola popisuje jejich účel, funkce a další podrobnosti, které správce systému musí mít na paměti.

Co je virtuální paměť?

Operační systém Linux podporuje *virtuální paměť*, to znamená, že používá disk jako rozšíření paměti RAM. Tím se efektivní velikost využitelné paměti odpovídajícím způsobem zvětší. Jádro systému zapisuje obsah právě nevyužívaných paměťových bloků na pevný disk a paměť se tak může využívat pro jiné účely. Když pak přijde požadavek na její původní obsah, bloky z disku se načtou zpět do paměti. To vše probíhá z pohledu uživatele zcela transparentně. Programy běžící pod Linuxem vidí pouze, že je k dispozici hodně paměti a nestarají se o to, že její část je občas uložená na disku. Přirozeně, čtení a zápis na pevný disk je pomalejší (zhruba o tři řády), než využití reálné paměti, takže programy neběží tak rychle. Část disku, která se využívá jako virtuální paměť, se nazývá *odkládací prostor*.

Linux může použít jako odkládací prostor nejen normální soubor uložený v souborovém systému, ale i diskové oblasti. Předností diskových oblastí je rychlost, výhodou odkládacího souboru je jednodušší možnost změny celkové velikosti odkládacího prostoru. Není přitom totiž potřeba měnit rozdělení celého pevného disku, kdy navíc hrozí nutnost kompletní reinstalace systému. Víte-li, jak velký odkládací prostor budete potřebovat, zvolte odkládací prostor na zvláštní diskovou oblast. Pokud si nároky nejste zcela jisti, zvolte nejdřív odkládací prostor v souboru. Když budete systém nějakou dobu používat, budete schopni odhadnout, kolik odkládacího prostoru skutečně potřebujete. Až budete mít ohledně předpokládané velikosti požadovaného odkládacího prostoru jasno, vytvoříte pro něj zvláštní diskovou oblast.

Měli byste rovněž vědět, že Linux umožňuje využívat současně několik odkládacích oblastí, případně několik odkládacích souborů. Takže když potřebujete pouze příležitostně větší množství odkládacího prostoru, je lepší (místo trvale vyhrazené rezervy) nastavit další soubor navíc.

Poznámka k terminologii používané v oblasti operačních systémů: v odborných kruzích se obvykle rozlišuje mezi odkládáním (anglicky *swapping*), tedy zapsáním celého procesu do odkládacího prostoru na disku, a stránkováním (anglicky *paging*), tedy zapisováním pouhých částí pev-

16 Pozn. překladatele: Něco jako „Paměť, ztratil jsem paměť, jsem Švéd nebo Fin, nevzpomínám si...“ (Ale v originále je to prý vtipnější...)

né velikosti (obvykle několik kilobajtů) najednou. Stránkování je obecně výkonnější a je to metoda, kterou používá i operační systém Linux. Tradiční terminologie systému Linux ale používá pojem odkládání (swapping)[17].

Vytvoření odkládacího prostoru na disku

Odkládací soubor je běžný soubor a není pro jádro systému ničím zvláštní. Jediná vlastnost, která má pro jádro význam je, že odkládací soubor nemá díry a že je připraven pro použití programem **mkswap**. Musí být navíc (z důvodů implementace) uložen na lokálním disku, takže nemůže být uložen v souborovém systému, který je připojen pomocí NFS.

Zmínka o dírách je důležitá. Odkládací soubor rezervuje určitý diskový prostor, takže jádro systému pak může rychle odložit stránku paměti bez toho, že by muselo absolvovat celou proceduru alokace diskového prostoru, která se používá pro běžný soubor. Jádro využívá pouze ty sektory, které byly odkládacímu souboru skutečně přiděleny. Protože díra v souboru znamená, že tomuto místu souboru nejsou přiděleny žádné diskové sektory, nemohlo by je jádro dost dobře využít.

Jeden ze způsobů, kterým lze vytvořit odkládací soubor bez prázdných míst, je:

```
$ dd if=/dev/zero of=/extra-swap bs=1024 count=1024
1024+0 records in
1024+0 records out
$
```

kde `/extra-swap` je jméno odkládacího souboru, jehož velikost je uvedena za parametrem `count=`. Ideální je zvolit velikost jako násobek 4, protože jádro systému zapisuje do odkládacího prostoru *stránky paměti*, které jsou 4 kilobajty velké. Nebude-li velikost násobkem 4, může být posledních pár kilobajtů nevyužitých.

Samostatná odkládací disková oblast rovněž není ničím neobvyklým. Vytvoří se stejně, jako každá jiná disková oblast. Jediným rozdílem je, že se používá jako holé zařízení, tedy bez souborového systému. Je dobré označit ji jako typ 82 („Linux swap“). I když to jádro systému striktně nevyžaduje, vnese to do seznamu diskových oblastí řád.

Poté, co vytvoříte diskovou oblast pro odkládací prostor nebo odkládací soubor, je potřeba zapsat na jejich začátek signaturu, která obsahuje některé administrativní informace a s níž jádro pracuje. Provede se to příkazem **mkswap** tímto způsobem:

```
$ mkswap /extra-swap 1024
Setting up swapspace, size = 1044480 bytes
$
```

Odkládací prostor zatím není využitý. Sice existuje, ale jádro systému jej jako virtuální paměť zatím nezná.

Při zadávání příkazu **mkswap** byste měli být velice opatrní, protože program nekontroluje, zda se soubor nebo disková oblast nevyužívá k jiným účelům. *Příkazem **mkswap** proto můžete lehce přepsat důležité soubory nebo celé diskové oblasti!* Naštěstí budete tento příkaz potřebovat pouze když instalujete operační systém.

17 A tak je spousta počítačových teoretiků zbytečně naštvaná.

Využívání odkládacího prostoru

Využívání nově vytvořeného odkládacího prostoru lze zahájit příkazem **swapon**. Příkaz sdělí jádru systému, že odkládací prostor, jehož úplná cesta se zadává jako parametr příkazu, lze od této chvíle používat. Takže když budete chtít začít využívat dočasný soubor jako odkládací prostor, zadáte tento příkaz:

```
$ swapon /extra-swap
$
```

Odkládací prostory lze využívat automaticky poté, co budou zapsány v souboru `/etc/fstab`, například:

```
/dev/hda8   none   swap   sw   0   0
/swapfile   none   swap   sw   0   0
```

Spouštěcí skripty vykonávají příkaz **swapon -a**, jenž zahájí odkládání do všech odkládacích prostorů uvedených v souboru `/etc/fstab`. Takže příkaz **swapon** se obvykle používá jenom když je potřeba použít odkládací prostor navíc.

Příkazem **free** lze monitorovat využívání odkládacích prostorů. Příkaz zobrazí celkové množství odkládacího prostoru, který je v systému využíván:

```
$ free
             total       used       free     shared   buffers
Mem:         15152      14896        256      12404      2528
-/+ buffers:            12368       2784
Swap:        32452       6684      25768
$
```

V prvním řádku výstupu (`Mem:`) se zobrazuje velikost fyzické paměti. Sloupec `total` neukazuje velikost fyzické paměti, kterou využívá jádro systému, ta má obvykle asi jeden megabajt. Ve sloupci `used` je zobrazeno množství využívané paměti (v druhém řádku je vynechána velikost vyrovnávací paměti). Sloupec `free` udává celkové množství nevyužité paměti. Sloupec `shared` ukazuje paměť sdílenou několika procesy – platí čím více, tím lépe. Ve sloupci `buffers` je zobrazena aktuální velikost vyrovnávací paměti.

V posledním řádku (`Swap:`) jsou analogické informace pro odkládací prostor. Když jsou v tomto řádku samé nuly, není odkládací prostor systému aktivovaný.

Stejné informace lze získat příkazem **top**, nebo z údajů v souborovém systému `proc`, přesněji v souboru `/proc/meminfo`. Obvykle je ale dost obtížné získat informace o využití jednoho konkrétního odkládacího prostoru.

Odkládací prostor lze vyřadit z činnosti příkazem **swapoff**. Příkaz pravděpodobně využijete pouze pro vyřazení dočasných odkládacích prostorů. Všechny stránky, které jsou uloženy v odkládacím prostoru, se po zadání příkazu **swapoff** nejdříve načtou do paměti. Když není dostatek fyzické paměti, do které by se načetly, budou uloženy do některého z jiných odkládacích prostorů. Když není ani dostatek virtuální paměti na odložení všech načítaných stránek odkládacího prostoru, jenž má být vyřazen z činnosti, začnou problémy. Po delší době by se operační systém měl zotavit, ale mezitím bude prakticky nepoužitelný. Proto byste měli předtím, než vyřadíte některý odkládací prostor z činnosti, zkontrolovat (například příkazem **free**), jestli máte dostatek volné paměti.

Všechny odkládací prostory, které se aktivují automaticky příkazem **swapon -a**, lze vyřadit z činnosti příkazem **swapoff -a**. Příkaz vyřadí z činnosti odkládací prostory uvedené v souboru `/etc/fstab`. Odkládací prostory přidané ručně zůstanou nadále v činnosti.

Někdy mohou nastat situace, že se využívá příliš mnoho odkládacího prostoru, i když má systém dostatek volné fyzické paměti. Může se to stát například v situaci, kdy jsou v jednom okamžiku velké nároky na virtuální paměť, ale po chvíli je ukončen některý větší proces, který využívá větší část fyzické paměti, a ten paměť uvolní. Odložená data se ale nenačítají do paměti automaticky a zůstanou uložena na disku až do doby, než budou potřeba. Fyzická paměť by tak mohla zůstat dost dlouho volná, nevyužitá. Není potřeba se tím znepokojovat, ale je dobré vědět, co se v systému děje.

Sdílení odkládacího prostoru s jinými operačními systémy

Virtuální paměť používá mnoho operačních systémů. Vzhledem k tomu, že každý ze systémů využívá svůj odkládací prostor jenom když běží (tedy nikdy ne několik systémů současně), odkládací prostory ostatních operačních systémů pouze zabírají místo na disku. Pro operační systémy by bylo efektivnější sdílet jediný odkládací prostor. I to je možné, ale je potřeba si s tím pohrát. V textu *Tips – HOWTO* je uvedeno několik praktických rad, jak implementovat sdílení odkládacího prostoru.

Přidělování odkládacího prostoru

Možná někdy narazíte na radu, že máte přidělovat dvakrát tolik odkládacího prostoru, než máte fyzické paměti. To je ovšem pouhá pověra. Uvádíme proto správný postup:

- Zkuste odhadnout, jaké budete mít nároky na paměť, tedy největší množství paměti, které budete pravděpodobně v jednom okamžiku potřebovat. To je dané součtem paměťových nároků všech programů, které poběží současně.

  Když například chcete, aby běželo grafické uživatelské rozhraní X Window, měli byste mu přidělit asi 8 MB paměti. Kompilátor gcc vyžaduje několik megabajtů (kompilace některých souborů by mohla mít neobvykle velké nároky na paměť, jež by mohly dosáhnout až několika desítek megabajtů, ale běžně by měly stačit zhruba čtyři megabajty). Jádro samotné bude využívat asi jeden megabajt paměti, běžné interprety příkazů a jiné menší utility několik stovek kilobajtů (řekněme asi jeden megabajt dohromady). Není potřeba být v odhadech úplně přesný, stačí udělat velmi hrubý odhad, ale ten by měl být spíše pesimistický.

  Je důležité si uvědomit, že když bude systém současně používat více uživatelů, budou paměť RAM potřebovat všichni. Avšak budou-li současně ten samý program používat dva uživatelé, celková potřeba paměti obvykle nebude dvojnásobná, protože stránky kódu a sdílené knihovny budou v paměti pouze jednou.

  Pro správný odhad nároků na paměť jsou užitečné příkazy **free** a **ps**.

- K odhadu podle předchozího kroku připočítejte nějakou rezervu. To proto, že odhady paměťových nároků programů budou velmi pravděpodobně nedostatečné – je možné, že na některé aplikace, které budete chtít používat, zapomenete. Takto budete mít jistotu, že pro tento případ máte nějaké to místo navíc. Mělo by stačit pár megabajtů. (Je lepší vyčlenit příliš mnoho, než moc málo odkládacího prostoru. Ale není potřeba to přehánět a alokovat celý disk, protože nevyužitý odkládací prostor zbytečně zabírá místo. V dalších částech

této kapitoly bude uvedeno, jak přidat další odkládací prostor.) Vzhledem k tomu, že se lépe počítá s celými čísly, je dobré hodnoty zaokrouhlovat směrem nahoru, řádově na další celé megabajty.

- Na základě těchto výpočtů budete vědět, kolik paměti budete celkem potřebovat. Takže když odečtete velikost fyzické paměti od celkových nároků na paměť, dozvíte se, kolik odkládacího prostoru musíte celkem vyčlenit. (U některých verzí Unixu se musí vyčlenit i paměťový prostor pro obraz fyzické paměti, to znamená, že velikost paměti vypočítaná podle kroku 2 představuje skutečné nároky na odkládací prostor a neodečítá se velikost fyzické paměti.)
- Je-li vypočítaná velikost odkládacího prostoru o hodně větší než dostupná fyzická paměť (více než dvakrát), měli byste raději více investovat do fyzické paměti. Jinak bude výkon systému příliš nízký.

Vždy je dobré mít v systému alespoň nějaký odkládací prostor, i když podle výpočtů žádný nepotřebujete. Linux používá odkládací prostor poněkud agresivněji, snaží se mít tolik volné fyzické paměti, kolik je jenom možné. Linux navíc odkládá paměťové stránky, které se nepoužívají, i když se fyzická paměť zatím nevyužívá. Tím se totiž zamezí čekání na odložení v případě akutní potřeby, data se odkládají dříve, v době, kdy je disk jinak nevyužitý.

Odkládací prostor lze rozdělit mezi několik disků. To může v některých případech zlepšit výkon systému, jenž v tomto směru závisí na relativních rychlostech disků a jejich přístupových modelech. Lze samozřejmě experimentovat s několika variantami, ale mějte vždy na paměti to, že je velmi lehké u takovýchto pokusů pochybit. Neměli byste také věřit tvrzením, že některá z možností je lepší než ostatní, protože to nemusí být vždy pravda.

Vyrovnávací paměť

Čtení z disku[18] je ve srovnání s přístupem k fyzické paměti velmi pomalé. Navíc se v běžném provozu velmi často z disku načítají stejná data několikrát během relativně krátkých časových intervalů. Například v případě elektronické pošty se nejdříve musí načíst došlá zpráva; když na ni chcete odpovědět, načte se ten samý dopis do editoru; pak stejná data načte i poštovní program, který je kopíruje do souboru s došlou, případně odeslanou poštou a podobně. Nebo si zkuste představit, jak často se může zadávat příkaz **ls** na systému s mnoha uživateli. Jediným načtením informací z disku a jejich uložením do paměti do doby až je nebude potřeba, lze zrychlit všechny operace čtení z disku s výjimkou prvního čtení. Paměť vyhrazená pro tyto účely, tedy pro ukládání z disku načítaných a na disk zapisovaných dat, se nazývá *vyrovnávací paměť*.

Paměť je naneštěstí omezený a navíc vzácný systémový prostředek, takže vyrovnávací paměť nemůže být obvykle dost velká (nemohou v ní být uložena úplně všechna data, která by chtěl někdo používat). Když se vyrovnávací paměť zaplní, data, jež se nepoužívala nejdéle se zruší a takto uvolněná vyrovnávací paměť se využije na ukládání nových dat.

Vyrovnávací paměť funguje i při zápisu na disk. Jednak proto, že data, která se zapisují na disk, se velmi často brzo opakovaně načítají (například zdrojový kód nejprve uložíme do souboru a pak jej opět načítá kompilátor), takže je výhodné uložit data zapisovaná na disk i do vyrovnávací paměti. Druhou výhodou je to, že uložením dat do vyrovnávací paměti (aniž by se okamžitě zapsala na disk) se zlepší odezva programu, jenž data zapisuje. Zápis dat na disk pak může probíhat na pozadí bez toho, že by se tím zpomalovaly ostatní programy.

18 Samozřejmě s výjimkou RAM-disku, to je jasné.

Diskové vyrovnávací paměti používá většina operačních systémů (i když je možné, že se jim říká nějak jinak). Ne všechny ale pracují podle výše uvedených principů. Některé fungují jako *write-through*: data se zapisují na disk ihned (ovšem přirozeně zůstanou rovněž uložena ve vyrovnávací paměti). Je-li zápis odložen na pozdější dobu, označují se tyto vyrovnávací paměti jako *write-back*. Tento systém je samozřejmě efektivnější, je ale také o něco více náchylný k chybám. V případě havárie systému, případně výpadku proudu v nevhodném okamžiku, či vytažení diskety z disketové mechaniky předtím, než jsou data čekající ve vyrovnávací paměti na uložení zapsána, se změny uložené ve vyrovnávací paměti obvykle ztratí. Navíc by takováto událost mohla zapříčinit chyby v souborovém systému (je-li na disku či disketě vytvořen), jelikož mezi nezapsanými daty mohou být i důležité změny účetních informací samotného souborového systému.

Proto by se nikdy nemělo vypínat napájení počítače dřív, než správně proběhla procedura zastavení systému (viz kapitola 6). Rovněž by se neměla vytahovat disketa z mechaniky předtím, než byla odpojena příkazem **umount** (byla-li předtím připojená), případně předtím, než program, jenž přistupoval na disketu, signalizuje, že práci s disketovou mechanikou ukončil a než přestane svítit kontrolka mechaniky pružného disku. Příkaz **sync** vyprázdní vyrovnávací paměť, tedy uloží všechna nezapsaná data na disk. Lze jej použít vždy, když chcete obsah vyrovnávací paměti bezpečně uložit na disk. V tradičních systémech Unix existuje program **update**, který běží na pozadí a spouští příkaz **sync** každých 30 sekund. V těchto systémech proto není obvykle potřeba příkaz **sync** používat ručně. V systému Linux běží další démon **bdflush**, jenž dělá něco podobného jako program **sync**, ale častěji a ne v takovém rozsahu. Navíc se tímto způsobem vyhnete náhlému „zamrznutí“ systému, které může někdy příkaz **sync** při větším rozsahu vstupně-výstupních operací způsobit.

V systému Linux se program **bdflush** spouští příkazem **update**. V případě, že je démon **bdflush** z jakéhokoliv důvodu neočekávaně ukončen, jádro systému o tom podá varovné hlášení. Obvykle ale není důvod se ničeho obávat. Proces **bdflush** lze spustit ručně (příkazem **/sbin/update**).

Ve skutečnosti se do vyrovnávací paměti neukládají celé soubory, ale bloky, což jsou nejmenší jednotky při vstupně-výstupních diskových operacích (v Linuxu mají nejčastěji velikost 1 kB). Tímto způsobem se do vyrovnávací paměti ukládají i adresáře, superbloky, další účetní data souborového systému a data z disků bez souborových systémů.

O efektivitě vyrovnávací paměti prvotně rozhoduje její velikost. Malá vyrovnávací paměť je téměř zbytečná. Bude v ní uloženo tak málo dat, že se vyrovnávací paměť vždy vyprázdní předtím, než mohou být data opakovaně použita. Kritická velikost záleží na tom, jaké množství dat se z disků čte a na disky zapisuje a jak často se k těmto údajům opakovaně přistupuje. Jediným způsobem jak to zjistit, je experimentovat.

Má-li vyrovnávací paměť pevnou velikost, není výhodné ji mít příliš velkou. To by mohlo kriticky zmenšit dostupnou systémovou paměť a zapříčinit časté odkládání (jež je rovněž pomalé). Aby byla reálná paměť využívána co nejefektivněji, Linux automaticky využívá veškerou volnou paměť RAM jako vyrovnávací paměť. Naopak, operační systém vyrovnávací paměť automaticky zmenšuje, když běžící programy požadují víc fyzické paměti.

V systému Linux není potřeba dělat nic zvláštního pro to, aby bylo možné vyrovnávací paměť využívat. Systém ji totiž používá zcela automaticky. Kromě správného postupu při zastavení systému a vyjímání disket z mechaniky se prakticky o vyrovnávací paměť vůbec nemusíte starat.

Spouštění a zastavování systému

Start me up
Ah... you've got to... you've got to
Never, never never stop
Start it up
Ah... start it up, never, never, never
You make a grown man cry,
you make a grown man cry
(Rolling Stones)[19]

Tato kapitola popisuje, co se děje po tom, co je systém Linux spuštěn, a jak jej správně zastavit. Při nedodržení správných postupů může dojít k poškození nebo ztrátě souborů.

Zavádění a ukončení práce systému – přehled

Zapnutí počítače a následné zavedení[20] operačního systému se označuje jako *bootování*. Ten termín původně vznikl z anglické fráze „pull yourself up by your own bootstraps", tedy vytáhnout sebe sama za jazyk vlastních bot, což obrazně vystihuje proces, který probíhá při zapnutí počítače – nicméně realita je podstatně méně zábavná.

V průběhu zavádění počítač nejdříve nahraje krátký kód, takzvaný *zavaděč*, jenž pak zavede a spustí samotný operační systém. Zavaděč je obvykle uložen na předem určeném místě pevného disku nebo diskety. Důvodem pro rozdělení procedury zavádění systému do dvou kroků je to, že samotný operační systém je velký a složitý, kdežto samotný zavaděč je velmi krátký (má několik stovek bajtů). Tím se zamezí nežádoucímu komplikování firmware.

19 Pozn. překladatele: Volně přeloženo „Spusť mě / ... nesmíš... nesmíš / nikdy, nikdy, nikdy nesmíš zastavit / rozjeď to / ... rozjeď to, nikdy, nikdy, nikdy / nezklameš /nezklameš". Taky vypínáte počítač jenom když vypnou proud?

20 U starších počítačů nestačilo počítač zapnout, samotné zavedení operačního systému bylo taky nutné provést ručně. To až ty moderní vymyšlenosti dělají všechno samy.

Různé typy počítačů provádí úvodní sekvence zavádění systému různě. Pokud jde o počítače třídy PC, ty (přesněji jejich systém BIOS) načítají první sektor pevného disku nebo diskety, kterému se říká *zaváděcí sektor*. Zavaděč je uložen v tomto prvním – zaváděcím sektoru. Zavaděč pak načítá operační systém z jiného místa na disku, případně i z nějakého jiného média.

Poté, co se operační systém Linux zavede, inicializují se technické prostředky počítače a ovladače jednotlivých zařízení. Pak se spustí proces **init**, který spouští další procesy, umožňující uživatelům přihlásit se do systému a pracovat v něm. O podrobnostech této části zaváděcího procesu se pojednává dále.

Aby bylo možné systém Linux zastavit, je nejdříve nutné požádat všechny procesy, aby ukončily činnost (zavřely všechny otevřené soubory, popřípadě zařídily další důležité věci – obrazně řečeno „uklidily" po sobě). Potom se odpojí souborové systémy, odkládací prostory a nakonec se na konzole objeví zpráva, že lze počítač vypnout. Jestli se nedodrží správný postup, mohou se stát (a obvykle se stanou) dost nepříjemné věci. Nejzávažnějším problémem je v tomto případě nevyprázdněná vyrovnávací disková paměť souborového systému. Všechna data ve vyrovnávací paměti se totiž ztratí, takže souborový systém na disku bude nekonzistentní a pravděpodobně nepoužitelný.

Zavádění podrobněji

Systém Linux lze zavést buď z disket, z pevného disku nebo z CD. Příslušná kapitola Průvodce instalací uvádí návod, jak systém instalovat oběma z výše uvedených způsobů.

Když počítač PC startuje, systém BIOS provádí různé testy a kontroluje, zda je vše v pořádku[21]. Až pak zahájí skutečné zavádění operačního systému. Vybere zaváděcí diskovou jednotku. Typicky první disketovou mechaniku (je-li v mechanice zasunuta disketa), jinak první pevný disk, pokud je v počítači nainstalován. Pořadí prohledávání lze konfigurovat. Poté se načte první sektor tohoto disku, kterému se říká *zaváděcí sektor*, u pevných disků se označuje jako *hlavní zaváděcí sektor*, protože na pevném disku může být několik diskových oblastí, každá se svým vlastním zaváděcím sektorem.

Zaváděcí sektor obsahuje krátký program (dostatečně krátký na to, aby se vešel do jednoho bloku), jehož úkolem je načíst z disku operační systém a spustit jej. Při zavádění systému z diskety obsahuje její zaváděcí sektor kód, který načte jenom prvních několik stovek bloků (v závislosti na aktuální velikosti jádra) do předem určeného místa v paměti. Na linuxové zaváděcí disketě totiž nebývá vytvořen souborový systém, je na ní uloženo jenom jádro systému v několika po sobě jdoucích sektorech, což proces zavádění systému značně zjednodušuje. Systém lze zavést rovněž z diskety se souborovým systémem, a to pomocí zavaděče systému Linux (anglicky LInux LOader, zkráceně LILO).

Když se systém zavádí z pevného disku, kód obsažený v zaváděcím sektoru disku si prohlédne tabulku diskových oblastí, která je v tomto sektoru také uložená. Pak zaváděcí kód identifikuje aktivní diskovou oblast (oblast, která je označená jako zaváděcí), načte zaváděcí sektor této diskové oblasti a spustí kód v něm uložený. Kód v zaváděcím sektoru diskové oblasti pak dělá v podstatě to samé, co kód obsažený v zaváděcím sektoru diskety. Načte jádro systému ze své diskové oblasti a spustí jej. Avšak v detailech se tyto procedury poněkud liší, protože obecně není výhodné mít zvláštní diskovou oblast čistě pro obraz jádra systému, proto kód v zaváděcím sektoru nemůže jednoduše sekvenčně číst data z disku, jako při zavádění systému z diskety. Existuje několik způsobů řešení tohoto problému. Nejběžnější možností je použít zavaděč operačního systému Linux LILO.

21 Tato procedura se označuje jako *Power On Self-Test*, zkráceně *POST*.

(Další detaily tohoto postupu nejsou v této chvíli podstatné. Více informací najdete v dokumentaci zavaděče LILO, která je v tomto směru dokonalejší.)

Když se zavádí operační systém pomocí zavaděče LILO, pokračuje se načtením a spuštěním implicitního jádra. Je také možné nakonfigurovat LILO tak, aby zavedl některý z více obrazů jádra systému, nebo i jiný operační systém než Linux. Uživatel systému si tak při zavádění může vybrat, které jádro, případně operační systém, se implicitně zavede při spuštění počítače. Zavaděč LILO lze nakonfigurovat i tak, že při zmáčknutí kláves Alt, Shift nebo Ctrl v okamžiku zavádění systému (tedy při spouštění LILO) se nebude zavádět systém ihned. Místo toho se zavaděč zeptá, který operační systém se bude zavádět. LILO lze nastavit i tak, že se bude při zavádění systému ptát na požadovaný systém, ale s volitelným časovým prodlením, po kterém se zavede implicitně určené jádro.

Zavaděč LILO rovněž umožňuje předat jádru systému řádkové parametry, které se zadávají za jménem jádra nebo operačního systému, který se zavádí.

Zavádění systému z diskety i z pevného disku má své výhody. Obecně je zavádění z disku příjemnější, uživatel je ušetřen zbytečných nepříjemností v případě, že v disketách „zcela náhodou" zavládne nepořádek. Kromě toho je zavádění z disku rychlejší. Naopak konfigurace pro zavedení operačního systému z disku může být složitější. Proto mnoho uživatelů dává v první fázi přednost zavádění systému z diskety a až pak, když bude nainstalovaný systém fungovat, doinstalují zavaděč LILO a zavádějí systém z pevného disku.

Jakmile se jádro systému načte do paměti (ať už to znamená cokoliv) a dojde k jeho skutečnému spuštění, stanou se přibližně následující věci:

- Jádro Linuxu se instaluje v komprimovaném tvaru, takže se nejprve samo dekomprimuje. Stará se o to krátký program, jenž je obsažen v začátku obrazu jádra.
- Rozezná-li systém kartu Super VGA, která má nějaké zvláštní textové režimy (jako například 100 sloupců na 40 řádků), zeptá se vás, který z režimů budete používat. V průběhu kompilace jádra systému lze videorežim nastavit – pak se systém při startu nedotazuje. Stejného efektu lze dosáhnout konfigurací zavaděče systému LILO nebo příkazem **rdev**.
- V dalším kroku jádro zkontroluje, jaké další hardwarové komponenty jsou k dispozici (pevné disky, diskety, síťové adaptéry a podobě) a odpovídajícím způsobem nastaví některé ze svých ovladačů zařízení. V průběhu této „inventury" vypisuje jádro zprávy o tom, která zařízení byla nalezena. Například při zavádění systému, jenž používá autor, se vypisují tato hlášení:

```
LILO boot:
Loading linux.
Console: colour EGA+ 80x25, 8 virtual consoles
Serial driver version 3.94 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16450
tty01 at 0x02f8 (irq = 3) is a 16450
lp_init: lp1 exists (0), using polling driver
Memory: 7332k/8192k available (300k kernel code, 384k reserved, 176k da-
ta)
Floppy drive(s): fd0 is 1.44M, fd1 is 1.2M
Loopback device init
Warning WD8013 board not found at i/o = 280.
Math coprocessor using irq13 error reporting.
Partition check:
  hda: hda1 hda2 hda3
VFS: Mounted root (ext filesystem).
Linux version 0.99.pl9-1 (root@haven) 05/01/93 14:12:20
```

Přesný formát výstupů se na různých systémech liší, a to v závislosti na hardwarové konfiguraci výpočetního systému, verzi operačního systému a jeho konkrétním nastavení.

- Potom se jádro Linuxu pokusí připojit kořenový svazek. Přípojné místo lze nastavit při kompilaci jádra, později pak příkazem **rdev**, případně zavaděčem LILO. Typ souborového systému je detekován automaticky. Jestli připojení souborového systému selže (například proto, že jste při kompilaci jádra systému zapomněli uvést odpovídající ovladač souborového systému), jádro zpanikaří a systém se v tomto kroku zastaví (beztak mu nic jiného ani nezbývá).

  Kořenový svazek se obvykle připojuje pouze pro čtení (to lze nastavit stejným způsobem, jako místo připojení). Díky tomu se může provést kontrola souborového systému při jeho připojování. Není vhodné prověřovat systém souborů, jenž je připojen pro čtení i zápis.
- Poté spustí jádro systému na pozadí program **init**, jenž se nachází v adresáři `/sbin`. Program **init** bude vždy procesem číslo 1 a jeho úkolem jsou různé operace spojené se startem systému. Přesný postup toho, co jádro systému v tomto kroku dělá, závisí na tom, jak je konfigurováno. Více informací uvádí kapitola 7. Jádro systému v této fázi přinejmenším spustí na pozadí některé nezbytné démony.
- Proces **init** pak přepne do víceuživatelského režimu a spustí procesy **getty** pro virtuální konzoly a sériové linky. Proces **getty** je program, který umožňuje uživatelům přihlásit se prostřednictvím virtuální konzoly nebo sériových terminálů do systému. Proces **init** může rovněž spouštět některé další programy, a to podle toho, jak je konfigurován.
- Poté je zavádění systémů ukončeno a systém normálně běží.

Podrobněji o zastavení systému

I při zastavování operačního systému Linux je důležité dodržovat správný postup. Pokud se správná procedura zastavení systému nedodrží, budou souborové systémy pravděpodobně poškozeny a obsah jednotlivých souborů může být promíchán. To proto, že operační systém využívá diskovou vyrovnávací paměť, jež nezapisuje změny na disk ihned, ale v určitých časových intervalech. To sice významně zvyšuje výkon systému, ale následkem toho je, že pokud se z ničeho nic vypne napájení ve chvíli, kdy vyrovnávací paměť obsahuje množství nezapsaných změn, mohou být data na disku nekonzistentní a souborový systém zcela nefunkční, protože se na disk zapsala jenom některá změněná data.

Dalším z argumentů proti tvrdému vypnutí počítače je to, že v systému ve víceuživatelském režimu může běžet hodně programů na pozadí. Vypnutí ze sítě by pak mohlo mít katastrofální důsledky. Tím, že se při zastavení systému dodržuje korektní postup, se zajistí, že všechny procesy běžící na pozadí svá data včas uloží.

Běh systému Linux se správně ukončí příkazem **shutdown**. Zadává se obvykle jedním ze dvou způsobů.

Když jste přihlášeni v systému jako jediný uživatel, je potřeba před zadáním příkazu **shutdown** ukončit všechny běžící programy, odhlásit se ze všech virtuálních konzolí a přihlásit se na jednu z nich jako superuživatel. Pokud už tak jste přihlášení, je rozumné přepnout se buď do kořenového adresáře nebo do svého domovského adresáře, aby se předešlo problémům při odpojení souborových systémů. Pak můžete zadat příkaz **shutdown -h now**. (Místo parametru `now` můžete zadat symbol plus a nějaké číslo, pak se zastavení systému zpozdí o zadaný počet minut – ve většině případů totiž nejste jediný uživatel systému.)

Druhou alternativou – je-li v systému přihlášeno více uživatelů – je použití příkazu **shutdown -h +*time message***, kde *time* je čas v minutách zbývající do zastavení systému a *message* je stručné sdělení důvodu tohoto opatření.

```
# shutdown -h +10 'Bude instalován nový disk. Systém by
> měl být opět spuštěn za tři hodiny.'
#
```

Takto můžete každého uživatele varovat, že systém bude za deset minut zastaven a že bude lepší se odpojit, než ztratit neuložená data. Varování se zobrazí na každém terminálu, na kterém je někdo přihlášený, včetně všech terminálů **xterm** systému X Window:

```
Broadcast message from root (ttyp0) Wed Aug 2 01:03:25 1995...
Bude instalován nový disk. Systém by měl být
opět spuštěn za tři hodiny.
The system is going DOWN for system halt in 10 minutes !!
```


Varování se automaticky opakuje několik minut před zastavením systému v kratších a kratších intervalech, až stanovený čas vyprší.

Když se pak po určeném časovém prodlení rozjede procedura skutečného zastavení systému, odpojí se nejdříve všechny souborové systémy (kromě kořenového), uživatelské procesy (je-li někdo stále přihlášen) se ukončí, běžící démoni se zastaví, všechny připojené svazky se odpojí, obrazně řečeno – všechno ustane. Poté proces **init** vypíše zprávu, že lze počítač vypnout. Až pak lze sáhnout na síťový vypínač.

V některých případech (ovšem zřídkakdy u správně nakonfigurovaného systému) není možné zastavit systém korektně. Například když jádro systému zpanikaří, havaruje a nefunguje správně, může být zcela nemožné zadat jakýkoliv další příkaz. Pochopitelně, v takovéto situaci je správné zastavení systému poněkud obtížné. Nezbývá než doufat, že se neuložené soubory až tak moc nepoškodí a vypnout napájení. Když nejsou potíže až tak vážné (řekněme, že někdo jenom sekerou rozsekl klávesnici vašeho terminálu) a jádro systému i program **update** stále normálně běží, je obvykle dobré pár minut počkat (dát tím programu **update** šanci vyprázdnit vyrovnávací paměti) a potom jednoduše vypnout proud.

Někteří uživatelé rádi používají při zastavení systému příkaz **sync**[22] zadaný třikrát po sobě, pak počkají, než se ukončí diskové vstupně-výstupní operace a vypnou napájení. Neběží-li žádné programy, je tento postup téměř ekvivalentní zadání příkazu **shutdown** s tím rozdílem, že se neodpojí žádný ze souborových systémů, což ale může způsobit problémy s nastavením příznaku korektního ukončení u souborových systémů ext2fs. Proto se metoda trojího zadání příkazu sync *nedoporučuje*.

(Pokud vás zajímá *proč* zrovna třikrát – v ranných dobách Unixu se všechny příkazy musely naťukat zvlášť, takže než to člověk udělal potřetí, bylo obvykle dost času na to, aby se ukončila většina diskových vstupně-výstupních operací.)

Znovuzavedení systému

Pod znovuzavedením operačního systému se rozumí jeho opakované zavedení, tedy jeho správné zastavení, vypnutí a opětovné zapnutí napájení počítače. Jednodušší cestou je žádost programu **shutdown** o znovuzavedení systému (místo jeho pouhého zastavení), a to použitím parametru **-r**, tedy například zadáním příkazu **shutdown -r now**.

22 Příkaz **sync** ukládá obsah vyrovnávacích pamětí.

Většina systémů Linux vykonává příkaz **shutdown -r now** i v případě, že se na systémové klávesnici současně zmáčknou klávesy Ctrl-Alt-Del. Tato trojkombinace obvykle vyvolá znovuzavedení operačního systému. Reakci na stisk kláves Ctrl-Alt-Del lze nastavit, a na víceuživatelském počítači by například bylo lepší před znovuzavedením systému povolit určité časové prodlení. Naopak systémy, které jsou fyzicky přístupné komukoliv, by bylo lepší nastavit tak, aby se při zmáčknutí kombinace kláves Ctrl-Alt-Del nedělo vůbec nic.

Jednouživatelský režim

Příkaz **shutdown** lze použít k přepnutí systému do jednouživatelského režimu. V něm se do systému nemůže přihlásit nikdo jiný než superuživatel, jenž může používat konzolu systému. Jednouživatelský mód lze s výhodou využít při plnění některých úkolů spojených se správou systému, které nelze dělat, pokud systém běží v normálním (víceuživatelském) režimu.

Záchranné zaváděcí diskety

Občas se stává, že není možné při zapnutí počítače zavést systém z pevného disku. Například tím, že uděláte chybu při nastavování parametrů zavaděče systému LILO, můžete zavinit, že systém Linux nebude možné zavést. V těchto situacích by přišel vhod nějaký jiný způsob zavedení systému, jenž by fungoval vždy (když samozřejmě funguje hardware). Pro počítače PC je takovou alternativou zavádění systému z diskety.

Většina distribucí operačního systému Linux umožňuje vytvořit takzvanou *záchrannou zaváděcí disketu* již při instalaci systému. Doporučujeme to udělat. Avšak některé takovéto záchranné diskety obsahují pouze jádro systému a předpokládá se, že při odstraňování vzniklých problémů budete používat programy uložené na instalačních médiích distribuce systému. Někdy ale tyto programy nestačí. Například v případě, že budete muset obnovit některé soubory ze záloh, jež jste dělali programem, který není na instalačních discích distribuce systému.

Proto by si správce měl vytvořit vlastní zaváděcí diskety, přizpůsobené konkrétním potřebám. Pokyny jak na to jsou obsaženy v příručce *Bootdisk HOWTO* Grahama Chapmana. Musíte mít přirozeně neustále na paměti, abyste měli záchranné zaváděcí diskety stále aktuální.

Disketovou mechaniku, která se využívá pro připojení superuživatelské zaváděcí diskety, nebudete moci použít k čemukoliv jinému. Tento problém může být obzvlášť nepříjemný v případě, že máte jenom jednu disketovou mechaniku. Když ale máte dostatek paměti, můžete nastavit zavádění z diskety tak, aby se obsah tohoto superuživatelského zaváděcího disku načítal do virtuálního ramdisku (je proto nutné zvlášť nakonfigurovat jádro systému na zaváděcí disketě). Když se podaří načíst superuživatelskou zaváděcí disketu na ramdisk, lze disketovou mechaniku využít pro připojení jiných disket.

init

Uuno on numero yksi[23]

Tato kapitola popisuje proces **init**, jenž je vždy prvním uživatelským procesem, který spouští jádro systému. Proces **init** má mnoho důležitých povinností. Spouští například program **getty** umožňující uživatelům přihlásit se do systému, implementuje úrovně běhu systému, stará se o osiřelé procesy a podobně. V této kapitole bude vysvětleno, jak se proces **init** konfiguruje a jak se zavádí různé úrovně běhu systému.

Proces init přichází první

Proces **init** je jedním z programů, jež jsou sice absolutně nezbytné k tomu, aby operační systém Linux fungoval, ale kterým obvykle nemusíte věnovat přílišnou pozornost. Součástí každé dobré distribuce systému je i předem nastavená konfigurace procesu **init**, která vyhovuje většině systémů. Správce pak už obvykle nemusí kolem procesu **init** nic dělat. O proces **init** se většinou staráte jenom pokud připojujete nové sériové terminály, modemy pro příchozí volání (takzvané *dial-in* modemy, nikoliv tedy modemy, pomocí kterých se budete připojovat do jiných systémů, to jsou *dial-out* modemy) a když potřebujete změnit implicitní úroveň běhu systému.

Když se zavede jádro systému (načte se do paměti, spustí se, inicializuje ovladače zařízení, datové struktury a podobně), ukončí svou roli v proceduře zavádění operačního systému tím, že spustí první program uživatelské úrovně – proces **init**. Takže program **init** je vždy prvním spuštěným procesem, má tedy vždy číslo procesu 1.

Jádro systému hledá z historických důvodů proces **init** na několika místech. Jeho správné umístění v systému Linux nicméně je `/sbin/init`. Nenajde-li jádro program **init**, pokusí se spustit program `/bin/sh` a pokud neuspěje, skončí neúspěšně i celá procedura startu systému.

Když proces **init** nastartuje, dokončí proces zavedení systému tím, že se provede několik administrativních úkolů, například kontrola souborových systémů, úklid v adresáři `/tmp`, start různých služeb a spuštění procesu **getty** pro každý terminál nebo virtuální konzolu, prostřednictvím nichž se uživatelé mohou přihlašovat do systému. (Viz kapitola 8.)

Po správném zavedení systému proces **init** po každém odhlášení uživatele restartuje procesy **getty** pro příslušný terminál. Umožní tím další přihlášení jiných uživatelů. Program **init** si také osvojuje všechny osiřelé procesy. Když některý z procesů spustí další proces (svého potomka) a později ukončí svou činnost dřív než potomek, vzniká sirotek, který se ihned stává potomkem procesu **init**. Adopce sirotků má význam především z různých technických důvodů, je ale dobré o ní

23 Pozn. překladatele: Česky *Uuno je jedničk*a, přičemž Uuno je figurka z jakéhosi finského seriálu, a vtip prý je v tom, že *uuno* zní podobně jako *uno*, což je italsky jedna.

vědět, protože je pak snadnější pochopit význam položek seznamu procesů a grafů hierarchického stromu běžících procesů[24].

Správce systému má na výběr několik verzí programu **init**. Většina distribucí operačního systému Linux používá program **sysvinit** (autorem programu je Miquel van Smoorenburg), jehož předlohou je program **init** pro Unix System V. Unix verze BSD používá odlišný program **init**. Primárním rozdílem mezi uvedenými verzemi programů jsou úrovně běhu systému. Unix System V je implementuje, kdežto verze BSD nikoliv (alespoň v klasické verzi). Tento rozdíl není podstatný, a tak se podíváme pouze na program **sysvinit**.

Konfigurace procesu init pro spouštění programu getty – soubor `/etc/inittab`

Když proces **init** startuje, načítá konfigurační soubor `/etc/inittab`. Když pak systém Linux běží, proces **init** tento konfigurační soubor opakovaně načítá pokaždé, když přijme signál HUP[25]. Tato vlastnost umožňuje měnit konfiguraci programu **init** a zajistit, aby se takováto změna projevila i bez toho, že by bylo nutné znovu zavést systém.

Soubor `/etc/inittab` je trochu složitější. Začneme tedy s jednoduchým příkladem konfigurace programu **getty**. Jednotlivé řádky souboru `/etc/inittab` sestávají ze čtyř polí oddělených dvojtečkou:

```
id:úrovně_běhu:akce:proces
```

Uvedené položky budou popsány níže. Soubor `/etc/inittab` může kromě řádkových záznamů obsahovat i prázdné řádky a řádky začínající znakem #. Ty program **init** ignoruje.

id — První položka identifikuje každý z řádků konfiguračního souboru. U řádků procesů **getty** se tak blíže specifikuje terminál, který daný proces obsluhuje (rozlišuje se číslem následujícím za příslušným názvem speciálního souboru `/dev/tty`). V řádcích pro ostatní procesy nemá toto pole žádný význam (kromě jeho omezení v délce), avšak mělo by být v celém souboru jedinečné.

úrovně_běhu — Úrovně běhu systému, které připadají pro daný řádek (proces) v úvahu. Úrovně se zadávají jako číslice bez oddělovače. Jednotlivé úrovně běhu systému budou popsány v následujícím odstavci.

akce — Akce, jež se má provést, například `respawn` (opakovaně spustí příkaz, jenž je uveden v dalším poli pokaždé, když je z různých důvodů ukončen), nebo `once` (spustí daný příkaz jenom jednou).

proces — Příkaz, který se má spustit.

Chcete-li spustit program **getty** pro první virtuální terminál (`/dev/tty1`) ve všech běžných víceuživatelských úrovních běhu (2–5), vložte do souboru `/etc/inittab` tento řádek:

```
1:2345:respawn:/sbin/getty 9600 tty1
```

První pole identifikuje řádek pro zařízení `/dev/tty1`. Druhá položka určuje, že proces uvedený v posledním poli lze spouštět na úrovni běhu systému číslo 2, 3, 4 a 5. Třetí pole znamená, že

24 Samotný proces **init** nesmí skončit. Nepodaří se vám jej ukončit ani signálem SIGKILL.

25 Zašlete jej například příkazem **kill –HUP 1** jako superuživatel.

tento příkaz by měl být vždy opakovaně spuštěn poté, co se ukončí (aby se po odhlášení uživatele mohl přihlásit kdokoliv jiný). V posledním poli je příkaz, který spouští proces **getty** pro první virtuální terminál[26].

Když budete potřebovat přidat do systému další terminály nebo modemové linky pro příchozí volání, měli byste rozšířit soubor `/etc/inittab` o další řádky, vždy jeden pro každý terminál, respektive modemovou linku. Více informací najdete v manuálových stránkách **init**, `inittab` a **getty**.

Nespustí-li se úspěšně příkaz uvedený v souboru `/etc/inittab` a je-li v konfiguraci procesu **init** nastavený jeho restart (akce `respawn`), bude proces zabírat značnou část systémových zdrojů. Proces **init** totiž požadovaný proces spustí, ten se neprovede úspěšně a ukončí se, **init** jej opakovaně spustí, proces se ukončí, **init** jej spustí, proces se ukončí a tak dál, až do nekonečna. Této situaci se předchází tím, že si proces **init** vede záznamy o tom, jak často se pokoušel určitý příkaz spustit. Je-li frekvence opakovaných pokusů o spuštění procesu příliš vysoká, **init** před dalším pokusem o provedení příkazu vyčká pět minut.

Úrovně běhu systému

Úrovní běhu systému se rozumí určitý stav procesu init i celého systému. Tento stav určuje, které ze služeb se poskytují. Jednotlivé úrovně se rozlišují čísly, uvedenými v tabulce 7.1. Pokud jde o uživatelsky definované úrovně (2 až 5), neexistují obecně platná pravidla pro jejich používání. Někteří správci systémů pomocí těchto úrovní určují, které subsystémy se spustí, zdali například poběží X, jestli bude systém připojený k síti a podobně. Jiní dávají přednost inicializaci všech subsystémů na všech uživatelsky definovaných úrovních, popřípadě některé ze subsystémů spouští a zastavují samostatně bez toho, že by se měnily úrovně běhu systému. To proto, že počet úrovní je poměrně malý a řízení konfigurace takovýchto systémů pomocí jednotlivých uživatelsky definovaných úrovní běhu systému je tedy příliš hrubé. Správce systému se v této otázce musí rozhodnout sám, avšak nejjednodušší bude řídit se způsobem, jenž implementuje distribuce, ze které byl systém Linux instalován.

Tabulka 7.1 – Čísla úrovní běhu

0	Zastavení systému.
1	Jednouživatelský režim (pro administraci systému).
2-5	Běžný provoz (definováno uživatelem).
6	Znovuzavedení systému.

Úrovně běhu systému se konfigurují v souboru `/etc/inittab` řádkem podobným tomuto:

```
l2:2:wait:/etc/init.d/rc 2
```

V prvním poli je uvedeno libovolné návěstí, druhé pole znamená, že tento záznam platí pro úroveň běhu systému číslo 2. Třetí položka (pole `wait`) říká procesu **init**, aby spustil příkaz uvedený ve čtvrtém poli jenom jednou, a to při startu dané úrovně běhu systému a pak vyčkal, než se příkaz provede. Samotný příkaz `/etc/init.d/rc` pak spustí všechny procesy a příkazy, které jsou potřebné pro spuštění a ukončení služeb, jimiž se implementuje úroveň běhu číslo 2.

26 Různé verze programu **getty** používají různé parametry. Podrobnosti zjistíte na správné manuálové stránce.

Všechnu dřinu spojenou s nastavováním určité úrovně běhu dělá samotný příkaz uvedený ve čtvrtém poli záznamu. Spouští služby, které zatím neběží, a pozastaví ty, které by na nové úrovni běhu již neměly být poskytovány. Záleží na konkrétní distribuci operačního systému Linux, který z příkazů bude v posledním políčku souboru `/etc/inittab` přesně uveden a které úrovně běhu budou v této konfiguraci implementovány.

Když proces **init** startuje, hledá ten řádek v souboru `/etc/inittab`, jenž specifikuje implicitní úroveň běhu systému:

```
id:2:initdefault:
```

Proces **init** lze také požádat o to, aby se spustil v jiné než běžné úrovni běhu, a to tak, že předáte jádru systému řádkový parametr `single` nebo `emergency`. Parametry lze jádru předat například programem LILO. Tímto způsobem spustíte systém na úrovni 1.

Když už systém běží, lze změnit aktuální úroveň běhu příkazem **telinit**. V případě, že se úroveň běhu systému mění, proces **init** spouští ten příkaz v souboru `/etc/inittab`, jenž odpovídá nové úrovni běhu systému.

Zvláštní konfigurace v souboru `/etc/inittab`

Soubor `/etc/inittab` má některé speciální funkce, jež umožňují procesu **init** reagovat i na některé zvláštní situace. Tyto speciální funkce definují zvláštní klíčová slova ve třetím poli záznamu konfiguračního souboru. Několik příkladů:

`powerwait` Umožní procesu **init** v případě výpadku napájení zastavit systém. Předpokládá se, že systém používá záložní zdroj UPS a software, jenž sleduje UPS a informuje proces **init** o případném výpadku napájení.

`ctrlaltdel` Nařizuje procesu **init** znovu zavést systém, když uživatel současně zmáčkne klávesy Ctrl-Alt-Del. Uvědomte si, že správce systému může reakci na tuto kombinaci nastavit tak, že se místo rebootu provede nějaká jiná akce, že se například – zvlášť když má k systému přístup širší veřejnost – tato klávesová zkratka ignoruje. (Nebo se spustí program **nethack**.)

`sysinit` Příkaz spouštěný při startu systému. Typicky se takto zajistí například smazání adresáře `/tmp`.

Výše uvedený seznam klíčových slov není úplný. Další možnosti i podrobnosti týkající se těch, o kterých se zmiňujeme, uvádí manuálová stránka souboru `inittab`.

Zavádění systému v jednouživatelském režimu

Důležitou úrovní běhu systému je takzvaný jednouživatelský režim (úroveň běhu číslo 1), v němž může počítač používat pouze správce systému. V tomto režimu běží jenom minimum systémových služeb (včetně možnosti přihlášení do systému). Jednouživatelský režim je nutný pro některé úlohy spojené s údržbou systému[27]. Například kontrola konzistence svazku `/usr` programem **fsck** vyžaduje, aby byla disková oblast se souborovým systémem odpojená. Toho ale nelze dosáhnout, pokud nejsou ukončeny téměř všechny systémové služby.

27 Asi jej nebudete používat, když si budete chtít zahrát **nethack**.

Běžící systém lze přepnout do jednouživatelského režimu příkazem **telinit** a požadavkem na přechod do úrovně běhu 1. Při zavádění systému lze do jednouživatelského režimu přejít zadáním parametru `single` nebo `emergency` na příkazové řádce jádra systému. Jádro předá parametry příkazové řádky procesu **init**. Program **init** podle tohoto parametru pozná, že nemá použít implicitní úroveň běhu. (Způsob, kterým se zadává parametr příkazové řádky jádra systému, je podmíněn způsobem, jakým se zavádí operační systém.)

Někdy je potřeba zavést systém v jednouživatelském režimu například proto, aby bylo možné ručně spustit program **fsck** dřív, než se systém pokusí připojit poškozený systém souborů `/usr`, nebo se s ním pokusí jiným způsobem manipulovat. Jakékoliv aktivity na defektním svazku jej s největší pravděpodobností poškodí ještě více, proto by se měla kontrola programem **fsck** udělat co nejdříve.

Zaváděcí skripty, které spouští proces **init**, automaticky přechází do jednouživatelského režimu pokaždé, když automatická kontrola programu **fsck** při zavádění systému neproběhne úspěšně. Pokouší se tak zabránit systému použít souborový systém, jenž je poškozený natolik, že jej nelze automaticky opravit zmiňovaným programem **fsck**. Takovéto poškození svazku je relativně málo frekventované a jeho příčinou bude pravděpodobně mechanické poškození pevného disku, případně nějaká chyba v experimentální verzi jádra systému. Bude ale lepší, když budete jako správce systému připraven i na tuto situaci.

Správně nakonfigurovaný systém se před spuštěním příkazového interpretu v jednouživatelském režimu zeptá na přístupové heslo superuživatele. Je to důležité bezpečnostní opatření, protože jinak by bylo možné jednoduše zadat vhodný parametr příkazové řádky zavaděči systému LILO a dostat se tak k systému s oprávněním superuživatele. (Takto nastavený systém se samozřejmě nezavede, když bude důsledkem defektů na systémovém svazku soubor `/etc/passwd` poškozený. Pro tento případ je dobré mít někde po ruce zaváděcí diskety.)

KAPITOLA 8

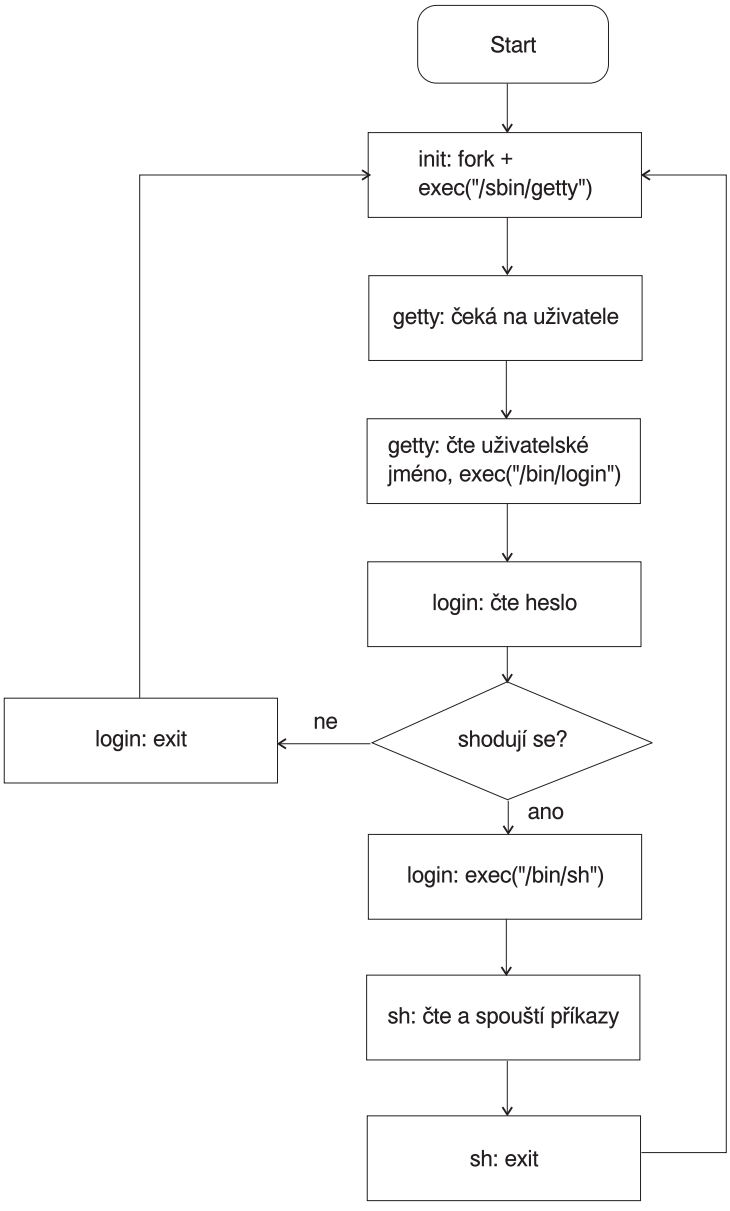
Přihlašování a odhlašování

Nestojím o to být členem klubu,
který přijímá lidi jako jsem já.
(Groucho Marx)

Tato část knihy popisuje, co se v systému děje poté, když se uživatel přihlásí nebo odhlásí. Dále budou detailněji popsány různé interakce některých procesů běžících na pozadí, při zahajování a ukončování sezení používané logovací soubory, konfigurační soubory a další.

Přihlašování přes terminály

Na obrázku 8.1 je graficky zobrazen algoritmus přihlášení uživatele do systému přes terminál. V prvním kroku si proces **init** ověří, zda běží program **getty** pro dané terminálové spojení (nebo konzolu). Program **getty** sleduje terminál a čeká na uživatele, jenž by mu sdělil, že se chce přihlásit do systému (obvykle tím, že stiskne některou klávesu na klávesnici terminálu). Když proces **getty** zjistí, že uživatel něco napsal na klávesnici, vypíše na obrazovku uvítací zprávu. Ta je uložená v souboru `/etc/issue`. Pak vyzve uživatele, aby zadal své uživatelské jméno a nakonec spustí program **login**. Program **login** dostane zadané uživatelské jméno jako parametr a následně vyzve uživatele, aby zadal přístupové heslo. Je-li heslo zadáno správně, program **login** spustí příkazový interpret vybraný podle nastavení konfigurace pro přihlášeného uživatele. V opačném případě se program **login** jednoduše ukončí, a tím se ukončí i celý proces přihlašování (většinou až poté, co uživatel dostane další možnost zadat správné uživatelské jméno a přístupové heslo). Proces **init** rozpozná, že byla procedura přihlašování ukončena a spustí pro daný terminál novou instanci programu **getty**.



Obrázek 8.1 – Přihlášení přes terminál. Interakce programů **init**, **getty**, **login** a shellu

Je důležité si uvědomit, že jediným novým procesem je ten, jenž vytvoří program **init** (použitím systémového volání `fork`). Procesy **getty** a **login** nahrazují právě tento nový proces (systémovým voláním `exec`).

V případě přihlašování po sériových linkách se pro sledování aktivity uživatelů používá zvláštní program proto, že někdy může být (a tradičně bývá) poměrně složité zjistit, kdy je terminál po ne-

činnosti opět aktivní. Program **getty** se rovněž přizpůsobuje přenosové rychlosti a dalším nastavením konkrétního spojení. Takovéto změny parametrů připojení jsou obzvlášť důležité v případě, že systém odpovídá na příchozí modemové žádosti o připojení, kdy se přenosové parametry běžně mění případ od případu.

V současnosti se používá několik různých verzí programů **getty** a **init**. Mají samozřejmě své výhody i nevýhody. Je dobré si přečíst dokumentaci k verzím, které jsou součástí vašeho systému, ale rozhodně neuškodí ani informace o jiných verzích. Další dostupné verze programu lze vyhledat pomocí „Mapy programového vybavení pro Linux“ (The Linux Software Map). V případě, že nemusíte obsluhovat příchozí volání se žádostmi o přihlášení, nebudete se pravděpodobně muset programem **getty** zabývat, avšak podrobnější informace o programu **init** pro vás budou i nadále důležité.

Přihlášení prostřednictvím sítě

Dva počítače, které jsou zapojeny v jedné síti, jsou obvykle propojeny jediným fyzickým kabelem. Když spolu stanice prostřednictvím sítě komunikují, programy, které běží na každé z nich a podílejí se na vzájemné komunikaci, jsou propojeny *virtuálními spojeními*, tedy jakousi sadou imaginárních kabelů. Když spolu aplikace na obou koncích virtuálního spojení komunikují, mají pro sebe vyhrazenou vlastní „linku“. Protože tato linka není skutečná, pouze imaginární, mohou operační systémy na obou počítačích vytvořit i několik virtuálních spojení sdílejících tutéž fyzickou linku. Takto spolu může s využitím jediného kabelu komunikovat několik programů bez toho, že by o ostatních spojeních věděly, nebo se o ně nějakým jiným způsobem staraly. Stejné fyzické médium může být sdíleno i několika počítači. Když pak existuje virtuální spojení mezi dvěma stanicemi, další počítače, které se komunikace neúčastní a sdílí tutéž fyzickou linku, toto spojení ignorují.

Toto byl komplikovaný a možná až příliš odtažitý popis reality. Měl by ale stačit k pochopení důležitého rozdílu mezi přihlášením prostřednictvím sítě a normálním přihlášením přes terminál. Virtuální spojení se vytvoří v případě, že existují dva programy na různých stanicích a přejí si spolu komunikovat. Vzhledem k tomu, že je principiálně možné připojit se z kteréhokoliv počítače v síti na kterýkoliv jiný, existuje velké množství potenciálních virtuálních spojení. Díky tomu není praktické spouštět proces **getty** pro každé potenciální síťové přihlášení do systému.

Proto také existuje jediný proces **inetd** (odpovídající procesu **getty**), který obsluhuje *všechna* síťová připojení. V případě, že proces **inetd** zaregistruje žádost o připojení ze sítě (tedy zaregistruje navázání nového virtuálního spojení s některým jiným počítačem zapojeným v síti), spustí nový proces obsluhující toto jediné přihlášení. Původní proces je nadále aktivní a dále čeká na nové požadavky o připojení.

Aby to nebylo až tak jednoduché, existuje pro připojení ze sítě víc komunikačních protokolů. Dva nejvýznamnější jsou **telnet** a **rlogin**. Kromě připojení do systému existuje i mnoho dalších druhů virtuálních spojení, která lze mezi počítači v síti navázat (síťové služby FTP, Gopher, HTTP a další). Bylo by neefektivní mít zvláštní proces, jenž by sledoval žádosti o navázání spojení pro každý typ připojení (službu). Místo toho existuje jediný proces, který umí rozeznat typ spojení a spustit správný program, jenž pak poskytuje odpovídající služby. Tímto procesem je právě proces **inetd**. Podrobnější informace uvádí *Příručka správce sítě*.

Co dělá program login

Program **login** se stará o autentizaci uživatele (kontroluje, zda bylo zadáno správné uživatelské jméno a přístupové heslo) a počáteční nastavení uživatelského prostředí nastavením oprávnění pro sériovou linku a spuštěním interpretu příkazů.

Částí procedury úvodního nastavení uživatelského prostředí je i vypsání obsahu souboru `/etc/motd` (zkratka pro „message of the day" – zprávu pro tento den) a kontrola nově příchozí elektronické pošty. Tyto kroky lze zakázat vytvořením souboru nazvaného `.hushlogin` v domovském adresáři uživatele.

Existuje-li soubor `/etc/nologin`, jsou přihlášení do systému zakázána. Tento soubor je typicky vytvářen příkazem **shutdown** nebo příbuznými programy. Program **login** kontroluje, jestli tento soubor existuje, a v případě, že je tomu tak, odmítne akceptovat přihlášení a předtím, než se definitivně ukončí, vypíše obsah tohoto souboru na terminál.

Program **login** rovněž zapisuje všechny neúspěšné pokusy o přihlášení do systémového logu (pomocí programu **syslog**). Rovněž zaznamenává úspěšné i neúspěšné pokusy o přihlášení superuživatele. Oba druhy záznamů jsou užitečné při pátrání po případných „vetřelcích".

Momentálně přihlášení uživatelé jsou zapsáni v seznamu `/var/run/utmp`. Tento soubor je platný jenom do dalšího znovuzavedení nebo zastavení systému, protože v průběhu zavádění systému se jeho obsah vymaže. Jinak jsou v souboru `/var/run/utmp` kromě seznamu všech přihlášených uživatelů a používaných terminálů (nebo síťových spojení) uvedeny i další užitečné informace. Příkazy **who**, **w** a další podobné se dívají právě do souboru `/var/run/utmp` a zjišťují, kdo je k systému připojený.

Všechna úspěšná přihlášení jsou zaznamenána do souboru `/var/log/wtmp`. Tento soubor se může bez omezení zvětšovat, proto je potřeba jej pravidelně mazat (například po týdnu) zadáním úkolu démonu **cron**[28]. Soubor wtmp lze procházet příkazem **last**.

Oba soubory `utmp` i `wtmp` mají binární formát (viz manuálová stránka **utmp**), takže je nelze prohlížet bez speciálních programů.

X a xdm

X implementují přihlášení prostřednictvím procesu **xdm**; viz též rovněž **xterm -ls**.

Řízení přístupu

Databáze uživatelů je tradičně uložena v souboru `/etc/passwd`. Některé systémy používají takzvaná *stínová hesla*. Přesouvají uživatelská přístupová hesla ze souboru `/etc/passwd` do souboru `/etc/shadow`. Sítě s velkým počtem počítačů, ve kterých se informace o uživatelských účtech sdílí pomocí systému NIS nebo nějakou jinou metodou, mohou databázi uživatelů automaticky kopírovat z jediného centrálního počítače na všechny ostatní stanice.

Databáze uživatelů obsahuje nejenom hesla, ale i některé další informace o uživatelích, například jejich skutečná jména, domovské adresáře a interprety příkazů, jež se implicitně spouští po přihlášení. Je potřeba, aby byly tyto informace o uživatelích v systému obecně dostupné a aby si je mohl každý přečíst. Kvůli tomu se heslo ukládá v zakódovaném tvaru. Má to ale jeden háček. Kdokoliv, kdo má přístup k databázi uživatelů, může s pomocí různých kryptografických metod zku-

28 Slušné distribuce Linuxu to dělají samy.

sit hesla uhodnout i bez toho, že by se musel přihlásit k hostitelskému počítači. Systém stínových hesel se snaží zamezit možnosti prolomení přístupových hesel tím, že se přesouvají do jiného souboru, jenž je přístupný pouze superuživateli (hesla se i tak ukládají v zakódovaném tvaru). Avšak s pozdější instalací systému stínových hesel na systému, který jej nepodporuje, mohou vznikat různé potíže.

Ať tak nebo onak, z bezpečnostních důvodů je důležité pravidelně ověřovat, jestli jsou všechna v systému používaná přístupová hesla netriviální, tedy taková, aby nebylo lehké je uhodnout. Lze použít například program **crack**, jenž zkouší hesla v `/etc/passwd` dekódovat. Heslo, které se mu podaří uhodnout, nelze podle výše uvedeného považovat za spolehlivé. Program **crack** mohou samozřejmě zneužít i případní vetřelci, ale správci systému může jeho pravidelné používání pomoci preventivně omezit výběr nevhodných přístupových hesel. K volbě netriviálního přístupového hesla lze uživatele donutit i programem **passwd**. To je metoda, která je efektivnější především z hlediska zatížení procesoru, protože zpětná analýza zašifrovaných hesel programem **crack** je výpočetně o hodně náročnější.

Databáze skupin uživatelů je uložena v souboru `/etc/group`, u systémů se stínovými hesly případně v souboru `/etc/shadow.group`.

Uživatel `root` se obvykle nemůže přihlásit z kteréhokoliv terminálu nebo počítače v síti, pouze z terminálu uvedeného v seznamu `/etc/securetty`. Pak je nutné mít k některému z těchto terminálů fyzický přístup. Avšak takovéto bezpečnostní opatření nelze považovat za dostatečné, protože je možné přihlásit se z kteréhokoliv jiného terminálu jako běžný uživatel a pro změnu uživatelských oprávnění použít příkaz **su**.

Spouštění interpretu příkazů

Při startu každý příkazový interpret automaticky spouští jeden či více předem určených souborů. Různé interprety spouští různé konfigurační soubory. Podrobnější informace najdete v dokumentaci k jednotlivým typům interpretů.

Většina příkazových interpretů nejdříve spustí některý globální soubor, například interpret Bourne shell (`/bin/sh`) a jeho klony spouští soubor `/etc/profile`, až poté spustí soubor `.profile`, jenž je uložen v uživatelově domovském adresáři. Soubor `/etc/profile` umožňuje správci systému nastavit běžné, implicitní uživatelské prostředí, například nastavením proměnné PATH, tak, aby zahrnovalo kromě obvyklých i lokální adresáře s příkazy. Soubor `.profile` zase umožňuje každému z uživatelů upravit si předem nastavené běžné prostředí podle svého vlastního vkusu.

Správa uživatelských účtů

Jaký je rozdíl mezi správcem systému a překupníkem drog?
Žádný, oba měří své prostředky v kilech a oba mají své klienty.
(Starý a otřelý počítačový vtip.)

Tato kapitola popisuje způsob vytváření nových uživatelských účtů, změny vlastností těchto účtů a způsoby jejich odstraňování. Různé distribuce systému Linux používají pro tyto úkoly různé nástroje.

Co je to účet?

Používá-li počítač více lidí, je obvykle nutné mezi jednotlivými uživateli rozlišovat. Například proto, aby jejich osobní soubory a data byly osobními v pravém smyslu slova. Identifikace uživatelů je ale důležitá i v případě, že systém využívá pouze jedna osoba, což se týká převážné většiny mikropočítačů[29]. Proto má každý uživatel systému přiděleno jednoznačné uživatelské jméno, které zadává při každém přihlášení.

Avšak pojem „účet" je poněkud širší a zahrnuje – kromě uživatelského jména – i některé další informace o uživateli. Pojmem *uživatelský účet* se označují všechny soubory, zdroje a informace, jež se vztahují k danému uživateli. Běžně se termín „účet" spojuje s bankovním sektorem. V komerčním systému se kolem každého účtu skutečně točí nějaké peníze. Ty mohou z účtu mizet různou rychlostí, podle toho, jak moc uživatel systém zatěžuje. Tak například diskový prostor lze ohodnotit cenou za megabajt uložených dat a den, čas procesoru může mít určitou cenu za sekundu využití a podobně.

Vytváření uživatelských účtů

Samotné jádro systému Linux považuje uživatele systému za pouhé číslo. Každého uživatele lze totiž identifikovat podle jednoznačného celého čísla, takzvaného *identifikačního čísla uživatele* (anglicky *user ID*, zkráceně *UID*). Je tomu tak proto, že počítač umí zpracovat čísla rychleji a jednodušeji než jména v textové formě. Jména v textové podobě (*uživatelská jména*) se udržují ve

29 Docela by mi vadilo, kdyby si má sestra četla mé milostné dopisy.

zvláštní databázi mimo vlastní jádro. Tato databáze obsahuje též další informace o uživatelích systému.

Když potřebujete vytvořit nový uživatelský účet, musíte přidat informace o novém uživateli do této uživatelské databáze a vytvořit pro něj vlastní domovský adresář. Kromě toho by měl každý nový uživatel absolvovat školení. Je též vhodné nastavit pro nové uživatele přiměřené počáteční nastavení prostředí.

Většina distribucí systému Linux se dodává s programy pro vytváření uživatelských účtů. Správce má dokonce k dispozici hned několik takovýchto programů. Dvě varianty, jejichž uživatelským rozhraním je příkazová řádka, jsou programy **adduser** a **useradd**. Existují i nástroje s grafickým uživatelským rozhraním. Ať už se jako správce systému rozhodnete pro kterýkoliv z programů, oceníte jejich hlavní přínos, tedy to, že omezují manuální práci s nastavováním na minimum, či dokonce úplně. I když na vás při administraci uživatelských účtů čeká mnoho dost spletitých detailů, díky těmto nástrojům vypadá jednoduše. Postup při „ručním" zakládání nových uživatelských účtů uvádí odstavec *Manuální vytváření účtů*.

Soubor `/etc/passwd` a další informační soubory

Základní databází uživatelů v systému Unix je textový soubor `/etc/passwd` (označovaný jako *password file*), v němž jsou uvedena platná uživatelská jména a další k nim přidružené informace. Každému uživateli odpovídá v souboru jeden záznam – řádek, který je rozdělen na sedm polí, jejichž oddělovačem je dvojtečka. Význam jednotlivých položek je následující:

- Uživatelské jméno.
- Heslo v zakódované podobě.
- Identifikační číslo uživatele.
- Identifikační číslo skupiny.
- Skutečné jméno uživatele, případně popis účtu.
- Domovský adresář.
- Příkazový interpret (nebo program), který se spustí po přihlášení.

Formát jednotlivých políček je podrobněji popsán v manuálové stránce souboru `passwd`.

Každý uživatel systému má k souboru `/etc/passwd` přístup (může jej číst). Může tedy například zjistit přihlašovací jména ostatních uživatelů. To ale znamená, že jsou všem přístupná i hesla ostatních uživatelů (druhé pole každého záznamu). Hesla uložená v souboru `/etc/passwd` jsou zakódovaná, takže teoreticky nevzniká žádný problém. Avšak použitý kódovací algoritmus lze prolomit[30], zvlášť je-li zvolené heslo jednoduché (například krátké slovo, jež lze najít v nějakém slovníku, jméno nebo příjmení uživatele a podobně). Proto z hlediska bezpečnosti není dobré mít hesla uložená přímo v souboru `/etc/passwd`.

Řada systémů Linux používá systém takzvaných *stínových hesel*. Jde o alternativní způsob uložení uživatelských přístupových hesel, jež se zašifrovaná ukládají do jiného souboru (`/etc/shadow`), který může číst pouze superuživatel. Soubor `/etc/passwd` pak obsahuje v druhém poli pouze speciální znak. Programy, které si potřebují ověřit totožnost uživatele a mají propůjčená přístupová práva vlastníka souboru (pomocí volání jádra setuid), mohou soubor `/etc/shadow` číst. Běžné programy, které používají pouze některé z dalších položek souboru `/etc/passwd`, přístup k heslům uživatelů systému nemají[31].

30 Pozn. překladatele: Pokud je mi známo, ještě pořád nikdo nepřišel na to, jak zmíněný algoritmus „prolomit" (ve smyslu, že ze zašifrované podoby hesla zjistím původní heslo). Ale existuje řada programů, které prostě šifrují spousty běžných slov a testují, jestli se náhodou výsledek neshoduje s tím, co najdou v souboru.

31 Ano, znamená to, že soubor hesel obsahuje o uživateli všechny informace kromě hesla. Není ten pokrok nádherný?

Výběr čísel uživatelského ID a ID skupiny

Ve většině systémů nezáleží na tom, jaké jsou hodnoty UID a GID. Když ale používáte síťový souborový systém NFS, musíte mít stejná UID a GID na všech systémech v síti. To proto, že i systém NFS identifikuje uživatele podle hodnoty UID. Jestliže systém NFS nepoužíváte, můžete vybírat identifikační čísla uživatele a skupiny podle automatického návrhu některého z nástrojů pro správu uživatelských účtů.

Když v síti využíváte NFS, budete si muset zvolit některý z mechanismů synchronizace informací o uživatelských účtech. Jednou z možností je systém NIS – Network Information Service.

Měli byste se také vyvarovat opakovanému přidělování stejných uživatelských čísel (i textových uživatelských jmen), protože nový vlastník UID (případně uživatelského jména) by tak mohl mít přístup k souborům bývalého vlastníka, k jeho elektronické poště a dalším informacím.

Nastavení uživatelského prostředí: adresář `/etc/skel`

Když jste již pro nového uživatele vytvořili vlastní domovský adresář, můžete nastavit vlastnosti uživatelského prostředí tak, že do domovského adresáře nového uživatele nakopírujete některé soubory z adresáře `/etc/skel`. Správce systému totiž může v adresáři `/etc/skel` vytvořit konfigurační soubory, jež novým uživatelům vytvoří příjemné základní uživatelské prostředí. Administrátor může například vytvořit soubor `/etc/skel/.profile`, v němž lze nastavením proměnné prostředí EDITOR vybrat některý z textových editorů, jenž by měl pro méně zkušené uživatele přátelské ovládání.

Avšak obvykle se doporučuje mít v adresáři `/etc/skel` co nejméně souborů, protože by jinak bylo téměř nemožné upravit na větších víceuživatelských systémech při každé změně konfigurační soubory v již existujících uživatelských adresářích. Když se například změní název onoho uživatelsky příjemného editoru, všichni současní uživatelé si musí upravit svůj vlastní soubor `.profile`. Správce systému by se to mohl pokusit udělat automaticky, například pomocí skriptu, ale takovéto akce téměř pravidelně končí tak, že se nechtěně přepíše či poškodí nějaký jiný soubor.

Vždy když to situace dovolí, je lepší nastavovat globální konfigurace v globálních souborech, jako je `/etc/profile`. Pak lze nastavení pohodlně upravovat bez toho, že by se muselo měnit vlastní nastavení jednotlivých uživatelů systému.

Manuální vytváření účtů

Nový uživatelský účet lze vytvořit ručně tímto postupem:

- Upravte soubor `/etc/passwd` příkazem **vipw**, tak, že do souboru hesel přidáte nový řádek pro nový uživatelský účet. Je nutné dodržovat správnou syntaxi. *Není vhodné upravovat tento soubor přímo běžným editorem!* Program **vipw** soubor `/etc/passwd` uzamkne, takže jej ostatní programy nemohou změnit. Do pole pro heslo vložte znak *, takže zatím nebude možné se prostřednictvím tohoto účtu do systému přihlásit.
- Je-li třeba vytvořit i novou pracovní skupinu, upravte soubor `/etc/group` rovněž programem **vigr**.
- Příkazem **mkdir** pro nového uživatele vytvoříte domovský adresář.
- Do nově vytvořeného domovského adresáře nakopírujte konfigurační soubory z adresáře `/etc/skel`.
- Příkazy **chown** a **chmod** upravte jejich vlastnická a přístupová práva. Užitečný je v tomto případě jejich parametr `-R`. Správná přístupová práva se mohou trochu lišit, a to podle typického využití toho kterého systému, nicméně příkazy v níže uvedeném příkladu obvykle vyhovují většině případů:

```
cd /home/newusername
chown -R username.group .
chmod -R go=u,go-w .
chmod go= .
```

- Zadejte heslo programem **passwd**.

Poté, co bylo v posledním kroku nastaveno heslo, bude nový účet přístupný. Heslo by se skutečně mělo nastavovat až v posledním kroku, jinak by se mohl uživatel nepozorovaně přihlásit do systému například v době, kdy kopírujete konfigurační soubory.

Někdy je potřeba vytvořit „falešný" účet, který se nebude používat pro přihlašování běžných uživatelů. Například při konfigurování anonymního serveru FTP je výhodné vytvořit účet s uživatelským jménem ftp. Pak si ze serveru může stahovat soubory kdokoliv a pro budoucí (anonymní) klienty se nemusí zřizovat vlastní uživatelské účty. V takovýchto případech obvykle není třeba nastavovat heslo (vynechá se poslední krok výše uvedeného postupu). Je lepší zřizovat takovéto anonymní účty bez nastaveného hesla. Jinak by k nim měl přístup pouze uživatel, který by předtím musel získal oprávnění superuživatele, protože pouze uživatel root má přístup k ostatním uživatelským účtům.

Změny vlastností uživatelských účtů

Je několik příkazů, kterými lze měnit různé vlastnosti uživatelských účtů (tedy příslušných položek v souboru `/etc/passwd`):

chfn Mění pole, ve kterém je uloženo skutečné jméno uživatele.

chsh Mění nastavený příkazový interpret.

passwd Mění přístupové heslo.

Superuživatel může pomocí těchto programů změnit vlastnosti kteréhokoliv účtu. Ostatní neprivilegovaní uživatelé mohou měnit pouze vlastnosti svého vlastního účtu. V některých případech je vhodnější běžným uživatelům zakázat možnost používat uvedené příkazy (programem **chmod**), například v případě, že systém používá větší počet méně zkušených začátečníků.

Ostatní změny položek souboru `/etc/passwd` se musí dělat ručně. Když například potřebujete změnit uživatelské jméno, musíte přímo upravit databázi uživatelů `/etc/passwd` (připomínáme, že pouze příkazem **vipw**). Analogicky, když potřebujete přidat či odebrat uživatele z nebo do některé z pracovních skupin, musíte upravit soubor `/etc/group` (příkazem **vigr**). Avšak takovéto úkoly se dělají zřídka a musí se dělat opatrně, protože když například změníte některému z uživatelů jeho uživatelské jméno, nebude mu docházet elektronická pošta a musíte pro něj vytvořit poštovní alias[32].

Zrušení uživatelského účtu

Potřebujete-li zrušit uživatelský účet, smažte nejdříve všechny soubory, které patří uživateli rušeného účtu, včetně poštovní schránky, aliasů pro elektronickou poštu, tiskových úloh, úkolů spouštěných démony **cron** a **at** a všechny další odkazy na tohoto uživatele. Pak odstraňte odpovídající řádek v souborech `/etc/passwd` a `/etc/group` (nezapomeňte odstranit uživatelské jméno i ze všech skupin, do nichž byl uživatel zařazen). Předtím, než začnete mazat vše ostatní, je lepší zakázat přístup

32 Jméno uživatele (respektive uživatelky) se může změnit například po svatbě.

k rušenému účtu (viz níže). Uživatel tak nebude mít možnost se připojit do systému v době, kdy je jeho účet odstraňován.

Nezapomeňte, že uživatelé systému mohou mít některé soubory uloženy i mimo svůj domovský adresář. Najdete je pomocí příkazu **find**:

```
find / -user username
```

Pamatujte na to, že když má váš systém velké disky, poběží výše uvedený příkaz dost dlouho. V případě, že máte v souborovém systému připojeny síťové disky, musíte dávat pozor, abyste tím nezpůsobili problémy s odezvou v síti nebo na serveru.

Součástí některých distribucí systému Linux jsou i zvláštní příkazy, které lze použít při rušení uživatelských účtů. Zkuste na vašem systému vyhledat programy **deluser** či **userdel**. Každopádně není zase tak složité to udělat ručně, navíc tyto programy nemusí najít a odstranit všechny souvislosti.

Dočasné zablokování uživatelského účtu

Občas je potřeba dočasně zablokovat přístup k některému z účtů bez toho, že by bylo nutné jej smazat. Například když uživatel nezaplatil poplatky za využívání systému, nebo když má správce systému podezření, že neznámý hacker prolomil přístupové heslo uživatele některého účtu.

Nejvhodnějším způsobem jak zamezit přístup k podezřelému účtu, je zaměnit nastavený příkazový interpret na zvláštní program, který na terminál pouze vypíše nějaké hlášení. Když se pak kdokoliv pokusí přihlásit do systému přes zablokovaný účet, neuspěje a dozví se proč. Zprávou lze sdělit uživateli, že se má spojit se správcem systému a domluvit se s ním na řešení vzniklého problému.

Alternativou je změna uživatelského jména, případně hesla. Uživatel se tak ale nedozví, co se vlastně děje. A zmatení uživatelé znamenají i více práce pro správce systému.

Nejjednodušší způsob jak vytvořit program, který by blokoval přístup k účtu, je napsat skript pro program **tail**:

```
#!/usr/bin/tail +2
Tento účet byl z důvodů porušení bezpečnostních opatření zablokován.
Volejte prosím číslo 555-1234 a vyčkejte příjezdu mužů v černém.
```

Podle prvních dvou znaků (#!) pozná jádro systému, že zbytek řádku je příkaz, který je třeba spustit, aby se skript provedl. V tomto případě je to příkaz **tail**, jenž vypíše vše kromě prvního řádku na standardní výstup.

Je-li uživatel billg podezřelý, že porušuje bezpečnostní opatření, může správce systému udělat něco jako:

```
# chsh -s /usr/local/lib/no-login/security billg
# su - billg
Tento účet byl z důvodů porušení bezpečnostních opatření zablokován.
Volejte prosím číslo 555-1234 a vyčkejte příjezdu mužů v černém.
#
```

Pomocí příkazu **su** ve výše uvedeném příkladu se pochopitelně pouze testuje, zda je změna původně nastaveného interpretu funkční.

Takovéto skripty by měly být uloženy ve zvláštním adresáři, aby jejich jména nekolidovala s příkazy jednotlivých uživatelů.

Zálohování

Hardware je nedeterministicky spolehlivý.
Software je deterministicky nespolehlivý.
Lidé jsou nedeterministicky nespolehliví.
Příroda je deterministicky spolehlivá.

Tato kapitola vysvětluje proč, jak a kdy zálohovat a jak ze záloh obnovit data.

Jak je důležité míti zálohy

Vaše data mají určitou cenu. Jejich cena je daná hodnotou vašeho času a cenou úsilí potřebného pro nové vytvoření dat v případě jejich ztráty. To vše lze vyjádřit v penězích, nebo přinejmenším vyvážit zármutkem a slzami. Někdy již totiž ztracené informace nelze obnovit, například když data pocházejí z nějakých neopakovatelných experimentů. Vzhledem k tomu, že informace a data jsou v jistém slova smyslu investicí, měli byste tuto investici chránit a učinit kroky, jež by zabránily jejímu znehodnocení.

V zásadě existují čtyři důvody, jež by mohly vést ke ztrátě dat: závady hardwaru, chyby v programech, jednání lidí nebo přírodní katastrofy[33]. I když je v současnosti hardware relativně spolehlivý, ještě stále se může v podstatě bez příčiny porouchat. Pokud jde o ukládání dat, je nejkritičtější součástí výpočetního systému pevný disk. Ve světě plném elektromagnetického šumu se bláhově spoléhá na to, že miniaturní magnetická políčka na povrchu disku, ve kterých jsou cenné informace uloženy, zůstanou v neporušeném stavu. U programů o spolehlivosti prakticky nelze mluvit, robustní a spolehlivý software je spíše výjimkou, než pravidlem. I lidé jsou dost nespolehliví. Buďto udělají nějakou chybu, nebo jsou zlomyslní a zničí data úmyslně. Přírodu obecně bychom neměli považovat za zlo, ale i když je na nás hodná, může způsobit zkázu. Takže je malým zázrakem, když všechno funguje, jak má.

Řekli jsme, že zálohování je způsobem ochrany investic do dat a informací. Máte-li několik kopií dat, nestane se zase až tak moc, když se některá z verzí zničí (cenou za takovou ztrátu je pouze to, že data musíte obnovit ze zálohy).

Je důležité zálohovat správně. Jako všechno ostatní v reálném světě, dříve nebo později selžou i zálohy. Součástí správného zálohování je i to, že se kontroluje, jestli vůbec funguje. Jistě byste si nechtěli jednoho dne „všimnout“, že se zálohy nedělaly správně[34]. Neštěstí v neštěstí – může se stát, že dojde k nějaké vážné havárii právě v okamžiku, kdy zálohujete a máte pouze jediné zálohovací médium. To se může rovněž poškodit a z úmorné práce zbude (někdy doslova) hromád-

33 A pátá příčina je „něco jiného“.

34 Nesmějte se, to už se stalo spoustě lidí.

ka popela[35]. Může se také stát, že si v momentě, kdy se pokoušíte obnovit data ze záloh, uvědomíte, že jste zapomněli zálohovat něco důležitého, například databázi uživatelů systému, která může mít třeba 15 000 položek. To „nejlepší" nakonec – všechny zálohy mohou fungovat bezchybně, ale v poslední známé páskové jednotce, jež ještě umí přečíst typ pásky, který používáte, je vědro vody.

Když totiž dojde na zálohy, je paranoia v popisu práce.

Výběr média pro zálohování

Co se týče zálohování, je nejdůležitějším rozhodnutím výběr médií. Musíte zvážit náklady, spolehlivost, rychlost, dostupnost a použitelnost.

Cena je poměrně důležitá, protože byste měli mít několikrát větší kapacitu zálohovacích médií, než je velikost zálohovaných dat. Levné médium je obvykle nezbytnost.

Extrémně důležitá je spolehlivost, protože poškozená záloha by rozbrečela i dospělého. Data uložená na zálohovacích médiích musí vydržet bez poškození i několik let. I způsob, kterým médium používáte, má vliv na jeho spolehlivost. Například pevný disk je typicky velmi spolehlivé médium, ovšem z hlediska zálohování je to k ničemu, pokud jej máte ve stejném počítači, ze kterého zálohujete (jistě – jeden blesk přece nezničí *oba* disky v tomtéž počítači).

Rychlost většinou není příliš důležitá, pokud zálohování může probíhat automaticky. Když nemusíte na zálohování dohlížet, pak nevadí, že trvá dejme tomu dvě hodiny. Ale naopak, nelze-li provést zálohování v době, ve které je počítač jinak nevyužitý (například v průběhu noci), pak může být i rychlost problémem.

Dostupnost je zjevně důležitá, protože nemůžete používat zálohovací médium, které není k dostání. Méně zjevný je důležitý požadavek, aby bylo zálohovací médium dostupné i v budoucnu a případně i pro jiné počítače než ty, které používáte dnes. Jinak se může stát, že nebudete schopni zálohy po nečekané události obnovit.

Použitelnost je nejvíce závislá na tom, jak často se zálohování provádí. Čím jednodušší je zálohování, tím lépe. Zálohování na zvolené médium nesmí být složité, pracné nebo otravné.

Typickými alternativami jsou diskety a pásky. Diskety jsou velmi levné, celkem spolehlivé, ne příliš rychlé, velmi dostupné, ale ne příliš použitelné v případě velkého množství dat. Magnetické pásky jsou levné i drahé, celkem spolehlivé, celkem rychlé, dost dostupné a – podle toho, jaká je jejich kapacita – z hlediska použitelnosti i docela pohodlné.

Existují i jiné možnosti. Obvykle nejsou nejlepší, pokud jde o jejich dostupnost, ale vznikne-li problém, oceníte je více než ostatní možnosti. Řeč je o magneto-optických discích, jež kombinují dobré vlastnosti disket (jejich náhodný, nesekvenční přístup k souborům, možnost rychlého obnovování jednotlivých souborů) a magnetických páskových médií (zálohování velkého objemu dat)[36].

Výběr nástroje pro zálohování

Existuje bezpočet nástrojů, jichž lze při zálohování využít. Mezi ty tradiční, používané při zálohování v systémech Unix, patří programy **tar**, **cpio** a **dump**. Kromě toho lze sáhnout i po velkém

35 Já tam byl, já to viděl...

36 Pozn. překladatele: Pro malé objemy dat je dnes optimální vypalování na CD disk, pro větší objemy dat stojí za to vypalovat na DVD disk.

množství balíků programů třetích výrobců (šířených zdarma jako freeware i komerčně). Výběr médií pro zálohování má často vliv i na výběr nástroje.

Programy **tar** a **cpio** jsou podobné a z hlediska zálohování prakticky stejné. Oba programy umí ukládat soubory na pásky a obnovovat je, oba jsou schopné využívat téměř všechna média, protože díky ovladačům jádra systému, které mají na starost obsluhu nízkoúrovňových zařízení, se takováto zařízení chovají vůči programům na uživatelské úrovni stejně. Některé unixové verze programů **tar** a **cpio** mohou mít problémy s neobvyklými soubory (symbolickými odkazy, speciálními soubory, soubory s velmi dlouhou cestou a podobně), avšak verze pro operační systém Linux by měly pracovat se všemi soubory korektně.

Program **dump** se liší v tom, že přistupuje k souborovému systému přímo a ne prostřednictvím jeho služeb. Navíc byl napsán speciálně pro zálohování, kdežto programy **tar** a **cpio** jsou primárně určeny pro archivaci souborů, i když je lze s úspěchem použít i jako zálohovací nástroje.

Přímý přístup k souborovému systému má několik výhod. Umožňuje zálohovat soubory bez toho, že by to mělo vliv na jejich časové značky. U programů **tar** a **cpio** byste museli nejdřív připojit souborový systém pouze pro čtení. Přímé čtení dat ze souborového systému je také efektivnější v případech, kdy je potřeba zálohovat všechna data na disku, protože v tomto případě proběhne zálohování s výrazně menším pohybem čtecích hlav. Největší nevýhodou přímého přístupu je, že zálohovací program, který jej používá, je specifický pro každý typ souborového systému. Program **dump** pro Linux rozumí pouze souborovému systému ext2.

Program **dump** má navíc zabudovanou podporu úrovní zálohování, o kterých bude řeč později. U programů **tar** a **cpio** musí být implementovány pomocí jiných nástrojů.

Srovnávání zálohovacích nástrojů třetích výrobců jde nad rámec této knihy. „Mapa programů pro Linux“ (The Linux Software Map) uvádí výčet těch, které jsou jako freeware šířeny zdarma.

Jednoduché zálohování

Při proceduře jednoduchého zálohování se v prvním kroku vytvoří zálohy všech dat a v dalších krocích se pak zálohuje jenom to, co se od posledního zálohování změnilo. Prvnímu kroku se říká *úplné zálohování*, v dalších krocích se tvoří *inkrementální zálohy*. Úplné zálohování je obvykle pracnější než inkrementální, protože se na médium zapisuje víc dat, a proto se úplná záloha nemusí vždy vejít na jednu pásku (nebo disketu). Naopak, obnovování dat z inkrementálních záloh bývá pochopitelně mnohokrát pracnější, než obnovování z úplných záloh. Nicméně inkrementální zálohování lze optimalizovat tak, že se vždy zálohují všechny změny od poslední úplné zálohy. Takto je sice o něco pracnější proces zálohování, ale pak by nemělo být nikdy potřeba obnovovat víc než jednu úplnou a jednu inkrementální zálohu.

Uveďme příklad, kdy chcete data zálohovat každý den a máte k dispozici šest páskových kazet, můžete první z nich použít (řekněme v pátek) pro uložení první úplné zálohy a pásky 2 až 5 pro inkrementální zálohování (od pondělí do čtvrtka). Pak vytvoříte novou úplnou zálohu na pásku číslo 6 (druhý pátek) a začnete znovu s inkrementálním zálohováním na pásky 2 až 5. Nepřepisujte pásku číslo jedna do doby, než se ukončí nové úplné zálohování – v jeho průběhu by se totiž mohlo stát něco neočekávaného. Poté, co vytvoříte novou úplnou zálohu na pásce číslo 6, měli byste uschovat pásku 1 někam jinam pro případ, že by se ostatní zálohovací pásky zničily – například při požáru. Takto vám v případě neočekávané události zůstane alespoň něco. Když pak potřebujete vytvořit další úplnou zálohu, dojdete pro pásku číslo 1 a pásku číslo 6 necháte na jejím místě.

Máte-li víc než šest kazet, můžete ukládat na ty, jež jsou navíc, další úplné zálohy. Pokaždé, když budete dělat úplnou zálohu, použijete tu nejstarší pásku. Takto budete mít úplné zálohy z něko-

lika předchozích týdnů, což je dobré, když potřebujete obnovit například některý starší, omylem smazaný soubor, případně starší verzi nějakého souboru.

Zálohování programem tar

Pomocí programu **tar** lze jednoduše vytvořit úplnou zálohu tímto způsobem:

```
# tar -create -file /dev/ftape /usr/src
tar: Removing leading / from absolute path names in the archive
#
```

Ve výše uvedeném příkladu byla použita syntaxe programu **tar** verze GNU s dlouhými tvary přepínačů. Tradiční verze programu **tar** rozumí pouze jednopísmenným volbám. Verze GNU ale umí pracovat i se zálohami, které se nevejdou na jednu pásku nebo disketu, a s velmi dlouhými cestami k zálohovaným souborům. Ne všechny klasické verze programu **tar** tyto věci zvládají. (Operační systém Linux používá výhradně program **tar** ve verzi GNU.)

Jestli se záloha nevejde na jednu pásku, je potřeba zadat přepínač `-multi-volume` (`-M`):

```
# tar -cMf /dev/fd0H1440 /usr/src
tar: Removing leading / from absolute path names in the archive
Prepare volume #2 for /dev/fd0H1440 and hit return:
#
```

Nezapomeňte, že před zálohováním je potřeba diskety zformátovat. Stačí, když to uděláte v jiném okně, případně na jiném virtuálním terminálu ve chvíli, kdy program **tar** čeká na vložení další diskety.

Pokaždé, když ukončíte zálohování, měli byste zkontrolovat, zda proběhlo správně. Zadejte příkaz **tar** s parametrem `-compare` (`-d`):

```
# tar -compare -verbose -f /dev/ftape
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
....
#
```

Chyba v záloze znamená, že nic nepoznáte až do okamžiku, než budete potřebovat data ze zálohy obnovit.

Je-li kontrola záloh neúspěšná, nebudete moci v případě potřeby původní data z takovýchto záloh obnovit.

Inkrementální zálohování dělá program **tar** s parametrem `-newer` (`-N`):

```
# tar -create -newer '8 Sep 1995' -file /dev/ftape /usr/src -verbose
tar: Removing leading / from absolute path names in the archive
usr/src/
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/modules/
usr/src/linux-1.2.10-includes/include/asm-generic/
usr/src/linux-1.2.10-includes/include/asm-i386/
usr/src/linux-1.2.10-includes/include/asm-mips/
```

```
usr/src/linux-1.2.10-includes/include/asm-alpha/
usr/src/linux-1.2.10-includes/include/asm-m68k/
usr/src/linux-1.2.10-includes/include/asm-sparc/
usr/src/patch-1.2.11.gz
#
```

Bohužel program **tar** neumí zjistit změny informací v inode souboru, například změny bitu vyhrazeného pro přístupová práva nebo změny jména souboru. Lze to obejít pomocí programu **find** a srovnávat stav dat uložených v souborovém systému se seznamem souborů, které byly posledně zálohovány. Skripty a programy, které implementují tento způsob zálohování, najdete na různých linuxových FTP serverech.

Obnovování souborů programem tar

Zálohované soubory lze obnovit příkazem **tar** s parametrem `-extract` (`-x`):

```
# tar -extract -same-permissions -verbose -file /dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```

Můžete rovněž obnovovat jenom vybrané soubory či adresáře (a všechny soubory a podadresáře, které jsou v nich uloženy) tak, že je uvedete v příkazové řádce:

```
# tar xpvf /dev/fd0H1440 usr/src/linux-1.2.10-includes/include/li-
nux/hdreg.h usr/src/linux-1.2.10-includes/include/linux/hdreg.h
#
```

Když vás zajímá jenom to, které soubory záloha obsahuje, zadejte příkaz **tar** s parametrem `-list` (`-t`):

```
# tar -list -file /dev/fd0H1440
usr/src/
usr/src/linux
usr/src/linux-1.2.10-includes/
usr/src/linux-1.2.10-includes/include/
usr/src/linux-1.2.10-includes/include/linux/
usr/src/linux-1.2.10-includes/include/linux/hdreg.h
usr/src/linux-1.2.10-includes/include/linux/kernel.h
...
#
```

Uvědomte si, že program **tar** čte zálohu vždy sekvenčně, takže je při práci s většími objemy dat dost pomalý. Ale jestli používáte páskové jednotky nebo jiná média se sekvenčním přístupem, stejně nemůžete využít různé techniky, jež využívají náhodný přístup.

Program **tar** neumí správně zacházet se smazanými soubory. Potřebujete-li obnovit souborový systém z úplné a inkrementální zálohy a mezi těmito zálohami jste některý ze souborů smazali, po

obnovení dat ze záloh bude smazaný soubor znovu existovat. To může být dost závažný problém například v případě, že soubor obsahuje citlivá data, která by již neměla být k dispozici.

Víceúrovňové zálohování

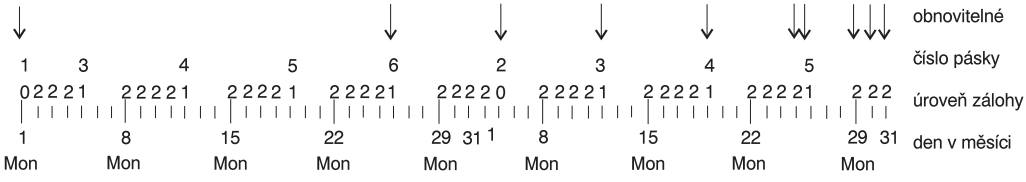
Metoda jednoduchého zálohování, jež byla načrtnuta v předchozím odstavci, je vhodná pro osobní potřebu nebo systémy s malým počtem uživatelů. Pro náročnější podmínky je vhodnější víceúrovňové zálohování.

Metoda jednoduchého zálohování má dvě úrovně: úplné a inkrementální zálohování. Ty lze zobecnit do libovolného počtu dalších úrovní. Úplná záloha by mohla být úrovní 0 a další následné inkrementální zálohy úrovněmi 1, 2, 3 a tak dále. Na každé úrovni inkrementálního zálohování se zálohuje vše, co se změnilo od poslední zálohy stejné nebo nižší úrovně.

Účelem víceúrovňového zálohování je levněji prodloužit historii zálohování dat. V příkladu v předchozím odstavci šla historie zálohování jenom k poslední úplné záloze. Kdybyste měli víc médií, mohli byste ji prodloužit, ale s každou další novou páskou jenom o týden, a to by bylo příliš nákladné. Delší historie vytváření záloh je užitečná, protože omylem smazaných nebo poškozených souborů si často všimnete až po delší době. Navíc jakákoliv verze souboru, i když není zrovna aktuální, je obvykle lepší než žádná.

Víceúrovňovým zálohováním lze historii vytváření archivů prodloužit levněji. Když si například koupíme deset pásek, můžeme používat pásky 1 a 2 na měsíční zálohy (první pátek každého měsíce), pásky 3 až 6 na týdenní zálohy (všechny ostatní pátky – uvědomte si, že v některém měsíci může být pět pátků, takže potřebujete o čtyři pásky víc) a pásky 7 až 10 na denní zálohy (od pondělí do čtvrtka). Takto jsme schopni – pouhým zakoupením čtyř dalších pásek navíc – prodloužit historii zálohování ze dvou týdnů (s využitím všech denních záloh) na dva měsíce. Pravda, během těchto dvou měsíců nemůžeme obnovit všechny verze zálohovaných souborů, avšak to, co obnovit lze, obvykle postačí.

Z obrázku 10.1 je patrné, která úroveň zálohování se používá v ten který den a z kterých záloh je možno na konci měsíce data obnovit.



Obrázek 10.1 – Příklad časového rozvrhu víceúrovňového zálohování

Víceúrovňové zálohování navíc snižuje i čas potřebný k obnovení celého souborového systému. Když potřebujete obnovit celý systém souborů a máte mnoho inkrementálních záloh s monotónně rostoucí řadou úrovní, musíte data pracně obnovovat postupně ze všech médií. Když ale budete používat méně záloh a větší počet úrovní, snížíte počet úrovní potřebných k obnovení celého svazku.

Chcete-li snížit počet médií potřebných k obnovení dat, používejte pro inkrementální zálohy menší rozsah úrovní. Tak se ovšem prodlouží čas zálohování, protože při každém zálohování se ukládá vše, co se změnilo od předchozí úplné zálohy. O něco lepší plán zálohování se uvádí v manuálové stránce programu **dump**. Jeho schéma je patrné z tabulky 10.1. Zkuste použít posloupnost

úrovní zálohování: 3, 2, 5, 4, 7, 6, 9, 8, 9 a tak dále. Snížíte tím čas zálohování i obnovování dat. Nejdelší záloha se bude pořizovat po dvou dnech práce. Počet pásek, z nichž se budou obnovovat data, závisí na intervalu mezi úplnými zálohami, ale je rozhodně nižší než u procedury jednoduchého zálohování.

Tabulka 10.1 – Efektivní zálohovací schéma při více úrovních

Páska	Úroveň	Zálohuje se (dní)	Obnova z pásek
1	0	-	1
2	3	1	1, 2
3	2	2	1, 3
4	5	1	1, 2, 4
5	4	2	1, 2, 5
6	7	1	1, 2, 5, 6
7	6	2	1, 2, 5, 7
8	9	1	1, 2, 5, 7, 8
9	8	2	1, 2, 3, 7, 9
10	9	1	1, 2, 5, 7, 9, 10
11	9	1	1, 2, 5, 7, 9, 10, 11
...	9	1	1, 2, 5, 7, 9, 10, 11, ...

Tyto sofistikované postupy zálohování obvykle redukují pracnost, ale na druhou stranu je víc věcí, které jako správce systému musíte uhlídat. Sami se musíte rozhodnout, jestli se vám to vyplatí.

Program **dump** má zabudovanou podporu víceúrovňového zálohování. U příkazů **tar** a **cpio** je potřeba víceúrovňové zálohování implementovat pomocí skriptů.

Co zálohovat

Je pochopitelné, že budete chtít zálohovat vše, co se vám na zálohovací média vejde. Výjimkou je software, jenž lze snadno obnovit z instalačních médií[37]. Ale aplikace mohou používat mnoho různých konfiguračních souborů a ty je lepší zálohovat, protože si tím ušetříte nelehkou práci, kterou zabere jejich opakované nastavování. Další významnou výjimkou je souborový systém `/proc`. Obsahuje pouze data, která jádro vytváří automaticky, a proto nemá žádný smysl je zálohovat. Zvláště nežádoucí je zálohovat soubor `/proc/kcore`, protože je to pouze aktuální obraz fyzické paměti, takže je dost velký.

Nerozhodné jsou zprávy news, poštovní schránky, logovací soubory a mnoho dalších dat uložených v adresáři `/var`. Sami se musíte rozhodnout, co pokládáte za důležité.

Typickým příkladem toho, co se musí zálohovat, jsou uživatelské soubory (adresář `/home`) a systémové konfigurační soubory (adresář `/etc` a další konfigurační údaje, které jsou často rozeseté po celém souborovém systému).

37 Záleží na tom, co chápete pod slovem *snadno*. Někomu nevadí instalovat program z 35 disket...

Komprimované zálohy

Zálohy zabírají hodně místa na disku, což může stát dost peněz. Nároky na diskový prostor lze minimalizovat komprimováním záloh. Existuje několik způsobů, jak to udělat. Některé programy přímo podporují kompresi, například po zadání parametru `-gzip` (`-z`) programu **tar** verze GNU se jeho výstup spojí pomocí roury s kompresním programem **gzip**. Záloha se pak komprimuje před tím, než se zapíše na zálohovací médium.

Bohužel komprimované zálohy mohou způsobit vážné komplikace. Z povahy toho, jak komprimace funguje, vyplývá, že když se zapíše jediný bit špatně, bude nepoužitelný i celý zbytek komprimovaných dat. Některé zálohovací programy sice používají zabudované opravné algoritmy, ale žádná z těchto metod si neumí poradit s větším počtem chyb. Je-li tedy záloha komprimovaná způsobem, jakým to dělá program **tar** verze GNU, který svůj výstup komprimuje jako celek, může jediná chyba způsobit poškození celé zálohy. Stěžejním rysem zálohování musí být spolehlivost, takže tato metoda komprese není příliš dobrá.

Alternativou je komprimace jednotlivých záloh (souborů). Když se pak stane, že bude některý z nich poškozen, můžete použít jinou zálohu. Pravděpodobnost, že data, která tím ztratíte, se mohou poškodit i jiným způsobem, je stejná, takže na tom nebudete o moc hůř, než kdyby se nezálohovalo vůbec. Metodu komprimace jednotlivých souborů využívá například program **afio** (varianta programu **cpio**).

Komprese nějakou dobu trvá, takže zálohovací program nemusí být schopen data dostatečně rychle zapisovat na pásku[38]. Tento problém lze řešit využitím vyrovnávacích pamětí pro výstup zálohovacích programů (buď interních, je-li zálohovací program natolik chytrý, nebo pomocí jiných programů). Ale i tak by se mohlo stát, že zálohování nebude fungovat správně. S tímto problémem byste se měli setkat jen u velmi pomalých počítačů.

38 Pokud páska nemá co zapisovat, musí se zastavit. Tím se zálohování podstatně zpomalí, navíc to nedělá dobře ani pásce, ani mechanice.

Časové údaje

Čas je iluze. Čas oběda dvojnásob.
(Douglas Adams)

V této kapitole bude vysvětleno, jakým způsobem systém Linux udržuje správný čas a co je potřeba dělat, abyste potížím s nesprávným systémovým časem předešli. Obvykle nebudete muset dělat se systémovým časem nic, ale je užitečné chápat, oč jde.

Časové zóny

Měření času je založeno na převážně pravidelných přírodních jevech jako je střídání světla a tmy, jehož příčinou je rotace planety. Celkový čas mezi dvěma po sobě následujícími periodami je stejný, ale délka denní a noční doby se mění. Jedinou konstantou je poledne.

Poledne je denní doba, ve které se Slunce nachází na své nejvyšší pozici. Vzhledem k tomu, že Země ke kulatá[39], je poledne na různých místech zeměkoule v jinou dobu. Z toho vychází představa *místního času*. Lidstvo měří čas v různých jednotkách, jichž většina je rovněž svázána s přírodními jevy, jako je ona kulminace Slunce v poledne. Pokud se nacházíte stále na stejném místě, nezáleží na tom, že je lokální čas na jiných místech jiný.

Když ale potřebujete komunikovat se vzdálenějšími místy, uvědomíte si zároveň, že potřebujete jakýsi společný čas. V dnešní moderní době potřebuje hodně míst komunikovat s různými jinými místy na celé planetě. Proto byl zaveden společný standard měření času. Tomuto společnému času se říká *univerzální čas* (anglicky *universal time*, zkráceně UT nebo UTC), dříve označovaný jako greenwichský čas (Greenwich Mean Time nebo GMT), protože je místním časem v Greenwichi v Anglii. Když spolu potřebují komunikovat lidé, kteří mají různý lokální čas, mohou používat společný univerzální čas. Nevzniká tak zmatek kolem toho, kdy se která věc stala nebo měla stát.

Místním časům se říká časové zóny. I když se z pohledu geografie zdá logické, aby byla místa, která mají poledne ve stejnou dobu, začleněna do stejného časového pásma, neumožňují to hlediska politická. Mnoho zemí z různých důvodů zavádí *letní čas* (anglicky *daylight savings time*, zkráceně DST). Posouvají ručičky hodinek tak, aby měly více denního světla v době, kdy se pracuje, pak je v zimě zase přetáčí zpět. Jiné státy to tak pro změnu nedělají. No a ty, které tak činí, nejsou zajedno v tom, kdy má být čas posunut a mění pravidla z roka na rok. To je důvod, proč jsou konverze časů mezi pásmy poměrně netriviální.

Časová pásma je nejlepší určovat podle polohy nebo podle rozdílu mezi místním a univerzálním časem. Ve Spojených státech a některých dalších zemích mají místní časové zóny své jméno a odpovídající třípísmennou zkratku. Ale tato zkratka není jedinečná a neměla by se používat bez sou-

39 Jak naznačují poslední výzkumy.

časného označení země. Je lepší mluvit o lokálním čase například v Helsinkách, než o východoevropském čase (zkratka EET, East European Time), protože ne všechny státy východní Evropy leží ve stejném pásmu a nemusí mít stejná pravidla přechodu na letní čas.

Linux má balík programů pro práci s časovými pásmy. Tento software zná všechny existující časové zóny, a navíc jej lze jednoduše přizpůsobit v případě, že se změní pravidla určování času. Takže jediné, co musí správce systému udělat, je vybrat správné časové pásmo. Také každý z uživatelů si může nastavit své vlastní časové pásmo – to je důležité proto, že mnoho z nich používá prostřednictvím sítě Internet počítače v různých zemích. Když se ve vašem místním časovém pásmu změní pravidla přechodu na letní čas, budete muset aktualizovat tomu odpovídající část subsystému pro určování času. Stačí obvykle nastavit časové pásmo systému a upravit datové soubory zvoleného pásma, zbytek už nestojí za řeč.

Hardwarové a softwarové hodiny

Osobní počítač má hardwarové systémové hodiny napájené z baterií. Díky těmto bateriím hodiny fungují, i když je zbytek počítače bez proudu. Hardwarové systémové hodiny lze nastavovat pomocí programu setup v BIOSu, nebo přímo z operačního systému.

Jádro systému Linux udržuje vlastní čas nezávisle na hardwarových hodinách. Při zavádění operačního systému Linux nastaví své vlastní softwarové hodiny na stejný čas, jaký je v tom momentě na systémových hardwarových hodinách. Pak běží oboje hodiny nezávisle. Systém Linux udržuje vlastní čas pomocí softwarových hodin proto, že využívání systémových hardwarových hodin je dost pomalé a složité.

Hodiny jádra systému ukazují vždy univerzální čas. Proto jádro nemusí vůbec nic vědět o časových zónách – jednoduchost přispívá k vyšší spolehlivosti a ulehčuje možnost změny informací o vybraném časovém pásmu. Každý proces si převádí časová pásma sám (pomocí standardních nástrojů, jež jsou součástí balíku pro konverze časových zón).

Hardwarové systémové hodiny mohou být nastaveny na místní či univerzální čas. Je obvykle lepší mít je nastaveny na univerzální čas, protože pak není potřeba měnit nastavení hardwarových systémových hodin na začátku a konci období letního času (univerzální čas je totiž „pořád zimní“). Bohužel některé operační systémy pro PC – včetně systémů MS-DOS, Windows, OS/2 – počítají s tím, že hardwarové hodiny ukazují lokální čas. Systém Linux si umí poradit s oběma způsoby nastavení, ale v případě, že hardwarové hodiny ukazují místní čas, bude potřeba měnit jejich nastavení při přechodu z a na letní čas (jinak by totiž neukazovaly místní čas).

Zobrazení a nastavování času

V systému Debian je systémové časové pásmo určeno symbolickým odkazem `/etc/localtime`. Tento odkaz odkazuje na datový soubor pro dané časové pásmo, popisující místní časovou zónu. Jednotlivé datové soubory pro časová pásma jsou uloženy v adresáři `/usr/lib/zoneinfo`. Jiné distribuce Linuxu mohou parametry pro časová pásma nastavovat odlišně.

Uživatel může změnit své vlastní časové pásmo nastavením proměnné prostředí TZ. Není-li hodnota TZ nastavena, předpokládá se, že uživatel používá nastavení časového pásma systému. Syntaxe nastavení hodnoty proměnné TZ je popsána v manuálové stránce příkazu `tzset`.

Příkaz **date** ukáže aktuální datum a čas[40]. Například:

40 Pozor na příkaz **time**, ten aktuální čas nezobrazuje.

```
$ date
Sun Jul 14 21:53:41 EET DST 1996
$
```

Znamená to, že je neděle 14. července 1996, asi za deset minut deset večer, to všechno v časovém pásmu označeném „EET DST“, což by mohlo v angličtině znamenat „East European Daylight Savings Time“, tedy východoevropský letní čas. Příkazem **date** můžeme rovněž zjistit univerzální čas:

```
$ date -u
Sun Jul 14 18:53:42 UTC 1996
$
```

Příkazem **date** se také nastavují softwarové hodiny jádra systému:

```
# date 07142157
Sun Jul 14 21:57:00 EET DST 1996
# date
Sun Jul 14 21:57:02 EET DST 1996
#
```

Více podrobností hledejte v manuálové stránce příkazu **date** – syntaxe příkazu je tak trochu tajemná. Čas může nastavovat pouze superuživatel. I když může mít každý z uživatelů nastaveno své vlastní časové pásmo, systémový čas je pro všechny stejný.

Příkaz **date** ukazuje nebo nastavuje jenom softwarové hodiny. Příkaz **clock** synchronizuje systémové hardwarové a softwarové hodiny. Spouští se při zavádění systému, kdy se zjišťuje nastavení hardwarových systémových hodin. Podle nich se pak nastavují hodiny softwarové. Potřebujete-li nastavit oboje, nastavte nejprve softwarové hodiny příkazem **date**, následně hardwarové příkazem **clock -w**.

Parametrem `-u` sdělíte programu **clock**, že hardwarové hodiny ukazují univerzální čas. Přepínač `-u` je potřeba používat správně. V opačném případě bude mít váš systém mírný zmatek v tom, jaký je vlastně přesný čas.

Nastavení hodin se musí dělat opatrně. Mnoho částí operačního systému Unix spoléhá na to, že hodiny fungují správně. Například démon **cron** spouští pravidelně různé příkazy. Změníte-li nastavení hodin, může se stát, že nebude vědět, zda je potřeba některý z programů spustit, či nikoliv. Když na některém starším unixovém systému někdo nastavil hodiny o dvacet let dopředu, démon **cron** se snažil spustit všechny periodicky vykonávané příkazy za celých dvacet let naráz. Aktuální verze programu **cron** si s tímto problémem umí poradit. Přesto byste měli být při změnách času opatrní. Velké časové skoky a skoky vzad jsou nebezpečnější než menší změny a posuny dopředu.

Co když jdou hodiny špatně

Softwarové hodiny systému Linux nejdou vždy přesně. V chodu je udržují pravidelná *přerušení časovače* generovaná hardwarem PC. Běží-li v systému příliš mnoho procesů, může trvat obsluha přerušení časovače příliš dlouho a softwarové hodiny se začnou zpožďovat. Hardwarové systémové hodiny běží nezávisle na operačním systému a jsou obvykle přesnější. Jestliže často vypínáte a zapínáte počítač (to je případ většiny systémů, které neslouží jako servery), budete mít obvykle i přesnější systémový čas.

Potřebujete-li upravit nastavení hardwarových hodin, je obyčejně nejjednodušší restartovat systém, spustit setup BIOSu a udělat to tam. Tak lze předejít všem potížím, které by mohly být zapříčiněny změnou systémového času za běhu systému. Nemáte-li možnost upravit čas v nastaveních systému BIOS, nastavte jej příkazy **date** a **clock** (v tomto pořadí), ale připravte se na to, že se možná některé části systému budou chovat podivně, takže budete muset systém opět restartovat.

Počítač připojený k síti (i když jenom pomocí modemu) může automaticky kontrolovat správnost nastavení vlastních hodin tak, že je bude srovnávat s nastavením hodin nějakého jiného počítače. Jestli se o tomto jiném počítači ví, že jeho hodiny jdou velmi přesně, budou pak mít po připojení přesný čas oba systémy. Systémové časy dvou systémů lze seřídit příkazy **rdate** a **netdate**. Oba uvedené programy kontrolují čas na vzdáleném počítači (příkaz **netdate** i na několika vzdálených stanicích) a podle toho nastaví místní čas lokálního počítače. Počítač, na kterém se bude pravidelně spouštět některý z těchto příkazů, bude mít tak přesný čas, jak přesné budou hodiny vzdáleného počítače.

Slovníček

Knihovník Neviditelné univerzity se rozhodl sám napomoci obecnému porozumění tak, že napíše orangutansko-lidský slovník. Pracoval na něm celé tři měsíce. Nebylo to lehké. Zatím se dostal jenom ke slůvku „oook“.
(Terry Pratchett, Muži ve zbrani)

Zde je stručný seznam slovních definicí pojmů spojovaných s operačním systémem Linux, zvláště pak se správou systému.

ambice

Sepsání několika zábavných vět v naději, že se dostanou mezi linuxové soubory „cookie“.

aplikační program

Software, který dělá něco užitečného. Výsledek používání aplikačního programu je v podstatě důvodem zakoupení počítače. Viz také systémový program, operační systém.

démon

Proces číhající v pozadí – obvykle nenápadně – až do chvíle, než jej něco „rozjede“. Například démon **update** se probudí každých třicet sekund (nebo tak nějak) a vyprázdní buffer vyrovnávací paměti, démon **sendmail** se probere pokaždé, když někdo odesílá poštu.

jádro systému

Část operačního systému, která implementuje interakce s technickými prostředky výpočetního systému a sdílení zdrojů. Viz také systémový program.

operační systém

Software, jenž umožňuje sdílení zdrojů počítačového systému (procesor, paměť, diskový prostor, síť a tak dál) mezi uživateli a aplikačními programy, které uživatelé spouští. Nabízí zabezpečení systému řízením přístupu k němu. Viz také jádro systému, systémový program, aplikační program.

slovníček

Seznam slov a vysvětlení jejich významu. Nezaměňovat se slovníkem, jenž je rovněž seznamem slov a jejich vysvětlení.

souborový systém

Metody a datové struktury, které používá operační systém pro ukládání záznamů souborů na disk či diskovou oblast; způsob, kterým jsou soubory na disku organizovány. Pojem se používá též pro označení diskové oblasti či disku, kam se soubory ukládají, případně pro určení typu souborového systému.

systémový program

Programy, které implementují vyšší úrovně funkcionality operačního systému, tedy ty vlastnosti systému, které nejsou přímo závislé na hardwaru. Někdy mohou vyžadovat ke spuštění zvláštní oprávnění (například doručování elektronické pošty), ale velmi často jsou považovány za běžnou součást systému (například překladač). Viz také aplikační program, jádro systému, operační systém.

volání systému

Služby, které poskytuje aplikačním programům jádro systému, a způsob, jímž jsou volány. Viz sekci 2 manuálových stránek.

ČÁST III

Příručka správce sítě

Zdrojová podoba a předformátované verze

Zdrojový kód této knihy a další počítačově čitelné formáty můžete získat na Internetu na stránkách Linux Documentation Project na adrese *http://linuxdoc.org*, nebo na domovské stránce této knihy na adrese *http://www.iki.fi/viu/linux/sag*. K dispozici jsou minimálně verze PostScript a DVI.

Českou verzi této části naleznete na adrese *http://www.cpress.cz/knihy/ldp2/ldp-castIII.pdf*

Úvod

Internet dnes v řadě zemí představuje v mnoha domácnostech běžnou věc. Stále více jinak normálních lidí nastupuje na cestu po Informační superdálnici a počítačové sítě se tak dostávají na stejnou úroveň jako televizory nebo mikrovlnné trouby. Internet zažívá neuvěřitelnou míru mediální popularizace a řada společenskovědních oborů začíná považovat Web, Usenet a virtuální realitu za nový fenomén internetové kultury.

Počítačové sítě jsou samozřejmě poměrně stará věc. Propojování počítačů a vytváření lokálních sítí bylo běžné už i v prostředích jen s několika počítači, a stejně obvyklé je využití telekomunikačních linek k realizaci dálkových propojování různých systémů. Prudce se rozrůstající společenství celosvětové sítě je nyní běžně dostupné i nekomerčním domácím uživatelům. Cenově přístupné je vytvořit si internetový server nabízející poštovní a konferenční služby prostřednictvím vytáčené nebo ISDN linky, s nástupem technologie DSL (Digital Subscriber Line) a kabelových modemů se tento trend bezpochyby jen rozšíří.

Mluvit o počítačových sítích velmi často znamená mluvit zároveň o Unixu. Unix samozřejmě není jediným operačním systémem se síťovými možnostmi, nebude také navždy představovat vedoucí standard v této oblasti, nicméně již dlouhou dobu se v prostředí počítačových sítí používá a jistě se ještě nějakou dobu používat bude.

Co je na Unixu zejména zajímavé pro domácí uživatele je to, že existuje řada aktivit jak zajistit zdarma dostupný operační systém unixového typu pro platformu PC. Mezi tyto aktivity patří například 386BSD, FreeBSD a Linux.

Linux je zdarma distribuovaný klon operačního systému Unix, určený pro osobní počítače. V současné době funguje na řadě různých platforem, jako jsou rodina procesorů Intel, ale i procesory Motorola 680x0 a tedy počítače Commodore Amiga a Apple Macintosh, na počítačích Sun SPARC a Ultra-SPARC, Compaq Alpha, MIPS, PowerPC (například nová generace Apple Macintosh) a StrongARM (například rebel.com Netwinder nebo 3Com Palm). Linux byl přenesen i na některé méně rozšířené platformy, například Fujitsu AP-1000 a IBM System 390. Stále probíhá vývoj na přenesení Linuxu na další zajímavé platformy.

Linux byl vyvinut velkým týmem dobrovolníků spolupracujících po Internetu. Celý projekt zahájil v roce 1990 Linus Torvalds, finský vysokoškolák, jako projekt do předmětu Operační systémy. Od té doby se Linux stal plnohodnotným unixovým klonem, na němž běží tak rozdílné aplikace jako jsou simulační a modelační programy, textové editory, systémy rozpoznávání řeči, webové prohlížeče a celá řada dalších programů včetně řady vynikajících her. Podporována je velmi široká škála hardware, a navíc Linux obsahuje úplnou implementaci protokolu TCP/IP včetně SLIP, PPP a firewallů, úplnou implementaci protokolu IPX a celou řadu dalších funkcí a protokolů, které v jiných operačních systémech nejsou. Linux je výkonný, rychlý a zadarmo. Jeho popularita v celém světě silně narůstá.

Samotný operační systém Linux je distribuován za podmínek licence GNU General Public Licence, tedy pod stejnou licencí, pod jakou jsou distribuovány programy vyvíjené Free Software Foundation. Tato licence komukoliv umožňuje programy dále distribuovat a modifikovat (ať už zdarma nebo za úplatu) za podmínky, že všechny modifikace a distribuce budou rovněž dále distribuovatelné zdarma. Termín „free software“ znamená možnost aplikaci volně šířit, nikoliv nutnost šířit ji zdarma.

Účel a orientace této knihy

Tato kniha vznikla se záměrem poskytnout síťovým administrátorům v prostředí Linuxu jednoduchou referenční příručku. Informace pokrývající prakticky všechny důležité operace prováděné síťovým administrátorem zde naleznou jak začátečníci, tak zkušení uživatelé. Samozřejmě rozsah pokrývané problematiky je tak široký, že není prakticky možné zmínit se o všem, co může být za nějakých okolností významné. Snažili jsme se popsat nejdůležitější a nejběžnější oblasti. Zjistili jsme, že i začátečníci pracující s Linuxem v síti byli schopni podle tohoto textu úspěšně nakonfigurovat síťové prostředí a dozvěděli se dost, aby se mohli věnovat i další specializované problematice.

Existuje celá řada knih a dalších zdrojů informací, kde se můžete dozvědět více o jakékoliv problematice pokryté v této knize (možná s výjimkou některých specificky linuxových funkcí, jako je například nové firewallové rozhraní, které ještě není dobře dokumentováno nikde). Připravili jsme pro vás přehled literatury, která by vám měla pomoci v dalším studiu.

Zdroje informací

Pokud s Linuxem pracujete nově, může pro vás být zajímavá celá řada dalších informací. Pomůže vám přístup k Internetu, i když není nezbytný.

Příručky Linuxového dokumentačního projektu

Linuxový dokumentační projekt (LDP) je skupina dobrovolníků, kteří pracují na tvorbě knih (příruček), dokumentů HOWTO a manuálových stránek, pokrývající širokou oblast témat od instalace po programování jádra. Práce skupiny LDP zahrnuje mimo jiné:

Linux Installation and Getting Started

Matt Welsh a kolektiv. Tato kniha popisuje jak Linux získat a nainstalovat. Obsahuje základní seznámení s Linuxem, s problematikou administrace systému, s X-Windows System a se sítěmi.

Linux System Administrators Guide

Lars Wirzenius a Joanna Oja. Tato kniha představuje obecnou příručku administrace linuxového systému a pokrývá témata jako je vytváření a konfigurace uživatelů, zálohování dat, konfigurace hlavních softwarových balíků a instalace a aktualizace různých programů.

Linux System Administration Made Easy

Steve Frampton. Tato kniha popisuje denní úkony administrace a údržby systému vztahující se k uživatelům.

Linux Programmers Guide

Scott Burkett, Sven Goldt, John D. Harper, Sven van der Meer a Matt Welsh. Tato kniha pokrývá tematiku týkající se vývoje aplikačního software pro Linux.

The Linux Kernel

David A. Rusling. Tato kniha představuje úvod do problematiky jádra Linuxu, jak je vystavěno a jak funguje. Seznamte se s jádrem svého systému.

The Linux Kernel Module Programming Guide

Ori Pomerantz. Tato kniha vysvětluje jak vytvářet moduly jádra.

Další knihy jsou ve stádiu vzniku. Přesnější informace naleznete na webových stránkách LDP na adrese *http://www.linuxdoc.org* nebo na některém z četných zrcadel.

Dokumenty Linux HOWTO představují vyčerpávající sérii dokumentů, které pokrývají různé aspekty systému – například jak instalovat a nakonfigurovat systém X Windows, nebo jak pod Linuxem programovat v assembleru. Můžete je obecně najít v adresáři *HOWTO* na dále uvedených FTP serverech, nebo na webových stránkách některého z mnoha zrcadlících serverů LDP. Seznam dostupných dokumentů naleznete v přehledu literatury na konci této příručky nebo v souboru *HOWTO-INDEX*.

Mohou vás zajímat například dokumenty *Installation HOWTO*, popisující jak Linux nainstalovat, *Hardware Compatibility HOWTO*, obsahující seznam hardware, se kterým Linux pracuje, nebo *Distribution HOWTO*, který obsahuje seznam dodavatelů, jež prodávají Linux na disketách a CD discích.

V přehledu literatury v této příručce jsou uvedeny dokumenty HOWTO, které se vztahují k síťové problematice.

Linux Frequently Asked Questions

Linux Frequently Asked Questions with Answers (FAQ) obsahují široký okruh otázek a odpovědí týkajících se systému. Tento dokument představuje nezbytnou literaturu pro všechny začátečníky.

Dokumentace dostupná přes FTP

Pokud můžete používat anonymní FTP službu, veškerou výše uvedenou dokumentaci můžete získat na řadě různých serverů, například *metalab.unc.edu:/pub/Linux/docs* nebo *tsx-11.mit.edu:/pub/linux/docs*.

Dokumentace dostupná přes WWW

Existuje celá řada webových stránek věnovaná problematice Linuxu. Domovskou stránku dokumentačního projektu najdete na adrese *http://www.linuxdoc.org*.

Open Source Writers Guild (OSWG) je projekt, jehož rozsah zahrnuje mimo jiné i Linux. Výsledkem projektu OSWG je například i tato příručka, cílem projektu je podpora volně distribuované dokumentace. Domovskou stránku tohoto projektu najdete na adrese *http://www.oswg.org:8080/oswg*.

Na obou uvedených adresách najdete hypertextové (a jiné) verze řady dokumentů týkajících se Linuxu.

Dokumentace dostupná komerčně

Literaturu týkající se Linuxu a LDP vydává celá řada nakladatelství a softwarových společností. Dvě z nich jsou:

Specialized Systems Consultants, Inc. (SSC)
http://www.ssc.com/
P.O. Box 55549 Seattle, WA 98155-0549
1-206-782-7733
1-206-782-7191 (FAX)
sales@ssc.com

a

Linux Systems Labs
http://www.lsl.com/
18300 Tara Drive
Clinton Township, MI 48036
1-810-987-8807
1-810-987-3562 (FAX)
sales@lsl.com

Obě společnosti vydávají v knižní podobě vybrané dokumenty HOWTO a další dokumentaci týkající se Linuxu.

Celou sérii knih věnovaných Linuxu vydalo nakladatelství O'Reilly & Associates. Některé z nich vycházejí z LDP, většina je však vydávána nezávisle na LDP. Patří sem například:

- *Running Linux* – Instalační a uživatelská příručka, popisující jak co nejlépe využít osobní počítač se systémem Linux.
- *Learning Debian GNU/Linux, Learning Red Hat Linux* – Ještě základnější příručka než *Learning Linux*. Tyto knihy obsahují uvedené populární distribuce na CD discích a uvádějí podrobný návod pro jejich instalaci, nastavení a použití.
- *Linux in a Nutshell* – Další kniha z úspěšné série „v kostce", představuje referenční text týkající se Linuxu.

Linux Journal a Linux Magazine

Linux Journal a *Linux Magazine* jsou měsíčně vydávané časopisy pro linuxovou komunitu, které píše a vydává řada linuxových aktivistů. Obsahují články od návodů pro začátečníky až po podrobné popisy interních vlastností jádra. Pokud máte přístup k Usenetu, představují tyto časopisy dobrý způsob jak být v kontaktu s linuxovou komunitou.

Linux Journal je nejstarší a je vydáván společností S.S.C. Incorporated, o níž jsme se už zmínili. Tento časopis najdete i na webu na adrese *http://www.linuxjournal.com*.

Linux Magazine je novější, vydávaný nezávisle. Jeho domovskou stránku najdete na adrese *http://www.linuxmagazine.com*.

Linux Usenet Newsgroups

Pokud máte přístup k Usenetu, můžete využít následujících linuxových skupin:

- *comp.os.linux.announce* – Moderovaná diskusní skupina obsahující informace o nových programech, distribucích, chybách a dalších novinkách. Tuto skupinu by měli sledovat všichni uživatelé Linuxu. Přihlásit se můžete e-mailem na adrese *linux-announce@news.ornl.gov*.
- *comp.os.linux.help* – Skupina věnovaná obecně problematice instalace a použití Linuxu.
- *comp.os.linux.admin* – Diskuse věnovaná administraci systému Linux.
- *comp.os.linux.networking* – Diskuse týkající se síťové problematiky v Linuxu.
- *comp.os.linux.development* – Diskuse věnovaná vývoji jádra a celého systému.
- *comp.os.linux.misc* – Skupina pro všechna témata, která nezapadají do předchozích kategorií.

Existuje i několik diskusních skupin týkajících se Linuxu a vedených v jiných jazycích než v angličtině, například *fr.comp.os.linux* ve francouzštině nebo *de.comp.os.linux* v němčině.

Linuxové poštovní konference

Existuje celá řada specializovaných linuxových poštovních konferencí, kde můžete potkat mnoho lidí ochotných odpovědět vám na vaše otázky.

Nejznámější jsou konference provozované univerzitou Rutgers. Přihlásit se můžete odesláním e-mailu, který bude vypadat takto:

```
To: majordomo@vger.rutgers.edu
Subject: libovolný
Body:

subscribe název-konference
```

Některé z konferencí věnovaných problematice sítí jsou:

- *linux-net* – Diskuse týkající se sítí v Linuxu.
- *linux-ppp* – Diskuse týkající se implementace PPP v Linuxu.
- *linux-kernel* – Diskuse týkající se vývoje jádra.

Online podpora

Existuje celá řada možností, jak kontaktovat online podporu, kterou poskytují dobrovolníci po celém světě a nabízejí rady a služby, kterými pomáhají uživatelům s jejich otázkami a problémy.

OpenSource IRC Network je IRC služba věnovaná Open projektům jako jsou Open Source a Open Hardware. Některé kanály této služby nabízejí online podporu pro Linux. IRC je zkratka z Internet Relay Chat, což je síťová služba, která vám umožňuje interaktivně si povídat s jinými lidmi na Internetu. IRC servery nabízejí různé kanály, na nichž si povídají různé skupiny lidí. Cokoliv, co v daném kanálu napíšete, si budou moci přečíst všichni uživatelé téhož kanálu.

Na OpenSource IRC existuje celá řada aktivních kanálů, kde 24 hodin denně, 7 dní v týdnu najdete lidi, kteří jsou ochotni vám pomoci vyřešit jakékoliv problémy týkající se Linuxu, nebo se kterými si budete moci prostě jen popovídat. Tuto službu můžete využívat, když si nainstalujete IRC klienta jako je například *irc-II*, připojíte se na server *irc.openprojects.org:6667* a vstoupíte do kanálu *#linpeople*.

Uživatelské skupiny

Po celém světě existuje řada uživatelských skupin, které nabízejí přímou podporu. Řada těchto skupin poskytuje aktivity jako jsou instalační dny, schůzky a semináře, demonstrační noci a jiné společenské události. Uživatelské skupiny představují vynikající způsob jak se kontaktovat s dalšími uživateli Linuxu ve vašem okolí. Existuje celá řada seznamů těchto skupin. Mezi nejznámější patří:

- Groups of Linux Users Everywhere – *http://www.ssc.com/glue/groups*
- LUG list project – *http://www.nllgg.nl/lugww*
- LUG registry – *http://www.linux.org/users*

Získání Linuxu

Neexistuje žádná jednotná podoba Linuxu, namísto toho existuje řada různých distribucí, jako jsou Debian, RedHat, Caldera, Corel, SuSE nebo Slackware. Každá z těchto distribucí obsahuje vše, co

potřebujete ke spuštění úplného systému: jádro, základní nástroje, knihovny, podpůrné soubory a aplikační software.

Distribuce Linuxu můžete získat různými způsoby, například pomocí Internetu. Každá z hlavních distribucí nabízí vlastní FTP a webové servery. Například:

- Caldera – *http://www.caldera.com*, *ftp://ftp.caldera.com*
- Corel – *http://www.corel.com*, *ftp://ftp.corel.com*
- Debian – *http://www.debian.org*, *ftp://ftp.debian.org*
- RedHat – *http://www.redhat.com*, *ftp://ftp.redhat.com*
- Slackware – *http://www.slackware.com*, *ftp://ftp.slackware.com*
- SuSE – *http://www.suse.com*, *ftp://ftp.suse.com*

Na řadě všeobecných archivních FTP serverů jsou zrcadleny různé distribuce Linuxu. Mezi nejznámější servery patří:

metalab.unc.edu:/pub/Linux/distributions/
ftp.funet.fi:/pub/Linux/mirrors/
tsx-11.mit.edu:/pub/linux/distributions/
mirror.aarnet.edu.au:/pub/linux/distributions/

Řadu moderních distribucí je možné instalovat přímo z Internetu. Nicméně pro typickou instalaci musíte nahrát celou řadu programů, takže tuto variantu můžete zvolit pouze pokud máte rychlé a trvalé internetové připojení, nebo pokud chcete aktualizovat stávající instalaci[1].

Linux si můžete koupit na CD disku od celé řady prodejců software. Pokud jej nemají ve vašem oblíbeném počítačovém obchodě, měli byste je požádat, ať si jej objednají. Na CD discích můžete dostat všechny populární distribuce. Někteří prodejci nabízejí celé sady disků, kde každý obsahuje jinou distribuci. Toto řešení je ideální, pokud si chcete vyzkoušet různé distribuce předtím, než se rozhodnete pro svou oblíbenou.

Standardy souborového systému

V minulosti byly různé distribuce a programové balíky určené pro Linux ovlivňovány tím problémem, že neexistoval žádný všeobecně uznávaný způsob organizace souborového systému. To vedlo k nekompatibilitám mezi různými balíky a komplikovalo to uživatelům a administrátorům práci při hledání různých souborů a programů.

K vyřešení této situace byla v srpnu 1993 založena skupina Linux File System Standard Group (FSSTND). Po půl roce debat vydala skupina návrh definující koherentní strukturu souborového systému a udávající umístění základních programů a konfiguračních souborů.

Bylo doporučeno tento standard implementovat v hlavních distribucích Linuxu. Trochu smůla spočívá v tom, že sice ve většině distribucí bylo dosaženo jisté shody s tímto standardem, nicméně jen velmi málo distribucí jej implementuje plně. V celé knize budeme předpokládat, že soubory, o nichž budeme hovořit, jsou umístěny tam, kde mají být podle standardu FSSTND. O možných alternativních umístěních se zmíníme pouze tam, kde existuje dlouhá tradice rozcházející se se standardem.

Standard FSSTND se dále vyvíjel a v roce 1997 byl nahrazen standardem Linux File Hierarchy Standard (FHS). Tento standard se zabývá i multiarchitekturovou problematikou, což standard FSST-

1 A nebo pokud jste extrémně netrpěliví a uvědomujete si, že 24 hodin, které může trvat stažení souborů z Internetu, je podstatně kratší než 72 hodin, které může trvat dodání objednaného CD.

ND neřešil. FHS můžete nalézt v dokumentaci na všech větších FTP serverech věnovaných Linuxu a na jejich zrcadlech, nebo přímo na domovské stránce na adrese *http://www.pathname.com/fhs*. Koordinátora skupiny, Daniela Quinlana, můžete kontaktovat na adrese *quinlan@transmeta.com*.

Linux Standard Base

Velké množství distribucí Linuxu představuje výbornou možnost volby pro uživatele, nicméně to zároveň představuje problém pro programátory – zejména pro ty, kteří vyvíjejí komerční programy.

Každý distribuční balík obsahuje své základní knihovny, konfigurační nástroje, systémové aplikace a konfigurační soubory. Bohužel rozdíly v jejich verzích, názvech a umístěních velmi komplikují snahu zjistit, co která distribuce obsahuje. Tím se stává velice obtížné vyvíjet binární aplikace, které budou spolehlivě fungovat ve všech distribucích.

Jako pomoc při řešení tohoto problému byl založen další projekt, pojmenovaný „Linux Standard Base“. Má za cíl definovat standardní složení distribuce, kterého by se měly všechny hlavní distribuce držet. Pokud programátor vytvoří aplikaci využívající tuto standardní základní platformu, bude aplikace fungovat na všech distribucích, které standard dodržují.

Informace o stavu projektu můžete najít na jeho domovských stránkách na adrese *http://www.linuxbase.org*.

Pokud vám záleží na univerzální použitelnosti programů, zejména u komerčně dodávaných programů, ověřte si, že vámi používaná distribuce se snaží vyhovět standardům podle tohoto projektu.

O této knize

Když se Olaf v roce 1992 přidal k práci na dokumentačním projektu, napsal dvě krátké kapitoly o UUCP a **smailu**, které měly být příspěvkem k příručce systémového administrátora. Vývoj TCP/IP byl v začátcích a když se tyto dvě malé kapitoly začaly rozrůstat, navrhl, že by možná bylo dobré vytvořit samostatnou příručku síťového administrátora. Všichni ho za ten nápad chválili a poradili mu, ať jej realizuje. Tak to udělal a v září roku 1993 vyšla první verze příručky síťového administrátora.

Olaf pokračoval v práci na této příručce a posléze vydal podstatně obsáhlejší verzi. Vince Skahan doplnil původní kapitolu o programu **sendmail**, která je v tomto vydání úplně přepracována vzhledem k novému konfiguračními rozhraní programu.

Příručka kterou čtete je revize a aktualizace ohlášená nakladatelstvím O'Reilly & Associates a provedená Terry Dawsonem[2]. Terry byl přes 20 let radioamatérem a více než 15 let pracoval v telekomunikačním průmyslu. Byl spoluautorem původního dokumentu NET-FAQ a je autorem a správcem řady dokumentů týkajících se sítí. Terry vždy nadšeně podporoval příručku síťového administrátora a doplnil do ní několik kapitol týkajících se různých síťových oblastí a podílel se i na její postupné aktualizaci tak, jak probíhal vývoj síťových možností Linuxu.

Kapitolu o programu **exim** napsal Philip Hazel[3], který je hlavním vývojářem a správcem tohoto balíku.

2 Terryho Dawsona můžete kontaktovat na adrese *terry@linux.org.au*.

3 Philipa Hazela můžete kontaktovat na adrese *ph10@cus.cam.ac.uk*.

Tato kniha je organizována přibližně chronologicky podle postupu, kterým se konfigurují síťové vlastnosti systému. Začíná popisem základní síťové problematiky, zejména sítí na bázi protokolů TCP/IP. Pak přechází od konfigurace TCP/IP na úrovni zařízení přes nastavení firewallu, účtování a konfigurace maškarády, až k nastavení obvyklých aplikací jako jsou **rlogin** a **friends**, Network File System a Network Information System. Dále následuje kapitola věnovaná tomu, jak počítač nakonfigurovat jako uzel UUCP. Zbývající část knihy je věnována převážně dvěma hlavním aplikacím, které běží na protokolech TCP/IP a UUCP: elektronické poště a zprávám. Zvláštní kapitoly jsou věnovány protokolu IPX a souborovému systému NCP, protože se používají v řadě prostředí, kde bývá instalován i Linux.

Část věnovaná elektronické poště představuje úvod do obsáhlé problematiky přenosu a směrování pošty a různých adresačních schémat, která můžete potkat. Popisuje konfiguraci a správu programu **exim**, přenosového agenta, který je ideální ve většině případů kdy se nepoužívá UUCP a programu **sendmail**, který umožňuje řešení i velmi složitých směrovacích situací zahrnujících protokol UUCP.

Kapitola o zprávách představuje přehled toho, jak fungují Usenet news. Hovoří o programech INN a C, v současné době dvou nejrozšířenějších programech pro přenos zpráv a o protokolu NNTP, který umožňuje přístup ke zprávám na lokální síti. Kniha končí kapitolou popisující údržbu a práci s nejrozšířenějšími programy pro čtení zpráv.

Samozřejmě žádná kniha nemůže nikdy úplně zodpovědět všechny otázky, které se mohou vyskytnout. Pokud se tedy stane, že se budete držet popsaného postupu a něco stále nebude fungovat, buďte trpěliví. Některé vaše problémy mohou být způsobeny chybou na naší straně (viz část *Oznamování změn* dále v tomto úvodu), mohou být ale způsobeny i změnami v používaných programech. Proto byste měli vždy nejprve využít uvedených informačních pramenů. Existuje velká šance, že se svým problémem nebudete osamoceni a že tedy bude známo řešení nebo alespoň obejití daného problému. Pokud máte možnost, měli byste vždy používat nejnovější jádro a síťové programy, které můžete získat na některém FTP serveru nebo BBS ve vašem okolí. Řada problémů bývá způsobena programy v různém stádiu vývoje, které spolu nedokáží korektně pracovat. Koneckonců, práce na vývoji Linuxu stále probíhají.

Oficiální tištěná verze

Na podzim roku 1993 požádal Andy Oram, který se účastní na práci na LDP od samého počátku, Olafa, aby tuto příručku vydal v nakladatelství O'Reilly & Associates. Ten s tím souhlasil, protože nikdy netušil, že se příručka setká s tak velkým úspěchem. Nakonec tedy Olaf a Andy souhlasili s tím, že O'Reilly vydá rozšířenou oficiální verzi této příručky s tím, že Olaf si ponechal svá autorská práva k ní a příručka proto může být volně distribuována. Můžete si tedy vybrat: různé verze tohoto dokumentu můžete najít na serverech zrcadlících obsah LDP a ty si můžete vytisknout, nebo si můžete koupit tištěnou verzi v nakladatelství O'Reilly.

Proč byste tedy měli platit za něco, co můžete mít zadarmo? Že by se Tim O'Reilly definitivně zbláznil a vydal něco, co si může kdokoliv vytisknout a prodávat sám[4]. Jsou nějaké rozdíly mezi různými verzemi?

Odpovědi na jednotlivé otázky jsou „to záleží", „ne, určitě ne" a „ano i ne". O'Reilly & Associates na sebe vzali riziko vydání této příručky a zdá se, že se jim to vyplatilo, protože nás požádali o spolupráci na novém vydání. Domníváme se, že tento projekt může sloužit jako příklad toho,

4 Pozor: Elektronickou verzi příručky si samozřejmě můžete stáhnout a vytisknout, nicméně knihu z nakladatelství O'Reilly nesmíte prohnat kopírkou a eventuálně prodávat její kopie.

jak může svět free software a komerční společnost spolupracovat na něčem, z čeho mají prospěch oba. Podle nás služba, kterou O'Reilly poskytuje linuxové komunitě (kromě toho, že si můžete koupit tuto knihu) spočívá také v tom, že přispívá k tomu, aby byl Linux chápán jako něco seriózního – jako životaschopná a užitečná alternativa ke komerčním operačním systémům. Špatné je to knihkupectví, které nemá alespoň jednu polici plnou O'Reillyho knih o Linuxu.

Proč tuto knihu vydali? Zdálo se jim, že to je „jejich" typ knihy. Je to kniha, která by vznikla, kdyby požádali nějakého autora o její napsání. Tempo výkladu, míra podrobností a styl odpovídá tomu, co běžně nabízejí.

Smyslem licence LDP je to, aby nikdo nezůstal „mimo mísu". Tuto příručku může vytisknout kdokoliv a nikdo se na vás nebude zlobit, pokud si ji pořídíte jinak. Pokud jste ale vytištěnou verzi neviděli, podívejte se na ni v knihkupectví nebo u kamaráda. Domníváme se, že se vám bude líbit a rádi si ji koupíte.

Jaké jsou rozdíly mezi tištěnou a elektronickou verzí? Andy Oram odvedl skvělou práci při přepisu našich nesourodých poznámek v něco, co stojí za vytištění. (Kromě toho revidoval i další knihy vydané v rámci LDP a svou prací odvedl cenný přínos pro celou linuxovou komunitu.)

Jakmile Andy začal původní podobu této příručky upravovat, kniha se rychle vzdalovala od své původní podoby a s každou další konzultací se měnila více a více. Možnost pracovat s profesionálním redaktorem je něco, co by se mělo využívat. V mnoha směrech byl Andyho přínos stejný jako přínos autorů. To platí samozřejmě i o dalších lidech, kteří se podíleli na tom, aby kniha získala podobu, kterou vidíte. Všechny tyto úpravy byly promítnuty zpátky do elektronické verze, takže v jejich obsahu žádné rozdíly nejsou.

Nicméně tištěná verze *je* rozdílná. Je to profesionálně vytištěná vázaná kniha a i když si dáte tu práci s vytištěním elektronické verze, nedosáhnete stejné kvality a pravděpodobně vás to vyjde dráž než koupě knihy. Navíc je v ní náš amatérský ilustrátorský styl nahrazen profesionální prací pracovníků nakladatelství O'Reilly. Kniha obsahuje rejstřík, který vám usnadní práci s ní. Pokud zamýšlíte celou příručku od začátku do konce přečíst, rozhodně vám doporučujeme její knižní podobu.

Přehled

Kapitola 1 hovoří o historii Linuxu a pokrývá základní síťovou problematiku protokolů UUCP, TCP/IP a dalších, hardware i bezpečnosti. Následující kapitoly se věnují konfiguraci Linuxu pro práci v sítích TCP/IP a práci s některými hlavními aplikacemi. Podrobněji o protokolu IP hovoříme v **kapitole 2**, a pak už se pouštíme do opravdové práce. Pokud víte, jak funguje směrování v protokolu IP a jak se provádí zjišťování adres, můžete tuto kapitolu přeskočit.

Kapitola 3 hovoří o úplně základní konfigurační problematice, jako je například sestavení jádra a nastavení síťové karty. Konfigurace sériových portů je popsána samostatně v **kapitole 4**, protože tato problematika se týká nejen TCP/IP, ale vztahuje se i k UUCP.

Kapitola 5 popisuje nastavení počítače pro síť TCP/IP. Obsahuje návod pro konfiguraci samostatného počítače pouze s rozhraním *loopback*, i konfiguraci počítačů připojených k Ethernetu. Popisuje také několik užitečných nástrojů pro testování nastavení. **Kapitola 6** popisuje jak nakonfigurovat rozlišování jmen a jak vytvořit name server.

Kapitola 7 popisuje jak vytvořit SLIP spojení a nabízí podrobný popis programu **dip**, který vám umožňuje automatizovat většinu nezbytných kroků. **Kapitola 8** hovoří o PPP a programu **pppd**, démonu služby PPP.

Kapitola 9 navazuje na problematiku bezpečnosti a popisuje jak vytvořit na Linuxu firewall a jak jej nakonfigurovat pomocí nástrojů **ipfwadm**, **ipchains** a **iptables**. Firewall umožňuje přesně nastavit to, kdo může k vaší síti přistupovat a které její počítače mu budou dostupné.

Kapitola 10 popisuje, jak nastavit IP účtování, takže budete moci zjistit, kde k jakým přenosům dochází a co je vyvolává.

Kapitola 11 hovoří o funkci nazvané IP maškaráda, která umožňuje připojit k Internetu celou síť prostřednictvím jediné IP adresy, takže všechny interní systémy budou před vnějškem skryty.

Kapitola 12 představuje úvod do nastavení některých nejvýznamnějších síťových aplikací jako jsou **rlogin**, **ssh** a další. Hovoří také o tom, jak jsou různé služby obsluhovány superdémonem **inetd** a jak můžete některé z hlediska bezpečnosti významné služby omezit jen na důvěryhodné počítače.

Kapitola 13 a **kapitola 14** popisují NIS a NFS. NIS je nástroj pro distribuci administrativních informací, například uživatelských hesel, v rámci lokální sítě. NFS umožňuje sdílet mezi počítači v síti souborový systém.

V **kapitole 15** popisujeme IPX a souborový systém NCP. Tím je umožněna integrace Linuxu do prostředí Novell NetWare a sdílení souborů a tiskáren s nelinuxovými počítači.

Kapitola 16 představuje vyčerpávající úvod do administrace Taylor UUCP, zdarma dostupné implementace UUCP služby.

Zbytek knihy představuje podrobného průvodce elektronickou poštou a zprávami. **Kapitola 17** představuje základy problematiky elektronické pošty, například jak vypadá elektronická adresa a jak poštovní systém obsluhuje doručení pošty od odesílatele k adresátovi.

Kapitola 18 a **kapitola 19** hovoří o programech **sendmail** a **exim**, tedy o dvou transportních agentech, které se v Linuxu používají. Věnujeme se záměrně oběma, protože **exim** je jednodušší na instalaci a nastavení, zatímco **sendmail** obsahuje podporu UUCP.

Kapitola 20 až **kapitola 23** hovoří o tom, jak v Usenetu pracují zprávy a jak instalovat a používat C News, **nntpd** a INN: tři oblíbené nástroje pro práci s Usenetovými zprávami. Po stručném úvodu v kapitole 20 si můžete přečíst kapitolu 21, která pro vás bude zajímavá, pokud budete chtít zprávy přenášet programem C News, který se typicky používá v prostředí UUCP. Následující kapitoly hovoří o moderních alternativách k programu C News, které umožňují přenos zpráv internetovým protokolem NNTP (Network News Transfer Protocol). Kapitola 22 popisuje jak nainstalovat jednoduchého NNTP démona, program **nntpd**, který umožní přístup ke zprávám z lokální sítě, zatímco kapitola 23 popisuje instalaci mnohem robustnějšího serveru InterNet News (INN) pro velké provozy. Konečně kapitola 24 hovoří o práci s různými programy pro čtení zpráv.

Konvence používané v této části

Všechny příklady uváděné v této části předpokládají, že používáte **sh**-kompatibilní příkazový interpret (shell). Příkazový interpret **bash** je kompatibilní s **sh** a je standardním příkazovým interpretem ve všech distribucích Linuxu. Pokud používáte **csh**, budete muset příklady upravit.

Následuje přehled typografických konvencí, které v knize používáme:

Kurzíva se používá pro názvy souborů a adresářů, programů a příkazů, řádkových voleb, e-mailových adres, URL a pro zvýraznění nových pojmů.

Tučné písmo používáme pro názvy počítačů, sítí, uživatelská jména a identifikátory, a příležitostně ke zvýraznění něčeho.

`Neproporcionální` písmo používáme pro výpisy programů, výstupy příkazů a k indikaci proměnných prostředí a klíčových slov, která se v kódu objevují.

`Neproporcionální kurzíva` znázorňuje proměnné parametry, klíčová slova nebo text, který musí uživatel nahradit skutečnou hodnotou.

`Neproporcionální tučné` písmo znázorňuje příkazy nebo jiný text, který musí uživatel zapsat.

> **Varování** Takto označený text představuje varování. Popisuje situace, kdy chyba může vést k poškození systému nebo by se velmi obtížně opravovala.

Oznamování změn

Informace uvedené v knize jsme pečlivě testovali a ověřovali, nicméně můžete narazit na funkce, které se nějak změnily nebo i na naši chybu. Budeme rádi, když nás budete informovat o jakýchkoliv nalezených chybách i o doporučeních pro příští vydání. Můžete nám psát na adresu

O'Reilly & Associates
101 Morris Street
Sebastopol, CA 95472
1-800-988-9938 (USA a Kanada)
1-707-829-0515 (mezinárodní a lokální)
1-707-829-0104 (fax)

Kromě toho nám můžete psát i elektronicky. Pokud se chcete přidat na seznam nakladatelství nebo si chcete nechat poslat katalog, napište na adresu

info@oreilly.com

Technické otázky a komentáře ke knize posílejte na adresu

bookquestions@oreilly.com

Tato kniha má svou domovskou stránku, na které najdete příklady, opravy a plány do budoucna. Najdete ji na adrese

http://www.oreilly.com/catalog/linag2

Další informace o této i dalších knihách najdete na stránkách vydavatelství O'Reilly na adrese

http://www.oreilly.com

Poděkování

Toto vydání příručky síťového administrátora vděčí prakticky za všechno vynikající práci Olafa a Vinceho. Těžko se dá vysvětlit, jakou práci představuje napsání knihy jako je tato, to si musíte vyzkoušet sami. Aktualizace knihy byla náročná činnost, nicméně vycházeli jsme z dobrého základu, a proto to bylo příjemné.

Dále vděčíme mnoha lidem, kteří knihu přečetli a podíleli se na odstranění chyb, ať už technických nebo gramatických (nikdy jsem neslyšel o něčem takovém jako je přívlastek volný a těsný). Phil Hughes, John Macdonald a Erik Ratcliffe odvedli velmi záslužnou (a velmi konzistentní) práci nad obsahem knihy.

Chtěli bychom také poděkovat lidem z nakladatelství O'Reilly, se kterými byla radost pracovat. Byli to Sarah Jane Shangraw, která knize dala podobu, v níž ji vidíte, Maureen Dempsey, která redi-

govala text, Rob Romano, Rhon Porter a Chris Reilly, autoři obrázků, Hanna Dyer, která navrhla obálku, Alicia Cech, David Futato a Jennifer Niedherst, tvůrci sazby, Lars Kaufman, který navrhl na titulní stranu obrázek pařezu, Judy Hoer, autorka rejstříku a konečně Tim O'Reilly za odvahu tento projekt realizovat.

Vřelé poděkování si zaslouží Andres Sepúlveda, Wolfgang Michaelis, Michael K. Johnson a řada dalších programátorů, kteří strávili mnoho času nad ověřováním informací, které v knize najdete. Phil Hughes, John MacDonald a Eric Ratcliffe se řadou návrhů podíleli na obsahu druhého vydání. Děkujeme také všem, kteří četli první verzi této příručky a poslali nám opravy a připomínky. Snad úplný seznam přispěvatelů najdete v souboru `Thanks` v elektronické distribuci. A konečně, kniha by nemohla vzniknout bez Holgera Grotheho, který zajistil Olafovi připojení k Internetu, potřebné ke vzniku první verze knihy.

Olaf by chtěl poděkovat následujícím skupinám a lidem, kteří byli uvedeni v prvním vydání a sponzorovali ať už jej osobně nebo LDP jako celek. Byli to Linux Support Team, Erlangen, SRN; S.u.S.E. GmbH, Fuerth, SRN; Linux System Labs, Inc., Clinton Twp., USA a RedHat Software, North Carolina, USA.

Terry děkuje své manželce Maggie, která jej podporovala při práci na této knize i navzdory nárokům jejich čerstvě narozeného prvorozeného syna Jacka. Chtěl by také poděkovat všem lidem z linuxové komunity, kteří mu různými způsoby pomohli dostat se na úroveň, kdy sám může pomáhat jiným. „Pomůžu ti pokud slíbíš, že příště pomůžeš ty někomu jinému."

Síň slávy

Kromě už uvedených osob se na příručce administrátora podílela i celá řada dalších lidí tím, že ji četli a poslali nám různé opravy a doporučení. Všem děkujeme.

Následující seznam uvádí jména těch, jejichž příspěvky dorazily do našich e-mailových schránek: Al Longyear, Alan Cox, Andres Sepúlveda, Ben Cooper, Cameron Spitzer, Colin McCormack, D.J. Roberts, Emilio Lopes, Fred N. van Kempen, Gert Doering, Greg Hankins, Heiko Eissfeldt, J.P. Szikora, Johannes Stille, Karl Eichwalder, Les Johnson, Ludger Kunz, Marc van Diest, Michael K. Johnson, Michael Nebel, Michael Wing, Mitch D'Souza, Paul Gortmaker, Peter Brouwer, Peter Eriksson, Phil Hughes, Raul Deluth Miller, Rich Braun, Rick Sladkey, Ronald Aarts, Swen Thüemmler, Terry Dawson, Thomas Quinot a Yury Shevchuk.

KAPITOLA 1

Úvod do sítí

Historie

Myšlenka vytváření sítí je pravděpodobně stejně stará jako vlastní telekomunikace. Představte si lidi žijící v době kamenné, kteří si možná mezi sebou předávali zprávy pomocí bubnů. Dejme tomu, že jeskynní člověk A chtěl pozvat jeskynního člověka B na zábavnou hru, která spočívala v tom, že po sobě házeli kamením, ale žil příliš daleko na to, aby B slyšel jeho buben. Jaké měl tedy A volby? Mohl 1) dojít za B, 2) sehnat si větší buben nebo 3) požádat C, který žil v polovině vzdálenosti jejich obydlí, aby zprávu předal. V případě poslední možnosti se jedná o vytvoření sítě.

Samozřejmě, že od primitivních snah a prostředků našich předků jsme urazili již dlouhou cestu. Když si dnes chceme domluvit schůzku na sobotní fotbalový zápas[5], máme počítače, které spolu komunikují prostřednictvím soustavy drátů, optických vláken, mikrovln a podobně. V následujícím textu se budeme zabývat způsoby a metodami, kterými je to realizováno, opustíme však záležitosti kolem drátů i kolem fotbalu.

V tomto průvodci si popíšeme tři typy sítí: Nejpodrobněji budeme mluvit o sítích TCP/IP, které jsou nejpopulárnější jak pro lokální sítě (LAN) tak i pro rozsáhlé sítě (WAN), jako je Internet. Kromě toho budeme hovořit o protokolech UUCP a IPX. UUCP se používal pro přenos pošty a zpráv prostřednictvím telefonních spojení. Protokol IPX se nejčastěji používá v prostředí Novell NetWare a my si řekneme, jak linuxový počítač připojit do sítě Novell. Ve všech případech jde o síťové protokoly, které slouží k přenosu dat mezi hostitelskými počítači. Popíšeme si, jak se používají a na jakých principech jsou založeny.

Síť definujeme jako soubor *hostitelů*, kteří spolu mohou vzájemně komunikovat a často se přitom spoléhají na služby vyhrazených hostitelů, přenášejících data mezi jednotlivými účastníky. Hostitelé jsou často počítače, ale neplatí to vždy; někdy tak nazýváme i X-terminály nebo inteligentní tiskárny. Menším seskupením hostitelů říkáme *místa (site)*.

Komunikace by nebyla možná bez určitého druhu jazyka nebo kódu. V počítačových sítích se těmto jazykům souhrnně říká *protokoly*. Neměli byste si však představovat písemné protokoly, ale spíše vysoce formalizovaný kód chování, které lze pozorovat například při setkání hlav států. Podobně protokoly používané v počítačových sítích nejsou ničím jiným, než velmi přísnými pravidly pro výměnu zpráv mezi dvěma nebo více hostiteli.

5 Který se ještě ve své původní podobě hrává v Evropě.

Sítě TCP/IP

Moderní síťové aplikace vyžadují sofistikovaný způsob přenosu dat z jednoho počítače na jiný. Pokud spravujete linuxový stroj, který používá řada uživatelů, přičemž všichni současně mohou chtít připojit se ke vzdálenému počítači na jiné síti, musíte mít způsob, jakým budou moci sdílet vaše síťové připojení, aniž by se vzájemně ovlivňovali. Řešením používaným v řadě moderních síťových protokolů je *přepínání paketů* (packet-switching). *Paket* je malý blok dat, který je přenášen z jednoho počítače na jiný prostřednictvím sítě. K přepínání dochází, když datagram[6] přechází mezi různými linkami v síti. Síť s přepínáním paketů používá jedinou síťovou linku pro mnoho uživatelů, kteří mohou v různých okamžicích posílat pakety od jednoho uživatele k jinému.

Řešení realizované v systému Unix a následně i v celé řadě jiných systémů je nazýváno TCP/IP. Když se mluví o sítích TCP/IP, často narazíte na termín *datagram*, který má svůj specifický technický význam, nicméně často se používá jako ekvivalent slova *paket*. V této části budeme hovořit o základech, na nichž protokoly TCP/IP stojí.

Úvod do sítí TCP/IP

Protokol TCP/IP byl vyvinut v rámci výzkumného projektu financovaného americkou společností DARPA (Defense Advanced Research Projects Agency) v roce 1969. Jednalo se o experimentální síť zvanou ARPANET, která byla uvedena do operačního provozu v roce 1975, poté co se ukázalo, že jde o úspěšné řešení.

V roce 1983 byla standardizována sada protokolů TCP/IP, kterou museli používat všichni hostitelé v této síti. Když se ze sítě ARPANET nakonec vyvinul Internet (přičemž síť ARPANET sama o sobě zanikla v roce 1990), rozšířilo se používání protokolu TCP/IP i v sítích mimo vlastní Internet. Řada společností má vlastní firemní sítě založené na protokolu TCP/IP a Internet se dostal do pozice, kdy může být klidně považován za obyčejnou konzumní technologii. Těžko dneska přečtete noviny nebo časopis, aby v nich nebyla zmínka o Internetu, který dnes může používat prakticky každý.

Abychom si ukázali konkrétnější využití protokolu TCP/IP, který budeme probírat v následujících statích, vezmeme si za příklad univerzitu GMU (Groucho Marx University), která se nachází někde v Fredlandu. Většina kateder zde má vlastní lokální sítě, některé sdílejí jedinou a jiné jich zase mají více. Všechny jsou vzájemně propojeny a k Internetu jsou připojeny jednou vysokorychlostní linkou.

Dejme tomu, že váš stroj pracující pod Linuxem je připojen k lokální síti unixových hostitelů na katedře matematiky a jmenuje se **erdos**. Budete-li se chtít připojit k hostiteli na katedře fyziky, který se jmenuje řekněme **quark**, zadáte následující příkaz:

```
$ rlogin quark.physics
Welcome to the Physics Department at GMU
(ttyq2) login:
```

Na výzvu zadáte vaše uživatelské jméno, třeba **andres,** a vaše heslo. Pak se vám spustí příkazový interpret[7] na hostiteli **quark**, se kterým můžete pracovat stejným způsobem, jako byste seděli u konzoly tohoto systému. Až tento interpret opustíte, vrátíte se do prostředí vašeho vlastního počítače. Právě jste použili jednu z okamžitých interaktivních aplikací, které poskytuje protokol TCP/IP: vzdálené přihlášení.

---

6 Pozn. překladatele: *Datagram* zde můžeme chápat jako jiný termín pro paket. Drobné a víceméně formální rozdíly v tom, co je co, nás nemusejí trápit.

7 Interpret nebo *shell* je řádkové rozhraní k operačnímu systému Unix. Podobá se příkazovému řádku DOSu v prostředí Microsoft Windows, je však mnohem výkonnější.

Když jste připojeni ke stroji **quark**, budete asi chtít spouštět také nějaké aplikace s grafickým uživatelským rozhraním, například textový editor, kreslící program nebo třeba prohlížeč WWW. Systém X Windows představuje plně síťově orientované grafické uživatelské prostředí, a je k dispozici pro řadu různých počítačových systémů. Aby příslušná aplikace věděla, že chcete mít její okna zobrazena na obrazovce vašeho hostitele, je třeba nastavit proměnnou prostředí `DISPLAY`:

```
$ DISPLAY=erdos.maths:0.0
$ export DISPLAY
```

Když nyní spustíte požadovanou aplikaci, spojí se s X-serverem na vašem stroji a ne na stroji **quark** a zobrazí všechna svá okna na vaší obrazovce. K tomu je samozřejmě nutné, aby na počítači **erdos** běžel systém X11. Pro nás je teď zajímavé, že protokol TCP/IP umožňuje hostitelům **quark** a **erdos** vzájemnou výměnu paketů, což budí dojem, že pracujete v jednom systému. Síť je zde takřka transparentní.

Další velice důležitou vlastností sítí TCP/IP je souborový systém NFS (Network File System). Také on přispívá k transparentnosti sítě, protože umožňuje připojovat adresářové struktury z jiných hostitelů, jako kdyby to byly lokální souborové systémy, takže pak vypadají jako normální adresáře na vašem počítači. Například všechny domovské adresáře uživatelů mohou být na centrálním serveru, ze kterého si je připojují všichni ostatní hostitelé v lokální síti. V důsledku toho se uživatelé mohou přihlásit na libovolný stroj a získají vždy stejný domovský adresář. Podobně je možné sdílet velké objemy dat (například databáze, dokumentaci nebo aplikační programy) mezi mnoha počítači tak, že jediná kopie dat bude udržována na serveru a ostatní počítače k ní budou mít přístup. K souborovému systému NFS se ještě vrátíme v kapitole 14.

Samozřejmě, že to jsou jen příklady využití sítí založených na protokolu TCP/IP. Možnosti jsou takřka neomezené.

Nyní se podrobněji podíváme na způsob, jakým funguje protokol TCP/IP. Jeho znalost vám pomůže lépe pochopit jak a proč máte nakonfigurovat váš počítač. Začneme hardwarem a pomalu budeme postupovat směrem vzhůru.

Ethernety

Typu hardwaru, který se nejčastěji používá v lokálních sítích, říkáme *Ethernet*. V nejjednodušším případě je tvořen jedním kabelem, ke kterému jsou jednotliví hostitelé připojeni prostřednictvím konektorů, odboček nebo transceiverů. Instalace jednoduchého Ethernetu není příliš drahá, což je společně s jejich přenosovou rychlostí 10, 100 nebo i 1000 megabitů za sekundu důvodem jejich oblíbenosti.

Ethernety existují ve třech provedeních, které nazýváme *tlustý*, *tenký* a *kroucená dvoulinka*. Tlustý a tenký Ethernet používají koaxiální kabel, který se liší šířkou a způsobem, jakým k němu lze hostitele připojit. Tenký Ethernet používá konektor „BNC“ ve tvaru písmene T, který vložíte do kabelu a zapojíte do zadní stěny vašeho počítače. Tlustý Ethernet vyžaduje vyvrtání malé díry do kabelu a připojení transceiveru za pomocí „vampire tap“. Tenký a tlustý Ethernet může být dlouhý maximálně 200 respektive 500 metrů, a proto se jim také říká 10Base-2 a 10Base-5. Termín „base“ je zkratka od „baseband modulation“ a jednoduše znamená, že data jsou posílána přímo na kabel bez použití modulace[8]. Číslo na začátku znamená přenosovou rychlost v megabitech za sekundu a číslo na konci je maximální povolená délka kabelu ve stovkách metrů.

8 Pozn. překladatele: Kromě „base“ přenosů se používají ještě „broad“ přenosy – od slovíčka broadband, při nich jsou data při odesílání modulována na nějakou nosnou frekvenci a při příjmu se musí demodulovat. Příkladem této technologie je 10Broad36, Ethernetová kabeláž vedená 75Ω koaxiálním kabelem (tedy klasickým „televizním kabelem“), používaná zejména v technologických systémech.

Kroucená dvoulinka je kabel tvořený dvěma dvojicemi měděných vodičů a obvykle vyžaduje použití specializovaného hardware – *aktivního rozbočovače*. Kroucená dvoulinka se označuje také jako 10BaseT, kde „T" znamená „twisted pair". 100megabitová verze se pak označuje jako 100BaseT.

Pro připojení nového počítače k tenkému Ethernetu je třeba alespoň na několik minut přerušit síťové spojení, protože musíte kabel přeříznout, abyste do něho mohli vložit konektor. Připojení počítače k tlustému Ethernetu je poněkud komplikovanější, typicky však nemusíte přerušit síť. Síť založená na kroucené dvoulince to má ještě jednodušší. Používá zařízení označované jako rozbočovač (hub), který slouží jako stykový uzel. Počítače můžete k rozbočovači připojovat a odpojovat, aniž byste jakkoliv ovlivnili ostatní.

Většina lidí upřednostňuje pro malé sítě tenký Ethernet, protože je velice levný: síťové karty seženete už za 500 korun (některé společnosti je dnes prakticky vyhazují) a kabel stojí jen několik korun za metr. Nicméně pro rozsáhlejší instalace je vhodnější tlustý Ethernet nebo kroucená dvoulinka. Ethernet na katedře matematiky univerzity GMU byl nejprve řešen tlustým koaxiálem, protože se jednalo o rozvod na poměrně velkou vzdálenost a nebylo nutné přerušovat síť při každém přidávání nového počítače. Ve většině instalací je dnes nejpopulárnější kroucená dvoulinka. Ceny rozbočovačů klesají a menší zařízení je dnes možné koupit za cenu přijatelnou i pro domácí uživatele. Instalace kroucenou dvoulinkou je pro rozsáhlejší sítě výrazně levnější a se samotným kabelem se pracuje mnohem snáze než s koaxiálními kabely v jiných typech instalací. Správci sítě na katedře matematiky GMU tak v příštím roce plánují přechod na kroucenou dvoulinku, protože se jedná o moderní technologii, která jim ulehčí práci při přidávání nových počítačů a odpojování starých.

Jednou z nevýhod ethernetové technologie je omezená délka kabelu, což vylučuje jeho použití pro jiné než lokální sítě. Nicméně pomocí opakovačů, mostů a směrovačů může být pospojováno několik ethernetových segmentů. Opakovače prostě jen kopírují signály mezi dvěma nebo více segmenty, takže všechny segmenty dohromady působí dojmem, jako by se jednalo o jediný segment. Vzhledem k požadavkům na časování nemůže mezi libovolnými dvěma počítači v síti ležet více než čtyři opakovače. Mosty a směrovače jsou chytřejší. Analyzují příchozí data a předávají je dále pouze v případě, kdy příjemce není připojen na stejném segmentu jako odesílatel.

Ethernet funguje jako sběrnice, po které může hostitel posílat pakety (nebo *rámce*) až do velikosti 1500 bajtů jinému hostiteli ve stejném Ethernetu. Adresa hostitele je dlouhá šest bajtů a je napevno zakódována do firmwaru jeho ethernetové desky. Takovéto adresy jsou zpravidla zapisovány ve tvaru dvojic hexadecimálních číslic oddělených dvojtečkami, například: **aa:bb:cc:dd:ee:ff**.

Rámec poslaný jednou stanicí vidí všechny připojené stanice, ale pouze cílová stanice si ho vyzvedne a zpracuje. Pokud se ve stejnou dobu pokusí vysílat dvě stanice, dojde ke *kolizi*. Elektronika síťových karet kolizi velmi rychle detekuje a řeší ji tak, že obě stanice vysílání přeruší a po náhodně zvolené době to zkusí znovu. Určitě uslyšíte hodně o tom, jak problematické jsou kolize na Ethernetu a že díky nim je využitelnost Ethernetu pouhých 30 % jeho přenosové kapacity. Kolize jsou na Ethernetu *normální* jev a na velmi zatížených sítích vás nemůže překvapit, že kolize představují až 30 % přenosů. Realističtější odhad využitelnosti přenosové kapacity je 60 % a pokud vám to stačí, nemusí vás kolize trápit[9].

9 O této problematice hovoří Ethernet FAQ na adrese *http://www.faqs.org/faqs/LANs/ethernet-faq/*, podrobné historické a technické informace naleznete na stránkách Charlese Spurgeona věnovaných Ethernetu na adrese *http://wwwhost.ots.utexas.edu/ethernet/*.

Další typy hardwaru

U větších instalací, jako je například Groucho Marx University, se zpravidla kromě Ethernetu používá ještě jiný typ vybavení. Existuje a používá se celá řada dalších komunikačních technologií.Všechny popsané protokoly jsou Linuxem podporovány, nicméně vzhledem k omezení délky tohoto textu je popíšeme jen velmi stručně. Pro řadu různých protokolů existují dokumenty HOWTO, které je popisují podrobně, takže pokud vás zajímá něco, o čem tady nebudeme hovořit, můžete vyzkoušet tyto dokumenty.

Na univerzitě GMU jsou všechny lokální sítě jednotlivých kateder připojeny k univerzitní páteři, což je optický kabel s rozhraním FDDI (*Fiber Distributed Data Interface*). Toto rozhraní používá při přenosu dat úplně jiný přístup, který spočívá v rozeslání několika *tokenů* a stanice může vysílat data pouze tehdy, pokud vlastní nějaký token. Hlavní výhodou technologie s předáváním tokenů je to, že nedochází ke kolizím. Protokol tak může mnohem snáze využít celou kapacitu přenosového média, která je v případě FDDI 100 megabitů za sekundu. Technologie FDDI založená na optickém kabelu může mít maximální délku kabelu mnohem větší než u klasických „drátěných" technologií. Délkový limit je 200 km, což je ideální při propojování více budov ve městě, nebo jako v případě GMU, k propojení mnoha budov univerzitního areálu.

Podobně pokud se používá vybavení společnosti IBM, velmi pravděpodobně bude instalována síť IBM Token Ring. Token Ring se v některých lokálních sítích používá místo Ethernetu a nabízí podobné výhody jako FDDI, protože plně využívá přenosovou kapacitu média (která je ovšem v tomto případě nižší, 4 Mb/s nebo 16 Mb/s), je ovšem levnější, protože místo optického kabelu používá klasický kabel. V Linuxu se sítě Token Ring konfigurují prakticky stejně jako Ethernet, proto se jim nebudeme specificky věnovat.

I když dnes už to je méně pravděpodobné než dříve, můžete potkat i jiné technologie lokálních sítí, například ArcNet nebo DECNet. Obě dvě Linux podporuje, nebudeme se však jimi zabývat.

Mnoho národních sítí provozovaných telekomunikačními operátory podporuje protokoly s přepínáním paketů. Pravděpodobně nejpopulárnější je standard nazvaný X.25. Tuto službu nabízí mnoho takzvaných veřejných datových sítí, jakou je například Tymnet v USA, Austpac v Austrálii nebo Datex-P v Německu. Standard X.25 definuje sadu protokolů, kterými komunikují terminálová zařízení (například počítače) s komunikačními zařízeními (přepínač X.25). X.25 vyžaduje synchronní datovou linku, a tedy i speciální synchronní sériové komunikační zařízení. Linku X.25 můžete používat i na normálním sériovém portu, pokud použijete speciální zařízení označované jako PAD (Packet Assembler Disassembler). PAD je externí zařízení, které obsahuje asynchronní sériový port a synchronní sériový port. Obstarává komunikaci protokolem X.25, takže se k němu mohou připojit i velmi jednoduchá terminálová zařízení. Protokol X.25 se často používá k přenosu jiných síťových protokolů, například TCP/IP. Protože IP pakety nelze jednoduše namapovat na pakety X.25 (nebo naopak), jsou IP pakety před odesláním jednoduše zapouzdřeny do paketů X.25 a pak odeslány po síti. Pro Linux je dostupná experimentální implementace protokolu X.25.

Novější protokol často nabízený telekomunikačními společnostmi se označuje *Frame Relay*. Protokol Frame Relay má s protokolem X.25 společnou řadu technických podrobností, jinak se ale více podobá protokolu IP. Stejně jako X.25 potřebuje Frame Relay speciální synchronní sériové zařízení. Vzhledem k této podobnosti podporuje řada karet oba dva protokoly. Alternativou nevyžadující žádný speciální interní hardware je opět použití externího zařízení označovaného jako Frame Relay Access Device (FRAD), které zapouzdřuje Ethernetové pakety do paketů Frame Relay a posílá je po síti. Frame Relay je ideální pro přenos TCP/IP mezi různými lokalitami. Linux obsahuje ovladače, které podporují některé typy interních Frame Relay karet.

Pokud potřebujete vysoce rychlý spoj, který bude přenášet různé typy dat, například digitalizovaný obraz nebo zvuk i normální data, bude vás zřejmě zajímat technologie ATM (Asynchronous Transfer Mode). ATM je nová síťová technologie navržená speciálně k poskytování dobře spravovatelných rychlých linek s malým zpožděním, které umožňují řízení kvality služby. Řada telekomunikačních společností buduje infrastrukturu linek ATM, protože jim umožní přenesení různých síťových služeb na jednu platformu a ušetří tak náklady na správu a údržbu. ATM se velmi často používá k přenosu protokolu TCP/IP. Podpoře ATM v Linuxu se věnuje dokument Networking-HOWTO.

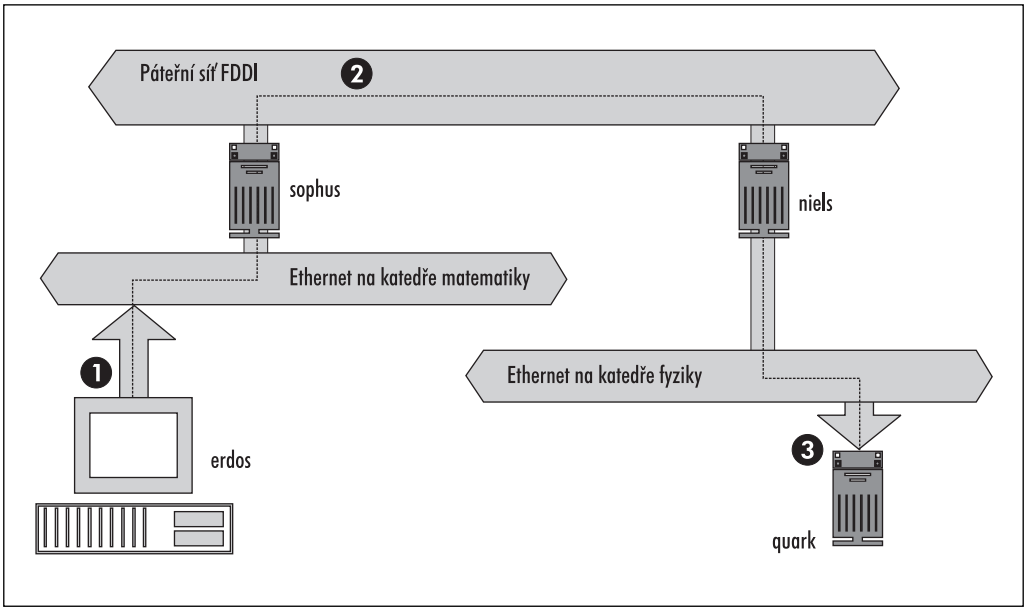
Radioamatéři často používají své vybavení k vzájemnému propojení svých počítačů; tomu říkáme *paketové rádio*. Jeden z protokolů používaný paketovými rádii se jmenuje AX.25, který je volným derivátem protokolu X.25. Protokol AX.25 se pak používá k přenosu protokolu TCP/IP i jiných. AX.25 stejně jako X.25 vyžaduje ke své práci synchronně pracující sériový hardware nebo externí zařízení označované jako Terminal Node Controller, který konvertuje pakety vysílané asynchronní linkou na linku synchronní. Existuje celá řada různých karet podporujících paketové rádio, obecně se označují jako „Z8530 SCC based“, což je název nejpopulárnějšího řadiče, od nějž byly odvozeny. Protokolem AX.25 se často přenášejí ještě protokoly NetRom a Rose, což jsou protokoly síťové vrstvy. Protože využívají protokolu AX.25, mají stejné požadavky na hardware. Linux obsahuje úplnou podporu protokolů AX.25, NetRom a Rose. Dobrým zdrojem informací o implementaci těchto protokolů je dokument AX.25-HOWTO.

Další typ spojení vyžaduje připojení k centrálnímu systému pomalou, nicméně levnou sériovou linkou (telefonní, ISDN a podobně). To vyžaduje další protokol pro přenos paketů, například SLIP nebo PPP, které si popíšeme dále.

Internet Protocol

Samozřejmě nechcete, aby byla vaše síť omezena jen na Ethernet nebo na dvoubodovou datovou linku. V ideálním případě budete chtít komunikovat s hostitelským počítačem bez ohledu na to, k jaké fyzické síti je připojen. Ve větších instalacích, jakou mají například na Groucho Marx University, existuje zpravidla několik oddělených sítí, které jsou nějakým způsobem propojeny. Na GMU fungují na katedře matematiky dva Ethernety: první je tvořen rychlými počítači, které využívají profesoři a absolventi, a druhý spojuje pomalejší počítače, na kterých pracují studenti. Obě sítě jsou připojeny k univerzitní páteřní síti FDDI.

Toto připojení zajišťuje vyhrazený hostitel, takzvaná *brána*, která obsluhuje příchozí a odchozí pakety tím způsobem, že je kopíruje mezi dvěma Ethernety a optickým kabelem sítě FDDI. Sedíte-li například na katedře matematiky a chcete se z vašeho linuxového stroje připojit k počítači **quark**, který je připojen do lokální sítě na katedře fyziky, nemůže síťový software poslat pakety přímo hostiteli **quark** na katedře fyziky, protože není připojen do stejného Ethernetu. Musí se spolehnout na bránu, která funguje jako dopravce. Brána (její jméno je **sophus**) pak za pomoci páteřní sítě tyto pakety předá partnerské bráně **niels** na katedře fyziky, která je doručí cílovému počítači. Na obrázku 1.1 je znázorněn tok dat mezi počítači **erdos** a **quark**.



Obrázek 1.1 – Tři kroky při posílání datagramu z počítače erdos počítači quark

Tomuto schématu doručování dat ke vzdálenému hostiteli se říká *směrování* a paketům v tomto kontextu říkáme *datagramy*. Aby to bylo jednodušší, je výměna datagramů řízena protokolem, který je nezávislý na použitém hardwaru: protokolem IP, *Internetovým protokolem*. Tímto protokolem a otázkami týkajícími se směrování se budeme podrobněji zabývat v kapitole 2.

Hlavní užitek protokolu IP spočívá v tom, že spojuje fyzicky rozdílné sítě do jedné na pohled homogenní sítě. Tomu říkáme internetworking (spojování více počítačových sítí do jedné) a výsledné „metasíti" říkáme *internet*. Všimněte si zde jemného rozdílu mezi slovy *internet* a *Internet*. Druhé z nich je název konkrétního globálního internetu.

Samozřejmě, že protokol IP vyžaduje adresové schéma, které je nezávislé na hardwaru. Toho dosáhneme tím, že každému hostiteli přidělíme jedinečné 32bitové číslo nazývané *IP adresa*. IP adresa se zpravidla zapisuje jako čtyři desítková čísla (každé z nich udává jednu 8bitovou část) oddělená tečkami. Například počítač **quark** by mohl mít IP adresu **0x954C0C04**, což bychom zapsali jako **149.76.12.4**. Tomuto formátu zápisu také říkáme *tečková notace*. Velmi často se označuje také jako IPv4 (jako Internet Protocol, Version 4), protože se připravuje nová verze IPv6, která nabízí podstatně pružnější možnosti adresování plus řadu dalších funkcí. Nicméně bude trvat ještě alespoň rok od vydání této knihy, než se protokol IPv6 dostane k normálnímu použití.

Všimněte si, že nyní tady máme tři různé typy adres: nejprve je zde název hostitele, **quark**, potom jeho IP adresa a konečně hardwarová adresa typu 6bajtové ethernetové adresy. Všechny tyto adresy si musí určitým způsobem odpovídat, takže když zadáte **rlogin quark**, může být síťovému softwaru předána IP adresa hostitele **quark** a když protokol IP doručí data do Ethernetu katedry fyziky, musí nějakým způsobem zjistit, která ethernetová adresa této IP adrese odpovídá.

K tomuto tématu se vrátíme v kapitole 2. Nyní nám bude stačit, když si zapamatujeme, že tomuto procesu získávání adresy mapováním názvu hostitele na IP adresu říkáme *rozlišování názvu hostitele* (*hostname resolution*) a mapování na hardwarovou adresu říkáme *rozlišování adresy* (*address resolution*).

Serial Line Internet Protocol

Na sériových linkách je de facto standardem protokol SLIP, neboli *Serial Line IP*. Modifikovaný typ protokolu SLIP se nazývá CSLIP, neboli *compressed SLIP*, a provádí kompresi IP hlaviček, aby bylo možné lépe využít relativně malou šířku pásma, kterou poskytují sériové linky. Dalším sériovým protokolem je PPP, neboli *Point-to-Point Protocol*. Jedná se o modernější verzi protokolu SLIP a obsahuje řadu funkcí, které jej dělají zajímavým. Jeho hlavní výhodou oproti protokolu SLIP je to, že může přenášet nejen datagramy protokolu IP, ale prakticky jakýkoliv protokol.

Transmission Control Protocol

Samozřejmě, že odesláním datagramů z jednoho hostitele na druhý celý proces nekončí. Přihlásíte-li se k počítači **quark**, chcete, aby bylo spojení mezi vaším procesem *rlogin* na hostiteli **erdos** a příkazovým interpretem na hostiteli **quark** spolehlivé. Proto musí odesílatel informaci posílanou z jednoho počítače na druhý rozdělit na pakety a příjemce pak z nich musí znovu sestavit proud znaků. Vypadá to sice jednoduše, ale celý tento proces v sobě zahrnuje několik náročných úkolů.

Předně je důležité vědět, že cílem protokolu IP nikdy nebylo zajistit spolehlivost. Představte si, že by deset lidí připojených k vaší ethernetové síti začalo stahovat zdrojový kód poslední verze prohlížeče Netscape z FTP serveru GMU. Síťový provoz, který by tyto operace vyvolaly, by brána nemusela zvládnout, protože je příliš pomalá a má nedostatek paměti. Když teď pošlete z počítače **quark** nějaký paket, nemusí mít brána **sophus** dostatek místa ve vyrovnávací paměti a tím pádem ho nebude moci předat dál. Protokol IP řeší tento problém tak, že příslušný paket zruší. Paket je tak neodvolatelně ztracen. Proto závisí na komunikujících hostitelích, aby kontrolovali integritu a úplnost dat a v případě výskytu chyby přenos opakovali.

K tomu se používá další protokol zvaný TCP, neboli *Transmission Control Protocol*, který vytváří spolehlivou službu nad protokolem IP. Základní vlastností protokolu TCP je, že za pomoci protokolu IP vytváří iluzi jediného spojení mezi příslušnými dvěma procesy na vašem hostiteli a vzdáleném počítači, takže se nemusíte starat jakým způsobem a po jaké trase vaše data ve skutečnosti putují. TCP spojení v podstatě funguje jako dvousměrná roura, do které mohou oba procesy zapisovat i z ní číst. Lze to přirovnat k telefonnímu hovoru.

Protokol TCP rozpozná koncové body takového spojení podle IP adresy příslušných dvou hostitelů a čísla takzvaného *portu* na každém z hostitelů. Na porty je možné se dívat jako na přípojky síťových spojení. Máme-li znovu použít podobnost s telefonním spojením, lze IP adresu přirovnat ke směrovému číslu (směrují čísla na určité město) a čísla portů pak k místním telefonním číslům (směrují čísla na jednotlivé telefony). Jeden hostitel může poskytovat mnoho různých služeb, které jsou od sebe odlišeny čísly portů.

V našem příkladu otevře klientská aplikace (**rlogin**) port na hostiteli **erdos** a spojí se s portem 513 na hostiteli **quark**, o kterém se ví, že na něm poslouchá server **rlogind**. Tím je navázáno TCP spojení. Za pomoci tohoto spojení provede server **rlogind** autorizaci uživatele a potom spustí příkazový interpret. Standardní vstup a výstup tohoto příkazového interpretu jsou přesměrovány na TCP spojení, takže cokoliv napíšete na vašem počítači, bude proudem TCP přeneseno a předáno na standardní vstup příkazového interpretu.

User Datagram Protocol

Samozřejmě, že protokol TCP není jediným uživatelským protokolem v sítích TCP/IP. I když je vhodný pro aplikace typu **rlogin**, jeho vysoká režie jej neumožňuje používat pro aplikace typu NFS. Místo toho se používá spřízněný protokol UDP, neboli *User Datagram Protocol*. Podobně jako TCP i protokol UDP dovoluje aplikaci kontaktovat službu na určitém portu na vzdáleném po-

čítači, ale nenaváže s ní spojení. Místo toho lze pomocí tohoto protokolu poslat cílové službě samostatné pakety – odtud je odvozen název tohoto protokolu.

Předpokládejme, že chcete přečíst nějaký malý objem dat z databázového serveru. Potřebujete minimálně tři datagramy k navázání spojení protokolem TCP, další tři k odeslání a potvrzení trochy dat a ještě tři k ukončení spojení. S protokolem UDP dosáhneme prakticky stejného efektu s pouhými dvěma datagramy. UDP je protokol bez navazování spojení, takže po nás nevyžaduje otevírat a zavírat relaci. Prostě zabalíme data do datagramu a pošleme je na server. Server vytvoří odpověď, rovněž ji zabalí do datagramu a pošle nám ji zpět. I když je to jednodušší a efektivnější než protokol TCP pro přenosy malých objemů dat, protokol UDP se neumí vyrovnat se ztrátou datagramu. Je čistě otázkou aplikace, například name serveru, aby se s takovou situací uměl vyrovnat.

Více o portech

Na porty je možné nahlížet jako na přípojky síťových spojení. Pokud chce nějaká aplikace nabízet určitou službu, připojí sama sebe k určitému portu a čeká na klienty (říká se tomu také *odposlouchávání* portu). Klient, který chce tuto službu využívat, si alokuje nějaký port na svém hostiteli a připojí se k příslušnému portu serveru na vzdáleném hostiteli. Jeden a týž port může být otevřen na více počítačích, na jednom počítači však může mít jeden port otevřena jen jedna aplikace.

Důležitou vlastností portů je, že jakmile bylo navázáno spojení mezi klientem a serverem, může se k portu serveru připojit jiná instance serveru a odposlouchávat další klienty. To umožňuje například současné přihlášení ke stejnému hostiteli při využití stejného portu 513. Protokol TCP je schopen tato spojení rozlišit, protože všechny přicházejí z různých portů nebo hostitelů. Pokud se například dvakrát přihlásíte k hostiteli **quark** z počítače **erdos**, potom bude první klient **rlogin** využívat port 1023 a druhý klient port 1022. Oba však budou připojeni ke stejnému portu 513 na hostiteli **quark**.

Tento příklad ukazuje využití portů jako přístupových bodů, kdy se klient připojuje ke specifickému portu, aby získal určitou službu. Aby klient znal správné číslo portu, musí mezi administrátory obou systémů existovat určitá dohoda ohledně přiřazování těchto čísel. U služeb, které jsou široce používány, jako je například rlogin, musí být tato čísla spravována centrálně. O to se stará organizace IETF (*Internet Engineering Task Force*), která pravidelně vydává RFC nazvané *Přiřazená čísla* (*Assigned Numbers*, RFC-1700). Kromě jiného popisuje čísla portů přiřazená všeobecně známým službám. K mapování služeb na čísla portů používá Linux soubor nazvaný `/etc/services`.

Je třeba poznamenat, že i když spojení využívající protokoly TCP i UDP spoléhají na porty, nedochází mezi nimi ke konfliktům. To znamená, že se například TCP port 513 liší od UDP portu 513. Ve skutečnosti slouží tyto konkrétní porty jako vstupní body pro dvě různé služby, jmenovitě **rlogin** (TCP) a **rwho** (UDP).

Knihovna soketů

V operačních systémech Unix je software, který obsluhuje veškeré výše popsané úkoly a protokoly, většinou součástí jádra, což platí i pro Linux. Ve světě Unixu je nejběžnějším programovým rozhraním knihovna *Berkeley Socket Library*. Její název je odvozen od oblíbené analogie, která nazírá na porty jako na zásuvky (socket) a připojování k portu chápe jako zapojování. Poskytuje volání `bind`, kterým se specifikuje vzdálený hostitel, přenosový protokol a služba, ke které se program může připojit nebo ji poslouchat (pomocí volání `connect`, `listen` a `accept`). Knihovna soketů je však obecnější v tom, že poskytuje nejenom třídu soketů založených na protokolu TCP/IP (sokety typu *AF_INET*), ale také třídu, která obsluhuje spojení s lokálním počítačem (třída *AF_UNIX*). Ně-

které implementace mohou obsluhovat také jiné třídy, jako například protokol XNS (*Xerox Networking System*) nebo X.25.

V Linuxu je knihovna soketů součástí standardní knihovny jazyka C zvané *libc*. V současné době podporuje sokety typu `AF_INET` a `AF_INET6` pro protokoly TCP/IP, `AF_UNIX` pro lokální komunikaci a dále `AF_IPX` pro protokoly sítě Novell, `AF_X25` pro protokol X.25, `AF_ATMPVC` a `AF_ATMSVC` pro sítě ATM, a `AF_AX25`, `AF_NETROM` a `AF_ROSE` pro paketové rádio. Vyvíjí se i podpora dalších protokolů, která bude dostupná časem.

Sítě UUCP

UUCP je zkratka spojení Unix-to-Unix Copy. Původně to byl balík programů pro přenos souborů po sériových linkách, plánování těchto přenosů a spouštění programů na vzdálených hostitelích. Od své první implementace koncem sedmdesátých let prodělal tento protokol velké změny, ale co se týče služeb, které nabízí, je stále poněkud strohý. Hlavní využití má stále v sítích WAN založených na periodicky vytáčených telefonních linkách.

Protokol UUCP byl vytvořen v Bellových laboratořích v roce 1977, kde měl sloužit ke komunikaci mezi místy podílejícími se na vývoji Unixu. V polovině roku 1978 spojovala tato síť již přes 80 míst. Používala se zde elektronická pošta i vzdálený tisk. Ovšem hlavní využití systému spočívalo v šíření nového softwaru a opravách chyb. Dnes již není protokol UUCP omezen jen na prostředí Unixu. Existují jak volně šiřitelné, tak i komerční přenosy na různé platformy, včetně AmigaOS, DOS a Atari TOS.

Jednou z hlavních nevýhod sítě UUCP je její dávkový provoz. Nevytváří se zde trvalé spojení mezi počítači, používají se dočasná spojení. UUCP systém se může s druhým UUCP systémem spojovat třeba jen jednou denně a to pouze na krátkou dobu. Po dobu trvání připojení se přenese veškerá pošta, zprávy a soubory, které byly zařazeny ve frontě a pak se spojení ukončí. Právě toto řazení do fronty představuje omezení, pro které aplikace je UUCP použitelné. V případě elektronické pošty může uživatel napsat zprávu a „odeslat“ ji. Zpráva nicméně čeká ve frontě do doby, než se počítač připojí k druhému UUCP počítači a zprávu mu předá. Je to použitelné pro služby jako je elektronická pošta, nehodí se to ale pro služby, jako je například **rlogin**.

Navzdory těmto omezením stále funguje po celém světě spousta sítí UUCP, o něž pečují zejména nadšenci, kteří za rozumné ceny nabízí soukromým uživatelům přístup k síti. Hlavním důvodem oblíbenosti protokolu UUCP je skutečnost, že ve srovnání s připojením počítače pevnou linkou je téměř „za babku“. K tomu, abyste z vašeho počítače udělali uzel UUCP, potřebujete jen modem, fungující implementaci protokolu UUCP a jiný uzel UUCP, který bude ochotný vás zásobovat poštou a zprávami. Řada různých společností je ochotná zprostředkovat UUCP připojování soukromým uživatelům, protože to pro ně nepředstavuje žádné enormní zatížení jejich sítě.

Konfiguraci protokolu UUCP probereme v samostatné kapitole později, nebudeme se jí však věnovat příliš podrobně, protože UUCP je stále více vytlačován protokolem TCP/IP, protože levný přístup k Internetu je dnes k dispozici prakticky všude.

Sítě v Linuxu

Jakožto výsledek soustředěného úsilí programátorů z celého světa by Linux nemohl existovat bez globální počítačové sítě. Takže nikoho nepřekvapí, že již v raných stádiích vývoje začalo několik lidí pracovat na síťových schopnostech tohoto operačního systému. Implementace protokolu UUCP běžela v Linuxu již od samého začátku a práce na začlenění protokolu TCP/IP začala na podzim roku 1992, když Ross Biro a další vytvořili projekt, který získal označení Net-1.

Poté co Ross v květnu 1993 opustil práci na vývoji tohoto protokolu, začal Fred van Kempen pracovat na nové implementaci, přičemž přepracoval hlavní části kódu. Výsledkem byla verze Net-2. První veřejná verze Net-2d byla hotová v létě 1992 (jako součást jádra verze 0.99.10) a od té doby byla udržována a rozšiřována různými lidmi, zejména Alanem Coxem[10], který stál u zrodu verze Net-2Debugged. Po důkladném otestování a mnoha vylepšeních kódu přišla po vydání Linuxu verze 1.0 změna názvu na Net-3. Kód Net-3 byl dále vyvíjen pro Linux 1.2 a Linux 2.0. Jádra verze 2.2 a vyšší používají knihovnu Net-4, která v současné době představuje standard.

Síťový kód Net-4 podporuje celou řadu zařízení a různých funkcí. Mezi standardní protokoly knihovny Net-4 patří SLIP a PPP (pro síťový provoz po sériových linkách), PLIP (pro paralelní linky), IPX (pro sítě kompatibilní s Novell NetWare, o této problematice budeme hovořit v kapitole 15), Appletalk (pro sítě počítačů Apple) a AX.25, NetRom a Rose (pro radiové sítě). Mezi další standardní funkce této knihovny patří IP firewally, IP účtování (budeme o nich hovořit v kapitole 9 a 10) a IP maškaráda (kapitola 11). Podporován je IP tuneling a různé techniky směrování. Je podporována celá řada ethernetových zařízení stejně jako některá FDDI, Token Ring, Frame Relay, ISDN a ATM zařízení.

Kromě toho je podporována celá řada dalších funkcí, které zvyšují univerzálnost Linuxu. Mezi tyto funkce patří implementace souborového systému SMB, který spolupracuje s aplikacemi jako jsou *lanmanager* a Microsoft Windows. Tuto implementaci, nazvanou Samba, napsal Andrew Tridgell. Dále Linux obsahuje implementaci novellového protokolu NCP[11].

Různé směry vývoje

V různých dobách probíhaly práce na vývoji různých síťových funkcí systému Linux.

Fred pokračoval ve vývoji knihovny Net2-Debugged poté, co byla prohlášena za oficiální síťovou implementaci. Vývoj vedl k verzi Net-2e, která měla přepracovanou strukturu síťové vrstvy. Největším přínosem Net-2e bylo rozhraní DDI (*Device Driver Interface*). V současné době už není knihovna Net-2e dále vyvíjena.

Další implementace protokolu TCP/IP pochází od Matthiase Urlichse, který napsal ovladač ISDN pro Linux a FreeBSD. Za tím účelem integroval do jádra Linuxu část síťového kódu BSD. Ani na tomto projektu se už dnes nepracuje.

V implementaci síťových funkcí v jádře Linuxu došlo k mnoha změnám, a protože vývoj stále probíhá, i nadále k nim dochází. Někdy to znamená, že změna se musí projevit i v jiných programech, například v nástrojích pro konfiguraci sítě. I když to dnes nepředstavuje takový problém jako v minulosti, i dnes se může stát, že pokud aktualizujete jádro, budete muset aktualizovat i nástroje pro konfiguraci síťových funkcí. Naštěstí vzhledem k velkému počtu dnes existujících distribucí je to poměrně snadná úloha.

Síťová implementace Net-4 je velmi vyzrálá a používá se na řadě míst po celém světě. Velké úsilí bylo věnováno vylepšení výkonu této implementace, takže dnes může být směle porovnána s nejvýkonnějšími implementacemi dostupnými pro srovnatelné platformy. Linux je velmi oblíben u poskytovatelů internetového přístupu a velmi často se v těchto firmách používá k vytvoření levných a spolehlivých webových serverů, poštovních serverů a serverů zpráv. O vývoj Linuxu je velký zájem, a tak probíhá paralelně se změnami síťových technologií, například současné verze jádra již podporují novou generaci protokolu IP, IPv6, tak, jak je navržen jako standard.

10 Alana můžete kontaktovat na adrese *alan@lxorguk.ukuu.org.uk*.

11 NCP je protokol, pomocí nějž komunikují souborové a tiskové servery v síti Novell.

Kde získáte kód

Dnes už zní velmi zvláštně, že v začátcích síťové implementace v Linuxu vyžadovalo standardní jádro k začlenění síťové podpory celou řadu různých patchů. V současné době probíhají práce na vývoji síťových funkcí současně s vývojem jádra. Poslední stabilní jádro lze získat na adrese *ftp.kernel.org* v adresáři `/pub/linux/kernel/v2.x`, kde *x* je sudé číslo. Poslední experimentální verzi naleznete na stejném místě, číslo této verze je ale liché. Na celém světě jsou zrcadla tohoto serveru. Jen velmi těžko si dnes někdo umí představit Linux bez integrované síťové podpory.

Údržba vašeho systému

V průběhu celé knihy se budeme zabývat především otázkami instalace a konfigurace. Správa však znamená mnohem víc – po zavedení služby je třeba ji také udržovat v provozu. Většina služeb vyžaduje jen nepatrnou údržbu, ale u některých z nich, jako je pošta a news, vyžaduje zachování aktuálnosti systému provádění rutinních úkolů. Tyto úkoly budeme probírat v dalších kapitolách.

Absolutní minimum údržby spočívá v pravidelné kontrole systémových a aplikačních logovacích souborů, zda neobsahují chybové stavy nebo neobvyklé události. Běžně se to dělá za pomoci dvojice administrativních skriptů, které se pravidelně spouštějí démonem **cron**. Tyto skripty obsahují distribuce zdrojových kódů některých hlavních aplikací, jako je například **inn** nebo C News. Je třeba si je jen upravit tak, aby vyhovovaly vašim potřebám a preferencím.

Výstup každé úlohy démona **cron** by měl být poslán poštou na účet **administrátora**. Mnoho aplikací implicitně posílá chybové zprávy, statistiky využití nebo souhrny logovacích souborů na účet **root**. To má smysl pouze tehdy, pokud se jako uživatel **root** přihlašujete často, ale výhodnější je za pomoci poštovního aliasu, který popisujeme v kapitolách 18 a 19, přesměrovat poštu pro uživatele **root** na váš osobní účet.

Bez ohledu na to, jak pečlivě váš systém nastavíte, v důsledku Murphyho zákonů se časem *zaručeně* vynoří nějaký problém. Proto znamená údržba systému i neustálou připravenost řešit stížnosti. Lidé zpravidla očekávají, že správce systému bude k zastižení přinejmenším prostřednictvím e-mailové adresy **root**, ale pro kontaktování lidí, kteří jsou odpovědní za určitý aspekt údržby, jsou běžně používány také jiné adresy. Například stížnosti na špatně fungující poštu budou adresovány uživateli **postmaster** a problémy se systémem news je možné oznamovat na adresách **newsmaster** nebo **usenet**. Pošta na účet **hostmaster** by měla být přesměrována člověku, který má na starosti základní síťové služby, a pokud provozujete name-server, tak také server služby DNS.

Bezpečnost systému

Dalším velice důležitým aspektem správy systému v síťovém prostředí je ochrana systému a uživatelů před vetřelci. Nedbale spravované systémy jsou vhodným cílem pro škodolibé uživatele, jejichž rozsah útoků je poměrně široký: od uhodnutí hesla až po sledování provozu na Ethernetu a způsobené škody mohou sahat od padělaných poštovních zpráv až po ztrátu dat nebo narušení soukromí vašich uživatelů. Když budeme hovořit o souvislostech, za kterých k nim může dojít, zmíníme se o některých konkrétních problémech a běžných ochranách proti nim.

Tato stať probírá některé příklady a základní postupy zajištění systémové bezpečnosti. Samozřejmě, že zde uvedená témata nemohou důkladně postihnout všechny bezpečností problémy, s nimiž se můžete setkat; slouží především k ilustraci problémů, které se mohou vyskytnout. Proto je nezbytně nutné přečíst si nějakou dobrou knihu pojednávající o bezpečnosti, zejména pak v systému propojeném do sítě.

Základem bezpečnosti systému je dobrá systémová administrace. Ta zahrnuje kontrolu vlastnictví a přístupových práv ke všem životně důležitým souborům a adresářům a sledování používání pri-

vilegovaných účtů. Program COPS například ověří, zda váš souborový systém a běžné konfigurační soubory neobsahují neobvyklá přístupová práva nebo jiné anomálie. Také je rozumné používat sadu programů pro správu hesel, které prosadí jistá pravidla pro zadávání hesel takovým způsobem, aby bylo těžké je uhodnout. Například sada shadow vyžaduje, aby bylo heslo dlouhé minimálně pět písmen a obsahovalo kromě malých a velkých písmen i jiné znaky.

Při zpřístupňování určité služby přes síť jí vždy přidělte nejnižší práva, což znamená, že jí nedovolíte provádět věci, které pro své fungování nepotřebuje. Programům byste měli například povolit přepnout se na účet **root** nebo některý jiný privilegovaný účet jen tehdy, pokud to je opravdu nezbytně nutné. V případě, že chcete používat některou službu jen omezeným způsobem, neváhejte ji nakonfigurovat s co největším omezením práv tak, jak to konkrétní situace dovolí. Chcete-li například povolit bezdiskovým počítačům zavádění operačního systému z vašeho počítače, musíte zprovoznit službu TFTP (*Trivial File Transfer Protocol*), aby tyto počítače mohly z adresáře `/boot` nahrát základní konfigurační soubory. Pokud byste ale používali službu TFTP bez omezení přístupu, umožnili byste komukoliv na světě nahrát si z vašeho systému libovolný soubor, který je veřejně čitelný. Proč tedy neomezit služby protokolu TFTP jen na adresář `/boot`[12]?

Možná také budete chtít omezit určité služby pouze na uživatele určitých počítačů, řekněme z vaší lokální sítě. V kapitole 12 si představíme nástroj **tcpd**, který to umožňuje pro různé síťové aplikace. O ještě pokročilejších metodách omezení přístupu na určité počítače nebo služby budeme hovořit také v kapitole 9.

Dále je nutné se vyhnout „nebezpečným“ aplikacím. Samozřejmě, že každý software, který používáte, může být svým způsobem nebezpečný, protože může obsahovat chyby, jichž mohou chytří lidé využít k získání přístupu k vašemu systému. K takovýmto věcem dochází, a není proti nim úplná ochrana. Tento problém se týká volně šiřitelného softwaru stejně jako komerčních produktů[13]. Nicméně programy, které vyžadují speciální přístupová práva, jsou logicky nebezpečnější než jiné, protože každá díra v nich může mít drastické důsledky[14]. Pokud nějaký síťový program instalujete jako setuid, dvakrát zkontrolujte dokumentaci, abyste na nic nezapomněli a náhodou tak nevytvořili bezpečnostní trhlinu.

Další oblast, kde byste měli být opatrní jsou programy, které umožňují přihlášení nebo provádění příkazů jen s omezenou autentikací. Programy **rlogin**, **rsh** a **rexec** jsou všechny velice užitečné, nicméně provádějí jen velmi jednoduchou autentikaci požadavku. Autentikace je založena na tom, že se důvěřuje jménu vzdáleného systému zjištěnému u name serveru (o name serverech budeme hovořit později). Tyto záznamy je však možné podvrhnout. V dnešní době by mělo být standardním postupem všechny **r** služby vypnout a nahradit je nástroji **ssh**. Tyto nástroje obsahují mnohem spolehlivější metody autentikace a nabízejí i další služby, jako je například šifrování a komprimace.

Nikdy nelze vyloučit, že vaše opatření selžou, bez ohledu na to, jak opatrní jste byli. Proto byste měli zajistit včasné odhalení vetřelců. Kontrola systémových logovacích souborů je dobrý začátek, ale vetřelec bude zřejmě natolik chytrý, že po sobě smaže všechny zřetelné stopy. Existují však nástroje jako například **tripwire**, napsaný Gene Kimem a Gene Spaffordem, které umožňují kontrolovat, zda nedošlo ke změně obsahu nebo přístupových práv životně důležitých systémových souborů. Program **tripwire** vypočítá pro tyto soubory několik silných kontrolních součtů a uloží je v databázi. Při dalším spuštění se kontrolní součty vypočítají znovu a kontroluje se, zda se oproti uloženým hodnotám nezměnily.

12 K tomuto tématu se vrátíme v kapitole 12.

13 Existovaly komerční unixové systémy (tedy systémy, za které se platí spousta peněz), které obsahovaly skript setuid-root, který umožňoval uživatelům získat práva uživatele **root** pomocí jednoduchého standardního triku.

14 V roce 1998 způsobil zablokování velké části Internetu červ RTM, který mimo jiné využíval chyby v různých programech včetně programu **sendmail**. Trvalo dlouho, než byly všechny tyto chyby odstraněny.

Problematika sítí TCP/IP

IP adresy

Již v předchozí kapitole jsme se zmínili o tom, že adresy, kterým rozumí síťový protokol IP, jsou 32bitová čísla. Každému počítači musí být v rámci celého síťového prostředí přiděleno jedinečné číslo[15]. Pokud provozujete lokální síť, která s ostatními sítěmi nekomunikuje na bázi protokolu TCP/IP, můžete tato čísla přidělit podle vlastního uvážení. Existují rozsahy IP adres, které jsou určeny právě pro přidělování na takovýchto privátních sítích. Tyto adresy jsou uvedeny v tabulce 2.1. Nicméně systémům připojeným k Internetu jsou čísla přidělována centrálně, konkrétně institucí *Network Information Center*, zkráceně NIC[16].

Tabulka 2.1 – Rozsahy IP adres rezervované pro privátní použití

Třída	Adresy
A	10.0.0.0 až 10.255.255.255
B	172.16.0.0 až 172.31.0.0
C	192.168.0.0 až 192.168.255.0

IP adresy jsou pro přehlednost rozděleny do čtyř 8bitových čísel, kterým říkáme *oktety*. Například počítač **quark.physics.groucho.edu** má IP adresu **0x954C0C04**, která je zapsána jako **149.76.12.4**. Tomuto formátu se také někdy říká *tečková notace*.

Dalším důvodem této symboliky je rozdělení IP adres na číslo *sítě*, které je obsaženo v levých oktetech, a na číslo *hostitele*, které je zapsáno v pravých oktetech. Když požádáte centrum NIC o přidělení IP adres, nebude vám přidělena adresa pro každého hostitele, kterého hodláte používat. Místo toho obdržíte jediné číslo sítě a pak na základě vlastního uvážení můžete přidělovat hostitelům ve vaší síti všechny IP adresy v rámci získaného rozsahu IP adres.

V závislosti na velikosti sítě je nutné mít i různě velkou část hostitele. Protože mají různí uživatelé různé nároky na velikost sítí, existuje několik tříd sítí, které definují různá rozdělení IP adres:

15 V současné době se na Internetu nejčastěji používá IP protokol verze 4. Velké úsilí bylo věnováno vytvoření náhrady, protokolu IP verze 6. IPv6 používá odlišné adresní schéma a delší adresy. Linux obsahuje implementaci protokolu IPv6, není však dosud ve stádiu, abychom ji zde mohli popsat. Samotná podpora protokolu IPv6 v jádře je v pořádku, nicméně řada aplikací musí být upravena, aby mohla s tímto protokolem pracovat. Zůstaňte s námi.

16 Typicky vám IP adresu přidělí poskytovatel internetového připojení, kterému za konektivitu k Internetu platíte. Můžete nicméně kontaktovat přímo NIC a vyžádat si své vlastní adresy prostřednictvím e-mailu na adresu *hostmaster@internic.net* nebo pomocí formuláře na adrese *http://www.internic.net*.

Třída A Třída A obsahuje sítě od **1.0.0.0** do **127.0.0.0**. Číslo sítě je obsaženo v prvním oktetu. Takto je k dispozici 24bitová část hostitele, která umožňuje až 1,6 milionů hostitelů.

Třída B Třída B obsahuje sítě od **128.0.0.0** do **191.255.0.0**; číslo sítě je obsaženo v prvních dvou oktetech. To umožňuje 16320 sítí, z nichž každá může mít až 65024 hostitelů.

Třída C Třída C zahrnuje sítě od **192.0.0.0** do **223.255.255.0**, kde číslo sítě je obsaženo v prvních třech oktetech. To umožňuje vytvořit téměř 2 miliony sítí s až 254 hostiteli.

Třída D, E a F Adresy, které spadají do intervalu od **224.0.0.0** do **254.0.0.0,** jsou buď experimentální, nebo jsou rezervovány pro speciální použití. Nespecifikují žádnou síť. Právě z tohoto rozsahu má přiděleny adresy například služba IP Multicast, která umožňuje posílat data současně na více počítačů.

Vrátíme-li se zpět k příkladu z první kapitoly, zjistíme, že adresa **149.76.12.4**, tedy adresa počítače **quark**, spadá do třídy B a označuje hostitele **12.4** na síti **149.76.0.0**.

Ve výše uvedeném seznamu jste si možná všimli, že v každém oktetu nebyly v příslušné části hostitele povoleny všechny hodnoty. Je to tím, že oktety s čísly **0** a **255** jsou rezervovány pro speciální účely. Adresa, ve které jsou všechny bity hostitelské části nulové, odkazuje na síť a adresa, v níž jsou všechny bity hostitelské části rovny jedné, se nazývá vysílací adresa. Vysílací adresa současně odkazuje na všechny hostitele v konkrétní síti. Tedy adresa **149.76.255.255** není konkrétní adresa nějakého hostitele, ale označuje všechny hostitele v síti **149.76.0.0**.

Dále existují ještě různé rezervované síťové adresy. Dvě z nich jsou **0.0.0.0** a **127.0.0.0**. První z nich se nazývá *implicitní směr*, druhá se nazývá *lokální adresa*. Implicitní směr souvisí se způsobem směrování IP datagramů, které budeme probírat dále.

Síť **127.0.0.0** je rezervována pro lokální přenosy na vašem počítači. Adresa **127.0.0.1** je typicky přiřazena speciálnímu rozhraní na vašem hostiteli, které se nazývá *loopback interface* a chová se jako uzavřený obvod. Každý IP paket předaný protokolem TCP nebo UDP je vrácen zpět, jakoby právě přišel z nějaké sítě. To umožňuje vyvíjet a testovat síťový software, aniž byste používali skutečnou síť. Další užitečnou aplikací je případ, kdy chcete použít síťový software na samostatném počítači. Není to zase tak neobvyklé, jak by se mohlo zdát; například mnoho systémů UUCP nemá vůbec žádné IP propojení, ale přesto chtějí provozovat systém zpráv INN. Aby systém INN pod Linuxem správně fungoval, vyžaduje použití loopback rozhraní.

Část adres z každé třídy je vyhrazena jako „rezervované“ nebo „privátní“ adresy. Tyto adresy jsou rezervovány pro použití na privátních sítích a v Internetu se nesměrují. Typicky se používají v organizacích, které si budují vlastní síť a jsou užitečné i pro malé domácí sítě. Rezervované adresy jsou uvedeny v tabulce 2.1.

Rozlišování adres

Když teď víte, jak se vytváří IP adresy, bude vás asi zajímat, jak se s jejich pomocí v Ethernetové nebo Token Ringové síti adresují různí hostitelé. Tyto protokoly koneckonců identifikují počítače vlastními adresami, které nemají nic společného s IP adresami, že? Správně.

Proto potřebujeme mechanismus pro mapování IP adres na adresy fyzické sítě. Tento mechanismus se nazývá *Address Resolution Protocol*, zkráceně ARP, a není omezen pouze na Ethernet nebo Token Ring, ale používá se i u jiných typů sítí, například u radiových sítí s protokolem AX.25. Myšlenka protokolu ARP je stejná jako když se někdo pokouší najít pana X mezi 150 různými lid-

mi: osoba která jej hledá prostě zakřičí dost nahlas, aby ji všichni slyšeli a spoléhá na to, že pan X se ozve (pokud je přítomen). Jakmile se ozve, víme, kdo to je.

Když chce ARP najít ethernetovou adresu odpovídající příslušné IP adrese, použije ethernetový mechanismus vysílání, při kterém je datagram současně adresován na všechny stanice v síti. Vyslaný datagram, odeslaný protokolem ARP, obsahuje dotaz na hledanou IP adresu. Každý hostitel, který tento datagram přijme, ji porovná se svou vlastní IP adresou a pokud souhlasí, vrátí dotazujícímu se hostiteli odpověď pomocí protokolu ARP. Nyní může dotazující se hostitel získat z odpovědi ethernetovou adresu počítače, který mu odpověděl.

Možná se ptáte, jak může nějaký počítač najít jiný jen podle jeho internetové adresy, když může jít o počítač někde na druhém konci světa. Odpověď na tuto otázku souvisí se *směrováním*, tedy s nalezením fyzického umístění počítače v síti. O tomto problému budeme hovořit v další části.

Podívejme se nyní na protokol ARP. Když hostitel získá ethernetovou adresu, uloží ji do vyrovnávací paměti protokolu ARP, takže když bude dotyčnému hostiteli posílat znovu nějaký datagram, nemusí se na ni znovu dotazovat. Tuto informaci by však nebylo rozumné uchovávat navždycky; ethernetová karta vzdáleného hostitele může být například z důvodu technických problémů vyměněna, čímž by byla data protokolu ARP neplatná[17]. Proto jsou po určité době záznamy z vyrovnávací paměti protokolu ARP vymazány a vynutí se tak nové dotazování.

Někdy je také nutné nalézt IP adresu, která odpovídá určité ethernetové adrese. K tomu dochází například v situaci, kdy chce bezdiskový počítač zavést operační systém ze síťového serveru, což je v lokálních sítích docela běžná situace. Bezdiskový klient však o sobě nemůže poskytnout vůbec žádné informace – kromě své ethernetové adresy! Jediné co může udělat je poslat bootovacímu serveru zprávu se žádostí o sdělení své IP adresy. Pro tento účel existuje další protokol, který se jmenuje *Reverse Adress Resolution Protocol*, zkráceně RARP. Společně s protokolem BOOTP slouží k zajištění procedury zavedení operačního systému bezdiskových klientů v síti.

Směrování protokolu IP

Nyní se budeme věnovat otázce jak nalézt počítač, jemuž mají být data poslána, pouze podle jeho IP adresy. Různé části adresy jsou zpracovávány různě a je tedy váš úkol nastavit správně příslušné soubory, které udávají, jak mají být jednotlivé části adresy chápány.

IP sítě

Když někomu píšete dopis, uvedete obvykle na obálce úplnou adresu, která obsahuje stát, město, kraj, PSČ a podobně. Poté co ho vhodíte do poštovní schránky, pošta jej doručí adresátovi: konkrétně ho pošle do uvedeného státu, zde ho místní pošta odešle do příslušného kraje a tak dále. Výhoda tohoto hierarchického schématu je poměrně zřejmá: ať už pošlete dopis kamkoliv, bude vždy místní poštmistr zhruba vědět směr, kterým má dopis poslat dál a přitom se nemusí starat o to, kudy bude dopis putovat z úřadu, na který jej doručí on.

Sítě IP jsou strukturovány obdobným způsobem. Celý Internet se skládá z mnoha sítí, kterým říkáme *autonomní systémy*. Každý takovýto systém provádí veškeré směrování v rámci svých členských systémů, takže se úkol doručení datagramu redukuje na nalezení cesty k síti cílového systému. Znamená to, že jakmile datagram zachytí *libovolný* hostitel v cílové síti, bude další zpracování provedeno výhradně touto sítí.

17 Pozn. překladatele: Ještě pravděpodobnější situace je ta, že typicky počítačům v lokální síti jsou IP adresy přidělovány dynamicky (například protokolem DHCP), takže fyzicky jeden a týž počítač se stejnou síťovou kartou může mít jednu IP adresu dnes a jinou zítra.

Podsítě

Tato struktura vyplývá z rozdělení IP adres na část hostitelskou a část síťovou tak, jak bylo popsáno dříve. Cílová síť je implicitně odvozena ze síťové části IP adresy. Hostitelé se shodnou síťovou částí adresy by se tak měli nacházet ve stejné síti[18].

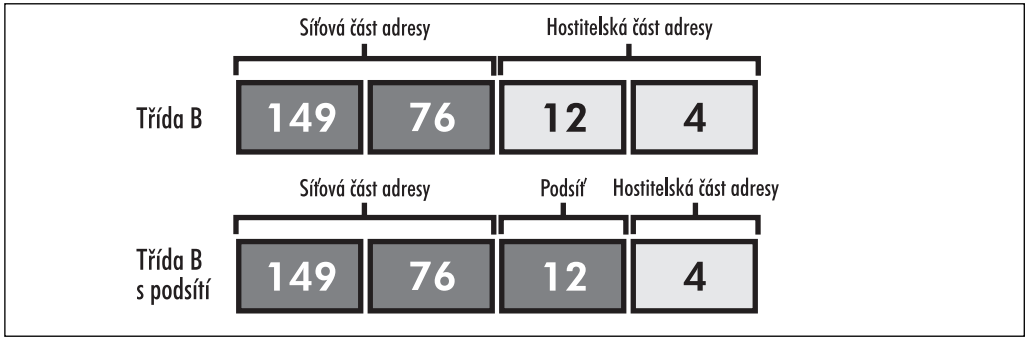
Obdobné schéma má smysl i *uvnitř* sítě, protože vlastní síť se může skládat ze stovek menších sítí, kde jsou nejmenšími jednotkami fyzické sítě, například Ethernety. Protokol IP proto umožňuje rozdělit síť IP na několik *podsítí*.

Podsíť na sebe přebírá odpovědnost za doručení datagramů pro daný rozsah IP adres. K její identifikaci slouží síťová část IP adresy, jako tomu bylo u tříd A, B nebo C. Nyní je však síťová část rozšířena tak, aby obsahovala i některé bity z hostitelské části. Počet bitů, které jsou interpretovány jako číslo podsítě, udává takzvaná *maska podsítě* nebo *síťová maska*. Jedná se také o 32bitové číslo, které určuje bitovou masku síťové části IP adresy[19].

Příkladem takové sítě je školní síť Groucho Marx University. Má síťovou adresu třídy B 149.76.0.0, a proto je její síťová maska 255.255.0.0.

Vnitřně je školní síť GMU složena z několika menších sítí, které tvoří například lokální sítě různých kateder. Proto přidělený rozsah IP adres je rozdělen na 254 podsítí, od adresy 149.76.1.0 po adresu 149.76.254.0. Například katedra teoretické fyziky má přidělenu adresu sítě 149.76.12.0. Páteř školní sítě je samostatná síť s adresou 149.76.1.0. Tyto podsítě sdílejí stejnou síťovou adresu v rámci třídy B a třetí oktet slouží k jejich rozlišení. Používají tedy masku podsítě 255.255.255.0.

Obrázek 2.1 ukazuje rozdílnou interpretaci adresy 149.76.12.4, tedy adresy počítače **quark**, je-li chápána jako běžná adresa třídy B, nebo je-li chápána jako adresa v rámci podsítí.



Obrázek 2.1 – Rozdělení adresy třídy B na podsítě

Je vhodné poznamenat, že vytváření podsítí slouží pouze k *vnitřnímu dělení* sítě. Podsítě jsou vytvářeny vlastníkem sítě (nebo jejím správcem). Podsítě jsou často vytvářeny kvůli existující struktuře, ať už fyzické (mezi dvěma Ethernety), administrativní (mezi dvěma katedrami) nebo geografické (mezi různými lokalitami), a pravomocemi nad jednotlivými podsítěmi je pověřena nějaká

18 Autonomní systémy jsou poněkud obecnější. Mohou v sobě zahrnovat více jednotlivých IP sítí.

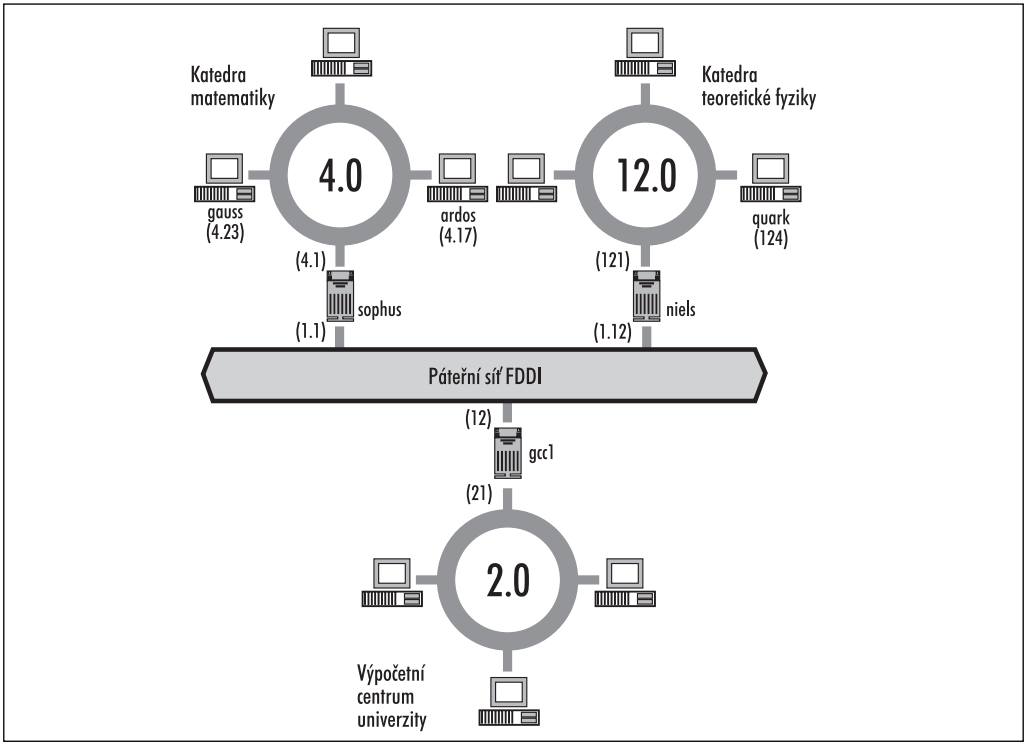
19 Pozn. překladatele: Síťová maska (představíme-li si ji ve dvojkové soustavě) začíná zleva jedničkami přes všechny bity, které označují síťovou část adresy a pokračuje až do konce nulami pro hostitelskou část adresy. Proto máme pro adresu třídy B (první dva bajty jsou síť a druhé dva bajty hostitel) masku 255.255.0.0 (první dva bajty jedniček a druhé dva bajty nul). Rozdělení mezi síťovou a hostitelskou částí adresy je možné nejen na hranici oktetů. Můžeme tak adresu třídy C (tři bajty síť, jeden bajt hostitel) rozdělit maskou například 255.255.255.224 na osm podsítí (první tři bity posledního oktetu jsou adresa podsítě), kde má každá k dispozici 32 hostitelských adres (méně významných pět bitů posledního oktetu). Ve shodě s tím co už bylo řečeno je nicméně z těchto 32 adres fyzicky využitelných jen 30, protože adresa s nulovými posledními pěti bity bude adresou celé konkrétní podsítě a adresa s posledními pěti bity jedničkovými bude vysílací adresou dané podsítě.

kontaktní osoba. Nicméně tato struktura však odráží pouze vnitřní chování sítě a pro okolní svět je neviditelná.

Brány

Vytváření podsítí nemá jen organizační výhody, je často přirozeným důsledkem hardwarové struktury. „Výhled" hostitele dané fyzické sítě, jako například Ethernetu, je velmi omezený: jedinými hostiteli, se kterými lze komunikovat přímo, jsou hostitelé v dané síti. Ke všem ostatním hostitelům může být přistupováno jen s pomocí specializovaných počítačů, takzvaných *bran*. Brána je hostitel, který je současně spojen se dvěma nebo více fyzickými sítěmi a který je nastaven pro přenášení paketů mezi nimi.

Obrázek 2.2 znázorňuje část síťové topologie sítě univerzity GMU. Počítače připojené ke dvěma různým podsítím jsou znázorněny se dvěma IP adresami.



Obrázek 2.2 – Část síťové topologie univerzity GMU

Aby protokol IP poznal, že se hostitel nachází ve fyzicky lokální síti, musí být různým fyzickým sítím přiřazeny různé IP sítě. Například síťová adresa 149.76.4.0 je vyhrazena pro hostitele lokální sítě katedry matematiky. Když se posílá datagram na počítač **quark**, síťový software na serveru **erdos** okamžitě z IP adresy 149.76.12.4 pozná, že cílový hostitel je připojen k jiné fyzické síti, a proto je dosažitelný pouze pomocí brány (implicitní bránou je **sophus**).

Vlastní brána **sophus** je propojena se dvěma rozdílnými podsítěmi: s katedrou matematiky a s páteřní sítí univerzity. Ke každé z nich přistupuje za pomoci různých rozhraní *eth0*, respektive *fddi0*. Jakou adresu mu ale přidělíme? Máme mu dát adresu z podsítě 149.76.1.0 nebo 149.76.4.0?

Odpověď zní: obě. Brána **sophus** musí mít přiřazenu adresu 149.76.1.1 pro použití na podsíti 149.76.1.0 a rovněž adresu 149.76.4.1 pro použití na podsíti 149.76.4.0[20]. Brána musí mít přidělenu jednu IP adresu pro každou síť, ke které je připojena. Tyto adresy – společně s odpovídající síťovou maskou – jsou svázány s rozhraním, přes které se s danou sítí komunikuje. Proto bude mít brána **sophus** přiřazena rozhraní a adresy tak, jak to ukazuje následující tabulka.

Rozhraní	Adresa	Síťová maska
`eth0`	149.76.4.1	255.255.255.0
`fddi0`	149.76.1.1	255.255.255.0
`lo`	127.0.0.1	255.0.0.0

Poslední řádek popisuje lokální rozhraní *lo*, o kterém jsme se již zmínili.

Jemnou odlišnost mezi přidělením adresy hostiteli nebo jeho rozhraní je možné ignorovat. Na hostitele, kteří se nachází jen v jedné síti, například hostitel **erdos**, se budete obecně odkazovat pomocí jeho IP adresy, i když třeba přísně vzato tato adresa odpovídá ethernetovému rozhraní, jemuž je daná IP adresa přidělena. Tento rozdíl je ale důležitý v případě, kdy se odkazujete na bránu.

Směrovací tabulka

Nyní se zaměříme na způsob, jakým protokol IP vybírá bránu, kterou použije k doručení datagramu do vzdálené sítě.

Ukázali jsme si, že u datagramu určeného serveru **quark** zkontroluje odesílající hostitel **erdos** cílovou adresu, načež zjistí, že se nenachází na stejné lokální síti. Proto jej pošle na implicitní bránu **sophus**, která je teď postavena před stejný úkol. Brána **sophus** pozná, že hostitel **quark** není na žádné ze sítí, s nimiž je přímo spojena, takže musí najít další bránu, na kterou by datagram poslala. Správnou volbou bude brána **niels**, což je brána katedry fyziky. Nicméně aby mohla brána **sophus** správně asociovat cílovou síť s odpovídající branou, potřebuje k tomu nějaké dodatečné informace[21].

Protokol IP pro tento účel používá tabulku, která asociuje brány se sítěmi, jež lze prostřednictvím dané brány dosáhnout. Navíc musí tabulka obsahovat záznam „pro všechno ostatní“ (takzvanou *implicitní trasu*) – tou bude brána přiřazená síti 0.0.0.0. Této adrese vyhovují všechny neznámé sítě a tak jsou pakety pro síť, u níž není trasa explicitně definována, posílány implicitní trasou. Pro bránu **sophus** by mohla směrovací tabulka vypadat takto[22]:

---

20 Pozn. překladatele: První (a analogicky i druhá) adresa samozřejmě nemusí být 146.76.1.*1*, ale může to být 146.76.1.*cokoliv*. Bývá nicméně zvykem, že se branám přiřazují adresy „z některého kraje“ rozsahu adres dané podsítě, tedy buď od *x.x.x.1* nahoru nebo od *x.x.x.254* dolů.

21 Pozn. překladatele: I když to na obrázku 2.2 nebylo, nezapomínejme, že na páteřní síti univerzity nejsou jen brány **sophus** a **niels**, ale i padesát dalších bran Katedry elektrických strojů a přístrojů, Katedry automatizace a dokonce i Katedry jazyků, Katedry společenských věd a všeho dalšího včetně rektorátu. A chudák **sophus** se přece nebude ptát každé z nich, zda ve své „diecézi“ nemají požadovaný cílový stroj.

22 Pozn. překladatele: Z tabulky je vidět, jak důležité je přidělovat branám adresy *inteligentně* a hlavně *konzistentně*. Na síti s padesáti podsítěmi a odpovídajícím počtem bran byste se totiž jinak z vytváření směrovacích tabulek zbláznili.

Síť	Síťová maska	Brána	Rozhraní	Poznámka
149.76.1.0	255.255.255.0	-	`fddi0`	páteřní síť, jsme připojeni přímo
149.76.2.0	255.255.255.0	149.76.1.2	`fddi0`	síť počítačového centra a její brána
149.76.3.0	255.255.255.0	149.76.1.3	`fddi0`	síť nějaké katedry (třeba Automatizace)
149.76.4.0	255.255.255.0	-	`eth0`	síť naší katedry, k té jsme připojeni přímo přes Ethernet
149.76.5.0	255.255.255.0	149.76.1.5	`fddi0`	síť další cizí katedry (třeba Elektrických pohonů)
...	...	...	`...`	tady budou odkazy na sítě a brány všech kateder univerzity
149.76.1.0	0.0.0.0	149.76.1.2	`fddi0`	přes počítačové centrum máme přístup na Internet, jejich brána proto slouží i jako implicitní

Při uvádění trasy pro síť, na niž je brána **sophus** připojena přímo, nepotřebujete žádnou další bránu, proto je na těchto místech místo adresy brány uvedena pomlčka.

Proces zjištění, které trase odpovídá konkrétní cílová adresa je čistě matematická operace. Celý proces je velmi jednoduchý, jeho pochopení však vyžaduje znalost aritmetiky a logiky ve dvojkové soustavě. Trasa danému cíli odpovídá tehdy a jen tehdy, jestliže adresa sítě binárně ANDována se síťovou maskou je rovna cílové adrese binárně ANDované se síťovou maskou.

Překlad předchozí věty: Trasa odpovídá tehdy, jestliže všechny bity adresy sítě definované síťovou maskou (začínáme zleva a zajímají nás ty bity, kde je v síťové masce jednička) jsou stejné jako bity na shodných pozicích v adrese cílového počítače.

Když protokol IP hledá nejlepší trasu k danému cíli, může ve směrovací tabulce najít více vyhovujících položek. Přinejmenším víme, že implicitní trasa vyhovuje pro jakýkoliv cíl, nicméně pro nám známé cíle vyhovují i jiné trasy. Jak protokol pozná, kterou trasu použít? V této situaci hraje významnou roli síťová maska. I když danému cíli vyhovuje více tras, jedna má síťovou masku delší a druhá kratší (rozuměj: jedna má v síťové masce více jedniček, druhá méně). Čím delší je maska, tím přesněji vyhovuje cílové adrese, a proto se při směrování datagramů vždy volí trasa s nejdelší síťovou maskou. Implicitní trasa má masku se všemi bity nulovými, zatímco masky lokálních podsítí mají 24 jedničkových bitů. Proto budou datagramy pro lokální podsítě předávány příslušným branám, protože jejich síťová adresa je delší. Jediné datagramy odesílané implicitní branou budou ty, které nevyhovují žádné jiné položce směrovací tabulky.

Směrovací tabulky mohou být vytvářeny různými způsoby. U malých sítí LAN je většinou nejefektivnější vytvořit je ručně a při procesu zavádění systému je předat protokolu IP pomocí příkazu **route** (viz kapitola 5). V rozsáhlejších sítích jsou vytvářeny a modifikovány za běhu systému pomocí *směrovacích démonů*; tyto démony běží na centrálních hostitelích sítě a prostřednictvím směrovacích protokolů si vyměňují informace pro výpočet optimálních tras mezi členskými sítěmi.

V závislosti na velikosti sítě mohou být použity různé směrovací protokoly. Ke směrování uvnitř autonomních systémů (jako je školní síť Groucho Marx) budou použity *interní směrovací protokoly*. Nejvýznamnějším z nich je protokol RIP, *Routing Information Protocol*, který je implementován v BSD démonu **routed**. Ke směrování mezi autonomními systémy slouží *externí směrovací protokoly*, jako je například EGP (*External Gateway Protocol*) nebo BGP (*Border Gateway Protocol*). Tyto protokoly (stejně jako RIP) jsou implementovány v démonu **gated** z Cornellské univerzity.

Hodnoty metrik

Dynamické směrování založené na protokolu RIP vybírá nejlepší směrování k cílovému hostiteli nebo síti na základě počtu skoků, to znamená počtu bran, kterými musí datagram projít, než do-

sáhne cíle. Čím méně má trasa skoků, tím lépe ji protokol RIP ohodnotí. Dlouhé trasy s 16 nebo více skoky jsou považovány za nepoužitelné a jsou zrušeny.

Chcete-li použít protokol RIP na vnitřní správu směrovacích informací ve vaší místní síti, musíte mít na všech hostitelích spuštěn program **gated**. Při procesu zavádění systému zkontroluje program **gated** všechna aktivní síťová rozhraní. Pokud existuje více než jedno aktivní rozhraní (nepočítaje lokální rozhraní), bude předpokládat, že „jeho" počítač předává pakety mezi několika sítěmi a bude aktivně vyměňovat a vysílat směrovací informace. V opačném případě bude jen pasivně přijímat všechny aktualizace protokolem RIP a bude aktualizovat lokální směrovací tabulku.

Při vysílání informace z lokální směrovací tabulky vypočte program **gated** délku trasy z takzvané *metriky*, která je ve směrovací tabulce součástí dat jednotlivých položek. Metrika je hodnota, kterou při konfiguraci směrování nastavuje správce systému a měla by odrážet skutečnou „cenu" použití dané trasy[23]. Proto by měla být metrika trasy na podsíť, s níž je brána přímo propojena vždy rovna nule, zatímco trasa procházející dvěma bránami by měla mít metriku rovnu dvěma. S metrikami se nicméně nemusíte zatěžovat, pokud nebudete používat protokol RIP nebo program **gated**.

Protokol ICMP

Součástí protokolu IP je ještě další protokol, o kterém jsme doposud nemluvili. Jde o protokol ICMP (*Internet Control Message Protocol*), jehož prostřednictvím odesílá síťový kód jádra zprávy o chybách jiným počítačům. Předpokládejme například, že jste znovu přihlášeni na hostiteli **erdos** a chcete se za pomoci telnetu připojit na port 12345 serveru **quark**, nicméně na tomto portu tohoto počítače žádný proces neposlouchá. Jakmile dorazí na tento port serveru **quark** nějaký paket, síťová vrstva to pozná a okamžitě vrátí prostřednictvím protokolu ICMP hostiteli **erdos** zprávu, v níž bude uvedeno chybové hlášení Port Unreachable (port je nedosažitelný).

Protokol ICMP obsahuje poměrně velký počet zpráv, z nichž se řada týká různých chybových stavů. Velmi zajímavou zprávou je zpráva Redirect message (přesměrování zprávy). Generuje ji směrovací modul v případě, že zjistí, že ho nějaký systém používá jako bránu v případě, že k požadovanému cíli existuje i kratší cesta. Například po restartu systému může být směrovací tabulka brány **sophus** neúplná a obsahuje pouze trasy na síť katedry matematiky, na páteř FDDI a implicitní směr na centrální bránu počítačového centra (**gcc1**). Proto bude každý paket pro server **quark** poslán na bránu **gcc1** místo aby byl poslán přímo na bránu **niels**, což je brána katedry fyziky. Při přijetí takového datagramu si brána **gcc1** všimne, že jde o špatnou volbu trasy, pošle paket na bránu **niels** a zároveň vrátí bráně **sophus** pomocí protokolu ICMP zprávu o přesměrování, která bude obsahovat výhodnější cestu.

Teď to vypadá, že jde o velmi chytrý způsob, jak se vyhnout manuálnímu nastavování všech tras s výjimkou těch nejzákladnějších. Ale ne vždy je dobré spoléhat na dynamická směrovací schémata, ať už jde o protokol RIP nebo ICMP přesměrování. Zpráva o přesměrování u protokolu ICMP nebo protokol RIP umožňují jen malou, případně nulovou kontrolu toho, zda je směrovací informace skutečně autentická. Takto mohou zlomyslní uživatelé přerušit dopravu v celé síti nebo případně provést i něco horšího. Proto všechny verze linuxového síťového kódu chápou zprávy o přesměrování pouze jako zprávy týkající se konkrétního počítače, nikoliv celé sítě. Tím se minimalizuje dopad eventuálního útoku pouze na směrování k jedinému počítači a nikoliv na směrování pro celou síť. Na opačné straně to ale vede k nadbytečnému provozu, pokud k přesměro-

23 Cenu trasy můžeme v nejjednodušším případě odvodit z počtů skoků v jednotlivých trasách. Ve složitějších případech (alternativní propojení více linkami s různou rychlostí, použití linek, kde platíme za objem přenesených dat a podobně) představuje optimální nastavení metrik mimořádné složité umění.

vání dochází oprávněně, protože pro každý počítač na cílové síti bude generována samostatná ICMP zpráva o přesměrování. Obecně je dnes považováno za nevychovanost spoléhat při směrování na ICMP a jeho zprávy o přesměrování trasy.

Rozlišení jmen hostitele

Jak už jsme řekli, je adresování v sítích na bázi protokolu TCP/IP (přinejmenším ve verzi IPv4) prováděno prostřednictvím 32bitových čísel. Pokud byste si však chtěli zapamatovat více takovýchto adres, strávili byste nad nimi poměrně hodně času. Proto se hostitelé označují „běžnými" jmény, jako je třeba hostitel **gauss** nebo hostitel **strange**. Povinností aplikace pak je, aby našla IP adresu odpovídající danému jménu. Tento proces se označuje jako *rozlišení jména hostitele*.

Když chce aplikace IP adresu odpovídající jménu nějakého hostitele, použije knihovní funkce `gethostbyname(3)` a `gethostbyaddr(3)`. Typicky byly tyto a další příbuzné funkce poskytovány knihovnou *resolveu*, v Linuxu jsou součástí standardní knihovny *libc*. Hovorově se všechny funkce této kategorie označují jako „resolver" – „ten, který rozlišuje". Informace o konfiguraci resolveru najdete v kapitole 6.

U malých sítí jako je Ethernet nebo skupiny sítí Ethernet není příliš obtížné udržovat tabulky s názvy hostitelů a jejich adresami. Tyto informace jsou obvykle uchovávány v souboru pojmenovaném `/etc/hosts`. Při přidávání nebo odebírání hostitele nebo při změně adresy stačí na všech hostitelích v síti aktualizovat soubor `hosts`. Je jasné, že u sítí s více než jen několika počítači by byl takovýto proces dost náročný.

Jedním z řešení tohoto problému je systém NIS (*Network Information System*) vyvinutý firmou Sun Microsystems, hovorově označovaný jako YP, tedy *Žluté stránky* (*Yellow Pages*). Systém NIS ukládá soubor `hosts` (a některé další informace) do databáze na nějakém hlavním hostiteli, odkud ho mohou klienti podle potřeby získat. I tak je toto řešení vhodné pouze pro středně velké sítě, jako jsou sítě LAN, protože vyžaduje centrální správu celé databáze `hosts` a její distribuci na všechny servery. Konfigurace a instalace služby NIS je popsána v kapitole 13.

V Internetu byly adresy všech počítačů původně rovněž uchovávány v jediné databázi nazvané `HOSTS.TXT`. Tento soubor byl spravován institucí Network Information Center (NIC) a všechny zúčastněné systémy si ho musely stáhnout a nainstalovat. Jak se síť rozrůstala, objevilo se v souvislosti s tímto schématem několik problémů. Kromě administrativní režie spojené s pravidelnou instalací souboru `HOSTS.TXT` se neúnosně zvětšovalo i zatížení serverů, které ho distribuovaly. Mnohem horší ale bylo, že všechny názvy musely být registrovány v centru NIC, čímž se zajistilo, aby se některý název neobjevil dvakrát.

Proto bylo v roce 1984 přijato nové schéma pro rozlišování názvů. Stal se jím *Domain Name System*, DNS. Systém DNS byl navržen Paulem Mockapetrisem a současně řeší oba zmíněné problémy. O systému DNS podrobněji hovoříme v kapitole 6.

KAPITOLA 3

Konfigurace síťového hardware

Až dosud jsme se bavili o síťových rozhraních a všeobecně o problematice protokolu TCP/IP, ale zatím jsme si neřekli, co se doopravdy děje, když „síťový kód" jádra přistupuje k nějakému hardwarovému zařízení. Abychom si to mohli přesně popsat, musíme si nejprve něco říct o koncepci rozhraní a ovladačů.

Nejprve máte samozřejmě vlastní hardware, například ethernetovou, FDDI nebo Token Ringovou kartu: to je laminátová deska přecpaná spoustou mrňavých čipů, která je zasunuta v nějakém slotu vašeho počítače. Takové věci obecně říkáme fyzické zařízení.

Abyste mohli síťovou kartu používat, musí být v jádru Linuxu k dispozici speciální funkce, které rozumí způsobu, jakým je k tomuto zařízení přistupováno. Software, který tyto funkce implementuje, označujeme jako *ovladač zařízení*. Linux obsahuje ovladače zařízení pro řadu různých síťových karet: ISA, PCI, MCA, EISA, paralelní port, PCMCIA a nejnověji i USB.

Ale co myslíme tím, když říkáme, že ovladač „obsluhuje" zařízení? Vraťme se zpět k ethernetové kartě. Ovladač musí být nějakým způsobem schopen komunikovat s logikou hardwarové karty: musí jí posílat příkazy a data, a ta by mu měla na oplátku doručit všechna přijatá data.

U počítačů PC se tato komunikace odehrává v oblasti vstupně-výstupní paměti, která je namapována na registry karty, a/nebo prostřednictvím sdílené paměti nebo přímého přístupu do paměti. Všechny příkazy a data vyslaná jádrem do karty musí těmito registry projít. Vstupně-výstupní a paměťové adresy jsou obecně určeny zadáním počáteční neboli *bázové adresy*. Typické bázové adresy pro ethernetové karty na sběrnici ISA jsou `0x280` nebo `0x300`. Kartám na sběrnici PCI se obvykle jejich vstupně-výstupní adresy přidělují automaticky.

Obvykle není třeba si dělat příliš starosti s nastavením hardwarového zařízení, například jeho bázové adresy, protože jádro se při zavádění systému pokusí detekovat pozici karty. Tento proces se nazývá *automatické detekce* a znamená to, že jádro přečte několik paměťových nebo vstupně-výstupních adres a porovná přečtená data s tím, co by získalo, kdyby na nich byla nainstalována určitá ethernetová karta. Mohou však existovat ethernetové karty, které se nepodaří detekovat automaticky; bývá to případ levných ethernetových karet, které nejsou dokonalými klony standardních karet jiných výrobců. Jádro systému se navíc bude při zavádění pokoušet detekovat pouze jediné ethernetové zařízení. Pokud používáte více než jednu kartu, musíte jádru o ostatních kartách explicitně říci.

Dalším takovým parametrem, který možná budete muset jádru sdělit, je nastavení kanálu přerušení. Hardwarové komponenty obvykle vyvolají přerušení jádra v okamžiku, kdy potřebují být obsluhovány – například pokud dorazí nějaká data nebo pokud dojde k nějaké neobvyklé situaci.

U počítačů PC se sběrnicí ISA se mohou přerušení vyskytnout na jednom z 15 kanálů přerušení, které jsou očíslovány 0, 1 a 3 až 15. Číslo přerušení, které je hardwarové komponentě přiděleno, se nazývá *interrupt request number*, zkráceně IRQ[24].

V kapitole 2 jsme si řekli, že jádro přistupuje k zařízení pomocí speciální softwarové konstrukce, takzvaného *rozhraní*. Ta poskytují abstraktní množinu funkcí, totožnou pro různé typy hardwarových zařízení, například funkce pro posílání nebo příjem datagramu.

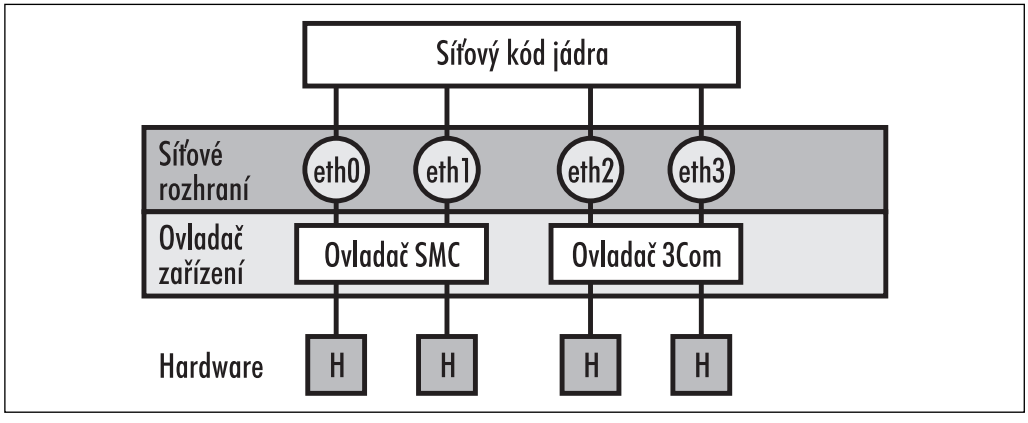
Rozhraní jsou identifikována prostřednictvím svých názvů. Ve většině operačních systémů unixového typu jsou síťová rozhraní implementována jako speciální soubory zařízení v adresáři `/dev/`. Zadáte-li příkaz `ls -las /dev/`, uvidíte, jak soubory zařízení vypadají. Ve sloupci přístupových práv (tedy ve druhém sloupci) si můžete všimnout, že údaj začíná písmenem a nikoliv pomlčkou, jako u normálních souborů. Toto písmeno indikuje typ zařízení. Nejběžnější zařízení jsou typu `b`, který znamená, že jde o *blokové zařízení*, které najednou obsluhuje celé bloky zapisovaných a čtených dat, a zařízení typu `c`, což jsou *znaková zařízení*, jež s daty pracují znak po znaku. Tam, kde ve výpisu příkazu `ls` normálně vidíte délku souboru, jsou v případě zařízení uvedena dvě čísla: hlavní a vedlejší číslo zařízení. Tato čísla označují skutečné zařízení, s nímž je soubor ovladače spjat.

Každý ovladač zařízení si v jádře registruje jednoznačné hlavní číslo zařízení. Každá *instance* konkrétního zařízení pak má v rámci tohoto hlavního čísla své jedinečné vedlejší číslo. Například pro rozhraní `tty` jsou soubory `/dev/tty*` znaková zařízení (to poznáme podle písmene `c`) a všechny mají hlavní číslo `4`, ale `/dev/tty1` má vedlejší číslo `1`, `/dev/tty2` má vedlejší číslo `2` a tak dále. Soubory zařízení jsou velmi užitečné pro řadu typů zařízení, mohou však situaci komplikovat, pokud se snažíme najít a otevřít nějaké nepoužívané zařízení.

Názvy rozhraní jsou interně definovány v jádru, a nejedná se o soubory zařízení v adresáři `/dev`.

Některá typická jména zařízení jsou uvedena dále v části *Prohlídka síťových zařízení v Linuxu*. Přiřazení rozhraní zařízením obvykle závisí na pořadí, v němž jsou zařízení konfigurována. Například první nainstalovaná ethernetová karta obdrží název *eth0*, další bude *eth1* a tak dále. Jinak jsou obsluhována rozhraní SLIP, která se přiřazují dynamicky. Kdykoliv se naváže spojení protokolem SLIP, bude sériovému portu přiřazeno rozhraní.

Schéma uvedené na obrázku 3.1 se pokouší ukázat vztahy mezi hardwarem, ovladači zařízení a rozhraními.



Obrázek 3.1 – Vztah mezi ovladači, rozhraními a hardwarem

24 Přerušení číslo 2 a 9 jsou totožná, protože architektura IBM PC používá dva řadiče přerušení s osmi kanály, zapojené v kaskádě. Sekundární řadič je připojen na kanál IRQ 2 primárního řadiče.

Při zavádění systému zobrazí jádro detekovaná zařízení a rozhraní, která se mu podařilo nainstalovat. Následuje část výpisu typické obrazovky při zavádění systému:

```
.
.
This processor honors the WP bit even when in supervisor mode.\
    Good.
Swansea University Computer Society NET3.035 for Linux 2.0
NET3: Unix domain sockets 0.13 for Linux NET3.035.
Swansea University Computer Society TCP/IP for NET3.034
IP Protocols: IGMP,ICMP, UDP, TCP
Swansea University Computer Society IPX 0.34 for NET3.035
IPX Portions Copyright (c) 1995 Caldera, Inc.
Serial driver version 4.13 with no serial options enabled
tty00 at 0x03f8 (irq = 4) is a 16550A
tty01 at 0x02f8 (irq = 3) is a 16550A
CSLIP: code copyright 1989 Regents of the University of California
PPP: Version 2.2.0 (dynamic channel allocation)
PPP Dynamic channel allocation code copyright 1995 Caldera, Inc.
PPP line discipline registered.
eth0: 3c509 at 0x300 tag 1, 10baseT port, address 00 a0 24 0e e4 e0, \
    IRQ 10.
3c509.c:1.12 6/4/97 becker@cesdis.gsfc.nasa.gov
Linux Version 2.0.32 (root@perf) (gcc Version 2.7.2.1)
#1 Tue Oct 21 15:30:44 EST 1997
 .
 .
```

Výpis ukazuje, že jádro bylo přeloženo se zapnutým protokolem TCP/IP, a obsahuje ovladače protokolů SLIP, CSLIP a PPP. Třetí řádek odspodu uvádí, že byla detekována ethernetová karta 3C509 a byla nainstalována jako rozhraní `eth0`. Máte-li jiný typ ethernetové karty – například kapesní adaptér D-Link – vypíše jádro zpravidla řádek začínající názvem tohoto zařízení – v případě karty D-Link to bude `dl0` – za ním pak následuje typ detekované karty. Pokud máte nainstalovanou síťovou kartu, ale nevidíte žádnou takovou zprávu, znamená to, že jádro není schopno správně vaši kartu detekovat. Tímto problémem se budeme zabývat později v části *Automatická detekce ethernetových zařízení*.

Konfigurace jádra

Většina distributorů Linuxu dodává zaváděcí diskety, které pracují se všemi běžnými typy hardware počítačů PC. Obecně je jádro na těchto disketách vysoce modulární a obsahuje prakticky všechny myslitelné ovladače. Je to vynikající řešení pro instalační diskety, nicméně dlouhodobě takové jádro asi nebudete chtít. Není nic vtipného na tom, mít disk plný ovladačů zařízení, které jsou vám na nic. Typicky si proto sestavíte své vlastní jádro, jež bude obsahovat pouze ty ovladače, které skutečně chcete nebo potřebujete. Tím ušetříte něco diskového prostoru a zkrátíte čas pro překlad jádra.

Každopádně chcete-li používat systém Linux, měli byste ovládat sestavování jádra. Chápejte to jako důkaz vaší způsobilosti, potvrzení toho, proč jsou volně dostupné programy tak výkonné – máte k dispozici jejich zdrojový kód. Nedívejte se na to jako na „musím zkompilovat jádro“, ale raději jako na „*můžu* zkompilovat jádro“. Základy jsou vysvětleny v knize Matta Welshe *Running*

Linux, vydané nakladatelstvím O'Reilly[25]. Proto se v této stati zmíníme pouze o těch konfiguračních volbách, které ovlivňují nastavení sítí.

Důležitá věc, kterou stojí za to zopakovat i zde, je způsob jakým funguje číslování verzí jádra. Jádra Linuxu jsou číslována následujícím způsobem: `2.2.14`. První číslo udává *hlavní verzi jádra*. Toto číslo se mění, když dojde k velké a významné změně architektury jádra. Hlavní verze se tak změnila z 1 na 2, když byla implementována podpora i jiných procesorů než jen Intel. Druhé číslo je *vedlejší číslo verze*. V zásadě to je nejdůležitější číslo, které nás zajímá. V komunitě linuxových vývojářů byl přijat standard, že *sudá* vedlejší čísla označují *produkční* nebo *stabilní* jádra, zatímco *lichá* čísla představují *vývojová* nebo *nestabilní* jádra. Na počítači na němž vám záleží byste měli používat stabilní verze, protože ty jsou mnohem pečlivěji testovány. Vývojová jádra používáte, pokud chcete experimentovat s nejnovějšími funkcemi Linuxu, mohou však obsahovat dosud nenalezené a neodstraněné problémy. Třetí číslo se jednoduše inkrementuje s každým novým uvolněním vedlejší verze[26].

Při spuštění příkazu **make menuconfig** se objeví textová nabídka s řadou konfiguračních voleb, například zda chcete v jádře emulovat matematický koprocesor. Jeden z těchto dotazů se bude týkat instalace podpory pro sítě TCP/IP. Aby bylo vaše jádro schopno pracovat se sítěmi, musíte na tento dotaz odpovědět `y` (yes – ano).

Síťové volby v jádře 2.0 a vyšším

Po skončení úvodní obecné části konfigurace budete dále dotazováni, zda si přejete přidat podporu různých funkcí, například ovladačů SCSI nebo zvukových karet. Výzva zároveň nabízí možné odpovědi. Pokud odpovíte `?`, objeví se popis toho, co vám daná volba konkrétně nabízí. Vždy budete mít možnost zvolit `y` pro statické zahrnutí komponenty do jádra nebo `n` pro její úplné vyloučení. U komponent, které mohou být přeloženy jako za běhu zaváděné moduly, se bude nabízet také volba `m`. Modulární ovladače musí být před použitím nahrány a jsou vhodné pro zařízení, která nepoužíváte často.

Následující seznam otázek se týká síťové problematiky. Přesná množina voleb se v důsledku probíhajícího vývoje neustále mění. Typický seznam nabízený většinou jader verze 2.0 a 2.1 vypadá takto:

```
*
* Network device support
*
Network device support (CONFIG_NETDEVICES) [Y/n/?]
```

Na tuto otázku musíte odpovědět y, pokud chcete používat *jakékoliv* síťové zařízení, ať už Ethernet, SLIP, PPP nebo cokoliv jiného. Odpovíte-li y, automaticky se zapne podpora ethernetových zařízení. Pokud chcete podporu i jiných zařízení, musíte kladně odpovědět i na další otázky:

```
PLIP (parallel port) support (CONFIG_PLIP) [N/y/m/?] y
PPP (point-to-point) support (CONFIG_PPP) [N/y/m/?] y
*
* CCP compressors for PPP are only built as modules.
```

25 Pozn. překladatele: A také v dokumentu *Linux Kernel HOWTO*, který je součástí této knihy.

26 Je dobré, když se používají vývojová jádra a uživatelé hlásí v nich nalezené chyby. Pokud můžete nějaký počítač použít jako testovací, uděláte tak velmi užitečnou věc. Instrukce pro hlášení chyb jsou uvedeny v souboru `/usr/src/linux/REPORTING-BUGS` ve zdrojovém kódu jádra.

```
*
SLIP (serial line) support (CONFIG_SLIP) [N/y/m/?] m
 CSLIP compressed headers (CONFIG_SLIP_COMPRESSED) [N/y/?] (NEW) y
 Keepalive and linefill (CONFIG_SLIP_SMART) [N/y/?] (NEW) y
 Six bit SLIP encapsulation (CONFIG_SLIP_MODE_SLIP6) [N/y/?] (NEW) y
```

Tyto otázky se týkají různých protokolů linkové vrstvy, které Linux podporuje. Protokoly PPP a SLIP umožňují přenos datagramů IP po sériové lince. PPP je vlastně skupina protokolů používaných pro přenos síťového provozu sériovou linkou. Některé z protokolů rodiny PPP řeší autentikaci u telefonního serveru, další řeší přenos konkrétních datových protokolů – PPP totiž není omezen na přenos IP datagramů, může přenášet i jiné protokoly, jako například IPX.

Pokud odpovíte `y` nebo `m` na dotaz na podporu protokolu SLIP, budete dále muset odpovědět ještě na tři otázky. Volba týkající se komprese hlaviček zapíná protokol CSLIP, v němž mohou být IP hlavičky komprimovány až na tři bajty. Všimněte si, že tato volba automaticky nezapíná použití protokolu CSLIP, pouze dodá do jádra funkce, které jeho použití dovolí. Volba `Keepalive and linefill` způsobí, že protokol SLIP bude na lince periodicky automaticky generovat nějakou aktivitu, aby nedocházelo k odpojení serverem kvůli neaktivitě. Konečně volba `Six bit SLIP encapsulation` umožňuje provozovat SLIP i na linkách, které nejsou schopny přenášet celou 8bitovou množinu znaků. Trochu se to podobá technice uuencoding nebo binhex, které se používají pro přenos binárních souborů elektronickou poštou.

Protokol PLIP dovoluje posílání IP datagramů přes paralelní port. Nejčastěji se používá pro komunikaci s dosovskými stanicemi. Na typickém PC bude PLIP rychlejší než SLIP nebo PPP, má však výrazně vyšší režijní zatížení procesoru, takže zatímco přenos bude rychlý, ostatní úlohy mohou být zpomaleny.

Další otázky se týkají síťových karet různých výrobců. S vývojem dalších a dalších ovladačů se budou otázky v této části rozrůstat. Pokud chcete sestavit jádro pro použití na více počítačích nebo pokud máte v počítači instalováno více síťových karet, můžete povolit více než jeden ovladač:

```
 .
 .
Ethernet (10 or 100Mbit) (CONFIG_NET_ETHERNET) [Y/n/?]
3COM cards (CONFIG_NET_VENDOR_3COM) [Y/n/?]
3c501 support (CONFIG_EL1) [N/y/m/?]
3c503 support (CONFIG_EL2) [N/y/m/?]
3c509/3c579 support (CONFIG_EL3) [Y/m/n/?]
3c590/3c900 series (592/595/597/900/905) "Vortex/Boomerang" support/
    (CONFIG_VORTEX) [N/y/m/?]
AMD LANCE and PCnet (AT1500 and NE2100) support (CONFIG_LANCE) [N/y/?]
AMD PCInet32 (VLB and PCI) support (CONFIG_LANCE32) [N/y/?] (NEW)
Western Digital/SMC cards (CONFIG_NET_VENDOR_SMC) [N/y/?]
WD80*3 support (CONFIG_WD80x3) [N/y/m/?] (NEW)
SMC Ultra support (CONFIG_ULTRA) [N/y/m/?] (NEW)
SMC Ultra32 support (CONFIG_ULTRA32) [N/y/m/?] (NEW)
SMC 9194 support (CONFIG_SMC9194) [N/y/m/?] (NEW)
Other ISA cards (CONFIG_NET_ISA) [N/y/?]
Cabletron E21xx support (CONFIG_E2100) [N/y/m/?] (NEW)
DEPCA, DE10x, DE200, DE201, DE202, DE422 support (CONFIG_DEPCA) [N/y/m/?]/
    (NEW)
EtherWORKS 3 (DE203, DE204, DE205) support (CONFIG_EWRK3) [N/y/m/?] (NEW)
EtherExpress 16 support (CONFIG_EEXPRESS) [N/y/m/?] (NEW)
HP PCLAN+ (27247B and 27252A) support (CONFIG_HPLAN_PLUS) [N/y/m/?] (NEW)
```

```
HP PCLAN (27245 and other 27xxx series) support (CONFIG_HPLAN) [N/y/m/?]/
    (NEW)
HP 10/100VG PCLAN (ISA, EISA, PCI) support (CONFIG_HP100) [N/y/m/?] (NEW)
NE2000/NE1000 support (CONFIG_NE2000) [N/y/m/?] (NEW)
SK_G16 support (CONFIG_SK_G16) [N/y/?] (NEW)
EISA, VLB, PCI and on card controllers (CONFIG_NET_EISA) [N/y/?]
Apricot Xen-II on card ethernet (CONFIG_APRICOT) [N/y/m/?] (NEW)
Intel EtherExpress/Pro 100B support (CONFIG_EEXPRESS_PRO100B) [N/y/m/?]/
    (NEW)
DE425, DE434, DE435, DE450, DE500 support (CONFIG_DE4X5) [N/y/m/?] (NEW)
DECchip Tulip (dc21x4x) PCI support (CONFIG_DEC_ELCP) [N/y/m/?] (NEW)
Digi Intl. RightSwitch SE-X support (CONFIG_DGRS) [N/y/m/?] (NEW)
Pocket and portable adaptors (CONFIG_NET_POCKET) [N/y/?]
AT-LAN-TEC/RealTek pocket adaptor support (CONFIG_ATP) [N/y/?] (NEW)
D-Link DE600 pocket adaptor support (CONFIG_DE600) [N/y/m/?] (NEW)
D-Link DE620 pocket adaptor support (CONFIG_DE620) [N/y/m/?] (NEW)
Token Ring driver support (CONFIG_TR) [N/y/?]
IBM Tropic chipset based adaptor support (CONFIG_IBMTR) [N/y/m/?] (NEW)
FDDI driver support (CONFIG_FDDI) [N/y/?]
Digital DEFEA and DEFPA adapter support (CONFIG_DEFXX) [N/y/?] (NEW)
ARCnet support (CONFIG_ARCNET) [N/y/m/?]
  Enable arc0e (ARCnet "Ether-Encap" packet format) (CONFIG_ARCNET_ETH)/
      [N/y/?] (NEW)
  Enable arc0s (ARCnet RFC1051 packet format) (CONFIG_ARCNET_1051)/
      [N/y/?] (NEW)
 .
 .
```

Konečně v části věnované souborovým systémům se vás konfigurační skript zeptá, zda budete chtít podporovat síťový souborový systém NFS. NFS vám umožňuje exportovat souborové systémy na jiné počítače, což vaše soubory na těchto počítačích zpřístupní stejně, jako kdyby byly jejich lokální:

```
NFS file system support (CONFIG_NFS_FS) [y]
```

O NFS budeme podrobněji hovořit v kapitole 14.

Síťové volby v jádře 2.0.0 a vyšším

Linux 2.0.0 znamenal v oblasti sítí podstatnou změnu. Standardní součástí jádra se stala řada funkcí, například podpora IPX. Kromě toho byla přidána řada dalších voleb. Mnohé z nich se používají jen za velmi specifických okolností a nebudeme se jimi zabývat. O čem nebudeme mluvit zde, naleznete pravděpodobně v dokumentu Networking HOWTO. V dalším textu si popíšeme řadu užitečných voleb a vysvětlíme si, kdy byste měli jednotlivé volby použít.

Základ

Abyste mohli používat síťovou podporu TCP/IP, musíte na následující otázku odpovědět `y`. I pokud na ni odpovíte `n`, budete později moci zapnout podporu protokolu IPX.

```
Networking options  --->
    [*] TCP/IP networking
```

Brány

Tuto volbu musíte zapnout, pokud váš systém funguje jako brána mezi dvěma sítěmi nebo mezi lokální sítí a linkou SLIP a podobně. Pokud volbu necháte zapnout vždy, nic se nestane, nicméně můžete ji vypnout a nakonfigurovat počítač jako *firewall*. Firewally jsou počítače připojené ke dvěma nebo více sítím, nepředávají však provoz mezi nimi. Typicky se používají k zajištění přístupu na Internet s minimalizací rizik pro interní síť. Uživatelé se mohou k firewallu přihlásit a používat internetové služby, nicméně interní počítače jsou chráněny před útoky z Internetu, protože firewall nepropustí dovnitř žádná spojení. (O firewallech budeme mluvit v kapitole 9.)

```
[*] IP: forwarding/gatewaying
```

Virtuální hostitelé

Následující volby vám umožní přiřadit jednomu rozhraní více IP adres. To může být užitečné, pokud provozujete takzvané „virtuální hostitele" – tedy v případě, kdy jeden počítač vypadá a chová se jako více samostatných počítačů, každý s vlastní síťovou identitou. O problematice IP aliasů budeme více hovořit za chvíli.

```
[*] Network aliasing
    <*> IP: aliasing support
```

Účtování

Tato volba umožňuje shromažďovat data o IP provozu (budeme o ní hovořit v kapitole 10).

```
[*] IP: accounting
```

PC záplata

Tato volba umožňuje obejít nekompatibilitu s některými verzemi PC/TCP, tedy s komerční implementací TCP/IP na dosovských počítačích. Při zapnutí této volby budete stále schopni komunikovat i s normálními linuxovými počítači, výkon linky se ale může lehce snížit.

```
--- (it is safe to leave these untouched)
[*] IP: PC/TCP compatibility mode
```

Bezdiskové bootování

Tato volba zapíná protokol RARP (*Reverse Address Resolution Protocol*). Protokol RARP používají bezdiskoví klienti a X terminály k vyžádání své IP adresy v době bootování. Pokud budete s tímto typem klientů pracovat, musíte tuto volbu zapnout. K přidávání záznamů do RARP tabulky jádra slouží krátký prográmek **rarp**, který je součástí standardních síťových nástrojů:

```
<*> IP: Reverse ARP
```

MTU

Při posílání dat protokolem TCP/IP musí jádro rozbít blok přenášených dat na menší kusy, které zpracovává protokol IP. Velikost těchto bloků se označuje jako *Maximum Transfer Unit*, MTU. Pro počítače přístupné lokální sítí jako je například Ethernet se MTU nastavuje na délku odpovídající nejdelší povolené délce paketu v Ethernetu – tedy na 1500 bajtů. Pokud přenášíte data rozsáhlou sítí jako je Internet, měli byste zvolit menší velikost MTU, abyste zajistili, že vaše datagramy nebu-

dou cestou dále děleny na menší kusy procesem zvaným *IP fragmentace*[27]. Jádro je schopno zjistit nejmenší MTU po trase a automaticky nakonfigurovat TCP spojení tak, aby tuto velikost používalo. Implicitně je toto chování zapnuto. Pokud na následující volbu odpovíte `y`, tato funkce se vypne.

Pokud budete chtít nastavit menší velikost datových bloků pro přenos na určité počítače (které jsou například připojeny linkou SLIP), můžete to udělat volbou **mss** příkazu **route**, o kterém budeme stručně mluvit na konci této kapitoly:

```
[ ] IP: Disable Path MTU Discovery (normally enabled)
```

Bezpečnostní funkce

Protokol IP podporuje funkci nazvanou *zdrojové směrování*. Tato funkce vám umožňuje definovat trasu pro přenos datagramu a zakódovat ji přímo v přenášeném datagramu. Bývalo to užitečné v dobách, než byly běžně rozšířeny směrovací protokoly jako RIP nebo OSPF. V současné době je tato funkce chápána spíše jako bezpečnostní ohrožení, protože umožňuje schopnému útočníkovi obejít některé typy firewallových ochran tím, že se přenos nebude řídit směrovací tabulkou brány. Za normálních okolností proto chcete datagramy se zdrojovým směrováním odfiltrovat, a proto je tato volba standardně zapnuta:

```
[*] IP: Drop source routed frames
```

Podpora Novellu

Tato volba zapíná podporu protokolu IPX, tedy transportního protokolu používaného v sítích Novell. Linux může klidně fungovat jako směrovač protokolu IPX a tato podpora je užitečná v sítích, kde používáte souborové servery Novell. Podpora protokolu NCP rovněž vyžaduje zapnutí podpory IPX, takže pokud chcete přistupovat k souborovým systém Novellových serverů, musíte tuto volbu zapnout. (O protokolu IPX a systému NCP budeme hovořit v kapitole 15.)

```
<*> The IPX protocol
```

Podpora rádia

Následující tři volby zapínají podporu tří protokolů podporovaných Linuxem a používaných v amatérských radiových sítích. Jsou to AX.25, NetRom a Rose (v této příručce se jim nevěnujeme, jsou ale popsány v dokumentu AX25-HOWTO):

```
<*> Amateur Radio AX.25 Level 2
<*> Amateur Radio NET/ROM
<*> Amateur Radio X.25 PLP (Rose)
```

Linux podporuje ještě další ovladač, takzvaný *dummy driver*. Na začátku části věnované volbě síťových ovladačů se objevuje následující otázka:

```
<*> Dummy net driver support
```

Ovladač dummy toho dohromady moc nedělá, je ale velmi užitečný na samostatných počítačích nebo počítačích s PPP/SLIP připojením. Jedná se vlastně o maškarádu lokálního ovladače. Pokud

27 Nezapomínejte, že protokol IP může být přenášen celou řadou různých typů sítí, a ne všechny podporují stejně velké pakety jako Ethernet.

máte počítač s připojením PPP/SLIP a žádným dalším rozhraním, budete zřejmě potřebovat rozhraní, které bude mít trvale přidělenu vaši IP adresu. Podrobněji o tom hovoříme v části *Rozhraní dummy* v kapitole 5. Dnes můžete stejného efektu dosáhnout pomocí IP aliasu s tím, že vám přidělenou IP adresu nastavíte jako alias lokálního rozhraní.

Prohlídka síťových zařízení v Linuxu

Jádro Linuxu podporuje množství ovladačů hardwaru pro různé typy zařízení. Tato stať obsahuje krátký přehled dostupných rodin ovladačů a názvů rozhraní, které ovladače používají.

V Linuxu existuje množství standardních názvů rozhraní, jejichž výčet následuje. Většina ovladačů podporuje více než jedno rozhraní. V takovém případě jsou rozhraní číslována, například *eth0*, *eth1* a podobně.

`lo`

Lokální zpětnovazebné rozhraní. Používá se pro testovací účely a pro dvojice síťových aplikací. Pracuje jako uzavřený obvod, kde všechny datagramy, které jsou na něj poslány, jsou okamžitě vráceny síťové vrstvě počítače. V jádru existuje vždy jedno toto lokální rozhraní a mít více těchto zařízení nemá smysl.

`eth0, eth1, ...`

Rozhraní ethernetových karet. Používá je většina standardních ethernetových karet včetně většiny ethernetových karet na paralelním portu.

`tr0, tr1, ...`

Rozhraní Token Ringových karet. Používá je většina Token Ringových karet včetně karet jiných výrobců než IBM.

`sl0, sl1, ...`

Rozhraní SLIP. Tato rozhraní se přidělují sériovým linkám v tom pořadí, jak jsou protokolem SLIP alokovány.

`ppp0, ppp1, ...`

Rozhraní PPP. Stejně jako rozhraní SLIP je i rozhraní PPP přiděleno sériové lince v okamžiku, kdy je sériová linka přepnuta do režimu PPP.

`plip0, plip1, ...`

Rozhraní PLIP. Protokol PLIP dopravuje datagramy po paralelních linkách. Tato rozhraní jsou přidělována ovladačem PLIP při procesu zavádění systému a jsou mapována na paralelní porty. V jádrech *2.0.x* je přímý vztah mezi názvem zařízení a číslem paralelního portu, u novějších jader jsou ovladače přidělovány sekvenčně jako u protokolů SLIP a PPP.

`ax0, ax1, ...`

Rozhraní AX.25. AX.25 je primární protokol používaný v radioamatérských sítích. Rozhraní protokolu AX.25 jsou alokována a mapována podobně jako zařízení protokolu SLIP.

Existuje celá řada dalších rozhraní pro jiné síťové ovladače. Zde jsme si uvedli pouze ty nejběžnější.

V následujícím textu se budeme podrobně věnovat použití výše popsaných ovladačů. O konfiguraci ostatních hovoří dokument Networking-HOWTO, o konfiguraci radioamatérských zařízení pak dokument AX25-HOWTO.

Instalace Ethernetu

Aktuální síťový kód Linuxu podporuje širokou skupinu ethernetových karet. Většinu ovladačů napsal Donald Becker, který je autorem rodiny ovladačů pro karty založené na čipu National Semiconductor 8390; tato rodina je známá pod názvem Becker Series Drivers. Celá řada dalších programátorů napsala další ovladače, takže dnes prakticky neexistuje ethernetová karta, kterou by Linux nepodporoval. Seznam podporovaných karet se navíc stále rozrůstá, takže pokud není vámi používaná karta podporována právě teď, může se to brzy změnit.

V minulosti jsme se pokusili o uvedení seznamu všech podporovaných ethernetových karet, dnes by do zabralo příliš mnoho práce a času. Naštěstí Paul Gortmaker udržuje dokument Ethernet-HOWTO, který obsahuje seznam všech podporovaných karet a uvádí u jednotlivých karet užitečné informace o tom, jak je v Linuxu rozběhnout[28]. Tento dokument se každý měsíc objevuje ve skupině *comp.os.linux.answers* a můžete jej také získat z některého zrcadla LDP.

I když jste si jisti že víte, jak konkrétní ethernetovou kartu nainstalovat, i tak je užitečné se do dokumentu Ethernet-HOWTO podívat, co se v něm o dané kartě říká. Dozvíte se informace nad standardní konfigurační údaje. Můžete si například ušetřit spoustu trápení, když budete vědět, že některé ethernetové karty používající DMA používají standardně stejný DMA kanál jako SCSI řadič Adaptec 1542. Pokud kanál některého z těchto zařízení nezměníte, zjistíte, že vám síťová karta volně zapisuje přijímaná data na náhodná místa disku.

Chcete-li použít kteroukoliv podporovanou kartu, můžete využít předkompilované jádro kterékoliv z hlavních distribucí Linuxu. Obvykle obsahují moduly pro všechna podporovaná zařízení a instalační proces vám umožní zvolit, které ovladače chcete nahrát. Z dlouhodobého hlediska je ale lepší sestavit si vlastní jádro a přeložit pouze ty ovladače, které budete skutečně potřebovat. Ušetříte tak diskový prostor i paměť.

Automatická detekce Ethernetu

Řada ovladačů ethernetových karet je dostatečně chytrých na to, aby dokázaly najít vaši kartu. Díky tomu nemusíte jádru říkat, jak máte kartu nastavenu. V dokumentu Ethernet-HOWTO je uvedeno, zda jednotlivé ovladače provádějí automatickou detekci a v jakém pořadí prohledávají vstupně-výstupní adresy.

Automatická detekce má tři omezení. Za prvé jádro nemusí korektně rozpoznat všechny karty. To platí zejména pro některé levnější klony běžných karet. Druhým problémem je, že jádro nebude automaticky detekovat výskyt více než jedné karty, pokud mu to explicitně nenařídíte. Toto chování je záměrné, protože se předpokládá, že budete osobně chtít nastavit, které rozhraní bude které kartě přiřazeno. Nejspolehlivější metoda jak toho dosáhnout je nastavit konfiguraci karet ručně. A konečně, ovladač nemusí testovat tu adresu, na níž máte kartu nastavenu. Obecně platí, že ovladače automaticky zkoušejí všechny adresy, pro něž lze dané zařízení nakonfigurovat, někdy však některé adresy vynechávají, aby se předešlo hardwarovým konfliktům s jinými zařízeními, která mohou pracovat na stejných adresách.

Karty na sběrnici PCI by měly být detekovány spolehlivě. Pokud ale používáte více než jednu kartu nebo pokud se automatická detekce karty nezdaří, je třeba jádru explicitně nastavit bázovou adresu a název karty.

28 Paula můžete kontaktovat na adrese *gpg109@rsphy1.anu.edu.au*.

Při zavádění systému můžete uvést parametry a informace, které si může přečíst kterákoliv komponenta jádra. Tento mechanismus vám dovoluje předat jádru informace, které ethernetový ovladač použije k nalezení ethernetové karty, aniž by prováděl automatickou detekci.

Pokud pro zavádění systému používáte program `lilo`, můžete předat jádru parametry pomocí volby append v souboru `lilo.conf`. Chcete-li jádro informovat o ethernetovém zařízení, předejte mu následující parametr:

```
ether=irq,base_addr,[param1,][param2,]name
```

První čtyři parametry jsou číselné, zatímco poslední představuje název zařízení. Hodnoty *irq*, *base_addr* a *name* jsou povinné, dvě hodnoty *param* jsou nepovinné. Kterákoliv z číselných hodnot může být nulová, v takovém případě jádro použije autodetekci.

První parametr nastavuje IRQ kanál, který bude zařízení přidělen. Jádro se implicitně pokusí automaticky detekovat IRQ kanál daného zařízení. Ovladač 3c503 například disponuje speciální vlastností, která vybere volný IRQ kanál z kanálů 5, 9, 3 a 4 a nakonfiguruje kartu tak, aby tento kanál používala. Parametr *base_addr* udává bázovou vstupně-výstupní adresu karty. Zadáte-li hodnotu nula, jádro se pokusí hodnotu zjistit automaticky.

Zbývající dva parametry mohou různé typy ovladačů používat odlišným způsobem. U karet sdílejících paměť, jako například WD80x3, určují počáteční a koncovou adresu oblasti sdílené paměti. Ostatní karty používají zpravidla parametr *param1* k nastavení úrovně zobrazení ladicích informací. Hodnoty od 1 do 7 ukazují zvyšující se úroveň rozsahu výpisů, zatímco hodnota 8 ji vypíná; 0 udává implicitní volbu. Ovladač 3c503 používá *param2* k výběru vnitřního transceiveru (implicitně) nebo vnějšího transceiveru (hodnota 1). Nulová hodnota způsobí použití BNC konektoru na kartě; hodnota 1 vyvolá použití portu AUI. Pokud nepotřebujete nic zvláštního nastavit, hodnoty *param* nemusí být vůbec uvedeny.

Pokud máte dvě ethernetové karty, může Linux nechat automaticky detekovat jednu z nich a parametry druhé karty mu pak předáte pomocí **lilo**. Pokud se ale rozhodnete pro automatickou detekci jedné a ruční zadání druhé karty, je třeba zajistit, aby jádro náhodou nenašlo nejprve druhou kartu, protože v takovém případě by první vůbec nefungovala. Lze tak učinit pomocí volby *reserve* v programu **lilo**, která explicitně řekne jádru, aby nedetekovalo ve vstupně-výstupním prostoru obsazeném druhou kartou. Chcete-li například, aby Linux nainstaloval druhou ethernetovou kartu na adrese `0x300` jako `eth1`, musíte jádru předat následující parametry:

```
reserve=0x300,32 ether=0,0x300,eth1
```

Volba *reserve* zajistí, že žádný ovladač nebude při detekci „svého“ zařízení prohlížet vstupně-výstupní oblast této druhé karty. Pomocí parametrů jádra můžete potlačit automatickou detekci i pro první kartu `eth0`:

```
reserve=0x340,32 ether=0,0x340,eth0
```

Automatickou detekci můžete vypnout úplně. Dělá se to například v případě, kdy nechcete aby jádro automaticky detekovalo kartu, kterou jste dočasně odstranili. Vypnutí autodetekce se dosáhne tím, že parametru *base_addr* přiřadíte hodnotu –1:

```
ether=0,-1,eth0
```

Chcete-li parametry jádru předat v době spouštění systému, zadáváte je na výzvu „`boot:`“ programu **lilo**. Aby program tuto výzvu zobrazil, musíte v době spouštění systému stisknout některou z kláves Control, Alt nebo Shift. Pokud na tuto výzvu stisknete tabulátor, objeví se seznam jader,

která můžete nabootovat. Chcete-li nabootovat nějaké jádro s nějakými parametry, zadejte jméno jádra, mezeru a požadované parametry. Po stisknutí klávesy Enter se nahraje příslušné jádro a předají se mu zadané parametry.

Pokud chcete, aby se nastavené parametry automaticky uplatnily při každém spuštění systému, zadejte je v souboru `/etc/lilo.conf` pomocí klíčového slova `append=`. Příklad může vypadat takto:

```
boot=/dev/hda
root=/dev/hda2
install=/boot/boot.b
map=/boot/map
vga=normal
delay=20
append="ether=10,300,eth0"

image=/boot/vmlinuz-2.2.14
label=2.2.14
read-only
```

Po úpravě souboru `lilo.conf` musíte znovu spustit příkaz **lilo**, aby se změny uplatnily.

Ovladač PLIP

Protokol PLIP znamená *IP-protokol po paralelní lince* (*Parallel Line IP*) a představuje levnou alternativu sítě, pokud chcete propojit pouze dva počítače. Používá paralelní port společně se speciálním kabelem a dosahuje rychlosti od 10 KB/s do 20 KB/s.

Protokol PLIP původně vyvinula firma Crynwr, Inc. Jeho původní návrh byl poměrně prostý: po dlouhou dobu byly u počítačů PC paralelní porty používány pouze jako jednosměrné porty pro tiskárny; to znamená, že osm datových linek tohoto portu mohlo být použito pouze pro posílání informací z počítače do periferního zařízení, přenos opačným směrem nebyl možný. Návrh protokolu PLIP společnosti Crynwr se s tím vypořádal tak, že pět stavových linek používá pro vstup dat, což sice omezuje přenos pouze na poloviny bajtů najednou, nicméně umožňuje to obousměrnou komunikaci. Tento pracovní režim se nazývá režim 0 protokolu PLIP. Dnešní paralelní porty zvládají úplnou obousměrnou osmibitovou komunikaci a protokol PLIP byl rozšířen o režim 1, který toho využívá.

Jádra Linuxu do verze 2.0 podporovala pouze PLIP režim 0, a s rozšířením výskytu obousměrných paralelních portů byl vyvinut patch pro jádro 2.0, které tento port využívá. Od jádra 2.2 je již režim 1 podporován standardně[29]. Na rozdíl od předchozích verzí kódu PLIP je nynější ovladač kompatibilní s implementací společnosti Crynwr i s ovladačem protokolu PLIP v NCSA[30] **telnetu**. K propojení dvou počítačů za pomoci protokolu PLIP potřebujete speciální kabel, který seženete v některých obchodech pod označením kabel „Null Printer“ nebo „Turbo Laplink“. Poměrně jednoduše si ale můžete vyrobit kabel vlastní. Příloha A ukazuje, jak na to.

Linuxový ovladač protokolu PLIP je výsledkem práce obrovského množství lidí. Momentálně ho spravuje Niibe Yutaka[31]. Pokud je ovladač protokolu PLIP vestavěn do jádra, nastaví následující síťové rozhraní pro každý dostupný paralelní port, přičemž rozhraní `plip0` bude odpovídat para-

29 Patch podporující obousměrné porty v jádře 2.0 je dostupný na adrese *http://www.cyberelk.demon.co.uk/parport.html*

30 NCSA **telnet** je oblíbený program pro DOS, který používá TCP/IP přes Ethernet nebo PLIP a podporuje služby telnet a FTP.

31 Niibeho můžete kontaktovat na adrese *gniibe@mri.co.jp*.

lelnímu portu `lp0`, rozhraní `plip1` paralelnímu portu `lp1` a tak dále. Mapování rozhraní portům se liší v jádrech verze 2.0 a 2.2. V jádrech 2.0 bylo mapování napevno zakódováno v souboru `drivers/net/Space.d` zdrojového kódu jádra. Standardní mapování v tomto souboru vypadá takto:

Rozhraní	Port	IRQ
`plip0`	`0x3BC`	7
`plip1`	`0x378`	7
`plip2`	`0x278`	5

Pokud máte porty tiskárny nastaveny jinak, musíte tyto hodnoty v souboru `drivers/net/Space.c` upravit a znovu sestavit jádro.

V jádrech 2.2 využívá ovladač PLIP sdílení paralelního portu mechanismem „parport" vyvinutým Philipem Blundellem[32]. Nový ovladač alokuje jména síťových zařízení PLIP postupně stejně jako u ovladačů ethernetových karet nebo PPP. První vytvořené PLIP zařízení tedy bude `plip0`, druhé bude `plip1` a tak dále. Fyzická paralelní rozhraní se rovněž alokují postupně. Standardně se ovladač paralelního portu snaží nalézt fyzické paralelní porty autodetekcí a očísluje je v tom pořadí, v jakém je nalezne. Lepší je jádru explicitně sdělit vstupně-výstupní parametry jednotlivých fyzických portů. Můžete to provést buď předáním parametrů modulu `parport_pc.o` při jeho nahrání, nebo pokud máte tento ovladač napevno vestavěn v jádře, můžete programem **lilo** předat jádru příslušné parametry při spouštění systému. Nastavení přerušení kteréhokoliv zařízení je možné později změnit zapsáním nové hodnoty přerušení do odpovídajícího souboru `/proc/parport*/irq`.

Konfigurace fyzických vstupně-výstupních parametrů v jádře 2.2 při nahrávání modulu je poměrně jednoduchá. Chcete-li ovladači říct, že máte dva paralelní porty na adresách `0x278` a `0x378` s přerušeními `5` a `7`, nahrajete modul s následujícími parametry:

```
modprobe parport_pc io=0x278,0x378 irq=5,7
```

Stejné parametry předávané jádru pro vestavěný ovladač budou vypadat takto:

```
parport=0x278,5 parport=0x378,7
```

Pomocí klíčového slova *append* můžete tyto parametry předávat jádru automaticky při každém spouštění systému.

Jakmile se inicializuje ovladač PLIP, ať už při spouštění systému, je-li vestavěný do jádra, nebo po nahrání modulu `plip.o`, bude každému paralelnímu rozhraní přiřazeno odpovídající zařízení `plip`. Tedy `plip0` bude přiřazeno prvnímu paralelnímu rozhraní, `plip1` druhému a tak dále. Tato automatická nastavení můžete změnit ručně dalšími parametry jádra. Pokud budete chtít například přiřadit `parport0` síťovému zařízení `plip1` a `parport1` síťovému zařízení `plip0`, použijete následující parametry jádra:

```
plip=parport1 plip=parport0
```

Nicméně toto mapování neznamená, že nemůžete používat paralelní porty obvyklým způsobem. Ovladač protokolu PLIP přistupuje k paralelním portům pouze v případě, že je k němu nakonfigurováno odpovídající rozhraní.

32 Philipa můžete kontaktovat na adrese *Philip.Blundell@pobox.com*.

Ovladače SLIP a PPP

Protokoly PPP (Point-to-Point Protocol) a SLIP (Serial Line IP) se hojně využívají k přenášení IP paketů po sériové lince. Množství firem nabízí přístup k Internetu prostřednictvím telefonického připojení protokoly SLIP nebo PPP. Tímto způsobem se obvykle připojují soukromé osoby, pro něž je jiný typ připojení cenově nedostupný.

Abyste mohli používat ovladače SLIP nebo PPP, nejsou nutné žádné hardwarové úpravy; stačí použít libovolný sériový port. Protože konfigurace sériových portů není specifickou problematikou sítí na bázi TCP/IP, bude jim věnována samostatná kapitola. Více informací o PPP tak naleznete v kapitole 8 a o SLIP v kapitole 7.

Další typy sítí

Řada jiných sítí se konfiguruje podobně jako Ethernet. Parametry předávané zaváděným modulům budou jiné a některé ovladače nepodporují více karet než jednu, nicméně všechno ostatní je prakticky stejné. Dokumentaci k různým kartám obecně najdete v adresáři `/usr/src/linux/Documentation/networking/` ve zdrojovém kódu jádra.

Konfigurace sériových zařízení

Internet se rozrůstá neuvěřitelným tempem. Velká část tohoto růstu jde na vrub uživatelům, kteří si nemohou dovolit vysokorychlostní trvalá připojení a kteří používají protokoly jako SLIP, PPP nebo UUCP k telefonickému spojení s poskytovatelem internetového připojení, od kterého si tak stáhnou svou denní dávku konferencí a pošty.

Tato kapitola má pomoci všem lidem, kteří jsou odkázáni pouze na modemy. Nebudeme se pouštět do popisů mechanismů jak nakonfigurovat modem (o tom vám podstatně více než my řekne manuál, který k modemu dostanete), nicméně pokryjeme většinu linuxově zaměřené problematiky týkající se práce se zařízeními, která komunikují sériovým portem. Budeme hovořit o programech pro sériovou komunikaci, o vytvoření souboru sériového zařízení, o sériovém hardware a o konfiguraci sériových zařízení příkazy **setserial** a **stty**. Řada dalších témat je uvedena v dokumentu Serial-HOWTO Davida Lawyera[33].

Komunikační software pro modemové linky

V Linuxu je k dispozici mnoho komunikačních balíků. Spoustu z nich tvoří *terminálové programy*, které umožňují uživateli propojení s jiným počítačem a které se tváří, jakoby uživatel seděl před jednoduchým terminálem. Tradičním unixovým terminálovým programem je **kermit**. Dnes už však je značně zastaralý a pravděpodobně byste jej považovali za příliš složitý na používání. Dnes jsou dostupné mnohem komfortnější programy, které podporují různé další funkce jako adresář s telefonními čísly, skriptové jazyky pro automatické volání a připojování ke vzdáleným počítačovým systémům a řada různých souborových komunikačních protokolů. Jeden z nich se nazývá **minicom** a byl vyvinut podle některých nejpopulárnějších dosových terminálových programů. Spokojeni budou i uživatelé X11, plně vybaveným komunikačním programem na bázi X11 je například **seyon**.

Terminálové programy nicméně nejsou jediné dostupné programy pro sériovou komunikaci. Existují jiné programy, které vám dovolují připojit se ke vzdálenému počítači a stáhnout zprávy a poštu najednou, přičemž jejich přečtení a zodpovězení provedete později. Tím můžete ušetřit hodně času a bude to pro vás výhodné zejména pokud žijete někde, kde jsou i místní hovory časově tarifikovány. Čtení a zodpovězení provedete offline a poté se znovu připojíte a všechny odpovědi najednou odešlete. Toto řešení samozřejmě vyžaduje trochu více diskového prostoru, protože všechny zprávy musíte mít před přečtením uloženy na disku, při současných cenách pevných disků to ale nepředstavuje zásadní nevýhodu.

33 Davida můžete kontaktovat na adrese *bf347@lafn.org*.

Typickým představitelem tohoto typu softwaru je UUCP. Jedná se o programový balík, který kopíruje data z jednoho hostitele na druhý a spouští programy na vzdáleném hostiteli. Často se používá pro přenos pošty nebo konferencí v soukromých sítích. Balík UUCP od Iana Taylora, který běží také v prostředí Linuxu, bude popsán v kapitole 16. Další neinteraktivní komunikační software se používá například v sítích Fidonet. Jsou k dispozici i aplikace přenesené na platformu Linuxu, například **ifmail**, i když si nemyslíme, že by je používalo hodně lidí.

Protokoly PPP a SLIP stojí někde mezi, protože umožňují jak interaktivní, tak i neinteraktivní použití. Mnoho lidí využívá protokol PPP nebo SLIP pro připojení ke školní síti nebo k nějakému komerčnímu poskytovateli. Mohou tak využívat služby protokolu FTP a jiných. Protokoly PPP a SLIP se ale také často používají k propojení několika lokálních sítí pevnými nebo občasnými linkami. Toto využití je však zajímavé pouze pokud použijeme zařízení ISDN nebo jiné rychlé zařízení.

Úvod k sériovým zařízením

Zařízení, pomocí nichž umožňuje jádro Unixu přístup k sériovým zařízením se typicky nazývají *tty* (vyslovuje se to stejně jako při hláskování v angličtině, tedy *tí-tí-uaj*). Je to zkratka názvu společnosti *Teletype*, která bývala v počátcích unixové éry jedním z hlavních výrobců terminálů. V současnosti se tento termín používá pro jakékoliv znakově orientované datové terminály. V této kapitole budeme tento termín používat výhradně k označení zařízení jádra, nikoliv fyzického terminálu.

Linux rozlišuje tři třídy tty zařízení: sériová zařízení, virtuální zařízení (k nimž můžete na lokální konzole přistupovat stiskem kláves Alt-F1 až Alt-F*nn*) a pseudoterminály (podobné obousměrnému potrubí, které používají aplikace jako je X11). První zmíněná třída je rovněž považována za tty zařízení, protože původní znakově orientované terminály byly k linuxovému počítači připojovány sériovým kabelem nebo telefonní linkou a modemem. Druhé dvě třídy jsou klasická tty zařízení, protože se z pohledu programátora chovají stejně.

Protokoly SLIP a PPP bývají velmi často implementovány přímo v jádře. Jádro nechápe *tty* zařízení jako síťové zařízení, se kterým byste mohli pracovat stejně jako s ethernetovou kartou například příkazem **ifconfig**. Chápe nicméně *tty* zařízení jako místa, k nimž lze síťová zařízení připojit. Provede se to tak, že jádro změní takzvanou „linkovou disciplínu“ tty zařízení. SLIP i PPP jsou linkové discplíny, které mohou být na tty zařízeních povoleny. Hlavní myšlenka je ta, že sériový ovladač obsluhuje příchozí data různě podle toho, pro jakou disciplinu je linka nakonfigurována. Při použití standardní discipliny ovladač jednoduše předává každý znak, který obdrží. Když je zvolena disciplina SLIP nebo PPP, ovladač místo toho čte bloky dat, přidá k nim speciální hlavičky, které umožní vzdálenému zařízení identifikovat blok dat v datovém proudu a tento nový blok odešle. Není nijak důležité, abychom tomu nyní rozuměli, o protokolech SLIP a PPP budeme hovořit v dalších kapitolách a stejně se to všechno děje automaticky.

Přístup k sériovým zařízením

Stejně jako ke všem unixovým zařízením, je i k sériovým zařízením přistupováno pomocí speciálních souborů zařízení, které se nacházejí v adresáři `/dev`. Existují dvě skupiny souborů zařízení, které se vztahují k ovladačům sériových zařízení, přičemž každé sériové zařízení má přiřazen jeden soubor z každé skupiny. Zařízení se bude chovat různě v závislosti na typu souboru, kterým toto zařízení otevřeme. Hned si tento rozdíl vysvětlíme, protože to může být užitečné pro snazší pochopení konfigurace a některých doporučení, s nimiž se můžete při práci se sériovými porty

potkat. Nicméně v praxi stejně budete používat pouze jeden z těchto typů a v budoucnu možná ten druhý úplně vymizí.

Důležitější ze zmíněných dvou tříd jsou zařízení s hlavním číslem 4 a soubory těchto speciálních zařízení jsou pojmenovány `ttyS0`, `ttyS1` a tak dále. Druhá skupina má hlavní číslo 5 a používá se pro vytáčení směrem z počítače; soubory zařízení se jmenují `cua0`, `cua1` a tak dále. V unixovém světě se obecně počítá od nuly, i když „normální lidi" mají tendenci počítat od jedničky. To přináší lehké zmatky, protože port `COM1:` je reprezentován zařízením `/dev/ttyS0`, port `COM2:` zařízením `/dev/ttyS1` a tak dále. Kdokoliv se trochu orientuje v problematice počítačů IBM PC ví, že porty `COM3:` a vyšší stejně nebyly nikdy pořádně standardizovány.

Zařízení *cua* (callout) byla vyvinuta kvůli odstranění problémů s konflikty na sériovém zařízení při práci s modemy, které musí obsluhovat příchozí i odchozí spojení. Bohužel vytvořily své vlastní nové problémy a dnes se již prakticky nepoužívají. Podívejme se nyní, v čem problém spočívá.

Linux, stejně jako Unix, umožňuje každému zařízení nebo souboru, aby jej otevřelo více procesů současně. U tty zařízení to typicky není dobré, protože oba procesy se budou prakticky trvale rušit. Naštěstí byl vyvinut mechanismus umožňující, aby proces mohl před pokusem o otevření tty zařízení ověřit, zda je nemá otevřeno už jiný proces. Tento mechanismus se označuje jako *zámkové soubory*. Celá myšlenka spočívá v tom, že chce-li proces otevřít nějaké tty zařízení, na určitém místě hledá soubor pojmenovaný stejně jako zařízení, které chce otevřít. Pokud tento soubor neexistuje, proces jej vytvoří a otevře zařízení. Pokud by soubor existoval, proces bude předpokládat, že zařízení již bylo otevřeno jiným procesem a podle toho se zachová. Posledním chytrým trikem, který zajišťuje chod mechanismu zamykání prostřednictvím souborů je to, že proces, který zámkový soubor vytvoří, do něj zapíše svůj vlastní identifikátor procesu (pid). Budeme o tom hovořit za chvíli.

Mechanismus souborových zámků bezvadně funguje v situaci, kdy je přesně definováno umístění těchto souborů a kdy všechny procesy vědí, kde je mají hledat. To ovšem nebyla vždycky pravda. Neplatilo to do doby, dokud standard Linux Filesystem Standard nedefinoval pevné umístění těchto souborů. Jednu dobu existovaly přinejmenším čtyři (ne-li více) míst, kde různé programy tyto soubory ukládaly: `/usr/spool/locks`, `/var/spool/locks`, `/var/lock/` a `/usr/lock`. Tato nejednotnost vedla k chaosu. Programy vytvářely zámkové soubory téhož zařízení na různých místech, což bylo stejné, jako kdyby tento mechanismus nepoužívaly vůbec.

Jako řešení tohoto problému byla vytvořena zařízení `cua`. Namísto aby se při řešení sporů o sériovou linku spoléhalo na zámkové soubory, rozhodlo se, že tyto spory bude velmi jednoduše řešit přímo jádro. Pokud bylo zařízení `ttyS` otevřeno, pokus o otevření zařízení `cua` vedl k chybě, podle níž mohl volající proces poznat, že zařízení již je otevřeno. Pokud bylo otevřeno zařízení `cua` a došlo k pokusu o otevření zařízení `ttyS`, žádost byla zablokována – to znamená, že byla umístěna do fronty a čekala do doby, než bude zařízení `cua` uvolněno. Pracovalo to poměrně dobře, pokud jste používali jeden modem, kterým jste obsluhovali příchozí volání a čas od času jste chtěli tímto zařízením provést odchozí volání. Nefunguje to ale tehdy, pokud máte více programů, které chtějí všechny realizovat odchozí volání jedním zařízením. Jediná možnost řešení tohoto problému – používat zámkové soubory!

Naštěstí standard Linux Filesystem Standard nadefinoval umístění zámkových souborů do adresáře `/var/lock` a stanovil konvenci, že například zámkový soubor zařízení `ttyS1` bude pojmenován `LCK..ttyS1`. Do stejného adresáře se ukládají i zámkové soubory zařízení `cua`, nicméně použití tohoto typu zařízení se dnes nedoporučuje.

Zařízení `cua` budou pravděpodobně ještě nějakou dobu existovat kvůli zpětné kompatibilitě, postupem času však vymizí. Pokud nevíte, které zařízení použít, použijte `ttyS` a ověřte si, že používáte Linux kompatibilní se standardem FSSTND nebo že alespoň všechny programy pracující se

sériovým zařízením používají stejné umístění zámkových souborů. Většina programů, které se sériovou linkou pracují, obsahují parametr pro sestavení, kterým můžete umístění zámkových souborů definovat. Vesměs se jedná o proměnnou `LOCKDIR` v souboru `Makefile` nebo v konfiguračním hlavičkovém souboru. Pokud program překládáte, je dobré tento parametr nastavit tak, aby byl kompatibilní s FSSTND. Pokud používáte již přeložený program a nejste si jisti, kam zámky ukládá, zkuste to zjistit následujícím příkazem:

```
strings binaryfile | grep lock
```

Pokud nalezené umístění neodpovídá tomu, co používá zbytek systému, můžete zkusit vytvořit symbolický odkaz z adresáře používaného tímto programem na `/var/lock`. Není to moc pěkné, ale funguje to.

Soubory sériových zařízení

Vedlejší čísla zařízení jsou pro oba typy stejná. Máte-li modem připojen na jednom z portů COM1: až COM 4:, bude vedlejší číslo rovno číslu COM portu plus 63. Pokud používáte speciální sériový hardware, například kartu, která podporuje několik sériových linek, budete mu muset zřejmě vytvořit vlastní soubor zařízení, pravděpodobně nebude pracovat se standardním souborem. Příslušné podrobnosti byste měli najít v dokumentu Serial-HOWTO.

Budeme předpokládat, že váš modem je připojen na port COM2:. Jeho vedlejší číslo tak bude mít hodnotu 65 a hlavní číslo bude při normálním použití rovno 4. Mělo by existovat zařízení `ttyS1`, které bude mít tato čísla. Vypište si sériová tty zařízení z adresáře `/dev`. Pátý a šestý sloupec by měly obsahovat hlavní a vedlejší číslo:

```
$ ls -l /dev/ttyS*
0 crw-rw----   1 uucp     dialout    4,  64 Oct 13  1997 /dev/ttyS0
0 crw-rw----   1 uucp     dialout    4,  65 Jan 26 21:55 /dev/ttyS1
0 crw-rw----   1 uucp     dialout    4,  66 Oct 13  1997 /dev/ttyS2
0 crw-rw----   1 uucp     dialout    4,  67 Oct 13  1997 /dev/ttyS3
```

Jestliže neexistuje zařízení s hlavním číslem 4 a vedlejším číslem 65, musíte je vytvořit: přihlaste se jako superuživatel a napište:

```
# mknod -m 666 /dev/ttyS1 c 4 65
# chown uucp.dialout /dev/ttyS1
```

Různé distribuce Linuxu používají různé strategie pro to, kdo by měl sériové zařízení vlastnit. Někdy je vlastní uživatel *root*, jindy nějaký jiný uživatel, například *uucp*. Moderní distribuce používají skupinu uživatel s přístupem k sériovým zařízením a kdokoliv je má používat, přidá se do této skupiny.

Někteří lidé doporučují vytvořit soubor `/dev/modem`, který by sloužil jako symbolický odkaz na váš modem, takže si příležitostní uživatelé nemusí pamatovat poněkud neintuitivní název například `ttyS1`. Nemůžete pak ale v jednom programu používat soubor `modem` a ve druhém skutečný název souboru zařízení. Jejich zámkové soubory by pak měly různá jména a mechanismus zamykání by nefungoval.

Sériový hardware

Nejrozšířenějším standardem pro sériovou komunikaci ve světě PC je dnes RS-232. K přenosu jednotlivých bitů a synchronizaci využívá několik elektronických obvodů. Další linky lze využít k signalizaci výskytu nosné (kterou používají modemy) a k inicializačnímu handshakingu.

Hardwarový handshaking je nepovinný, je ale velice užitečný. Umožňuje oběma stanicím signalizovat, zda jsou schopny přijmout další data, nebo zda by měla druhá strana počkat, až přijímající strana dokončí zpracování přijatých dat. K tomuto účelu se používají linky „Clear to Send" (CTS) a „Ready to Send" (RTS), z jejichž názvů je odvozen hovorový název celého hardwarového handshakingu: „RTS/CTS". Druhým známým typem handshakingu je „XON/XOFF". Tento mechanismus používá dva speciální znaky, typicky Ctrl-S a Ctrl-Q, kterými se vzdálené straně signalizuje, že má zastavit nebo zahájit odesílání dat. Tato metoda je jednoduchá na implementaci a použitelná v hloupých terminálech, může způsobit problémy, pokud přenášíte binární data, protože tyto znaky můžete odeslat jako součást dat a nebudete je chtít považovat za řídící znaky. Navíc je toto řešení poněkud pomalejší než hardwarové řešení. Hardwarové řešení je čistší, rychlejší a pokud máte na výběr, doporučuje se je upřednostnit před mechanismem XON/XOFF.

V původních počítačích PC je rozhraní RS-232 obvykle řízeno čipem UART pojmenovaným 8250. Počítače zhruba od doby 486 používají UART pojmenovaný 16450. Ten je poněkud rychlejší než 8250. Většina počítačů Pentium obsahuje ještě novější verzi UART čipu, pojmenovanou 16550. Některá zařízení (typicky interní modemy firmy Rockwell) používají úplně jiný čip, který emuluje chování čipu 16550 a může být obsluhován stejně. Linux ve standardním ovladači sériového portu podporuje všechny tyto čipy[34].

Čip 16550 je oproti čipům 8250 a 16450 zásadně vylepšen, protože obsahuje vyrovnávací paměť FIFO o velikosti 16 bajtů. Čip 16550 vlastně představuje celou rodinu UART zařízení, do které patří 16550, 16550A a 16550AFN (později přejmenovaný na PC16550DN). Rozdíl spočívá v tom, zda vyrovnávací paměť opravdu funguje – což se s jistotou ví pouze o čipu 16550AFN. Existoval i čip NS16550, ale jeho vyrovnávací paměť nikdy nefungovala.

Použití konfiguračních nástrojů

Nyní se podívejme na dva nástroje, které se pro konfiguraci sériových linek nejčastěji používají. Jsou to programy **setserial** a **stty**.

Příkaz setserial

Jádro se velmi snaží o správné zjištění konfigurace sériových zařízení, nicméně různé rozdíly v jejich vlastnostech znemožňují dosažení stoprocentní spolehlivosti. Dobrým příkladem jsou například problémy s interními modemy, o kterých jsme už hovořili. Jimi používaný čip UART má 16bajtový buffer, nicméně tváří se jako čip 16450 – pokud tedy jádru explicitně neřekneme, že jde o čip 16550, nebude tento buffer využívat. Dalším příkladem mohou být hloupé čtyřportové karty, které sdílejí jedno přerušení pro všechna sériová zařízení. Musíme jádru přesně říct, která přerušení se používají a že jsou případně sdílená.

Program **setserial** slouží k nastavení sériových portů za běhu. Tento příkaz se typicky spouští v době spouštění systému skriptem, který se na některých distribucích jmenuje `0setserial`, na

34 Všimněte si, že se vůbec nezmiňujeme o zařízení WinModem(TM)! Tato zařízení jsou velice jednoduchá a k provedení veškeré potřebné práce plně spoléhají na procesor a nikoliv na specializovaný hardware. Pokud si kupujete modem, vřele vám doporučujeme *nekupovat* nic takového, kupte si opravdový modem. Podpora zařízení WinModem existuje i v Linuxu, to ale neznamená, že by je měl někdo používat.

jiných `rc.serial`. Tento skript zodpovídá za nastavení sériových zařízení tak, aby byl správně podchycen všechen nestandardní nebo neobvyklý hardware v počítači.

Obecná syntaxe příkazu **setserial** je

```
setserial device [parametry]
```

kde parametr `device` představuje některé sériové zařízení, například *ttyS0*.

Program **setserial** používá celou řadu parametrů. Nejběžnější z nich jsou shrnuty v tabulce 4.1. Informace o dalších parametrech najdete na manuálových stránkách programu **setserial**.

Tabulka 4.1 – Řádkové parametry programu setserial

Parametr	Popis
`port port_number`	Udává adresu vstupně-výstupního portu sériového zařízení. Hodnota se zadává hexadecimálně, například 0x2f8.
`irq num`	Udává číslo použitého přerušení
`uart uart_type`	Specifikuje UART čip zařízení. Možné hodnoty jsou 16450, 16550 a podobně. Nastavení této hodnoty na *none* vypne dané sériové zařízení.
`fourport`	Tento parametr jádru říká, že daný port je jedním z portů karty AST Fourport.
`spd_hi`	Naprogramuje UART tak, aby pracoval rychlostí 57,6 kb/s pokud proces požaduje rychlost 38,4 kb/s.
`spd_vhi`	Naprogramuje UART tak, aby pracoval rychlostí 115,2 kb/s pokud proces požaduje rychlost 38,4 kb/s.
`spd_normal`	Naprogramuje UART na použití standardní rychlosti 38,4 kb/s. Tento parametr se používá k odstranění účinku parametrů spd_hi nebo spd_vhi na daném zařízení.
`auto_irq`	Tento parametr způsobí, že se jádro pokusí automaticky detekovat IRQ daného zařízení. Výsledek nemusí být úplně spolehlivý, takže se na tuto operaci dívejme spíše jako na „hádání" přerušení. Pokud znáte číslo přerušení, které zařízení používá, měli byste je zadat pomocí parametru irq.
`autoconfig`	Tento parametr může být zadán jen společně s parametrem port. Způsobí, že jádro bude automaticky detekovat typ UART na daném portu. Pokud je zadán i parametr auto_irq, bude se automaticky detekovat i přerušení.
`skip_test`	Tento parametr jádru říká, aby se nepokoušelo o automatickou detekci UART čipu. Používá se tehdy, pokud jádro stejně čip správně nedetekuje.

Typický a jednoduchý `rc` soubor pro nastavení sériových portů při spouštění systému vypadá tak, jako na příkladu 4.1. Většina distribucí Linuxu obsahuje něco lepšího, než je jen tento skript:

Příklad 4.1 – Příklad konfiguračního skriptu rc.serial

```
# /etc/rc.serial - konfigurační skript sériové linky.
#
# Konfigurace sériových zařízení
/sbin/setserial /dev/ttyS0 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS1 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS2 auto_irq skip_test autoconfig
/sbin/setserial /dev/ttyS3 auto_irq skip_test autoconfig
#
```

```
# Zobrazení konfigurace sériových zařízení
/sbin/setserial -bg /dev/ttyS*
```

Parametr `-bg /dev/ttyS*` v posledním příkazu způsobí vytištění pěkně formátovaného přehledu hardwarové konfigurace všech aktivních sériových zařízení. Výstup bude vypadat podobně jako v příkladu 4.2:

Příklad 4.2 – Výstup příkazu setserial `-bg /dev/ttyS*`

```
/dev/ttyS0 at 0x03f8 (irq = 4) is a 16550A
/dev/ttyS1 at 0x02f8 (irq = 3) is a 16550A
```

Příkaz stty

Název příkazu **stty** zřejmě znamená „set tty“, nicméně příkazem **stty** je možné také zobrazit konfiguraci terminálů. Program **stty** umožňuje nastavení ještě rozsáhlejšího souboru vlastností než program **setserial**. Za chvilku si popíšeme nejdůležitější z nich. Zbytek pak najdete na manuálových stránkách programu **stty**.

Příkaz **stty** se nejčastěji používá ke konfiguraci parametrů terminálů, například zda mají být lokálně zobrazovány odesílané znaky nebo která klávesa má generovat přerušovací signál. Říkali jsme si už, že sériová zařízení jsou zařízení typu tty, takže program **stty** je použitelný i pro ně.

Jedno z nejdůležitějších použití programu **stty** pro sériová zařízení spočívá v povolení hardwarového handshakingu. O hardwarovém handshakingu jsme už stručně mluvili. Standardní nastavení sériových zařízení má hardwarový handshaking vypnut. Toto nastavení umožňuje použití třívodičových sériových kabelů, které nepodporují signály potřebné pro hardwarový handshaking, takže pokud bychom jej standardně zapnuli, nebyli bychom schopni odeslat žádný znak a toto nastavení změnit.

Překvapivě hardwarový handshaking nezapínají ani některá sériová komunikační zařízení, takže pokud váš modem hardwarový handshaking podporuje, nakonfigurujte jej tak, aby jej používal (jak to udělat najdete v návodu k modemu) a stejně nakonfigurujte i sériové zařízení. Příkaz **stty** má příznak `crtscts`, který zapíná hardwarový handshaking daného zařízení, takže jej budete muset uvést. Tento příkaz se nejlépe spouští ze souboru `rc.serial` (nebo jeho ekvivalentu) v době spouštění systému například tak, jak to ukazuje příklad 4.3.

Příklad 4.3 – Příklad skriptu rc.serial s programem stty

```
#
stty crtscts < /dev/ttyS0
stty crtscts < /dev/ttyS1
stty crtscts < /dev/ttyS2
stty crtscts < /dev/ttyS3
#
```

Příkaz **stty** standardně pracuje s aktuálním terminálem, pokud ale použijeme přesměrování vstupu („<“), můžeme příkazem **stty** nastavit jakékoliv zařízení. Běžnou chybou je zapomenutí, zda se má použít symbol „<“ nebo „>“, moderní verze příkazu **stty** používají jasnější syntaxi. Při použití nové syntaxe bude stejný příklad vypadat tak, jako příklad 4.4:

Příklad 4.4 – Příklad skriptu rc.serial s programem stty a moderní syntaxí

```
#
stty crtscts -F /dev/ttyS0
stty crtscts -F /dev/ttyS1
stty crtscts -F /dev/ttyS2
stty crtscts -F /dev/ttyS3
#
```

Zmínili jsme se už, že příkazem **stty** můžete zobrazit konfigurační parametry tty zařízení. Pokud chcete zobrazit všechna nastavení daného tty zařízení, použijte příkaz

```
$ stty -a -F /dev/ttyS1
```

Výstup tohoto příkazu, který vidíte v příkladu 4.5, obsahuje hodnoty všech stavových příznaků daného zařízení. Příznaky uvedené symbolem „minus“, například `-crtscts` udávají, že tento příznak je vypnut.

Příklad 4.5 – Výstup příkazu stty –a

```
speed 19200 baud; rows 0; columns 0; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D; eol = <undef>;
       eol2 = <undef>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
       werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 hupcl -cstopb cread clocal -crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr -igncr -icrnl -ixon
        -ixoff -iuclc -ixany -imaxbel
-opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0 tab0
        bs0 vt0 ff0
-isig -icanon iexten echo echoe echok -echonl -noflsh -xcase -tostop
      -echoprt echoctl echoke
```

Přehled nejdůležitějších příznaků obsahuje tabulka 4.2. Jednotlivé příznaky se zapnou tak, že se příkazu **stty** předají a vypnou tak, že se mu předají s uvádějícím symbolem minus. Pokud tedy budeme chtít na zařízení `ttyS0` vypnout hardwarový handshaking, zadáme

```
$ stty -crtscts -F /dev/ttyS0
```

Tabulka 4.2 – Důležité příznaky příkazu stty pro konfiguraci sériových zařízení

Příznak	Popis
`N`	Nastavuje rychlost linky na *N* bitů za sekundu.
`crtscts`	Zapíná/vypíná hardwarový handshaking.
`ixon`	Zapíná/vypíná XON/XOFF handshaking.
`clocal`	Zapíná/vypíná signály pro řízení modemu jako DTR/DTS a DCD. Je to nutné při použití třívodičového sériového kabelu, protože ten tyto signály nepřenáší.
`cs5 cs6 cs7 cs8`	Nastavuje počet datových bitů na 5, 6, 7 respektive 8.
`parodd`	Zapíná lichou paritu. Je-li příznak vypnut, používá se sudá parita.
`parenb`	Zapíná paritu. Není-li příznak zapnut, kontrola parity se neprovádí.
`cstopb`	Zapíná použití dvou stop bitů. Pokud je příznak vypnut, používá se jen jeden stop bit.
`echo`	Zapíná/vypíná lokální echo odesílaných znaků.

Následující příklad kombinuje některé z těchto příznaků a nastavuje zařízení `ttyS0` pro práci rychlostí 19200 b/s, s osmi datovými bity, bez parity, s hardwarovým handshakingem a bez lokálního echa:

```
$ stty 19200 cs8 -parenb crtscts -echo -F /dev/ttyS0
```

Sériová zařízení a výzva login:

Bývalo poměrně obvyklé, že Unix byl nainstalován s jedním serverem a řadou „hloupých" znakových terminálů nebo modemů. Dnes je tento typ instalace vzácnější, což je dobrá zpráva pro všechny, kteří si jej chtějí vyzkoušet, protože „hloupé" terminály jsou dnes velice levné. Modemové instalace sice vzácnější nejsou, ale obvykle dnes používají přihlášení protokolem SLIP nebo PPP (o kterých budeme hovořit v kapitolách 7 a 8). Nicméně všechny tyto konfigurace mohou využívat jednoduchý program nazvaný **getty**.

Termín **getty** je zřejmě zkratka z „get tty". Program **getty** otevře sériové zařízení, správně je nakonfiguruje, případně nakonfiguruje modem, a potom čeká na spojení. Aktivní spojení na sériové lince je typicky indikováno signálem DCD (Data Carrier Detect). Jakmile je detekováno spojení, program **getty** vypíše výzvu `login:` a pak spustí program **login**, který zajistí samotné přihlášení. Na každém z virtuálních terminálů (například `/dev/tty1`) v Linuxu tento program běží.

Existuje řada různých implementací programu **getty**, každá z nich obsluhuje některé konfigurace lépe než jiná. Program **getty**, o kterém budeme hovořit my, se jmenuje **mgetty**. Je velice oblíbený, protože obsahuje všechny funkce, pro něž je velmi přívětivý k modemům včetně podpory automatických faxových programů a hlasových modemů. Zaměříme se na konfiguraci programu **mgetty** tak, aby reagoval na klasické datové volání a zbytek už necháme na vás.

Konfigurace démona mgetty

Zdrojový kód démona **mgetty** je k dispozici na adrese `ftp://alpha.greenie.net/pub/mgetty/source/` a bývá součástí prakticky všech distribucí Linuxu. Démon **mgetty** se od ostatních démonů **getty** liší zejména v tom, že je navržen specificky pro podporu modemů kompatibilních se standardem Hayes. Podporuje i přímá terminálová připojení, lépe se však hodí pro telefonické aplikace. K detekci příchozího volání nepoužívá signál DCD, ale čeká na zprávu `RING`, kterou moderní modemy generují, když přijde hovor a nejsou nakonfigurovány na automatickou odpověď.

Hlavní spustitelný program se jmenuje `/usr/sbin/mgetty` a odpovídající hlavní konfigurační soubor je `/etc/mgetty/mgetty.config`. Kromě toho existuje ještě řada dalších binárních a konfiguračních souborů, které pokrývají všechny funkce programu **mgetty**.

U většiny instalací se celá konfigurace programu redukuje na úpravu souboru `/etc/mgetty/mgetty.config` a přidání příslušné položky do souboru `/etc/inittab` tak, aby se program **mgetty** spouštěl automaticky.

Příklad 4.6 znázorňuje velmi jednoduchý konfigurační soubor programu **mgetty**. V tomto příkladu konfigurujeme dvě sériová zařízení. První z nich, `/dev/ttyS0` podporuje modem kompatibilní s Hayes a rychlostí 38400 b/s. Druhé, `/dev/ttyS1`, podporuje přímo připojený terminál VT100 s rychlostí 19200 b/s.

Příklad 4.6 – Příklad konfiguračního souboru /etc/mgetty/mgetty.config

```
#
# konfigurační soubor programu mgetty
#
# toto je příklad konfigurace, podrobnosti viz mgetty.info
#
# komentáře začínají "#", prázdné řádky se ignorují
#
# ----- globální sekce -----
#
# V této sekci nastavujeme globální vlastnosti, nastavení jednotlivých
# portů budou následovat
#
# přístup k modemům rychlostí 38400 b/s
speed 38400
#
# nastavení ladicí úrovně na "4" (implicitní dle policy.h)
debug 4
#

# ----- části pro jednotlivé porty -----
#
# Tady se uvádějí věci platné jen pro jednu linku, ne pro všechny
#
#
# Modem Hayes připojený na ttyS0: bez faxu, menší logování
#
port ttyS0
  debug 3
  data-only y
#
# přímé spojení s terminálem VT100, který nepoužívá DTR
#
port ttyS1
  direct y
  speed 19200
  toggle-dtr n
#
```

Konfigurační soubor obsahuje globální část a část specifickou pro jednotlivé porty. V našem příkladu nastavujeme v globální části rychlost na 38400 b/s. Toto nastavení zdědí port `ttyS0`. Všechny porty obsluhované programem **mgetty** budou tuto rychlost používat, ledaže by v části konkrétního portu byla nastavena rychlost jiná tak, jak jsme to udělali pro zařízení `ttyS1`.

Klíčové slovo `debug` řídí „ukecanost“ logovacích záznamů programu **mgetty**. Klíčové slovo `data-only` nařizuje programu **mgetty** ignorovat všechny faxmodemové funkce a pracovat se zařízením jako s čistým modemem. Klíčové slovo `direct` v konfiguraci portu `ttyS1` programu **mgetty** říká, aby se na tomto portu nepokoušel inicializovat modem. Konečně klíčové slovo `toggle-dtr` programu nařizuje, aby neukončoval spojení pomocí signálu DTR (Data Terminal Ready) sériového rozhraní, protože některé terminály to nemají rády.

Můžete se také rozhodnout nechat soubor `mgetty.config` prázdný a stejné parametry specifikovat pomocí řádkových voleb příkazu. Dokumentace dodávaná s programem obsahuje úplný popis parametrů konfiguračního souboru i řádkových parametrů. Viz následující příklad.

Abychom konfiguraci aktivovali, musíme do souboru `/etc/inittab` přidat dva záznamy. Soubor `inittab` je konfiguračním souborem příkazu **init** na Unixu System V. Příkaz **init** provádí inicializaci systému, představuje způsob, jak zajistit automatické spuštění příkazů při spouštění systému i jak je spustit znovu poté, co budou ukončeny. To je ideální řešení pro programy jako je **getty**.

```
T0:23:respawn:/sbin/mgetty ttyS0
T1:23:respawn:/sbin/mgetty ttyS1
```

Každý řádek souboru `/etc/inittab` obsahuje čtyři údaje oddělené dvojtečkou. První údaj je identifikátor, který jednoznačně definuje každý ze záznamů v tomto souboru. Klasicky to bývaly dva znaky, novější verze umožňují čtyři znaky. Druhá položka představuje seznam běhových úrovní, na nichž má být program spouštěn. Běhové úrovně představují metodu jak nabídnout různé konfigurace stejného počítače a implementují se pomocí stromů spouštěcích skriptů umístěných v adresářích `/etc/rc1.d`, `/etc/rc2.d` a podobně. Tato funkce bývá typicky implementována velmi jednoduše a vlastní záznamy buď vytvářejte podle ostatních, nebo prostudujte dokumentaci k systému, kde se dozvíte více. Třetí údaj říká, kdy se má akce provést. Pro potřeby programů jako je **getty** používáme hodnotu `respawn`, což znamená, že program má být znovu spuštěn v okamžiku, kdy dokončí svou činnost. Lze zde použít i různé další volby, ale pro naše potřeby nám nyní žádná jiná nevyhovuje. Čtvrtým údajem je samotný spouštěný příkaz, což je tedy **mgetty** s případnými parametry, které mu předáváme. V našem jednoduchém případě spouštíme (a znovu spouštíme) program **mgetty** vždy na běhových úrovních 2 nebo 3 a jako parametr uvádíme pouze jméno zařízení, které chceme používat. Příkaz **mgetty** automaticky předpokládá cestu `/dev/`, takže ji nemusíme uvádět.

V tomto textu jsme se snažili o stručné představení programu **mgetty** a toho, jak zajistit přihlašovací výzvu u sériových zařízení. Podstatně podrobnější informace naleznete v dokumentu Serial-HOWTO.

Po provedení úprav v konfiguračních souborech musíte znovu spustit program **init**, aby se změny projevily. Stačí poslat procesu **init** signál HANGUP. Tento proces má vždy identifikátor 1, takže můžete jednoduše použít následující příkaz:

```
# kill -HUP 1
```

KAPITOLA 5

Konfigurace sítí TCP/IP

V této kapitole si projdeme kroky, které jsou nezbytné pro konfiguraci sítí na bázi protokolu TCP/IP. Začneme s přidělením IP adresy, pak si projdeme nastavení rozhraní TCP/IP a představíme si několik užitečných nástrojů pro hledání problémů ve vaší síťové instalaci.

Většinu úkolů probraných v této kapitole budete muset obvykle provést pouze jedenkrát. Pozdější změny v konfiguračních souborech budou nutné jen v případech, kdy budete chtít do sítě přidat nový systém; nebo budete-li chtít systém kompletně překonfigurovat. Nicméně některé z příkazů pro konfiguraci protokolu TCP/IP musí být spouštěny pokaždé při zavádění systému. To se zpravidla provádí jejich voláním ze systémových skriptů v adresáři `/etc/rc`.

Síťová část zaváděcí procedury je typicky obsažena v nějakém skriptu. Názvy tohoto skriptu se liší na různých distribucích Linuxu. Ve většině starších distribucí se používají skripty `rc.net` nebo `rc.inet`. Někdy se také můžete setkat se dvěma skripty s názvy `rc.inet1` a `rc.inet2`. První inicializuje síťovou část jádra systému, zatímco druhý spouští základní síťové služby a aplikace. V moderních distribucích jsou soubory `rc` strukturovány mnohem promyšlenějším způsobem, v adresáři `/etc/init.d/` (nebo `/etc/rc.d/init.d/`) najdete skripty, které vytvářejí síťová rozhraní a další soubory `rc`, které spouštějí síťové aplikační programy. Příklady v této knize vycházejí z tohoto posledního řešení.

V této kapitole budeme hovořit o skriptech konfigurujících síťová rozhraní, o aplikacích budeme hovořit v dalších kapitolách. Po přečtení této kapitoly byste měli dokázat vytvořit posloupnost příkazů, která na vašem počítači správně nakonfiguruje síť s protokolem TCP/IP. Pak byste měli nahradit všechny vzorové příkazy ve vašich konfiguračních skriptech svými vlastními příkazy, zkontrolovat, že je skript při startu systému spouštěn základním `rc` skriptem, a restartovat počítač. Síťové `rc` skripty dodávané společně s vaší oblíbenou verzí Linuxu vám mohou posloužit jako dobré příklady.

Připojování souborového systému /proc

Některé z konfiguračních nástrojů ve verzích NET-2 a NET-3 Linuxu spoléhají při komunikaci s jádrem na souborový systém `/proc`. Toto rozhraní umožňuje přístup k aktuálním informacím jádra za pomoci mechanismu, jenž je podobný souborovému systému. Když je tento systém připojen, můžete stejně jako u ostatních souborových systémů vypisovat jeho soubory nebo zobrazovat jejich obsah. Typickými položkami jsou soubory `loadavg`, které obsahují průměrné zatížení systému nebo soubor `meminfo`, který zobrazuje aktuální obsazení fyzické paměti a využití odkládacích souborů.

Síťový kód k tomu přidává adresář `net`. Obsahuje množství souborů s informacemi jako například tabulky ARP jádra systému, stav TCP spojení a směrovací tabulky. Většina síťových administrativních nástrojů čerpá informace z těchto souborů.

Souborový systém `proc` (někdy se používá označení `procfs`) je obvykle připojen při zavádění systému jako adresář `/proc`. Nejlépe toho dosáhnete přidáním následujícího řádku do souboru `/etc/fstab`:

```
# připojovací bod procfs
none /proc proc defaults
```

Potom ze svého skriptu `/etc/rc` spusťte příkaz **mount / proc**.

V současné době je souborový systém `procfs` implicitně začleněn do většiny jader operačního systému. Není-li souborový systém `procfs` ve vašem jádru, objeví se zpráva jako `mount: fs type procfs not supported by kernel`. V tom případě budete muset přeložit jádro systému a na dotaz, zda si přejete nainstalovat podporu souborového systému `procfs`, odpovědět „yes“.

Instalace binárních souborů

Používáte-li jednu z předkompilovaných distribucí Linuxu, bude velmi pravděpodobně obsahovat hlavní síťové aplikace a utility, a také kompletní sadu vzorových souborů. Jediným případem kdy budete chtít získat a nainstalovat nové utility bude instalace nové verze jádra operačního systému. Protože to občas vede ke změnám v síťové vrstvě jádra, budete muset aktualizovat i základní konfigurační nástroje. To znamená přinejmenším rekompilaci binárních souborů, ale někdy budete potřebovat i nejnovější sady binárních souborů. Tyto soubory jsou dostupné na oficiální adrese *ftp.inka.de/pub/com/Linux/networking/NetTools/*, kde jsou sbaleny v archivu `net-tools-XXX.tar.gz`, kde `XXX` označuje číslo verze. Linuxu 2.0 odpovídá balík `net-tools-1.45`.

Pokud si chcete zkompilovat a nainstalovat standardní síťové aplikace na bázi protokolu TCP/IP sami, získáte jejich zdrojový kód na většině linuxových FTP serverů. Většina moderních distribucí Linuxu obsahuje poměrně kompletní balík síťových aplikací TCP/IP, například webový prohlížeč, programy pro **telnet** a **ftp** a další síťové programy, jako je například **talk**. Narazíte-li na něco, co musíte přeložit sami, je dobrá šance, že se vám to bez potíží podaří; obvykle stačí dodržet instrukce uvedené ve zdrojovém balíku.

Nastavení jména hostitele

Většina, ne-li všechny aplikace spoléhají na to, že je název místního hostitele nastaven na nějakou rozumnou hodnotu. To se obvykle provede při zavádění systému pomocí příkazu **hostname**. Chcete-li nastavit název hostitele na *jméno*, zadejte:

```
# hostname jméno
```

U tohoto příkazu je běžné používat nekvalifikované názvy hostitele bez jakéhokoliv názvu domény. Například hostitelé v pivovaru Virtual Brewery (který je popsán v příloze A) se mohou jmenovat **vale.vbrew.com** nebo **vlager.vbrew.com**. Toto jsou jejich oficiální, *plně kvalifikovaná doménová jména* (FQDN). Názvy místních hostitelů budou tvořit pouze první část z plně kvalifikovaného jména, například **vale**. Protože je však název lokálního hostitele často používán pro vyhledání IP adresy hostitele, musíte zajistit, aby knihovna resolveru byla schopná nalézt IP adresu tohoto hostitele. To obvykle znamená, že musíte název hostitele specifikovat v souboru `/etc/hosts`.

Někteří lidé doporučují používat příkaz **domainname**, kterým se jádro operačního systému dozví o doméně ve zbývající části FQDN. U tohoto způsobu můžete zkombinovat hodnoty **hostname** a **domainname** a získat tak plně kvalifikované jméno. Tento způsob je však v pořádku maximálně z poloviny. Příkaz **domainname** se všeobecně používá k nastavení NIS domény hostitele, která může být zcela jiná než DNS doména, ke které patří váš hostitel. Pokud chcete zajistit, aby byla krátká podoba jména vašeho hostitele rozlišitelná všemi staršími verzemi příkazu **hostname**, přidejte je namísto toho jako záznam do místního DNS serveru nebo v souboru `/etc/hosts` uveďte plně kvalifikované jméno. Pak můžete použít parametr `--fqdn` příkazu **hostname** a získat tak plně kvalifikované doménové jméno.

Přiřazení IP adresy

Chcete-li nakonfigurovat síťový software na hostiteli, který není určen pro práci v síti (aby na něm mohl být například spouštěn software síťových konferencí INN), můžete tuto stať s klidným svědomím přeskočit, protože k tomu budete potřebovat pouze IP adresu svého lokálního rozhraní, a ta je vždy 127.0.0.1.

Mírně složitější je tento problém v reálných sítích typu Ethernet. Chcete-li připojit svého hostitele k existující síti, musíte požádat jejího správce, aby vám v této síti přidělil IP adresu. Nastavujete-li celou síť sami, musíte si přidělit IP adresy sami.

Hostitelé, kteří patří do místní sítě, by měli obvykle sdílet adresy ze stejné logické IP sítě. Z toho důvodu musíte přidělit síti síťovou IP adresu. Máte-li několik fyzických sítí, musíte jim buď přidělit odlišné síťové adresy, anebo musíte použít podsítě, čímž rozdělíte svůj rozsah IP adres do několika podsítí. O podsítích budeme hovořit v další části, *Vytváření podsítí*.

Při výběru IP adres sítí záleží na tom zda v blízké době plánujete připojení k Internetu. Pokud ano, měli byste *hned* požádat o přidělení oficiální IP adresy. Požádejte o pomoc svého poskytovatele síťových služeb. Pokud chcete získat oficiální síťovou adresu čistě pro případ, že byste se někdy k Internetu připojovali, vyžádejte si na adrese *hostmaster@internic.net* žádost o síťovou adresu, nebo se obraťte na národní NIC, pokud u vás existuje[35].

35 Pozn. překladatele: V České republice zajišťuje přidělování IP adres jednak řada takzvaných LIR (Local Internet Registry) – společností, které mají delegován nějaký rozsah adres a rozdělují je dále. Nadřazenou autoritou je RIR (Regional Internet Registry), kterým je pro celou Evropu organizace RIPE NCC (*www.ripe.net*). Ti se s vámi ovšem bavit nebudou a podstatně jednodušší je domluvit se se svým poskytovatelem internetového připojení, který obvykle bez zbytečného otálení poskytne určitý rozsah IP adres za jednorázový poplatek (a pokud to nedělá, neprodleně přejděte k jinému poskytovateli).

Není-li vaše síť připojena k Internetu a v dohledné době její připojení neplánujete, můžete si vybrat libovolnou platnou síťovou adresu. Musíte ovšem zajistit, aby žádné pakety z vaší interní sítě „neutekly" na Internet. Abyste zaručili, že nedojde k žádné škodě dokonce i když nějaký paket uteče, měli byste použít některou ze síťových adres, rezervovaných pro privátní sítě. Agentura IANA (Internet Assigned Number Agency) rezervovala několik sítí tříd A, B i C, které můžete použít i bez registrace. Tyto adresy platí pouze ve vaší privátní síti a mezi internetovými systémy nedochází k jejich směrování. Rozsahy těchto adres jsou definovány v dokumentu RFC 1597 a jsou uvedeny v tabulce 2.1 v kapitole 2. Všimněte si, že druhý a třetí blok obsahují 16, respektive 256 sítí.

Volba adres z těchto rozsahů není výhodná jen pro zcela nepřipojené sítě, pomocí jednoho počítače používaného jako brána budete pořád schopni zajistit do jisté míry omezený přístup k Internetu. Ve vaší interní síti bude tato brána známa pod její interní adresou, zatímco okolní svět ji uvidí pod její oficiální adresou (kterou vám přidělí poskytovatel). K této problematice se vrátíme v souvislosti s IP maškarádou v kapitole 11.

Ve zbytku knihy budeme předpokládat, že správce sítě v pivovaru používá adresu třídy B, řekněme 172.16.0.0. Samozřejmě potřebám pivovaru i vinařství by postačovala adresa třídy C, nicméně třídu B používáme kvůli jednoduchosti. Příklady rozdělení na podsítě v dalším textu budou díky tomu o něco názornější.

Vytváření podsítí

Abyste mohli provozovat více ethernetových (nebo jiných sítí), musíte svou síť rozdělit na podsítě. Rozdělení na podsítě je nutné pouze v případě, že máte více *vysílaných sítí*, spojení uzel-uzel se nepočítají. Pokud máte například jeden Ethernet a jednu nebo více SLIP linek do světa, nepotřebujete podsítě. Podrobněji je to vysvětleno v kapitole 7.

Kvůli správě dvou ethernetů se správce pivovaru rozhodl použít 8 bitů hostitelské části adresy jako podsíťovou adresu. Tím zůstává dalších 8 bitů volných pro adresu hostitele, což nám umožní připojit na každou podsíť 254 počítačů. Pak přiřadil podsíť 1 pivovaru a podsíť 2 vinařství. Tomu budou odpovídat síťové adresy 172.16.1.0 a 172.16.2.0. Maska podsítě bude 255.255.255.0.

Počítač **vlager**, což je brána mezi oběma sítěmi, má na obou sítích přidělenu adresu 1, takže jeho IP adresy na jednotlivých sítích budou 172.16.1.1 a 172.16.2.1.

Všimněte si, že v našem příkladu používáme kvůli názornosti adresu třídy B, realističtější přístup by byla adresa třídy C. S novým síťovým kódem není rozdělení na podsítě omezeno na hranice bajtů, takže i síť třídy C můžeme rozdělit na několik podsítí. Můžeme například použít dva bity hostitelské části pro masku sítě, čímž dostáváme čtyři podsítě s 64 počítači na každé z nich[36],[37].

36 První číslo na každé podsíti je adresa samotné podsítě a poslední číslo je rezervováno jako vysílací adresa, takže budeme mít pouze 62 možných počítačů na každé podsíti.

37 Pozn. překladatele: Předpokládejme například, že budeme mít síť třídy C s adresou 192.168.10.0. Použijeme-li síťovou masku 255.255.255.192 (ono 192 dekadicky je 11000000 binárně), dostáváme dva bity pro adresaci podsítě a 6 bitů pro adresaci počítačů na podsítích. Pak máme čtyři podsítě: 192.168.10.0, 192.168.10.64, 192.168.10.128 a 192.168.10.192 (do dvojkové soustavy už si to převádějte sami – je to práce, ale jen tak to lze pochopit). Pro podsítě platí omezení známé u hostitelských adres (tedy že nelze použít „samé nulové" a „samé jedničkové" adresy), takže počítače na první (respektive nulté) podsíti budeme číslovat 192.168.10.1, 192.168.10.2,... a na druhé (respektive první) pak 192.168.10.65, 162.168.10.66...

Pouze se musíte ujistit, že jste vybrali jednu ze tříd A, B nebo C, protože jinak by pravděpodobně vše nefungovalo zcela správně. Plánujete-li v blízké době připojení k Internetu, měli byste si už *nyní* obstarat oficiální IP-adresu. Nejlépe je požádat o pomoc svého poskytovatele síťových služeb. Pokud si chcete obstarat číslo sítě jenom proto, že se někdy v budoucnosti budete chtít připojit na Internet, vyžádejte si na adrese **hostmaster@internic.net** formulář Network Address Application Form.

Budete-li spravovat několik ethernetových sítí (nebo jiných sítí, jakmile pro ně budou dostupné ovladače), musíte rozdělit vaši síť na podsítě. Všimněte si, že vytváření podsítí je požadováno pouze v případě, že vlastníte více než jednu *vysílací síť* (*broadcast network*); nepočítaje v to spojení pomocí protokolu PPP. Máte-li například jeden Ethernet a jedno nebo více spojení s vnějším světem pomocí protokolu SLIP, nepotřebujete ve své síti vytvářet podsítě. Důvod bude objasněn v kapitole 7.

Síťový správce pivovaru požádal například centrum NIC o přidělení čísla sítě třídy B, obdržel adresu **191.72.0.0**. Aby si správce přizpůsobil své dva Ethernety k obrazu svému, rozhodl se použít osm bitů z části hostitele jako doplňující bity podsítě. Tato úprava poskytla části hostitele dalších osm bitů, což dovoluje až 254 hostitelů v každé podsíti. Dále správce přidělil pivovaru číslo podsítě 1 a vinárně číslo podsítě 2. Tedy jejich příslušné síťové adresy jsou **191.72.1.0** a **191.72.2.0**. Maska podsítě je **255.255.255.0**.

Bráně **vlager**, což je brána mezi dvěma sítěmi, bylo u obou podsítí přiděleno číslo hostitele 1, takže její IP-adresy jsou **191.72.1.1**, resp. **191.72.2.1**.

Vytvoření souborů hosts a networks

Po vytvoření podsítí byste měli za pomoci souboru `/etc/hosts` nastavit jednoduchou variantu rozlišení názvu hostitele. Pokud nehodláte k rozlišení adres používat DNS nebo NIS, musíte do souboru `hosts` vložit všechny hostitele.

Dokonce i v případě kdy budete za normálních okolností používat pro rozlišení názvů služby DNS nebo NIS, budete muset v souboru `/etc/hosts` specifikovat přinejmenším určitou vybranou skupinu hlavních hostitelů. Nějaký druh rozlišení názvů hostitelů budete totiž potřebovat i tehdy, nepoběží-li žádné síťové rozhraní – například při zavádění operačního systému. Není to jen otázka pohodlí, ale umožní vám to také používat v `rc` skriptech symbolické názvy hostitelů. Takže při změně IP adres vám postačí pouze zkopírovat aktualizovaný soubor `hosts` na všechny počítače a nemusíte se zatěžovat samostatnou editací velkého počtu `rc` souborů. Do souboru `hosts` obvykle přidáte všechny názvy a adresy lokálního hostitele, adresy všech používaných bran a eventuálních serverů NIS[38].

V průběhu úvodního testování byste měli také zajistit, že váš resolver používá pouze informace ze souboru `hosts`. Vzorové soubory služeb DNS a NIS mohou při jejich neúmyslném použití vést k velmi zajímavým efektům. Chcete-li, aby všechny aplikace při vyhledání IP adres hostitelů používaly výhradně soubor `/etc/hosts`, musíte upravit soubor `/etc/host.conf`. Všechny řádky, které začínají klíčovým slovem *order*, označte jako komentáře. Provedete to tak, že na první pozici příslušného řádku vložíte znak # a nakonec vložíte řádek:

```
order hosts
```

38 Adresy serverů NIC jsou zapotřebí pouze pokud používáte NIS Petera Erikssona. Jiné implementace služby NIS si své servery nacházejí za běhu prostřednictvím programu **ypbind**.

Konfigurace knihovny resolveru bude podrobně rozebrána v kapitole 6.

Soubor `hosts` obsahuje na každém řádku jednu položku, která se skládá z IP adresy, jména hostitele a z nepovinného seznamu aliasů hostitele. Jednotlivá pole jsou vzájemně oddělena mezerami nebo tabulátory a pole s adresou musí začínat v prvním sloupci. Cokoliv co následuje za znakem `#` na stejném řádku je považováno za komentář a ignoruje se.

Názvy hostitelů mohou být buď plně kvalifikované, nebo relativní vzhledem k místní doméně. U serveru **vale** byste typicky zadali jak plně kvalifikované jméno **vale.vbrew.com**, tak i relativní jméno **vale**. V souboru *`hosts`* tak bude server znám pod oficiálním i pod zkráceným jménem.

Následující příklad ilustruje, jak by mohl vypadat soubor `hosts` ve společnosti Virtual Brewery. Jsou v něm obsaženy dva speciální názvy, **vlager-if1** a **vlager-if2**, které definují adresy obou rozhraní používaných branou **vlager**.

```
#
# Soubor hosts pro Virtual Brewery/Virtual Winery
#
# IP            FQDN                 aliasy
#
127.0.0.1       localhost
#
172.16.1.1      vlager.vbrew.com      vlager vlager-if1
172.16.1.2      vstout.vbrew.com      vstout
172.16.1.3      vale.vbrew.com        vale
#
172.16.2.1      vlager-if2
172.16.2.2      vbeaujolais.vbrew.com vbeaujolais
172.16.2.3      vbardolino.vbrew.com  vbardolino
172.16.2.4      vchianti.vbrew.com    vchianti
```

Stejně jako u IP adres hostitelů budete také někdy chtít použít symbolický název pro síťové adresy. Proto má soubor `hosts` kamaráda nazvaného `/etc/networks`, který mapuje názvy sítí na jejich adresy a opačně. Ve společnosti Virtual Brewery můžeme nainstalovat následující soubor `networks`[39]:

```
# Soubor /etc/networks pro Virtual Brewery
brew-net      172.16.1.0
wine-net      172.16.2.0
```

39 Pozor na to, že názvy sítí souboru `networks` nesmějí kolidovat s názvy počítačů v souboru `hosts`, jinak by se některé programy mohly chovat nedefinovaným způsobem.

Konfigurace rozhraní pro protokol IP

Po nastavení hardwaru, které jsme probírali v kapitole 4, musíte o těchto zařízeních říci síťovému softwaru jádra. K nastavení síťových rozhraní a inicializaci směrovací tabulky se používá několik příkazů. Tyto úkoly se obvykle provádějí při každém startu počítače z inicializačního skriptu sítě. Základní konfigurační nástroje se nazývají **ifconfig** (kde „if" znamená rozhraní – interface) a **route**.

Příkaz ifconfig se používá pro zpřístupnění rozhraní síťové vrstvě jádra. Tento proces v sobě zahrnuje přidělení IP adresy a dalších parametrů a aktivaci rozhraní. Výraz „aktivní" znamená, že jádro pomocí takového rozhraní může vysílat a přijímat IP datagramy. Nejjednodušší způsob aktivace rozhraní je následující:

```
ifconfig rozhraní ip_adresa
```

Výše uvedený příkaz přidělí rozhraní IP adresu a aktivuje ho. Všechny ostatní parametry jsou nastaveny na své implicitní hodnoty. Například implicitní maska podsítě je odvozena z IP adresy podle třídy sítě, například adrese třídy B odpovídá maska 255.255.0.0. Příkaz **ifconfig** je detailně popsán později v části *Vše o příkazu ifconfig*.

Příkaz **route** umožňuje přidávat nebo odstraňovat trasy ze směrovací tabulky jádra systému. Může být volán následujícím způsobem

```
route [add|del] [-net|-host] cíl [rozhraní]
```

Parametry `add` a `del` určují, zda se má trasa pro zadaný cíl přidat nebo odebrat. Parametry `-net` a `-host` říkají, zda je cílem trasy síť nebo jednotlivý počítač (pokud nic neuvedete, předpokládá se, že je to počítač). Konečně nepovinný parametr `rozhraní` říká, přes které rozhraní má být trasa směrována. Pokud parametr neuvedete, jádro Linuxu provede obvykle správný odhad. O tomto tématu budeme podrobněji hovořit v následující části.

Lokální rozhraní

Jedním z prvních aktivovaných rozhraní je lokální zpětnovazebné rozhraní:

```
# ifconfig lo 127.0.0.1
```

Občas zjistíte, že se místo IP adresy používá fiktivní název hostitele **localhost**. Příkaz **ifconfig** vyhledá příslušné jméno v souboru `hosts`, kde by mu měla být přidělena adresa 127.0.0.1:

```
# Příklad záznamu pro localhost v /etc/hosts
localhost      127.0.0.1
```

Chcete-li si prohlédnout konfiguraci nějakého rozhraní, spusťte příkaz **ifconfig** a jako parametr zadejte pouze název tohoto rozhraní:

```
$ ifconfig lo
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:3924  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          Collisions:0
```

Jak vidíme, lokálnímu rozhraní byla přidělena síťová maska 255.0.0.0, protože adresa 127.0.0.1 je adresou třídy A.

Nyní můžete začít se svou miniaturní sítí experimentovat. Zatím však ve směrovací tabulce stále chybí položka, která řekne protokolu IP, že může toto rozhraní používat jako trasu k cílové adrese 127.0.0.1. Tu přidáte následujícím příkazem:

```
# route add 127.0.0.1
```

Opět můžete použít namísto IP adresy název hostitele **localhost** za předpokladu, že jej máte definován v souboru `/etc/hosts`.

Dále byste měli zkontrolovat, že vše správně funguje, například pomocí použití příkazu **ping**. Příkaz **ping** je síťovým ekvivalentem sonaru[40] a je používán k ověření dostupnosti příslušné adresy a k měření prodlev, které se vyskytují při posílání datagramu tam a zpět. Čas požadovaný na tuto operaci je často uváděn pod označením „round-trip time“.

```
# ping localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=32 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=32 time=0.4 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=32 time=0.4 ms
^C--- localhost ping statistics ---
3 packets transmitter, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.4/0.4/0.4 ms
```

Spustíte-li příkaz **ping** výše uvedeným způsobem, bude se tento příkaz snažit posílat pakety tak dlouho, dokud ho uživatel nepřeruší. Symbol `^C` označuje místo, kde byla stisknuta kombinace kláves Ctrl+C.

Výše uvedený příklad ukazuje, že pakety pro adresu 127.0.0.1 jsou doručovány správně a odpovědi se vracejí příkazu **ping** zpět téměř okamžitě. To naznačuje, že jste při svém prvním nastavení síťového rozhraní uspěli.

Pokud se výstup příkazu **ping** nepodobá výše uvedenému výpisu, pak je tu problém. Zkontrolujte případná chybová hlášení zda nenaznačují, že některý ze souborů nebyl korektně nainstalován. Zkontrolujte, zda jsou binární soubory příkazů **ifconfig** a **route** kompatibilní s vámi používanou verzí jádra operačního systému a hlavně se podívejte, zda bylo jádro zkompilováno s povolením síťových služeb (to poznáte podle přítomnosti adresáře `/proc/net`). Pokud obdržíte chybovou zprávu „Network unreachable“, pak bude zřejmě chyba v nastavení příkazu **route**. Ujistěte se, že používáte stejnou adresu, kterou jste předali příkazu **ifconfig**.

Výše popsané kroky stačí k tomu, abyste mohli na samostatném hostiteli používat síťové aplikace. Jakmile přidáte do síťového inicializačního skriptu uvedené řádky a ujistíte se, že se skript při startu systému skutečně spouští, můžete počítač restartovat a vyzkoušet různé aplikace. Například příkaz **telnet localhost** by měl s vaším hostitelem navázat telnetové spojení a nabídnout vám přihlašovací výzvu.

Lokální rozhraní je ale užitečné nejen jako příklad vhodný pro knihy o sítích nebo jako testovací prostředek při vývoji. Ve skutečnosti ho používají některé aplikace v průběhu normálních operací[41]. Proto ho musíte nakonfigurovat vždy, bez ohledu na to, zda váš počítač je či není připojen k síti.

---

40 Vzpomínáte si na „Echoes“ od Pink Floydů?

41 Například aplikace založené na RPC používají rozhraní při startu k registraci sama sebe u démona **portmapper**. Mezi tyto aplikace patří například NIS a NFS.

Ethernetová rozhraní

Konfigurace ethernetového rozhraní má s nastavením lokálního rozhraní mnoho společného, pouze ve spojitosti s podsítěmi vyžaduje několik dalších parametrů.

Ve společnosti Virtual Brewery jsme rozdělili síť IP, která byla původně sítí třídy B, na podsítě které jsou třídy C. Aby se v této změně vaše rozhraní vyznalo, bude příkaz **ifconfig** vypadat následovně:

```
# ifconfig eth0 vstout netmask 255.255.255.0
```

Tento příkaz přiřadí rozhraní `eth0` IP adresu počítače **vstout** (172.16.1.2)[42]. Pokud bychom vynechali síťovou masku, pak by si ji příkaz **ifconfig** odvodil z třídy sítě, ze které vyplyne síťová maska 255.255.0.0. Rychlé ověření vypíše následující text:

```
# ifconfig eth0
eth0      Link encap 10Mps Ethernet HWaddr  00:00:C0:90:B3:42
          inet addr 172.16.1.2 Bcast 172.16.1.255 Mask 255.255.255.0
          UP BROADCAST RUNNING  MTU 1500  Metric 1
          RX packets 0 errors 0 dropped 0 overrun 0
          TX packets 0 errors 0 dropped 0 overrun 0
```

Můžete si všimnout, že příkaz **ifconfig** automaticky nastavil vysílací adresu (výše uvedené pole Bcast) na obvyklou hodnotu, což je číslo sítě se všemi nastavenými bity v části hostitele. Také velikost přenosové jednotky (maximální velikost IP datagramu, kterou bude generovat jádro pro toto rozhraní) byla nastavena na maximální délku ethernetového paketu – na 1500 bajtů. Obvykle můžete tyto standardní hodnoty použít, nicméně všechny mohou být změněny speciálními parametry, které jsou popsány v části *Vše o příkazu ifconfig*.

Všechny tyto hodnoty mohou být změněny pomocí speciálních voleb, jež budou popsány dále.

Stejně jako tomu bylo u lokálního rozhraní, musíte nyní nainstalovat směrovací data, která budou jádro informovat o síti, jež je dosažitelná prostřednictvím rozhraní `eth0`. Pro firmu Virtual Brewery byste volali příkaz **route** následujícím způsobem:

```
# route add -net 191.72.1.0
```

Na první pohled to vypadá trochu magicky, protože není zcela zřejmé, jak příkaz **route** zjistí, přes které rozhraní má směrovat. Náš trik je ale celkem jednoduchý: jádro operačního systému si ověří všechna rozhraní, která již byla nakonfigurována a porovná cílovou adresu (v tomto případě 191.72.1.0) se síťovou částí adresy rozhraní (to znamená, že bude po bitech porovnávat hodnotu získanou z adresy rozhraní a síťové masky). Jediné vyhovující rozhraní bude `eth0`.

K čemu tedy slouží volba `-net`? Používá se z toho důvodu, že příkaz **route** umí obsluhovat jak směrování do sítí, tak i směrování k jednotlivým hostitelům (jak jsme si ukázali u příkladu s lokálním rozhraním a jeho adresou). Je-li mu předána adresa v tečkové notaci, pokusí se příkaz **route** odhadnout, zda se jedná o adresu sítě nebo hostitele tak, že se podívá na hostitelskou část adresy. Je-li hostitelská část adresy rovna nule, bude příkaz **route** předpokládat, že se jedná o síť, v opačném případě ji bude považovat za adresu hostitele. Teď by si ale příkaz **route** myslel, že je adresa 191.72.1.0 adresou hostitele a nikoliv adresou sítě, protože nemůže nic tušit o námi vytvořených podsítích. Tuto informaci mu musíme sdělit explicitně pomocí příznaku `-net`.

Samozřejmě, že vypisování výše uvedeného příkazu **route** je únavné a člověk při něm může udělat chyby. Mnohem pohodlnější je použít názvy sítí, které jsme již nadefinovali v souboru

42 Pozn. překladatele.: Definovanou v souboru `/etc/hosts`.

`/etc/networks`. Tento způsob výrazně zlepší čitelnost příkazu **route** a dokonce můžeme vynechat i parametr `-net`, protože příkaz **route** již bude vědět, že adresa 191.72.1.0 označuje síť.

```
# route add brew-net
```

Po skončení základních konfiguračních kroků se budeme chtít ujistit, že ethernetové rozhraní opravdu funguje správně. Vyberte si nějaký počítač na ethernetové síti, například **vlager,** a zadejte příkaz:

```
# ping vlager
PING vlager: 64 byte packets
64 bytes from 191.72.1.1: icmp_seq=0. time=11. ms
64 bytes from 191.72.1.1: icmp_seq=1. time=7. ms
64 bytes from 191.72.1.1: icmp_seq=2. time=12. ms
64 bytes from 191.72.1.1: icmp_seq=3. time=3. ms
^C
----vstout.vbrew.com PING Statistics
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms) min/avg/max = 3/8/12
```

Pokud neuvidíte výstup podobný výše uvedenému, bude zřejmě něco v nepořádku. Dochází-li k neobvykle vysokým ztrátám paketů, znamená to pravděpodobně hardwarový problém, například špatné nebo chybějící terminátory a podobně. Pokud nepřijmete vůbec žádné pakety, měli byste zkontrolovat konfiguraci rozhraní pomocí příkazu **netstat**, o kterém budeme hovořit později v části *Příkaz netstat*. Statistika paketů zobrazená příkazem **ifconfig** by vám měla říci, zda vůbec byly nějaké pakety rozhraním odeslány. Máte-li přístup i ke vzdálenému hostiteli, měli byste zkontrolovat statistiku rozhraní i na tomto počítači. Tak můžete přesně určit, kde se pakety ztrácejí. Kromě toho byste si měli pomocí příkazu **route** zobrazit směrovací informace a zkontrolovat, zda mají oba hostitelé správně nastavené směrování. Spustíte-li příkaz **route** bez parametrů, vytiskne kompletní směrovací tabulku jádra (parametr `-n` způsobí pouze to, že namísto jmen hostitelů budou vypisovány jejich IP adresy):

```
# route -n
Kernel routing table
Destination  Gateway  Genmask         Flags Metric Ref Use    Iface
127.0.0.1    *        255.255.255.255 UH    1      0      112 lo
172.16.1.0   *        255.255.255.0   U     1      0       10 eth0
```

Podrobný popis jednotlivých údajů naleznete dále v části *Příkaz netstat*. Sloupec Flag obsahuje seznam všech příznaků daného rozhraní. Aktivní rozhraní mají vždy nastaven příznak U (od „up“), příznak H znamená, že cílová adresa patří hostiteli a ne síti. Je-li příznak H zobrazen i u trasy, která má být trasou na síť, musíte položku tabulky upravit a spustit příkaz **route** znovu s parametrem `-net`. To, že se zadaná trasa vůbec používá zjistíte na základě položky Use ve druhém sloupci zprava. S každým voláním příkazu **ping** by se tato hodnota měla zvyšovat.

Směrování pomocí bran

V předešlém textu jsme hovořili o nastavení hostitele v jediné síti Ethernet. Poměrně často se však setkáváme se sítěmi, které jsou pomocí brány připojeny k jiné síti. Tyto brány mohou jednoduše spojovat dva nebo více Ethernetů, mohou ale také zajišťovat spojení s okolním světem, například s Internetem. Abyste mohli využít služeb bran, musíte síťové vrstvě poskytnout doplňující směrovací informace.

Například Ethernety ve společnostech Virtual Brewery a Virtual Winery jsou propojeny pomocí takovéto brány, konkrétně počítačem **vlager**. Za předpokladu, že brána **vlager** je již nakonfigurována nám zbývá pouze přidat do směrovací tabulky hostitele **vstout** další položku, která řekne jádru operačního systému, že všichni hostitelé sítě vinařství jsou dosažitelní přes bránu **vlager**. Příslušné volání příkazu **route** je uvedeno dále, klíčové slovo `gw` mu sděluje, že následující parametr definuje bránu.

```
# route add wine-net gw vlager
```

Samozřejmě že každý hostitel v síti vinařství musí mít odpovídající směrovací záznam k síti pivovaru. V opačném případě byste mohli posílat data pouze z pivovaru do vinařství, počítače v síti vinařství by však nebyly schopné odpovědět.

Tento příklad popisuje pouze bránu, která přenáší pakety mezi dvěma izolovanými Ethernety. Nyní předpokládejme, že brána **vlager** je také připojena k Internetu (například pomocí samostatné ISDN linky). V tom případě budeme chtít, aby *všechny* datagramy, které nejsou určeny přímo pro síť pivovaru, byly směrovány k bráně **vlager**. To lze provést tak, že označíme bránu **vlager** jako implicitní bránu pro hostitele **vstout**:

```
# route add default gw vlager
```

Název sítě **default** je zkratkou adresy 0.0.0.0, která označuje implicitní trasu. Implicitní trasa vede ke každému cíli a bude použita vždy, pokud se nenajde jiná specifičtější trasa. Název **default** nemusíte přidávat do souboru `/etc/networks`, protože je zabudován přímo v příkazu **route**.

Když při volání příkazu **ping** na hostitele za jednou nebo více branami dochází k velké ztrátě paketů, může to znamenat přetíženou síť. Ztráta paketů není ani tak způsobena technickými nedostatky, jako spíše dočasným špičkovým zatížením předávajících hostitelů, které pak přicházející datagramy obsluhují postupně nebo je dokonce zahazují.

Konfigurace brány

Konfigurace počítače pro předávání paketů mezi dvěma Ethernety je poměrně přehledná. Budeme předpokládat, že jsme u brány **vlager**, která je vybavena dvěma ethernetovými kartami, z nichž každá je připojena k jedné ze sítí pivovaru a vinařství. Pak stačí nakonfigurovat obě rozhraní, přidělit jim odpovídající IP adresy, nastavit trasy a to je vše.

Je rozumné přidat informace o obou rozhraních do souboru `hosts`, protože tak pro ně budeme mít jednoduché názvy. Ukazuje to následující příklad:

```
172.16.1.1      vlager.vbrew.com  vlager vlager-if1
172.16.2.1      vlager-if2
```

Sekvence příkazů pro nastavení obou rozhraní je následující:

```
# ifconfig eth0 vlager-if1
# route add brew-net
# ifconfig eth1 vlager-if2
# route add wine-net
```

Rozhraní PLIP

Propojení počítačů pomocí linky PLIP je poněkud odlišné od propojení Ethernetem. Linky PLIP jsou příkladem spojení *point-to-point*, protože na každém konci spojení je pouze jeden počítač.

Sítě jako Ethernet se označují jako *vysílající* sítě. Konfigurace linky point-to-point je odlišná od vysílající sítě, protože samotné point-to-point linky nepodporují vlastní síť.

Protokol PLIP nabízí levné a jednoduché propojení mezi počítači. Jako příklad budeme uvažovat laptop nějakého zaměstnance pivovaru, který je s branou **vlager** propojen protokolem PLIP. Tento laptop se jmenuje **vlite** a má pouze jediný paralelní port. Při zavádění systému bude tento port registrován jako rozhraní `plip1`. Abyste spojení aktivovali, musíte nakonfigurovat rozhraní `plip1` následujícími příkazy[43]:

```
# ifconfig plip1 vlite pointopoint vlager
# route add default gw vlager
```

První příkaz nastavuje rozhraní a sděluje jádru operačního systému, že se jedná o spojení typu point-to-point, u něhož má vzdálená strana přidělenu adresu **vlager**. Druhý řádek nainstaluje implicitní trasu s využitím hostitele **vlager** jako brány. Na straně brány **vlager** je nutná podobná konfigurace, která spojení zaktivuje (volání příkazu **route** není zapotřebí):

```
# ifconfig plip1 vlager pointopoint vlite
```

Zajímavé je, že rozhraní `plip1` brány **vlager** nemusí mít samostatnou IP adresu, ale může mu být také přidělena adresa 172.16.1.1. Sítě typu point-to-point nepředstavují samostatné sítě, takže jejich rozhraní nevyžadují přidělení samostatné adresy. Aby se předešlo nejasnostem, využívá jádro ve směrovací tabulce informace o použitém rozhraní[44].

Nyní máme vyřešeno směrování z laptopu do sítě pivovaru; zatím nám však chybí trasa ze všech počítačů sítě pivovaru na počítač **vlite**. Jeden z poměrně nešikovných způsobů spočívá v přidání této konkrétní trasy do směrovacích tabulek všech hostitelů, přičemž budeme **vlager** definovat jako bránu k počítači **vlite**:

```
# route add vlite gw vlager
```

Lepším řešením dočasných tras je dynamické směrování. Jeden z možných způsobů spočívá v použití směrovacího démona **gated**, který musí být nainstalován na každém hostiteli v síti, a ty si pak budou dynamicky předávat směrovací informace. Nejjednodušší způsob je ale použít *ARP proxy*. Při nainstalovaném ARP proxy bude brána **vlager** odpovídat na libovolné dotazy ohledně hostitele **vlite** zasláním své vlastní ethernetové adresy. Všechny pakety pro **vlite** tak skončí na bráně **vlager**, která je pošle na laptop. K problematice ARP proxy se vrátíme v části *Kontrola tabulek ARP*.

Současné verze balíku `net-tools` obsahují nástroj **plipconfig**, který umožuje nastavit parametry časování PLIP rozhraní. IRQ paralelního portu lze nastavit příkazem **ifcongif**.

Rozhraní SLIP a PPP

Ačkoliv jsou spojení typu SLIP nebo PPP pouze jednoduchými spojeními typu point-to-point, je třeba k nim říct daleko více. Použití SLIP spojení totiž v sobě obvykle zahrnuje vytočení čísla vzdálené sítě prostřednictvím modemu a přepnutí sériové linky do režimu SLIP. Podobně se používá i protokol PPP. Podrobněji budeme o protokolech SLIP a PPP hovořit v kapitolách 7 a 8.

---

43 Klíčové slovo **pointopoint** není překlep. Skutečně se zapisuje takto.

44 Kvůli jistotě byste měli linky PLIP nebo SLIP konfigurovat až poté, co budete mít vytvořeny kompletní směrovací tabulky pro celý Ethernet. U některých starších jader by jinak mohly později doplňované trasy skončit na lince point-to-point.

Fiktivní rozhraní

Fiktivní (*dummy*) rozhraní je trošku exotické, nicméně docela užitečné. Jeho hlavní výhoda vynikne u samostatných počítačů a u počítačů, jejichž jediné síťové spojení je připojení pomocí modemu. Koneckonců, tyto počítače představují po většinu času rovněž samostatné počítače.

Dilema u samostatných hostitelů spočívá v tom, že mají aktivní pouze jediné síťové zařízení, a to konkrétně lokální zařízení, které má obvykle přidělenu adresu 127.0.0.1. Nicméně v určitých situacích potřebujete posílat data na „oficiální" IP adresu místního hostitele. Uvažujme například laptop **vlite**, který je momentálně odpojen od sítě. Aplikace na laptopu **vlite** může chtít poslat nějaká data jiné aplikaci na tomtéž počítači. Prohledá-li soubor `/etc/hosts/`, získá IP adresu 172.16.1.65 a na ni se pokusí poslat data. Protože jediným aktivním rozhraním je v dané chvíli pouze lokální rozhraní, nemůže jádro vědět, že adresa 172.16.1.65 je jeho vlastní adresa! Jádro tedy datagram zruší a vrátí aplikaci chybové hlášení.

V tomto bodě vstupuje do hry fiktivní rozhraní. Vyřeší dilema tím, že bude sloužit jako „alernativní ego" lokálního rozhraní. V případě laptopu **vlite** bychom mu přidělili adresu 172.16.1.65 a přidali trasu, která na tuto adresu povede. Všechny datagramy na adresu 172.16.1.65 pak budou doručeny lokálně. Správné volání vypadá takto[45]:

```
# ifconfig dummy vlite
# route add vlite
```

IP aliasy

Nová jádra podporují funkci, kterou je možné použití fiktivního rozhraní úplně vyloučit, nemluvě o některých dalších užitečných možnostech. Pomocí *IP aliasů* je možné jednomu fyzickému zařízení přidělit více IP adres. V nejjednodušším případě můžete nahradit fiktivní rozhraní tak, že lokálnímu rozhraní přidělíte jako alias i oficiální IP adresu počítače. Ve složitějších případech můžete počítač nakonfigurovat tak, že se bude chovat jako několik různých počítačů s vlastními IP adresami. Této konfiguraci se někdy říká „virtuální hostitel" i když tento termín označuje i některé jiné techniky[46].

Chcete-li nějakému rozhraní přidělit alias, musíte nejprve ověřit, zda máte jádro přeloženo s podporou IP aliasů (zkontrolujte, zda existuje soubor `/proc/net/ip_alias`; pokud ne, musíte jádro znovu přeložit). Konfigurace IP aliasu je prakticky stejná jako konfigurace skutečného síťového zařízení, pouze použijete speciální název zařízení aby se vědělo, že jde o požadovaný alias. Například:

```
# ifconfig lo:0 172.16.1.1
```

Tímto příkazem přidělíte lokálnímu zařízení alias `172.16.1.1`. Na IP aliasy se odkazuje přidáním sekvence `:n` k názvu zařízení, kde *n* je celé číslo. V našem příkladu jsme vytvořili nultý alias zařízení `lo`. Jednomu fyzickému zařízení tak můžeme přiřadit více aliasů.

Každý z aliasů můžeme chápat jako samostatné fyzické zařízení a co se týče síťového sofwaru jádra, tak se tak i chovají – nicméně tato různá rozhraní sdílejí hardware s ostatními.

45 Pokud jste fiktivní zařízení nahráli jako samostatný modul a nemáte je vestavěno přímo v jádře, bude se jmenovat `dummy0`. Je to dáno tím, že těchto modulů můžete nahrát více a budou pak fungovat jako více fiktivních zařízení.

46 V poslední době se IP aliasy označují jako virtuální hostitelé na úrovni síťové vrstvy. U služeb WWW a SMTP se častěji používají virtuální hostitelé na úrovni aplikační vrstvy, kdy jedna IP adresa slouží více virtuálním hostitelům a v požadavích příslušné aplikační vrstvy se předává název konkrétního hostitele. Služby jako FTP tímto způsobem fungovat neumějí a vyžadují virtuální hostitele na úrovni síťové vrstvy.

```
# ifconfig dummy vlite
# route add vlite
```

Vše o příkazu ifconfig

Příkaz **ifconfig** má mnohem více parametrů, než jsme si zatím uvedli. Klasické volání tohoto příkazu vypadá takto:

```
ifconfig rozhraní [adresa [parametry]]
```

Parametr `rozhraní` představuje název rozhraní a parametr `adresa` představuje IP adresu přidělovanou tomuto rozhraní. Může to být buď IP adresa v tečkové notaci nebo jméno, které si příkaz **ifconfig** vyhledá v souboru `/etc/hosts`.

Je-li příkaz **ifconfig** vyvolán pouze s názvem rozhraní, zobrazí konfiguraci příslušného rozhraní. Je-li spuštěn bez parametrů, zobrazí všechna již dříve nakonfigurovaná rozhraní; volba `-a` vede ke zobrazení i neaktivních rozhraní. Příklad volání pro ethernetové rozhraní `eth0` vypadá takto:

```
# ifconfig eth0
eth0      Link encap 10Mbps Ethernet  HWaddr 00:00:C0:90:B3:42
          inet addr 172.16.1.2 Bcast 172.16.1.255 Mask 255.255.255.0
          UP BROADCAST RUNNING  MTU 1500  Metric 0
          RX packets 3136 errors 217 dropped 7 overrun 26
          TX packets 1752 errors 25 dropped 0 overrun 0
```

Údaje `MTU` a `Metrics` ukazují nastavené MTU a hodnotu metriky rozhraní. Hodnota metriky se v některých operačních systémech tradičně používá k výpočtu ceny dané trasy. Linux tuto hodnotu prozatím nevyužívá, nicméně z důvodů kompatibility ji definuje.

Řádky `RX` a `TX` ukazují, kolik paketů bylo bezchybně přijato nebo vysláno, ke kolika chybám došlo, kolik paketů bylo zahozeno (pravděpodobně z důvodu nedostatku paměti) a kolik paketů se ztratilo kvůli přetížení. K přetížení přijímače obvykle dojde v případě, kdy pakety přicházejí rychleji, než stačí jádro operačního systému obslouhovat přerušení. Hodnoty příznaků zobrazené příkazem **ifconfig** zhruba odpovídají názvům jeho voleb na příkazovém řádku, které si vysvětlíme později.

Následuje seznam parametrů rozeznávaných příkazem **ifconfig** společně s odpovídajícími názvy příznaků. Volby zapínající nějakou funkci umožňují i její vypnutí pomocí znaku minus (`-`) před názvem volby.

`up` — Označí rozhraní jako přístupné pro IP vrstvu. Tato volba je použita automaticky, je-li na příkazovém řádku definována adresa rozhraní. Lze ji také použít k opětovné aktivaci rozhraní, které bylo dočasně deaktivováno pomocí volby `down`. Tato volba odpovídá příznakům `UP` a `RUNNING`.

`down` — Označí rozhraní jako nepřístupné pro IP vrstvu. Tato volba vypne jakýkoliv IP provoz přes dané rozhraní. Volba rovněž automaticky smaže všechny směrovací záznamy používající dané rozhraní.

`netmask` *`maska`* — Tato volba přidělí masku podsítě, kterou bude rozhraní používat. Může být zadána jako 32bitové hexadecimální číslo s předponou `0x` nebo jako čtveřice dekadických čísel oddělených tečkou. I když je tečkový formát obvyklejší, s hexadecimálním se snáze pracuje. Síťová maska je z podstaty binární a převody mezi dvojkovou a šestnáctkovou soustavou jsou jednodušší než mezi dvojkovou a desítkovou.

`pointopoint` *adresa* — Tato volba se používá u IP spojení typu point-to-point, které zahrnuje pouze dva hostitele. Je nutná například při konfiguraci rozhraní SLIP nebo PLIP. (Byla-li nastavena adresa point-to-point, zobrazí příkaz **ifconfig** příznak `POINTOPOINT`.)

`broadcast` *adresa* — Vysílací adresa je obvykle vytvořena z čísla sítě nastavením všech bitů hostitelské části na jedničku. Některé implementace protokolu IP (například BSD 4.2) používají odlišné schéma, při němž jsou naopak všechny bity hostitelské části nulové. Volba `broadcast` slouží pro přizpůsobení těmto podivným prostředím. (Byla-li nastavena vysílací adresa, zobrazí příkaz **ifconfig** příznak `BROADCAST`.)

`irq` — Tento parametr umožňuje u některých zařízení nastavit číslo používaného přerušení. Je to užitečné zejména pro PLIP rozhraní, hodí se však i pro některé ethernetové karty.

`metric` *číslo* — Tato volba slouží k přiřazení metriky záznamu daného rozhraní ve směrovací tabulce. Hodnotu metriky používá protokol RIP (Route Information Protocol) k vytvoření směrovacích tabulek sítí[47]. Implicitní hodnota metriky používaná příkazem ifconfig je rovna nule. Pokud nepoužíváte démona RIP, je vám tato volba k ničemu; pokud ano, budete jen zřídka potřebovat tuto hodnotu měnit.

`mtu` *bajtů* — Tato volba nastavuje maximální přenosovou jednotku (MTU), která představuje maximální počet bajtů, které rozhraní dokáže obsloužit v jedné transakci. U sítí typu Ethernet je hodnota MTU implicitně nastavena na 1 500 (největší povolená délka ethernetového paketu); u rozhraní typu SLIP je MTU nastavena na hodnotu 296. (Pro nastavení MTU na linkách SLIP nejsou žádná omezení, hodnota 296 představuje vhodný kompromis.)

`arp` — Toto je volba typická pro sítě, jako je Ethernet nebo radiová síť. Povoluje použití protokolu ARP k detekci fyzických adres hostitelů, kteří jsou připojeni k síti. U vysílajících sítí je tato volba implicitně povolena. Je-li protokol ARP zakázán, zobrazí příkaz **ifconfig** příznak `NOARP`.

`-arp` — Zakáže na tomto rozhraní používat protokol ARP.

`promisc` — Přepne rozhraní do promiskuitního režimu. U vysílajících sítí to bude znamenat, že dané rozhraní bude přijímat všechny pakety bez ohledu na to, zda byly určeny pro jiného hostitele či nikoliv. Tato volba umožní analyzovat síťový provoz pomocí paketových filtrů (takzvané *sledování Ethernetu*). Je to dobrý způsob k nalezení síťových problémů, které jsou jinak těžko odhalitelné. Na druhou stranu umožní tato volba útočníkům kromě jiných věcí i zjištění potřebných hesel z vašeho síťového provozu. Jednou z ochran proti tomuto typu útoku je všeobecný zákaz připojení jakéhokoliv počítače do vaší ethernetové sítě. Můžete také použít zabezpečené autentikační protokoly, jako je například Kerberos nebo balík secure shell[48]. Této volbě odpovídá příznak `PROMISC`.

47 Protokol RIP vybírá optimální trasu k hostiteli na základě „délky“ tras. Délka se počítá jako součet jednotlivých metrik rozhraní, přes která trasa vede. Standardně má cesta přes jednu bránu délku 1, může však mít přiřazenu jakoukoliv celou hodnotu do 16. (Trasa o délce 16 odpovídá nekonečnu. Tyto trasy jsou považovány za nepoužitelné.) Cenu uzlu nastavuje právě parametr `metric`, a ta je pak vysílána směrovacím démonem.

48 ssh můžete získat na adrese **ftp.cs.hut.fi** v adresáři `/pub/ssh`.

`-promisc` Tato volba vypne promiskuitní režim.

`allmulti` Multicastové adresy se podobají vysílacím adresám, nepřijímají je však automaticky všechny počítače, ale pouze ty, které jsou tak nastaveny. Je to užitečné pro aplikace jako jsou ethernetové videokonference nebo přenos zvuku, kdy přijímají pouze ti, které to zajímá. Multicastové adresy jsou podporovány většinou ethernetových ovladačů, ne však všemi. Je-li tato volba zapnuta, rozhraní přijímá a předává ke zpracování multicastové pakety. Této volbě odpovídá příznak `ALLMULTI`.

`-allmulti` Tato volba vypne multicastové adresy.

Příkaz netstat

Příkaz **netstat** je užitečný nástroj pro kontrolu konfigurace a aktivity sítě. Jde vlastně o sbírku několika nástrojů spojených dohromady. V následujících částech probereme jeho jednotlivé funkce.

Zobrazení směrovací tabulky

Spustíte-li příkaz **netstat** s parametrem `-r`, zobrazí se směrovací tabulka jádra stejně, jako to dělá příkaz **route**. Na hostiteli **vstout** se zobrazí:

```
# netstat -nr
Kernel IP routing table
Destination   Gateway      Genmask         Flags  MSS Window  irtt Iface
127.0.0.1     *            255.255.255.255 UH       0 0          0 lo
172.16.1.0    *            255.255.255.0   U        0 0          0 eth0
172.16.2.0    172.16.1.1   255.255.255.0   UG       0 0          0 eth0
```

Volba `-n` způsobí, že příkaz **netstat** zobrazí adresy jako čísla a ne jako symbolické názvy hostitelů a sítí. To je užitečné zejména chcete-li se vyhnout vyhledávání adres po síti (například na DNS nebo NIS serveru).

Druhý sloupec výstupu příkazu **netstat** zobrazuje bránu, na níž trasa směruje. Není-li použita žádná brána, zobrazí se hvězdička. Třetí sloupec ukazuje „obecnost" trasy, tedy síťovou masku trasy. Při hledání odpovídající trasy pro nějakou IP adresu jádro projde všechna data směrovací tabulky, mezi adresou a maskou aplikuje binární AND a výsledek porovnává s cílem trasy v tabulce[49].

Čtvrtý sloupec zobrazuje různé symboly, které popisují trasu:

`G` Směrování používá bránu.

`U` Používané rozhraní je aktivní.

`H` Daná trasa vede pouze k jedinému hostiteli. To je například případ lokálního rozhraní 127.0.0.1.

`D` Trasa byla vytvořena dynamicky. Tento příznak se nastavuje, pokud byla položka směrovací tabulky vytvořena směrovacím démonem jako je **gated** nebo ICMP zprávou o přesměrování (viz část *Protokol ICMP* v kapitole 2).

`M` Příznak se nastavuje, byla-li trasa upravena ICMP zprávou o přesměrování.

`!` Jedná se o zamítnutou trasu a datagramy přes ni posílané budou zahazovány.

49 Pozn. překladatele: A samozřejmě vrátí tu trasu, kde je maskovaná adresa požadovaného cíle stejná jako adresa cíle v tabulce. Vyhovuje-li témuž cíli více tras (což je skoro vždycky), vrátí tu trasu, která má „nejdelší" masku (tedy masku s nejvíce jedničkami) – taková trasa je považována za „nejspecifičtější" a předpokládá se, že vede „nejlépe" k cíli.

Další tři sloupce zobrazují hodnoty MSS, Window a irtt, které budou použity pro TCP spojení přes danou trasu. Hodnota MSS znamená Maximum Segment Size a představuje velikost nejdelšího datagramu, který může jádro pro přenos po této trase vytvořit. Window představuje maximální objem dat, který je systém schopen ze vzdáleného hostitele přijmout „v jednom zátahu“. Zkratka `irtt` znamená „initial round trip time“ – počáteční čas pro přenos tam a zpět. Protokol TCP zaručuje, že data budou spolehlivě doručena a v případě ztráty datagramu zajistí jeho opakované odeslání. Protokol TCP si pamatuje jak dlouho trvá, než bude datagram doručen na vzdálený konec spojení a než od něj přijde potvrzení o jeho přijetí – tento údaj je právě round-trip time[50] a pokud se do té doby potvrzení neobjeví, protokol předpokládá ztrátu datagramu a odešle jej znovu. Jeho původní hodnotu používá protokol TCP při prvním navázání spojení. Ve většině sítí je standardně nastavená hodnota vyhovující, nicméně u některých pomalých sítí (například radioamatérských) bývá příliš krátká a vede ke zbytečnému opakovanému odesílání datagramů. Hodnotu `irtt` je možné nastavit příkazem **route**. Nastavením hodnoty `0` se indikuje, že se má použít standardní hodnota.

Konečně v posledním sloupci je zobrazeno síťové rozhraní, přes něž trasa vede.

Sloupec Ref ve výstupu příkazu netstat zobrazuje počet odkazů na dané směrování, to znamená, kolik dalších směrování (například přes brány) spoléhá na přítomnost daného směrování. Poslední dva sloupce zobrazují, kolikrát byla použita směrovací data a dále rozhraní, kterými při doručování procházejí datagramy.

Zobrazení statistik rozhraní

Spuštěním příkazu **netstat** s parametrem `-i` dojde k vypsání statistik všech nakonfigurovaných rozhraní. Pokud je uveden i parametr `-a`, zobrazí se statistiky *všech* rozhraní v jádře, nejenom těch doposud nakonfigurovaných. Na počítači **vstout** by mohl výstup příkazu **netstat** vypadat takto:

```
# netstat -i
Kernel Interface table
Iface MTU Met  RX-OK RX-ERR RX-DRP RX-OVR  TX-OK TX-ERR TX-DRP TX-OVR Flags
lo      0   0   3185      0      0      0   3185      0      0      0 BLRU
eth0 1500   0 972633     17     20    120 628711    217      0      0 BRU
```

Údaje `MTU` a `Met` uvádějí aktuální hodnoty MTU a metriky daného rozhraní. Sloupce `RX` a `TX` ukazují, kolik paketů bylo přijato a vysláno bezchybně (`RX-OK/TX-OK`), kolik bylo poškozeno (`RX-ERR/TX-ERR`), kolik bylo zahozeno (`RX-DRP/TX-DRP`) a kolik se ztratilo z důvodu přetížení (`RX-OVR/TX-OVR`).

Poslední sloupec zobrazuje příznaky daného zařízení. Jsou to jednoznakové ekvivalenty dlouhých názvů příznaků, které se vypisují při zobrazení konfigurace rozhraní pomocí příkazu **ifconfig**:

- `B` Byla nastavena vysílací adresa.
- `L` Dané rozhraní je lokální rozhraní.
- `M` Jsou přijímány všechny pakety (promiskuitní režim).
- `O` Pro dané rozhraní je vypnut protokol ARP.
- `P` Toto spojení je typu point-to-point.
- `R` Rozhraní právě běží.
- `U` Rozhraní je povoleno.

50 Pozn. překladatele: Round-trip asi opravdu nejde přeložit jinak než „tam a zpět“. Doslova to znamená „kruhový výlet“ a ve spojení „round-trip ticket“ to je zpáteční jízdenka. (Která je obvykle levnější než „one-way ticket“.)

Zobrazení spojení

Příkaz **netstat** podporuje skupinu voleb pro zobrazení aktivních nebo pasivních soketů. Volby `-t`, `-u`, `-w` a `-x` ukazují aktivní TCP, UDP, RAW nebo unix-soketová spojení. Pokud k nim doplníte i parametr `-a`, budou zobrazeny i sokety čekající na spojení (tedy naslouchající). Tato kombinace parametrů vám poskytne úplný výpis všech serverů, které právě běží ve vašem systému.

Při použití příkazu **netstat –ta** se na hostiteli **vlager** zobrazí následující výpis:

```
$ netstat -ta
Active Internet Connections
Proto Recv-Q Send-Q Local Address    Foreign Address    (State)
tcp        0      0 *:domain         *:*                LISTEN
tcp        0      0 *:time           *:*                LISTEN
tcp        0      0 *:smtp           *:*                LISTEN
tcp        0      0 vlager:smtp      vstout:1040        ESTABLISHED
tcp        0      0 *:telnet         *:*                LISTEN
tcp        0      0 localhost:1046   vbardolino:telnet  ESTABLISHED
tcp        0      0 *:chargen        *:*                LISTEN
tcp        0      0 *:daytime        *:*                LISTEN
tcp        0      0 *:discard        *:*                LISTEN
tcp        0      0 *:echo           *:*                LISTEN
tcp        0      0 *:shell          *:*                LISTEN
tcp        0      0 *:login          *:*                LISTEN
```

Tento výpis ukazuje, že většina serverů čeká na příchozí spojení. Nicméně čtvrtý řádek ukazuje příchozí spojení typu SMTP z hostitele **vstout** a šestý řádek vám sděluje, že existuje odchozí spojení typu telnet s hostitelem **vbardolino**[51].

Pokud bychom použili pouze argument –a, zobrazí se všechny sokety ze všech protokolových rodin.

Kontrola tabulek ARP

V určitých situacích je vhodné si prohlédnout, případně změnit obsah tabulek ARP jádra systému – například máte-li podezření, že příčinou občasných síťových problémů je duplicitní IP adresa. Pro takovéto situace byl vytvořen nástroj **arp**. Jeho volby jsou následující:

```
arp [-v] [-t hwtype] -a [hostname]
arp [-v] [-t hwtype] -s hostname hwaddr
arp [-v] -d hostname [hostname...]
```

Všechny parametry `hostname` (název hostitele) mohou být zadány jako IP adresy nebo jako symbolické názvy.

První typ volání příkazu **arp** zobrazí pro danou IP adresu nebo hostitele položku v ARP tabulce. Nebyl-li zadán název hostitele, zobrazí se všechny položky tabulky. Například na počítači **vlager** můžeme tímto příkazem dostat:

```
# arp -a
IP address       HW type                 HW address
```

51 Zda je spojení odchozí poznáte z čísla portu v lokální adrese. Čísla portů u *odchozího* spojení budou vždy nějaká obecná celá čísla. V případě *příchozího* spojení bude použito známé číslo portu dané služby a pro ně program **netstat** zobrazuje symbolické názvy služeb jako třeba `smtp`, které jsou definovány v souboru `/etc/services`.

```
172.16.1.3      10Mbps Ethernet         00:00:C0:5A:42:C1
172.16.1.2      10Mbps Ethernet         00:00:C0:90:B3:42
172.16.2.4      10Mbps Ethernet         00:00:C0:04:69:AA
```

Vidíme zde ethernetové adresy hostitelů **vlager**, **vstout** a **vale**.

Pomocí volby `-t` můžete omezit výpis pouze na zadaný typ hardwarových zařízení. Ten může být `ether`, `ax25` nebo `pronet`, což odpovídá (po řadě) 10 Mb Ethernetu, zařízením AMPR AX.25 a zařízení IEEE 802.5 Token Ring.

Parametr `-s` slouží k trvalému přidání ethernetové adresy hostitele do tabulky. Parametr `hwaddr` určuje hardwarovou adresu, u níž se implicitně předpokládá, že jde o ethernetovou adresu určenou šesti hexadecimálními bajty oddělenými dvojtečkami. Pomocí volby `-t` můžete nastavit hardwarové adresy pro jiné typy hardwaru.

Z různých důvodů mohou ARP dotazy selhávat, například je-li na vzdáleném hostiteli chybný ovladač ARP, nebo když v síti existuje další hostitel, který se chybně identifikuje IP adresou vzdáleného hostitele. V takovém případě musíte IP adresu problematického počítače přidat do tabulky ručně. Ruční zadávání údajů do ARP tabulky je rovněž (velice drastické) opatření, kterým se můžete chránit proti hostitelům, jež se vydávají za někoho jiného.

Použijete-li při spuštění příkazu **arp** parametr `-d`, smažou se z tabulky všechna data, která se vztahují k danému hostiteli. Pomocí tohoto parametru můžete rozhraní přinutit k tomu, aby se znovu pokusilo pomocí dotazu získat ethernetovou adresu odpovídající dané IP adrese. Je to užitečné v případě, kdy špatně nakonfigurovaný systém vyslal do ARP tabulky špatné informace (samozřejmě předtím musíte chybného hostitele znovu zkonfigurovat).

Volba `-s` slouží k implementaci techniky *ARP proxy*. Je to speciální technika, kdy se nějaký hostitel, řekněme **gate**, chová jako brána pro jiného hostitele, řekněme **fnord**, a předstírá, že IP adresy obou hostitelů odpovídají hardwarové adrese brány. Provede to tak, že brána zveřejní ARP položku hostitele **fnord**, která bude ukazovat na její vlastní ethernetové rozhraní. Když nyní nějaký hostitel pošle ARP dotaz na hostitele **fnord**, hostitel **gate** vrátí odpověď, která bude obsahovat jeho vlastní ethernetovou adresu. Dotazující se hostitel pak všechny datagramy pro **fnord** posílá na hostitele **gate**, který je samozřejmě musí směrovat na **fnord**.

Tyto záměny jsou nutné například v případě, kdy chcete přistupovat k hostiteli **fnord** z počítače s DOSem, který nemá zcela korektní implementaci TCP a nedokáže správě směrovat. Při použití ARP proxy se bude dosovskému počítači **fnord** jevit jako počítač na lokální síti a on proto nemusí umět směrovat přes brány.

Další velice užitečnou aplikací techniky ARP proxy je případ, kdy se jeden z vašich hostitelů chová jako brána vůči nějakému jinému hostiteli jen dočasně, například při připojení pomocí modemu. V předešlém příkladu jsme se již setkali s laptopem **vlite**, který je občas připojen k bráně **vlager** spojením typu PLIP. Samozřejmě, že tento způsob bude fungovat pouze v případě, kdy je adresa hostitele, na kterém chcete provozovat techniku ARP proxy, ve stejné podsíti jako vaše brána. Například hostitel **vstout** by mohl používat techniku ARP proxy pro libovolného hostitele sítě pivovaru (172.16.1.0), ale nikdy pro hostitele z podsítě vinařství (172.16.2.0).

Správné volání příkazu **arp**, kdy bude hostiteli **fnord** poskytnuto ARP proxy je uvedeno dále; zadaná ethernetová adresa musí samozřejmě odpovídat hostiteli **gate**:

```
# arp -s fnord 00:00:c0:a1:42:e0 pub
```

Položku ARP proxy můžete odstranit následujícím způsobem:

```
# arp -d fnord
```

KAPITOLA 6

Služby jmen a konfigurace resolveru

Ve druhé kapitole jsme si řekli, že sítě na bázi protokolu TCP/IP mohou používat různá schémata konverze názvů na adresy. Nejjednodušším způsobem je tabulka hostitelů v souboru `/etc/hosts`. Tento způsob je vhodný pouze pro malé lokální sítě, které spravuje jediný správce a které nepoužívají žádnou IP komunikaci s okolním světem. Formát souboru `hosts` jsme popsali v kapitole 5.

Další možností převodu názvů na IP adresy je použití služby BIND (*Berkeley Internet Name Domain*). Konfigurace služby BIND je skutečným oříškem, ale jakmile ji jednou zvládnete, bude zohlednění jakýchkoliv změn v síti velmi jednoduché. V systému Linux, stejně jako u dalších unixových systémů, poskytuje jmenné služby program **named**. Po svém spuštění si program načte do své vyrovnávací paměti obsah několika hlavních souborů a pak bude čekat na dotazy od vzdálených nebo místních uživatelských procesů. Existuje několik způsobů jak službu BIND nastavit a ne všechny vyžadují spuštěný jmenný server na každém hostiteli.

V této kapitole můžeme nabídnout jen o málo více než jen hrubý náčrt způsobu, jakým lze provozovat jmenný server. Náš popis by vám měl stačit, pokud provozujete malou lokální síť s připojením do Internetu. Nejnovější informace o službě BIND najete v jejím zdrojovém balíku, který obsahuje manuálové stránky, popis verze a Příručku operátora BIND. Nelekejte se názvu, jde o velice užitečný dokument. Podrobnější popis služby DNS a související problematiky najdete například v knize „DNS and BIND“ Paula Albitze a Cricketa Liua (vydalo nakladatelství O'Reilly). Otázkám systému DNS se věnuje i konference nazvaná **comp.protocols.tcp-ip.domains**. Technické podrobnosti o DNS naleznete v definičních dokumentech RFC 1033, 1034 a 1035.

Knihovna resolveru

Hovoříme-li o *resolveru*, nemáme na mysli žádnou speciální aplikaci, ale knihovnu resolveru. Jedná se o skupinu funkcí nacházející se ve standardní knihovně jazyka C. Centrálními funkcemi knihovny jsou procedury `gethostbyname(2)` a `gethostbyaddr(2)`, které dokáží nalézt IP adresu odpovídající danému názvu a naopak. Mohou být nastaveny tak, aby pouze využívaly soubor

hosts, aby se dotazovaly různých jmenných serverů nebo aby používaly databázi *hosts* služby NIS (Network Information Service).

Po svém volání si funkce resolveru přečtou své konfigurační soubory. Z těchto souborů poznají koho se mají dotazovat, v jakém pořadí a dozvědí se i další podrobnosti o prostředí, v němž pracují. Starší standardní knihovna Linuxu, libc, používala jako hlavní konfigurační soubor /etc/host.conf, verze 2 standardní knihovny GNU používá soubor /etc/nsswitch.conf. Popíšeme si oba dva, protože oba se běžně používají.

Soubor host.conf

Soubor /etc/host.conf říká starším verzím Linuxu, které služby má resolver používat a v jakém pořadí.

Jednotlivé volby musí být v souboru host.conf uvedeny na samostatných řádcích. Pole mohou být oddělena oddělovači (mezery nebo tabulátory). Symbol # označuje komentář, který končí u prvního znaku nové řádky. K dispozici jsou následující volby:

order — Tato volba určuje pořadí, ve kterém budou použity jednotlivé služby resolveru. Přípustné možnosti jsou *bind* pro použití dotazů na jmenný server, *hosts* pro vyhledávání v souboru /etc/hosts a *nis* pro vyhledávání pomocí služby NIS. Může být zadána buď jedna nebo více voleb. Pořadí, ve kterém jsou tyto volby uvedeny, bude určovat pořadí, v kterém budou volány jim odpovídající služby.

multi — Hodnota této volby může být on nebo off. Určuje, zda může mít hostitel v souboru /etc/hosts několik IP adres. Počítače tohoto typu se označují jako *multihomed* systémy. U dotazů systému DNS nebo NIS nemá tento přepínač žádný význam.

nospoof — O této volbě budeme hovořit v části *Reverzní hledání*. Systém DNS umí najít název hostitele náležející dané IP adrese prostřednictvím domény **in-addr.arpa**. Snaha jmenného serveru vrátit chybný název hostitele se označuje jako takzvaný *spoofing*. Obrana proti tomuto typu útoku spočívá v tom, že resolver nakonfigurujeme tak, že zpětným dotazem zjistí, že IP adrese vrácené jmenným serverem opravdu odpovídá název, který jsme hledali. Pokud neodpovídá, bude název odmítnut a resolver vrátí chybové hlášení. Toto chování se zapíná pomocí volby *nospoof on*.

alert — Tato volba pracuje s parametry on nebo off. Je-li zapnutá, pak veškeré pokusy o spoofing (viz výše) budou zaznamenány prostřednictvím funkce *syslog*.

trim — Tato volba má jako parametr název domény, která bude před vyhledáváním z názvů hostitelů odstraněna. Volba je užitečná u těch položek v souboru hosts, u kterých jsme zadali pouze názvy hostitelů bez názvu místní domény. Při vyhledávání lokálního hostitele u nějž je název místní domény uveden bude její název odstraněn a vyhledání v souboru /etc/hosts tak může uspět. Volbu *trim* lze použít vícekrát, takže počítač se může chovat jako lokální na více doménách.

V příkladu 6.1 je uveden příklad souboru host.conf pro počítač **vlager**.

```
# /etc/host.conf
# Používáme named, ale ne NIS
order   bind,hosts
# Povolujeme více adres
multi   on
# Ochrana před spoofingem
nospoof on
# Odříznutí lokální domény (není nutné).
trim    vbrew.com.
```

Proměnné prostředí resolveru

Nastavení v souboru `host.conf` lze přepsat pomocí několika proměnných prostředí resolveru:

`RESOLV_HOST_CONF`	Určuje soubor, který bude načten místo souboru `/etc/host.conf`.
`RESOLV_SERV_ORDER`	Přepíše nastavené volby *order* v souboru `host.conf`. Službami mohou být *hosts*, *bind* nebo *nis*. Odděleny mohou být mezerou, čárkou nebo středníkem.
`RESOLV_SPOOF_CHECK`	Určí opatření, která budou použita proti spoofingu. Při hodnotě *off* je kontrola úplně vypnutá. Hodnoty *warn* a *warn off* kontrolují spoofing, a zapíná se respektive vypíná se záznam detekovaných pokusů. Hodnota * bude kontrolovat spoofing, ale rozsah zaznamenávání chyb ponechá nastavený podle souboru `host.conf`.
`RESOLV_MULTI`	Přepisuje volbu *multi* v souboru `host.conf`. Proměnná může mít hodnoty *on* nebo *off*.
`RESOLV_OVERRIDE_TRIM_DOMAINS`	Tato proměnná určuje názvy domén a přepisuje nastavení volby *trim* v souboru `host.conf`.
`RESOLV_ADD_TRIM_DOMAINS`	Tato proměnná doplňuje nastavení volby *trim* o další odřezávané domény.

Soubor nsswitch.conf

Verze 2 standardní knihovny GNU Libc obsahuje mocnější a pružnější náhradu za původní mechanismus souboru `host.conf`. Služby jmen byly doplněny o možnost poskytovat i jiné druhy údajů. Konfigurační volby všech těchto různých funkcí prohlížejících různé databáze byly soustředěny v jednom konfiguračním souboru `nsswitch.conf`.

Pomocí souboru `nsswitch.conf` může administrátor nakonfigurovat různé databáze. My se omezíme pouze na ty parametry, které se týkají vzájemného převodu názvů a IP adres. Informace o dalších službách snadno naleznete v dokumentaci ke standardní knihovně GNU.

Volby v souboru `nsswitch.conf` musí být uvedeny na samostatných řádcích a jednotlivé údaje se od sebe oddělují mezerami nebo tabulátory. Znak # označuje začátek komentáře, který pokračuje až do konce daného řádku. Každý řádek popisuje jednu službu, rozlišování názvů je jednou z nich. Prvním údajem na řádku je název prohledávané databáze, ukončený dvojtečkou. Databáze zajišťující převod mezi jmény a adresami má název hosts. Souvisí s ní i databáze networks, která slouží k převodu síťových názvů na adresy sítí. Ve zbytku řádku jsou parametry udávající způsob, jakým se budou hodnoty databáze zjišťovat.

Můžeme použít následující volby:

`dns`	Adresy se zjišťují pomocí služby DNS. Tato volba má smysl pouze pro rozlišování názvů hostitelů, nikoliv názvů sítí. Tento mechanismus se dále nastavuje souborem `/etc/resolv.conf`, o kterém budeme hovořit dále.
`files`	Hledá adresy hostitelů a sítí v lokálních souborech `/etc/hosts` a `/etc/networks`.
`nis` nebo `nisplus`	K rozlišování názvů používá službu NIS. O službách NIS a NIS+ budeme podrobně hovořit v kapitole 13.

Pořadí v němž jsou volby na řádku uvedeny udává, v jakém pořadí budou jednotlivé služby použity. Jednotlivé služby se používají postupně zleva a standardně hledání končí, jakmile je některou službou název nalezen.

Jednoduché nastavení pro databáze hostitelů a sítí odpovídající tomu, co jsme nastavovali ve starším souboru `host.conf`, je uvedeno v příkladu 6.2.

Příklad 6.2 – Příklad souboru nsswitch.conf

```
# /etc/nsswitch.conf
#
# Příklad konfigurace služby jmen.
# Informace o tomto souboru jsou uvedeny v balíku 'libc6-doc'.

hosts:          dns files
networks:       files
```

Tento příklad způsobí, že se nejprve bude volat služba DNS a pokud té se nepodaří adresu zjistit, použije se soubor `/etc/hosts`. Názvy a adresy sítí budou hledány pouze v souboru `/etc/networks`. Způsob chování je možné definovat přesněji pomocí „položek akcí“, které udávají, co se má provést v závislosti na výsledku předchozího pokusu o nalezení. Položky akcí se uvádějí mezi názvy používaných služeb a jsou uzavřeny v hranatých závorkách [a]. Obecná syntaxe položky akce je

```
[[!] status = akce]
```

Existují dvě možné akce:

`return` Řízení se vrátí programu, který o zjištění názvu žádal. Pokud bylo hledání úspěšné, vrátí resolver požadované informace, v opačném případě vrátí prázdný výstup.

`continue` Resolver přejde na další službu a pokusí se název zjistit jejím prostřednictvím.

Nepovinný symbol „!“ znamená, že následující status má být před testováním invertován, chová se tedy jako podmínka typu „když ne“.

Stavy na něž můžeme reagovat jsou:

`succes` Požadovaný údaj byl bez chyby nalezen. Implicitní akcí pro tento status je `return`.

`notfound` Při vyhledávání nedošlo k chybě, nicméně název hostitele nebo sítě nebylo možné najít. Implicitní akce pro tento status je `continue`.

`unavail` Dotazovaná služba je nedostupná. Může to znamenat, že služba nemůže číst soubor `hosts` nebo `networks` nebo že vzdálený NIS nebo DNS server na požadavek neodpověděl. Implicitní akce pro tento status je `continue`.

`tryagain` Tento status znamená, že služba je momentálně nedostupná. U souborového rozlišování to typicky znamená, že soubory jsou momentálně zamčeny jiným procesem. U jiných služeb to může znamenat, že jmenný server nemůže dočasně přijímat požadavky. Implicitní akce pro tento status je `continue`.

Jednoduchý příklad použití tohoto mechanismu je uveden v příkladu 6.3.

Příklad 6.3 – Příklad souboru nsswitch.conf s definicí akcí

```
# /etc/nsswitch.conf
#
# Příklad konfigurace služby jmen.
# Informace o tomto souboru jsou uvedeny v balíku 'libc6-doc'.

hosts:          dns [!UNAVAIL=return] files
networks:       files
```

Tento příklad se pokouší zjistit název službou DNS. Pokud je návratový status jiný než „nedostupný“, resolver vrátí, co zjistil. Tehdy a jen tehdy vrátí-li služba DNS status „nedostupný“ se resolver pokusí o rozlišení pomocí souboru `/etc/hosts`. Znamená to, že soubor `/etc/hosts` budeme používat pouze v případě, že jmenný server není z nějakého důvodu dostupný.

Konfigurace vyhledávání souborem resolv.conf

Nakonfigurujete-li knihovnu resolveru tak, aby k vyhledávání hostitelů používala jmennou službu BIND, musíte jí sdělit, jaké má používat jmenné servery. K tomuto účelu se používá speciální soubor s názvem `resolv.conf`. Pokud tento soubor neexistuje nebo je prázdný, bude resolver předpokládat, že se jmenný server nachází na vašem místním hostiteli.

Provozujete-li jmenný server na svém místním hostiteli, musíte ho nastavit samostatně, což si vysvětlíme v dalším textu. Pokud se nacházíte v lokální síti a máte možnost používat existující jmenný server, měl byste dát této možnosti přednost. Pokud se k Internetu připojujete vytáčenou linkou, budete typicky v souboru `resolv.conf` specifikovat jmenný server vašeho poskytovatele konektivity.

Nejdůležitější volbou v souboru `resolv.conf` je volba *name server*, která udává IP adresu serveru, jež se má používat. Pokud opakovaným uvedením volby nadefinujete více jmenných serverů, budou se používat v uvedeném pořadí. Z toho důvodu byste měli na prvním místě uvést nejspolehlivější jmenný server. Současná implementace umožňuje zadat až tři jmenné servery. Pokud není žádný server definován, předpokládá resolver, že server běží na lokálním počítači.

Další dvě volby, *domain* a *search*, umožňují používat zkrácená jména hostitelů v lokální doméně. Pokud se chcete například telnetem připojit k počítači v lokální doméně, nebudete typicky chtít vypisovat celý její název a zadáte pouze samotný název počítače, například **gauss** a očekáváte, že resolver si sám doplní **mathematics.groucho.edu**.

To je funkce parametru *domain*. Umožňuje zadat název implicitní domény, která se má připojit k hledanému názvu v případě, že se jej nepodaří nalézt. Pokud například zadáte jméno **gauss**, pokusí se DNS najít adresu počítače **gauss.** a nepovede se jí to, protože nejde o jméno domény nejvyšší úrovně. Poté resolver připojí název domény **mathematics.groucho.edu.**, opakuje dotaz a tentokrát uspěje.

Možná si myslíte, že je to hezké, jenže jakmile jdete mimo doménu katedry matematiky, jste opět odkázáni na použití plně kvalifikovaných jmen. Jenže vy byste třeba chtěli používat zkrácené názvy jako **quark.physics** pro počítače katedry fyziky.

Zde přichází ke slovu *prohledávací seznam*. Tento seznam se definuje volbou *search*, která představuje zobecnění volby *domain*. Zatímco ta slouží k nastavení jediné implicitní domény, volba *search* umožňuje zadat celý seznam domén a postupně se zkoušejí všechny, dokud hledání neuspěje. Položky seznamu se oddělují mezerami nebo tabulátory.

Nastavují implicitní domény, které budou připojeny k názvu hostitele v případě, že se službě BIND nepodaří při prvním dotazu nalézt příslušný název hostitele. Volba *search* určuje seznam zkoušených názvů domén. Jednotlivé položky v seznamu jsou od sebe odděleny mezerami nebo tabulátory.

Příkazy *domain* a *search* se navzájem vylučují a mohou být uvedeny pouze jednou. Pokud není použit žádný z nich, pokusí se resolver zjistit jméno lokální domény z jména lokálního počítače voláním funkce `getdomainname(2)`. Pokud není v názvu lokálního počítače specifikována doména, bude jako implicitní doména použita kořenová doména.

Rozhodnete-li se tyto volby v souboru `resolv.conf` použít, musíte si dát pozor na to, které domény chcete do seznamu přidat. Knihovny resolveru do verze BIND 4.9 si v případě nezadání seznamu implicitních domén tento seznam vytvořily samy tak, že použily název lokální domény a všechny její rodičovské domény až po kořenovou. To způsobovalo určité potíže, protože požadavky se tak dostávaly k DNS serverům, kterým vůbec nepatřily. Řekněme, že jste ve virtuálním pivovaru a chcete se přihlásit k počítači **foot.groucho.edu**. Drobným překlepem zadáte namísto jména **foot** název **foo**, který neexistuje. Jmenný server vám tedy sdělí, že takovýto počítač nezná. Starší verze resolveru se nyní pokoušely požadavek obsloužit připojením domény **vbrew.com** a **com**. Ta druhá varianta je ovšem problematická, protože doména **groucho.edu.com** by klidně mohla existovat. Jejich jmenný server dokonce může znát i jejich počítač **foo** a vrátí vám IP adresu tohoto počítače – což je samozřejmě něco úplně jiného, než co jste chtěli.

U některých aplikací mohou takovéto chybně nalezené adresy představovat vážný bezpečnostní problém. Proto byste měli seznam prohledávaných domén omezit pouze na lokální domény nebo něco podobného. Na katedře matematiky univerzity Groucho Marx by mohl prohledávací seznam obsahovat například domény *mathematics.groucho.edu* a *groucho.edu*.

Pokud vám připadají implicitní domény zmatené, podívejte se na následující příklad souboru `resolv.conf` pro virtuální pivovar:

```
# /etc/resolv.conf
# Naše doména
domain         vbrew.com
#
# Jako centrální nameserver slouží vlager:
nameserver     172.16.1.1
```

Když budeme hledat název **vale**, pokusí se resolver nejprve najít **vale** a když se mu to nepodaří, pak **vale.vbrew.com**.

Robustnost resolveru

Pokud používáte menší lokální síť v rámci rozsáhlé sítě, měli byste rozhodně používat centrální jmenné servery, jsou-li tyto k dispozici. Výhoda tohoto způsobu spočívá v tom, že centrální jmenné servery si vytvoří velkou vyrovnávací paměť, protože na ně budou směrovány veškeré dotazy. Toto schéma má ale i nevýhody: pokud by shořel kabel páteřní sítě Olafovy univerzity, nedalo by se pracovat na žádné jejich lokální síti, protože by resolver nemohl nalézt žádný jmenný server. Tato situace způsobí problémy většině síťových služeb například přihlášení k X terminálům nebo tisku.

I když není příliš běžné, aby začala hořet páteř univerzitní sítě, budete asi chtít proti takovýmto případům učinit určitá opatření.

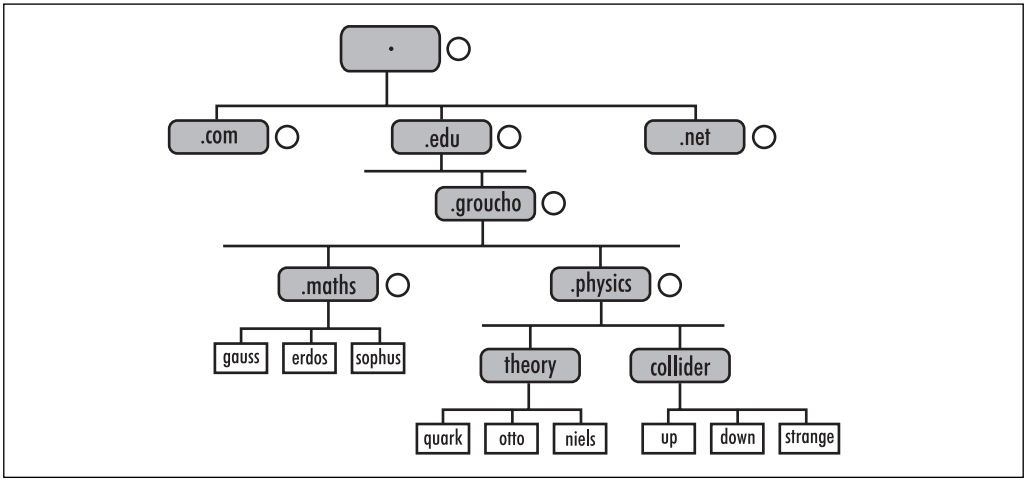
Jednou z možností je vytvořit místní jmenný server tak, aby hledal názvy hostitelů z vaší místní domény a všechny ostatní dotazy na další názvy hostitelů předával na hlavní server. Samozřejmě je to možné pouze v případě, že provozujete svou vlastní doménu.

Máte i druhou možnost – v souboru `/etc/hosts` udržovat záložní tabulku hostitelů vaší domény nebo lokální sítě. To není tak složité. Pak zajistíte, aby resolver nejprve používal DNS a pak tento soubor. V `/etc/host.conf` toho dosáhnete příkazem *order bind, hosts*, v souboru `/etc/nsswitch.conf` příkazem *hosts: dns files*.

Jak DNS funguje

DNS organizuje počítače do hierarchie domén. *Doména* představuje skupinu lokalit, které spolu nějak souvisejí – například tak, že vytvářejí jistou síť (všechny počítače univerzity, všechny počítače sítě BITNET a podobně), že všechny patří jedné instituci (například americké vládě) nebo že jsou geograficky pohromadě. Například univerzity jsou typicky sdruženy v doméně **edu**, přičemž každá univerzita nebo jiná vysoká škola používá vlastní *subdoménu*, v níž jsou sdruženi její hostitelé. Groucho Marx University má přidělenu doménu **groucho.edu** a lokální síť katedry matematiky má přidělenu subdoménu **maths.groucho.edu**. Hostitelé v síti katedry budou tento název domény připojovat ke svým názvům; takže hostitel **erdos** bude znám jako **erdos.maths.groucho.edu**. Tento název se označuje jako *plně kvalifikované doménové jméno* (FQDN) a jednoznačně identifikuje tohoto hostitele po celém světě.

Obrázek 6.1 ukazuje část doménového prostoru jmen. Vstup u kořene tohoto stromu, který je označen tečkou, je poměrně výstižně nazván *kořenovou doménou* a obsahuje v sobě všechny domény. Aby se naznačilo, že název hostitele je zadán plně kvalifikovaným doménovým jménem a nikoliv relativně pomocí nějaké (implicitní) subdomény, píše se někdy s tečkou na konci. Ta udává, že název poslední části představuje kořenovou doménu.



Obrázek 6.1 – Část doménového prostoru jmen

V závislosti na jejím umístění v hierarchii názvů může být doména nazvána doménou nejvyšší úrovně (primární doménou), doménou druhé úrovně nebo třetí úrovně. Více úrovní rozdělení je sice možné, ale vyskytuje se jen zřídka. Následuje přehled několika domén nejvyšší úrovně, s nimiž se můžete často setkat:

edu	Vzdělávací instituce (většinou v USA), jako jsou univerzity.
com	Komerční organizace a společnosti.
org	Nekomerční organizace. Často jsou v této doméně soukromé sítě typu UUCP.
net	Brány a další administrativní hostitelé na síti.
mil	Americké armádní instituce.
gov	Americké vládní instituce.
uucp	Oficiálně byly do této domény přesunuty názvy všech systémů, které byly dříve používány jako názvy UUCP bez domén.

Historicky byly první čtyři domény určeny jen pro USA, nicméně nedávné změny v politice použití domén vedly k tomu, že tyto domény nazvané globální domény nejvyšší úrovně (gTLD) jsou nyní chápány jako globální. V současné době probíhají jednání o rozšíření počtu gTLD domén a v blízké budoucnosti by měl jejich počet vzrůst[52].

Všechny státy mimo USA používají vlastní doménu nejvyšší úrovně[53], která je tvořena dvoupísmenným kódem země dle definice standardu ISO-3166. Například Finsko používá doménu **fi**, doména **fr** je přidělena Francii, doména **de** Německu, doména **au** patří Austrálii. Pod touto doménou nejvyšší úrovně může centrum NIC každé země libovolným způsobem organizovat názvy hostitelů[54]. Například Austrálie má doménu druhé úrovně podobnou mezinárodním doménám nejvyšší úrovně, tedy **com.au**, **edu.au** a podobně. Ostatní země, jako je například Německo nebo Česká republika, tuto speciální úroveň nepoužívají, ale využívají raději delší názvy, které odkazují přímo na organizace, jimž doména patří. Například není nic výjimečného setkat se s hostitelem, jehož název je například **ftp.informatik.uni-erlangen.de**. Zřejmě to nijak neovlivňuje německou výkonnost.

U těchto národních domén samozřejmě nemusí platit, že hostitel pod příslušnou doménou skutečně fyzicky leží v dané zemi; pouze to signalizuje, že hostitel byl registrován centrem NIC dané země. Švédský výrobce může mít filiálku v Austrálii a přesto bude mít všechny své hostitele registrovány pod doménou nejvyšší úrovně **se**.

Organizováním názvů do hierarchie názvů domén se tedy elegantně vyřeší problém jedinečnosti názvů; v systému DNS musí být název hostitele jedinečný pouze uvnitř vlastní domény, protože ta mu dává název odlišný od ostatních hostitelů po celém světě. Kromě toho jsou plně kvalifikované názvy poměrně lehce zapamatovatelné. Je rovněž vhodné rozdělit rozsáhlou doménu na několik subdomén.

Ale systém DNS toho umí ještě mnohem více: umožňuje pověřit správou subdomény její správce. Například správci ve výpočetním středisku univerzity Groucho Marx mohou vytvořit subdoménu pro každou katedru. Již jsme se setkali se subdoménami **math** a **physics**. Pokud se bude zdát správcům síť katedry fyziky pro správu zvenčí příliš rozsáhlá a chaotická (konec konců fyzici jsou známí jako neovladatelná skupina lidí), mohou jednoduše předat řízení nad doménou **phy-**

52 Pozn. překladatele: K 16. listopadu 2000 byla jednání ukončena a budou zavedeny následující nové globální domény (v závorce je uveden správce příslušné domény): **.aero** (Societe Internationale de Telecommunications Aeronautiques SC, SITA), **.biz** (JVTeam, LLC), **.coop** (National Cooperative Business Association, NCBA), **.info** (Afilias, LLC), **.museum** (Museum Domain Management Association, MDMA), **.name** (Global Name Registry, LTD) a **.pro** (RegistryPro, LTD). Zatím nicméně zůstává nevyjasněno, jaká pravidla budou pro přidělování adres v těchto doménách přesně platit.

53 Pozn. překladatele: I USA mají geografickou primární doménu **.us**, ale líní Američané ji moc nevyužívají a raději používají doménu **.com** a podobné.

54 Pozn. překladatele: NIC (Network Information Center) je instituce, která v dané zemi zajišťuje přidělování a registraci sekundárních domén a udržuje servery primární domény. V ČR je touto institucí CZ-NIC, z.s.p.o., *http://www.nic.cz*.

sics.groucho.edu správcům této sítě. Ti potom mohou používat libovolné názvy hostitelů a přidělovat jim IP adresy své sítě, aniž by do toho zvenčí někdo zasahoval.

Tím se celý prostor jmen rozpadá na *zóny*, které vždy začínají od nějaké domény. Mezi *zónou* a *doménou* je drobný rozdíl: doména **groucho.edu** obsahuje všechny hostitele univerzity Groucho Marx, zatímco *zóna* **groucho.edu** obsahuje pouze hostitele, které přímo spravuje výpočetní centrum, například mimo jiné hostitele na katedře matematiky. Hostitelé z katedry fyziky patří do odlišné zóny, konkrétně **physics.groucho.edu**. Na obrázku 6.1 je začátek zóny označen malým kolečkem napravo od názvu domény.

Vyhledávání názvů s pomocí DNS

Na první pohled se může zdát, že u všech těchto rozsáhlých domén a zón je provedení rozlišení názvu nesmírně komplikované. Konec konců, neřídí-li názvy, které jsou přidělovány daným hostitelům, žádná centrální správa, jak je má potom aplikace na nižší úrovni uhodnout?

Nyní přichází na řadu bezelstná vlastnost DNS. Chcete-li nalézt IP adresu serveru **erdos**, pak vám DNS odpoví, že se máte jít zeptat těch lidí, kteří ho spravují, a oni vám ji řeknou.

Systém DNS je vlastně obrovskou distribuovanou databází. Je implementován pomocí takzvaných jmenných serverů, které dodávají informace o dané doméně nebo o dané skupině domén. U každé zóny existují nejméně dva a nejvýše několik jmenných serverů, které uchovávají všechny závazné informace o hostitelích v příslušné zóně. Pokud chcete získat IP adresu serveru **erdos**, pak se stačí spojit s jmenným serverem zóny **groucho.edu**, který vám následně vrátí požadovaná data.

Možná si myslíte, že se o tom mnohem snadněji mluví, ale hůře se to provádí. Takže, jak mám vědět, jak se spojit s jmenným serverem na Groucho Marx University? V případě, že váš počítač není vybaven věštírnou pro hádání adres jmenných serverů, zvládne i to systém DNS za něj. Když chce vaše aplikace vyhledat informace o serveru **erdos**, spojí se s místním jmenným serverem, který aplikuje na požadované informace takzvaný iterační dotaz. Začne zasláním dotazu jmennému serveru kořenové domény, kde se zeptá na adresu **erdos.maths.groucho.edu**. Kořenový jmenný server pozná, že tento název nepatří do jeho správní zóny, ale patří do nějaké zóny pod doménou **edu**. Takže vám sdělí, že se máte spojit s jmenným serverem zóny **edu**, kde získáte více informací. Ke své odpovědi dále přibalí i seznam všech jmenných serverů domény **edu** společně s jejich adresami. Potom bude váš místní jmenný server pokračovat a pošle dotaz na některý z nich, například na **a.isi.edu**. Podobným způsobem jako u kořenového jmenného serveru i server **a.isi.edu** ví, že lidé z **groucho.edu** mají svou vlastní zónu a odkáže vás na její vlastní servery. Místní jmenný server bude pokračovat v dotazu na server **erdos** na jednom z těchto serverů, který konečně zjistí, že hledaný název patří do jeho zóny, a vrátí odpovídající IP adresu.

Teď to možná vypadá, že kvůli vyhledání jedné mizerné IP adresy bylo zapotřebí provést spoustu operací, ale ve skutečnosti je to pouhé minimum v porovnání s množstvím dat, která by musely být přenesena, kdyby se stále pracovalo se souborem `HOSTS.TXT`. Stále však existuje prostor, jak toto schéma dále vylepšit.

Aby zkrátil dobu odpovědi při dalších dotazech, uchovává jmenný server získané informace ve své místní *vyrovnávací paměti*. Takže když chce příště někdo z vaší lokální sítě vyhledat adresu hostitele v doméně **groucho.edu**, nemusí jmenný server znovu absolvovat celý výše popsaný proces, ale přímo se spojí s jmenným serverem pro doménu **groucho.edu**[55].

55 Kdyby se informace neukládaly do vyrovnávací paměti, byl by DNS neefektivní stejně jako jakýkoliv jiný systém, protože každý dotaz by bylo nutné řešit pomocí kořenových serverů.

Samozřejmě, že server nebude tyto informace uchovávat donekonečna, ale po určité době je smaže. Tato doba platnosti se nazývá *životnost*, zkráceně TTL. V databázi DNS přiděluje každé položce TTL správce, který je zodpovědný za danou zónu.

Typy jmenných serverů

Jmenné servery, které uchovávají všechny informace o hostitelích příslušné zóny, se nazývají *autoritativními jmennými servery* této zóny a někdy jsou také označovány jako *hlavní jmenné servery*. Jakýkoliv dotaz na hostitele uvnitř této zóny nakonec skončí u některého z hlavních jmenných serverů zóny.

Hlavní servery musí být velmi dobře synchronizovány. Toho dosáhneme tak, že jeden ze serverů pracuje jako *primární*. Ten bude načítat informace o své zóně z datových souborů a ostatní servery budou pracovat jako *sekundární* a budou v pravidelných intervalech přenášet data z primárního serveru.

Vytvořením několika jmenných serverů se rozkládá zatížení a zvyšuje se spolehlivost. Pokud některý ze serverů přestane pracovat, například když spadne nebo ztratí síťové spojení, přejdou všechny dotazy na ostatní servery. Samozřejmě vás toto schéma neochrání před chybami serveru, jejichž důsledkem budou špatné odpovědi na všechny požadavky systému DNS, například z důvodu softwarových chyb ve vlastním programu serveru.

Kromě toho můžete provozovat i jmenný server, který nebude autoritativním serverem žádné domény[56]. Tento typ serveru je důležitý, protože je schopen provádět DNS dotazy pro aplikace, které jsou spuštěny v lokální síti, a tyto informace ukládá do své vyrovnávací paměti. Proto se tento typ serveru nazývá *caching-only* server.

Databáze DNS

Ukázali jsme si, že systém DNS nejenom určuje IP adresy hostitelů, ale také vyměňuje informace mezi jmennými servery. Ve skutečnosti může databáze DNS obsahovat celou řadu různých typů záznamů.

V databázi DNS se jednotlivým informacím říká *zdrojový záznam*, zkráceně RR (resource record). Každý záznam má přidělen typ, který popisuje druh dat, jež obsahuje a třídu, určující typ sítě, k níž se záznam vztahuje. Třídy pokrývají potřeby různých adresových schémat, jako jsou IP adresy (třída IN) nebo adresy sítí Hesiod (používaných systémem Kerberos z MITu) a některé další. Obvyklým typem zdrojového záznamu je záznam A, který přiděluje plně kvalifikovanému doménovému jménu IP adresu.

Hostitel samozřejmě může mít více než jeden název. Můžete mít například server, který poskytuje služby FTP a WWW a máte pro něj dvě jména: **ftp.machine.org** a **www.machine.org**. Nicméně jen jedno z těchto jmen je označeno jako oficiální, neboli *kanonické*, ostatní jsou pak aliasy tohoto kanonického jména. Rozdíl spočívá v tom, že kanonický název hostitele je ten, jenž je definován záznamem typu A, zatímco ostatní jména jsou definována záznamem typu CNAME, který je odkazem na kanonický název.

Nebudeme si zde popisovat všechny typy záznamů, to si necháme do příští kapitoly. Nyní vám raději poskytneme krátký příklad. Příklad 6.4 ukazuje část doménové databáze, kterou používají jmenné servery zóny **physics.groucho.edu**.

56 Tedy skoro žádné. Jmenný server musí vždy obsluhovat alespoň název **locahost** a reverzní převod pro adresu 127.0.0.1.

Příklad 6.4 – Část souboru named.hosts pro katedru fyziky

```
; Autoritativní údaje o doméně physics.groucho.edu.
@  IN  SOA niels.physics.groucho.edu. janet.niels.physics.groucho.edu. (
                  1999090200       ; sériové číslo
                  360000           ; obnovování
                  3600             ; opakování pokusů
                  3600000          ; platnost
                  3600             ; implicitní ttl
                )
;
; Jmenné servery
              IN    NS       niels
              IN    NS       gauss.maths.groucho.edu.
gauss.maths.groucho.edu. IN A 149.76.4.23
;
; Teoretická fyzika (podsíť 12)
niels         IN    A        149.76.12.1
              IN    A        149.76.1.12
nameserver    IN    CNAME    niels
otto          IN    A        149.76.12.2
quark         IN    A        149.76.12.4
down          IN    A        149.76.12.5
strange       IN    A        149.76.12.6
...
; Laboratoř urychlovačů (podsíť 14)
boson         IN    A        149.76.14.1
muon          IN    A        149.76.14.7
bogon         IN    A        149.76.14.12
...
```

Kromě záznamů typu A a CNAME můžete v horní části souboru vidět speciální záznam, který je rozložen na několika řádcích. Je to zdrojový záznam typu SOA – *Start of Authority*, kde jsou umístěny všeobecné informace o zóně spravované daným serverem. Tento záznam například obsahuje implicitní hodnotu životnosti všech údajů.

Všimněte si, že všechny názvy v ukázkovém souboru, které nekončí tečkou, budou interpretovány vzhledem k doméně **physics.groucho.edu**. Speciální název „@“ použitý v záznamu typu SOA odkazuje na vlastní název domény.

Řekli jsme si už, že jmenné servery domény **groucho.edu** musí nějakým způsobem vědět o zóně **physics**, aby mohly odkazovat dotazy na její jmenné servery. Toho se obvykle dosáhne pomocí dvojice záznamů: záznam typu NS, který poskytne FQDN jmenného serveru, a záznam typu A, který tomuto názvu přidruží adresu. Protože právě tyto záznamy propojují celý prostor jmen dohromady, označují se často jako takzvané *glue records* (tmelicí záznamy). Jsou jedinými typy záznamů, kdy rodičovská zóna uchovává informace o hostitelích podřazené zóny. V příkladu 6.5 jsou uvedeny tmelicí záznamy s odkazy na jmenné servery katedry fyziky, které musí být uvedeny v databázi nadřazeného jmenného serveru (tedy serveru domény **groucho.edu**).

Příklad 6.5 – Část souboru named.hosts jmenného serveru univerzity Groucho Marx

```
; Data zóny groucho.edu.
 @  IN  SOA vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
                      1999070100       ; sériové číslo
                      360000           ; obnovování
                      3600             ; opakovaná pokusů
                      3600000          ; platnost
                      3600             ; implicitní ttl
               }
 ....
 ;
 ; Tmelicí záznamy zóny physics.groucho.edu
 physics         IN     NS        niels.physics.groucho.edu.
                 IN     NS        gauss.maths.groucho.edu.
 niels.physics   IN     A         149.76.12.1
 gauss.maths     IN     A         149.76.4.23
 ...
```

Reverzní hledání

Kromě vyhledávání IP-adresy, která náleží hostiteli, je někdy nutné vyhledat také název kanonického hostitele, který odpovídá nějaké adrese. Tento proces se označuje jako *reverzní mapování* a některé síťové služby jej používají k ověřování identity klienta. Pokud se používá soubor `hosts`, reverzní hledání představuje pouze otázku toho, nalézt v tomto souboru hostitele, který odpovídá dané adrese. U DNS samozřejmě nepřipadá v úvahu úplné prohledání jmenného prostoru. Místo toho byla vytvořena speciální doména **in-addr.arpa**, která obsahuje IP adresy všech hostitelů v obrácené tečkové notaci. Například IP adrese 149.76.12.4 odpovídá název **4.12.76.149.in-addr.arpa**. Sdružení těchto názvů s názvy kanonických hostitelů zajišťuje záznam typu PTR.

Vytvoření zóny obvykle znamená, že její správci mají plnou kontrolu nad způsobem, jakým přidělují jednotlivým názvům adresy. Protože většinou obsluhují jednu nebo více sítí nebo podsítí, existuje mezi DNS zónami a IP sítěmi jedno či více mapování. Například katedra fyziky obsahuje podsítě 149.76.8.0, 149.76.12.0 a 149.76.14.0.

V důsledku toho musí být v doméně **in-addr.arpa** společně se zónou **physics** vytvořeny nové zóny a ty musí být zpřístupněny správcům sítě katedry – půjde o zóny **8.76.149.in-addr.arpa**, **12.76.149.in-addr.arpa** a **14.76.149.in-addr.arpa**. Jinak by instalace nového hostitele v laboratoři urychlovačů vyžadovala, aby se správci spojili se svou nadřazenou doménou, kde se nová adresa zapíše do souboru zóny **in-addr.arpa**.

Zónová databáze podsítě 12 je uvedena na příkladu 6.6. Odpovídající tmelicí záznamy v databázi nadřazené zóny jsou uvedeny v příkladu 6.7.

Příklad 6.6 – Část souboru named.rev pro podsíť 12

```
; doména 12.76.149.in-addr.arpa.
 @  IN  SOA  niels.physics.groucho.edu. janet.niels.physics.groucho.edu. {
                      1999090200 360000 3600 3600000 3600
            }
 2         IN     PTR       otto.physics.groucho.edu.
 4         IN     PTR       quark.physics.groucho.edu.
 5         IN     PTR       down.physics.groucho.edu.
 6         IN     PTR       strange.physics.groucho.edu.
```

Příklad 6.7 – Část souboru named.rev pro síť 149.76

```
; doména 76.149.in-addr.arpa
 @  IN  SOA vax12.gcc.groucho.edu. joe.vax12.gcc.groucho.edu. {
                      1999070100 360000 3600 3600000 3600
                  }
 ...
 ; podsíť 4: katedra matematiky
 1.4         IN     PTR      sophus.maths.groucho.edu.
 17.4        IN     PTR      erdos.maths.groucho.edu.
 23.4        IN     PTR      gauss.maths.groucho.edu.
 ...
 ; podsíť 12: katedra fyziky, samostatná zóna
 12          IN     NS       niels.physics.groucho.edu.
             IN     NS       gauss.maths.groucho.edu.
 niels.physics.groucho.edu. IN  A 149.76.12.1
 gauss.maths.groucho.edu. IN  A   149.76.4.23
 ...
```

Zóny mohou být vytvářeny pouze jako nadmnožiny sítí IP. Co je ještě horší je to, že síťové masky podsítí musí být na hranicích celých bajtů. Všechny podsítě v univerzitě Groucho Marx mají síťovou masku 255.255.255.0, takže lze pro každou podsíť vytvořit zónu v doméně **in-addr.arpa**. Pokud by se používala síťová maska 255.255.255.128, nebylo by možné vytvořit zónu pro podsíť 149.76.12.128, protože by neexistoval žádný způsob jak sdělit systému DNS, že doména **12.76.149.in-addr.arpa** byla rozdělena na dvě zóny se samostatnou správou a s názvy hostitelů od 1 do 127, respektive od 129 do 254.

Provozování programu named

Program, který na většině unixových počítačů poskytuje doménové jmenné služby, se obvykle jmenuje named (čti *nejmdí*, ne *nejmd*). Jedná se o serverový program původně vyvinutý pro operační systém BSD, kde poskytuje služby jmen klientům a případně i dalším jmenným serverům. Nějakou dobu se nejvíce používal BIND verze 4 a byl součástí většiny distribucí Linuxu. Novou verzí používanou v dnešních distribucích je verze 8, která od minulé verze představuje podstatnou změnu[57]. Obsahuje mnoho nových funkcí, například podporu dynamické aktualizace DNS, oznamovaní změn v DNS, má výrazně vyšší výkon a používá novou syntaxi konfiguračního souboru. Podrobnosti naleznete v dokumentaci dodávané se zdrojovými soubory.

Tato stať vyžaduje určité znalosti toho, jakým způsobem systém doménových jmen funguje. Bude-li pro vás následující text tak trochu španělskou vesnicí, měli byste si znovu přečíst předcházející část *Jak DNS funguje*.

Program **named** je obvykle spuštěn při zavádění systému a běží tak dlouho, dokud počítač nevypnete. Do verze 8 získávaly informace z konfiguračního souboru `/etc/named.boot` a z mnoha dalších souborů, které obsahují data týkající se mapování názvů domén na adresy. Tyto soubory se označují jako *zónové soubory*. Verze BIND 8 používá namísto souboru `/etc/named.boot` soubor `/etc/named.conf`.

Program **named** spustíte z příkazový řádky příkazem:

```
# /usr/sbin/named
```

57 BIND 4.9 vyvinul Paul Vixie, *paul@vix.com*, nyní však BIND udržuje Internet Software Consortium, *bind-bugs@isc.org*.

Program **named** načte konfigurační soubor `named.conf` a všechny další v něm uvedené zónové soubory. Své identifikační číslo procesu zapíše ve formátu ASCII do souboru `/var/run/named.pid`, je-li to nutné, načte zónové soubory z primárních serverů a na portu 53 začne přijímat DNS dotazy.

Konfigurační soubor named.boot

Konfigurační soubor verzí starších než 8 měl velmi jednoduchou strukturu. Verze 8 používá úplně odlišnou syntaxi konfiguračního souboru, protože musí pracovat s řadou nových funkcí. Název konfiguračního souboru `/etc/named.boot` ve starších verzích se ve verzi 8 změnil na `/etc/named.conf`. Zaměříme se na nastavení souboru `/etc/named.boot`, protože ten doposud používá řada distribucí, nicméně pro ilustraci rozdílů budeme uvádět i adekvátní příkazy souboru `named.conf` a řekneme si, jak starý formát zkonvertovat do nového.

Soubor `named.boot` je zpravidla velmi malý a obsahuje pouze ukazatele na hlavní soubory s informacemi o zónách a ukazateli na další jmenné servery. V tomto zaváděcím souboru začínají komentáře znaky # nebo ; a končí u dalšího znaku nové řádky. Dříve než si probereme formát souboru `named.boot` podrobněji, podíváme se na vzorový soubor pro bránu **vlager**, který vidíte na příkladu 6.8:

Příklad 6.8 – Soubor named.boot brány vlager

```
;
; Soubor /etc/named.boot brány vlager.vbrew.com
;
directory      /var/named
;
;              doména                    soubor
;-----------------
cache          .                         named.ca
primary        vbrew.com                 named.hosts
primary        0.0.127.in-addr.arpa      named.local
primary        16.172.in-addr.arpa       named.rev
```

Podívejme se postupně na jednotlivé příkazy. Klíčové slovo *directory* říká programu **named**, že všechny soubory dále uvedené jsou uloženy v adresáři `/var/named`. Díky tomu si ušetříme něco psaní.

Klíčové slovo *primary* nahrává do démona **named** informace. Tyto informace se získávají z hlavních souborů, uvedených jako parametry. První příkaz říká programu **named**, že má fungovat jako primární server domény **vbrew.com** a data má načíst ze souboru `named.hosts`.

Klíčové slovo *cache* je velmi zvláštní a mělo by být použito na všech systémech, na nichž jmenný server běží. Říká programu **named**, že má aktivovat vyrovnávací paměť a že má načíst názvy primárních serverů ze zadaného souboru (v našem případě `named.ca`). K doporučením pro provoz jmenného serveru se vrátíme později.

Následuje seznam nejdůležitějších voleb, které můžete použít v souboru `named.boot`:

directory — Tato volba určuje adresář, ve kterém budou umístěny zónové soubory. Názvy souborů mohou být zadávány relativně vůči tomuto adresáři. Několikanásobným použitím volby directory lze zadat i více adresářů. Podle standardů souborového systému Linuxu by tímto adresářem měl být adresář `/var/named`.

primary — Tato volba používá jako parametry název domény a název souboru a nastavuje server jako autoritativní primární server této domény. Zónová data se nahrají ze zadaného hlavního souboru. Obecně bude v souboru `named.boot` vždy minimálně jedna položka s volbou primary a tou bude zpětné mapování sítě 127.0.0.0, což je lokální síť.

secondary — Tato volba používá jako parametr název domény, seznam adres a název souboru. Nastavuje server jako sekundární server dané domény. Sekundární server také uchovává autoritativní údaje o doméně. Nezískává je však ze souborů, ale snaží se je stáhnout z primárního serveru. Proto musí být v seznamu adres uveden minimálně jeden primární server. Místní server bude kontaktovat jeden po druhém, dokud se mu nepodaří úspěšný přenos zónové databáze, která se pak uloží jako záložní soubor se zadaným názvem. Neodpovídá-li ani jeden z primárních serverů, použijí se data získaná ze záložních souborů. Program **named** se v pravidelných intervalech pokouší obnovovat data zóny. Tato problematika je vysvětlena dále společně se zdrojovými záznamy typu SOA.

cache — Tato volba má jako parametry doménu a název souboru. Soubor obsahuje seznam záznamů ukazujících na kořenové jmenné servery. Jsou rozlišovány pouze záznamy typů NS a A. Parametr doména je obvykle název kořenové domény, tedy prostá tečka. Tyto informace jsou pro program **named** nezbytné. Pokud nebude příkaz cache v konfiguračním souboru uveden, program si nevytvoří vlastní lokální vyrovnávací paměť. To způsobí výrazné snížení výkonu a zvýšení zatížení sítě v případě, že se nadřazený server nenachází v místní síti. Kromě toho se nebude moci program **named** spojit s žádným kořenovým jmenným serverem, a tak nebude moci překládat jiné adresy kromě těch, pro něž je autoritativním serverem. (Výjimku z tohoto pravidla představují takzvané forwardery, o nichž hovoříme dále.)

forwarders — Tato volba používá jako parametr seznam adres. IP adresy v seznamu určují seznam jmenných serverů, kterých se může program **named** dotazovat v případě, že se mu nepodaří vyřešit dotaz pomocí své místní vyrovnávací paměti. Jmenné servery jsou v příslušném pořadí neustále dotazovány, dokud některý z nich neodpoví na dotaz.

slave — Tento příkaz označí jmenný server jako *podřízený* server. To znamená, že nikdy nebude sám provádět rekurzivní dotazy, ale bude je všechny postupovat na servery definované volbou forwarders.

Ještě jsme neuvedli dvě další volby, sortlist a domain. Kromě toho existují ještě dvě direktivy, které lze použít v databázových souborech zón. Jde o direktivy $INCLUDE a $ORIGIN. Protože jsou používány jen velmi zřídka, nebudeme se jimi dále zabývat.

Konfigurační soubor host.conf programu BIND verze 8

V programu BIND verze 8 byla uvedena řada nových funkcí, a proto byla zvolena i nová syntaxe konfiguračního souboru. Soubor `named.boot` s jednořádkovými konfiguračními příkazy byl nahrazen souborem `named.conf`, který používá podobnou strukturu jako program **gated**, která trochu připomíná kód jazyka C.

Nová syntaxe je složitější, existuje však naštěstí nástroj, který umožňuje konverzi starých konfiguračních souborů do nového formátu. Ve zdrojovém balíku BIND 8 existuje skript v Perlu, který se jmenuje **named-bootconf.pl** a který přečte stávající soubor `named.boot` ze standardního vstupu

a konvertuje jej na ekvivalentní soubor `named.conf` na stanadrdní výstup. Abyste mohli skript použít, musíte mít nainstalován interpret Perlu.

Skript se používá asi následujícím způsobem:

```
# cd /etc
# named-bootconf.pl <named.boot >named.conf
```

Program pak vygeneruje soubor `named.conf`, který bude vypadat podobně jako v příkladu 6.9. Odstranili jsme různé vysvětlující komentáře, které skript v souboru vytváří, abychom ukázali prakticky přímou souvislost nové a staré syntaxe.

Příklad 6.9 – Konfigurační soubor verze BIND 8

```
//
// Soubor /etc/named.boot brány vlager.vbrew.com
options {
        directory "/var/named";
};

zone "." {
        type hint;
        file "named.ca";
};

zone "vbrew.com" {
        type master;
        file "named.hosts";
};

zone "0.0.127.in-addr.arpa" {
        type master;
        file "named.local";
};

zone "16.172.in-addr.arpa" {
        type master;
        file "named.rev";
};
```

Když se podíváte pečlivě, zjistíte, že každý jedrn řádek souboru `named.boot` je v souboru `named.conf` převeden na příkaz uzavřený ve stylu jazyka C ve složených závorkách, { a }.

Komentáře uváděné v souboru `named.boot` středníkem jsou nyní označeny dvěma lomítky.

Příkaz *directory* byl převeden na blok *options* s klauzulí *directory*.
Příkazy *cache* a *primary* byly převedeny na bloky *zone* s klauzulí *type* o hodnotě *hint*, respektive *master*.

Samotné zónové soubory není třeba upravovat, jejich syntaxe se nezměnila.

Tato nová syntaxe konfiguračního souboru umožňuje použití celé řady různých nových voleb, o kterých jsme nemluvili. Pokud vás tyto nové možnosti zajímají, nejlepším zdrojem informací je dokumentace k balíku BIND, která je dodávána s jeho zdrojovými texty.

Databázové soubory systému DNS

Hlavní soubory, které využívá program **named**, například soubor `named.hosts`, jsou vždy sdruženy s nějakou doménou. Tato doména se nazývá *počátek*. Název domény je zadán parametry příkazů cache a primary. V rámci hlavního souboru můžete zadávat názvy domén a hostitelů relativně vůči této doméně. Název uvedený v konfiguračním souboru je považován za *absolutní*, pokud končí tečkou, v opačném případě je považován za relativní vůči počátku. Vlastní počátek může být odkazován symbolem @.

Všechna data obsažená v hlavním souboru jsou rozdělena do takzvaných *zdrojových záznamů*, RR. Vytvářejí nejmenší jednotky informace dostupné pomocí systému DNS. Každý zdrojový záznam je určitého typu. Například záznamy typu A mapují název hostitele na IP adresu a záznam typu CNAME přiřazuje přezdívky hostitele oficiálnímu názvu hostitele. Zajímá-li vás příklad, podívejte se na výpis 6.11, který představuje hlavní soubor `named.hosts` virtuálního pivovaru.

Položky zdrojových záznamů v hlavních souborech používají společný formát:

```
[doména] [ttl] [třída] typ rdata
```

Pole jsou navzájem oddělena pomocí mezer nebo tabulátorů. Položka může pokračovat na několika řádcích, pokud na konci prvního řádku uvedete levou závorku a pravou závorku umístíte na konec záznamu. Středníky představují začátek komentáře a cokoliv až do konce řádku se ignoruje. Následuje popis jednotlivých částí záznamu:

doména Název domény, ke které se záznam vztahuje. Není-li uveden žádný název domény, bude se předpokládat, že se záznam vztahuje k doméně uvedené v předchozím záznamu.

ttl Aby bylo po uplynutí určité doby možno donutit resolver k vyřazení určitých informací, je ke každému záznamu RR přiřazena životnost, *ttl*. Pole ttl udává čas v sekundách, po který budou ještě informace po stažení ze serveru považovány za platné. Čas ttl je desítkové číslo s maximálně osmi číslicemi. Nezadáte-li žádný časový údaj, bude jeho implicitní hodnota rovna hodnotě pole *minimum* předcházejícího záznamu SOA.

třída Tato volba určuje třídu adresy, například třídu IN pro IP adresy nebo HS pro adresy třídy Hesiod. U sítí založených na protokolu TCP/IP se používá třída IN. Není-li třída zadána, použije se třída z předchozího záznamu typu RR.

typ Tato volba popisuje typ záznamu RR. Nejběžnějšími typy jsou záznamy A, SOA, PTR a NS. V následující stati budeme popisovat různé typy záznamů RR.

rdata Tato položka obsahuje data záznamu RR. Formát tohoto pole závisí na typu záznamu. Dále si popíšeme data používaná u jednotlivých typů záznamů.

Následuje neúplný seznam typů zdrojových záznamů, které lze použít v hlavních souborech systému DNS. Existují i další typy, ale těmi se zabývat nebudeme, protože se jedná o experimentální typy s velmi řídkým využitím.

SOA Tento typ záznamu definuje zónu (SOA znamená Start of Authority). Tento záznam signalizuje, že následující záznamy budou obsahovat administrativní informace o doméně. Každý hlavní soubor definovaný příkazem *primary* musí obsahovat záznam typu SOA této zóny. Zdrojová data obsahují následující pole:

origin Kanonický název hostitele primárního jmenného serveru této domény. Obvykle je zadán jako absolutní název.

contact — E-mailová adresa osoby odpovědné za správu domény, u které je znak @ nahrazen tečkou. Je-li například ve virtuálním pivovaru odpovědnou osobou **janet**, potom bude toto pole obsahovat adresu *janet.vbrew.com*.

serial — Číslo verze souboru zóny vyjádřené jednou desítkovou číslicí. Kdykoliv v souboru zóny změníte data, měli byste inkrementovat i toto číslo. Klasická konvence obsahuje datum aktualizace společně s číslem verze pro případ více aktualizací v jednom dni, například 2000022600 je první aktualizace dne 26. 2. 2000. Sériová čísla používají sekundární jmenné servery k rozpoznávání změn v informacích o zónách. Aby měly sekundární servery aktuální informace, v pravidelných intervalech si po primárním serveru vyžádají záznam SOA a porovnávají jeho sériové číslo s číslem, které je obsaženo v jejich kopii zónového souboru. Změnilo-li se číslo, potom sekundární servery přenesou z primárního serveru celou databázi zóny.

refresh — Tato volba udává interval v sekundách, po který mají sekundární servery čekat, než provedou opětovnou kontrolu záznamu typu SOA s primárním serverem. Opět se jedná o desítkové číslo s maximálně osmi číslicemi. Síťová topologie se obecně příliš často nemění, takže by toto číslo mělo v rozsáhlejších sítích odpovídat zhruba dnům a v menších sítí by tento interval měl být ještě delší.

retry — Toto číslo určuje interval, po jehož uplynutí by se měl sekundární server znovu spojit s primárním serverem, když se nepodaří nějaký požadavek nebo aktualizace zónových informací. Tento interval by neměl být příliš malý, jinak by dočasný výpadek serveru nebo nějaký síťový problém mohl způsobit obrovské plýtvání se síťovými zdroji ze strany sekundárních serverů. Vhodnou hodnotou pro tento interval je jedna hodina nebo půl hodiny.

expire — Tato volba udává čas v sekundách, po jehož uplynutí by měl server skartovat všechna data o zónách, pokud se mu během tohoto intervalu nepodařilo spojit s primárním serverem. Normálně by měla být tato hodnota hodně velká, alespoň týden (604800 sekund), nicméně její prodloužení až na měsíc může být také užitečné.

minimum — Toto je implicitní hodnota ttl zdrojových záznamů, které nemají definován vlastní ttl. Volba přikazuje ostatním jmenným serverům, aby po předem dané době zrušily záznam ve vyrovnávací paměti. Tato hodnota nemá nic společného s časem, po kterém se budou snažit sekundární servery o aktualizaci svých informací. Hodnota by měla být dostatečně vysoká, zejména u sítí typu LAN, u nichž se prakticky nemění síťová topologie. Vhodné je použít hodnotu týden nebo dokonce měsíc. V případech, kde se některé záznamy mění často, můžete pro ně nastavit vlastní ttl.

A — Tento typ záznamu přiřazuje IP adresu k názvu hostitele. Pole zdrojových dat obsahuje adresu respektující tečkovou notaci. Každý hostitel musí mít pouze jeden záznam typu A. Název hostitele uvedený v tomto záznamu je považován za oficiální, neboli *kanonický* název hostitele. Všechny další názvy téhož hostitele jsou přezdívky a pomocí záznamu typu CNAME se mapují na kanonický název. Je-li kanonické jméno našeho hostitele **vlager**, uvedeme toto jméno v záznamu A společně s adresou hostitele. Budeme-li chtít téže adrese přiřadit další jména, například **news**, vytvoříme záznam typu CNAME, který bude asociovat alternativní jméno a kanonické jméno. O záznamech CNAME budeme hovořit za chvíli.

NS — Tento typ záznamu definuje primární a všechny sekundární jmenné servery zóny. Záznam ukazuje na hlavní jmenný server zóny, datová část obsahuje kanonické jméno tohoto serveru. Záznamy typu NS potkáváme ve dvou situacích: První z nich je případ, kdy delegujeme pravomoc na podřízenou zónu, druhý je v hlavní databázi samotné podřízené zóny. Skupiny serverů specifikované v nadřízené i podřízené zóně si musí odpovídat. Záznam typu NS definuje názvy primárního a sekundárních serverů zóny. Abychom tyto názvy mohli použít, musíme je být schopni převést na adresy. Často ovšem server patří přímo do té zóny, kterou obsluhuje a dostáváme tak klasický problém „kuřete a vejce“: adresu serveru nezjistíme, dokud se na ni serveru nezeptáme a nemůžeme se ho zeptat, dokud neznáme jeho adresu. K vyřešení tohoto dilematu musíme zavést speciální záznamy typu A přímo v databázi nadřazené zóny. Záznam A umožní serveru rodičovské zóny zjistit adresu jmenného serveru podřízené zóny. Tyto záznamy se často označují jako *tmelicí záznamy*, protože fungují jako „tmel“, který propojuje podřízenou zónu s jejím rodičem.

CNAME — Záznam přiřazuje kanonickému názvu hostitele přezdívku. Definuje alternativní názvy, kterými se můžeme odkazovat na hostitele, jehož kanonické jméno je uvedeno jako parametr. Kanonický název hostitele je ten název, pro který existuje v hlavní databázi záznam typu A; přezdívky jsou s tímto názvem jednoduše spojeny pomocí záznamu typu CNAME, ale jinak se k nim žádné další záznamy nevztahují.

PTR — Tento typ záznamu slouží ke sdružení názvů v doméně **in-addr.arpa** s názvy hostitelů. Záznam se používá ke zpětnému mapování IP adres na názvy hostitelů. Zadaný název hostitele musí být v kanonickém tvaru.

MX — Tento záznam definuje *poštovní server* dané domény. O poštovních serverech budeme hovořit v kapitole 17, *Směrování pošty na Internetu*. Syntaxe záznamu typu MX je: `[doména] [ttl] [třída] MX preference hostitel` Parametr *hostitel* definuje název poštovního serveru domény. Každému serveru odpovídá jeho *preference*. Poštovní agent snažící se o doručení pošty do domény se pokusí použít všechny hostitele definované záznamy MX cílové domény do doby, než u jednoho z nich uspěje. Pořadí, ve kterém je používá, závisí právě na jejich preferenci: první se zkusí hostitel s nejmenší preferencí a v případě neúspěchu postupně další a další podle rostoucí preference.

HINFO — Tento typ záznamu poskytuje informace o hardwaru a softwaru daného systému. Jeho syntaxe je: `[doména] [ttl] [třída] HINFO hardware software` Údaj *hardware* určuje typ hardwaru, který daný hostitel používá. Pro jeho specifikaci existují konvence, seznam platných „názvů hardwarových platforem“ definuje dokument RFC 1700, Assigned Numbers. Pokud údaj obsahuje mezeru, musí být uzavřen v uvozovkách. Údaj *software* obsahuje používaný operační systém. Opět by měl být vybrán korektní název podle dokumentu Assigned Numbers[58]. Záznam `HINFO` popisující linuxový stroj na platformě Intel by vypadal takto: `tao 36500 IN HINFO IBM-PC LINUX2.2` Záznamy `HINFO` popisující linuxový stroj na platformách Motorola-68000 budou vypadat takto: `cevad 36500 IN HINFO ATARI-104ST LINUX2.0 jedd 36500 IN HINFO AMIGA-3000 LINUX2.0`

58 Pozn. překladatele: Záznamy typu `HINFO` prakticky nikdo nepoužívá. Jsou to čistě informativní záznamy – jejich údaje ani DNS ani nikdo jiný k ničemu nepotřebuje. Řada lidí se domnívá, že zveřejněním hardwarové a softwarové platformy svých počítačů mohou usnadnit situaci útočníkům, kteří tak budou moci vést cílenější útok, jelikož přesně vědí, na co útočí.

Konfigurace typu „caching-only“

Existuje jeden speciální typ konfigurace programu **named**, o němž jsme se zmínili před tím, než jsme se pustili do podrobnějšího výkladu. Označuje se jako konfigurace *caching-only*. V této konfiguraci server neobsluhuje žádnou doménu, funguje však jako „zpracovatel“ všech DNS dotazů, které vaše počítače generují. Výhodou tohoto řešení je, že si server vybuduje vyrovnávací paměť a konkrétním jmenným serverům na Internetu se tak posílá vždy pouze první dotaz na určitého hostitele. Další stejné dotazy zodpoví přímo lokální server podle své vyrovnávací paměti. V této chvíli to nevypadá moc užitečně, výhody zjistíme při použití telefonického připojení k Internetu, které popisujeme v kapitolách 7 a 8.

Soubor `named.boot` serveru v režimu caching-only bude vypadat nějak takto:

```
; Soubor named.boot caching-only serveru
directory                          /var/named
primary       0.0.127.in-addr.arpa   named.local ; síť localhost
cache         .                      named.ca    ; kořenové servery
```

Kromě souboru `named.boot` musíte ještě vytvořit soubor `named.ca`, který bude obsahovat platný seznam kořenových serverů. Můžete jej vytvořit podle příkladu 6.10. Žádné další soubory nejsou v této konfiguraci zapotřebí.

Sestavení hlavních souborů

Příklady 6.10, 6.11, 6.12 a 6.13 obsahují příklady hlavních souborů jmenného serveru pivovaru na počítači **vlager**. Vzhledem k povaze diskutované sítě (jednoduchá síť typu LAN) je příklad poměrně průhledný.

Soubor `named.ca`, který vidíte na obrázku 6.10, ukazuje záznamy pro kořenové jmenné servery. Typický takovýto soubor obvykle definuje zhruba deset jmenných serverů. Aktuální seznam jmenných serverů kořenové domény můžete získat pomocí nástroje **nslookup**, který bude popsán v další části kapitoly[59].

Příklad 6.10 – Soubor named.ca

```
;
; /var/named/named.ca
;   Soubor kořenových serverů pro pivovar. Nejsme na
;   Internetu, tak je nepotřebujeme. V případě potřeby
;   stačí záznamy aktivovat odstraněním středníků.
;
;.                        3600000  IN  NS    A.ROOT-SERVERS.NET.
;A.ROOT-SERVERS.NET.      3600000      A     198.41.0.4
;.                        3600000      NS    B.ROOT-SERVERS.NET.
;B.ROOT-SERVERS.NET.      3600000      A     128.9.0.107
;.                        3600000      NS    C.ROOT-SERVERS.NET.
;C.ROOT-SERVERS.NET.      3600000      A     192.33.4.12
;.                        3600000      NS    D.ROOT-SERVERS.NET.
;D.ROOT-SERVERS.NET.      3600000      A     128.8.10.90
;.                        3600000      NS    E.ROOT-SERVERS.NET.
;E.ROOT-SERVERS.NET.      3600000      A     192.203.230.10
```

59 Uvědomte si, že se nemůžete vašeho jmenného serveru zeptat na kořenové servery, pokud v něm alespoň nějaké nemáte definovány. Problém můžete vyřešit tak, že se programem **nslookup** zeptáte jiného jmenného serveru, nebo použijete jako základ soubor 6.10 a pomocí něj získáte aktuální platný seznam všech kořenových serverů.

```
;.                          3600000      NS    F.ROOT-SERVERS.NET.
;F.ROOT-SERVERS.NET.        3600000      A     192.5.5.241
;.                          3600000      NS    G.ROOT-SERVERS.NET.
;G.ROOT-SERVERS.NET.        3600000      A     192.112.36.4
;.                          3600000      NS    H.ROOT-SERVERS.NET.
;H.ROOT-SERVERS.NET.        3600000      A     128.63.2.53
;.                          3600000      NS    I.ROOT-SERVERS.NET.
;I.ROOT-SERVERS.NET.        3600000      A     192.36.148.17
;.                          3600000      NS    J.ROOT-SERVERS.NET.
;J.ROOT-SERVERS.NET.        3600000      A     198.41.0.10
;.                          3600000      NS    K.ROOT-SERVERS.NET.
;K.ROOT-SERVERS.NET.        3600000      A     193.0.14.129
;.                          3600000      NS    L.ROOT-SERVERS.NET.
;L.ROOT-SERVERS.NET.        3600000      A     198.32.64.12
;.                          3600000      NS    M.ROOT-SERVERS.NET.
;M.ROOT-SERVERS.NET.        3600000      A     202.12.27.33
;
```

Příklad 6.11 – Soubor named.hosts

```
;
; /var/named/named.hosts
;   Lokální hostitelé pivovaru, doména vbrew.com
;
@               IN  SOA   vlager.vbrew.com. janet.vbrew.com. (
                          2000012601 ; sériové číslo
                          86400      ; obnovení: denně
                          3600       ; opakování: co hodinu
                          3600000    ; zneplatnění:  42 dní
                          604800     ; minimální ttl: týden
                          )
                IN  NS    vlager.vbrew.com.
;
; poštu obsahuje vlager
                IN  MX    10 vlager
;
; lokální adresa
localhost.      IN  A     127.0.0.1
;
; Ethernet pivovaru
vlager          IN  A     172.16.1.1
vlager-if1      IN  CNAME vlager
; vlager je rovněž server pro konference
news            IN  CNAME vlager
vstout          IN  A     172.16.1.2
vale            IN  A     172.16.1.3
;
; Ethernet vinařství
vlager-if2      IN  A     172.16.2.1
vbardolino      IN  A     172.16.2.2
vchianti        IN  A     172.16.2.3
vbeaujolais     IN  A     172.16.2.4
;
```

```
; (podřízený) Ethernet palírny
vbourbon         IN  A     172.16.3.1
vbourbon-if1     IN  CNAME vbourbon
```

Příklad 6.12 – Soubor named.local

```
;
; /var/named/named.local
;   Reverzní mapování 127.0.0 - doména 0.0.127.in-addr.arpa.
;
;
@             IN  SOA   vlager.vbrew.com. joe.vbrew.com. (
                        1          ; sériové číslo
                        360000     ; obnovení: 100 hodin
                        3600       ; opakování: co hodinu
                        3600000    ; zneplatnění:  42 dní
                        604800     ; minimální ttl: 100 hodin
)
              IN  NS    vlager.vbrew.com.
1             IN  PTR   localhost.
```

Příklad 6.13 – Soubor named.rev

```
;
; /var/named/named.rev
;   Reverzní mapování našich IP adres, doména 16.172.in-addr.arpa.
;
@             IN  SOA   vlager.vbrew.com. joe.vbrew.com. (
                          16         ; sériové číslo
                          86400      ; obnovení: denně
                          3600       ; opakování: co hodinu
                          3600000    ; zneplatnění:  42 dní
                          604800     ; minimální ttl: týden
                          )
              IN  NS    vlager.vbrew.com.
; pivovar
1.1           IN  PTR   vlager.vbrew.com.
2.1           IN  PTR   vstout.vbrew.com.
3.1           IN  PTR   vale.vbrew.com.
; vinařství
1.2           IN  PTR   vlager-if2.vbrew.com.
2.2           IN  PTR   vbardolino.vbrew.com.
3.2           IN  PTR   vchianti.vbrew.com.
4.2           IN  PTR   vbeaujolais.vbrew.com.
```

Kontrola nastavení jmenného serveru

Pro kontrolu činnosti jmenného serveru existuje vynikající nástroj **nslookup**. Lze jej používat jak interaktivně, tak i z příkazové řádky. Ve druhém případě ho spustíte takto:

```
$ nslookup hostname
```

Program se dotáže jmenného serveru zadaného v souboru `resolv.conf` na adresu hostitele, jehož název jsme zadali. (Je-li v souboru `resolv.conf` uveden více než jeden server, pak **nslookup** náhodně zvolí jeden z nich.)

Interaktivní režim je však mnohem zajímavější. Kromě vyhledávání jednotlivých hostitelů se můžete dotazovat na libovolný typ záznamu systému DNS a dále můžete přenést celou databázi zóny.

Spustíte-li **nslookup** bez parametrů, zobrazí používaný jmenný server a přepne se do interaktivního režimu. Na výzvu „>“ můžete zadat libovolný název domény, na který by se měl program dotazovat. Implicitně se ptá na záznamy typu A, což jsou ty, které obsahují IP adresy v dané doméně.

Tento typ je možné změnit příkazem

```
> set type = typ
```

kde parametr *typ* může být jeden z již popsaných typů záznamů, nebo klíčové slovo ANY.

Typická ukázka práce s programem **nslookup** může vypadat takto:

```
$ nslookup
Default Server:  tao.linux.org.au
Address:  203.41.101.121

> metalab.unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121

Name:    metalab.unc.edu
Address:  152.2.254.81

>
```

V odpovědích je nejprve uvedeno, kterého serveru jsme se ptali a pak odpověď na náš dotaz.

Pokusíte-li se dotazovat na název, který nemá přidělenu žádnou IP adresu, ale v databázi systému DNS byly nalezeny jiné záznamy, vrátí příkaz **nslookup** chybovou zprávu „`No type A records found`“. Programem **nslookup** je ale možné dotazovat se i na jiné typy záznamů, stačí typ změnit příkazem **set type**. Záznam SOA domény **unc.edu** získáme například takto:

```
> unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121

*** No address (A) records available for unc.edu
> set type=SOA
> unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121

unc.edu
        origin = ns.unc.edu
        mail addr = host-reg.ns.unc.edu
        serial = 1998111011
        refresh = 14400 (4H)
        retry   = 3600 (1H)
```

```
        expire  = 1209600 (2W)
        minimum ttl = 86400 (1D)
unc.edu name server = ns2.unc.edu
unc.edu name server = ncnoc.ncren.net
unc.edu name server = ns.unc.edu
ns2.unc.edu     internet address = 152.2.253.100
ncnoc.ncren.net internet address = 192.101.21.1
ncnoc.ncren.net internet address = 128.109.193.1
ns.unc.edu      internet address = 152.2.21.1
```

Podobně se můžeme zeptat i na záznamy typu MX:

```
> set type=MX
> unc.edu
Server:  tao.linux.org.au
Address:  203.41.101.121

unc.edu preference = 0, mail exchanger = conga.oit.unc.edu
unc.edu preference = 10, mail exchanger = imsety.oit.unc.edu
unc.edu name server = ns.unc.edu
unc.edu name server = ns2.unc.edu
unc.edu name server = ncnoc.ncren.net
conga.oit.unc.edu       internet address = 152.2.22.21
imsety.oit.unc.edu      internet address = 152.2.21.99
ns.unc.edu      internet address = 152.2.21.1
ns2.unc.edu     internet address = 152.2.253.100
ncnoc.ncren.net internet address = 192.101.21.1
ncnoc.ncren.net internet address = 128.109.193.1
```

Zadáme-li typ ANY, budou vráceny všechny záznamy libovolného typu, které se k požadovanému názvu vztahují.

Praktickou aplikací programu **nslookup** je kromě testování jmenného serveru i získání aktuálního seznamu kořenových jmenných serverů. Lze to provést tak, že se zeptáte na záznamy typu NS, odpovídající kořenové doméně:

```
> set type=NS
> .
Server:  tao.linux.org.au
Address:  203.41.101.121

Non-authoritative answer:
(root)  name server = A.ROOT-SERVERS.NET
(root)  name server = H.ROOT-SERVERS.NET
(root)  name server = B.ROOT-SERVERS.NET
(root)  name server = C.ROOT-SERVERS.NET
(root)  name server = D.ROOT-SERVERS.NET
(root)  name server = E.ROOT-SERVERS.NET
(root)  name server = I.ROOT-SERVERS.NET
(root)  name server = F.ROOT-SERVERS.NET
(root)  name server = G.ROOT-SERVERS.NET
(root)  name server = J.ROOT-SERVERS.NET
(root)  name server = K.ROOT-SERVERS.NET
(root)  name server = L.ROOT-SERVERS.NET
```

```
(root)  name server = M.ROOT-SERVERS.NET

Authoritative answers can be found from:
A.ROOT-SERVERS.NET      internet address = 198.41.0.4
H.ROOT-SERVERS.NET      internet address = 128.63.2.53
B.ROOT-SERVERS.NET      internet address = 128.9.0.107
C.ROOT-SERVERS.NET      internet address = 192.33.4.12
D.ROOT-SERVERS.NET      internet address = 128.8.10.90
E.ROOT-SERVERS.NET      internet address = 192.203.230.10
I.ROOT-SERVERS.NET      internet address = 192.36.148.17
F.ROOT-SERVERS.NET      internet address = 192.5.5.241
G.ROOT-SERVERS.NET      internet address = 192.112.36.4
J.ROOT-SERVERS.NET      internet address = 198.41.0.10
K.ROOT-SERVERS.NET      internet address = 193.0.14.129
L.ROOT-SERVERS.NET      internet address = 198.32.64.12
M.ROOT-SERVERS.NET      internet address = 202.12.27.33
```

Kompletní seznam příkazů programu **nslookup** získáte tak, že mu zadáte příkaz **help**.

Další užitečné nástroje

Existuje několik dalších nástrojů, které vám při správě služby BIND mohou pomoci. Krátce si popíšeme některé z nich. Podrobnější informace o jejich použití získáte v nápovědě, která se s nimi dodává.

Nástroj **hostcvt** pomáhá při prvotní konfiguraci služby BIND. Provádí konverzi souboru `/etc/hosts` do hlavních souborů pro program named. Vygeneruje data jak pro přímé mapování (typ záznamu A), tak i pro zpětné mapování (typ záznamu PTR) a postará se i o přezdívky. Samozřejmě, že nemůže udělat vše, protože si sami budete chtít nastavit časovací konstanty záznamu SOA nebo vytvořit záznam MX. Přesto vám však může ušetřit dost starostí. Nástroj **hostcvt** je částí zdrojového kódu služby BIND, ale je možné ho nalézt i jako samostatný balík na některých linuxových FTP serverech.

Jakmile nakonfigurujete vlastní jmenný server, budete si ho určitě chtít otestovat. Existuje několik dobrých nástrojů, které vám testování usnadní: jedním z nich je **dnswalk**, napsaný v Perlu. Dalším je **nslint**. Oba projdou databáze DNS serveru, hledají v nich obvyklé chyby a kontrolují, zda jsou údaje konzistentní. Dalšími užitečnými nástroji jsou **host** a **dig**, což jsou obecné programy pro dotazování se systému DNS. Pomocí nich můžete ručně zkoumat a diagnostikovat záznamy DNS.

Všechny nástroje jsou snadno dostupné v předpřipravených balících: **dnswalk** a **nslint** najdete na adresách **http://www.visi.com/~barr/dnswalk/** a **ftp://ftp.ee.lbl.gov/nslint.tar.Z**. Zdrojové kódy programů **host** a **dig** pak na adresách **ftp://ftp.nikhef.nl/pub/network/** a **ftp://ftp.is.co.za/networking/ip/dns/dig/**.

Linka SLIP

Paketové protokoly jako IP nebo IPX spolehájí na to, že přijímající hostitel pozná v přijímaném datovém proudu začátek a konec každého paketu. Mechanismus používaný k detekci začátků a konců paketů se označuje jako *deliminace*. V lokální síti tento mechanismus implementuje Ethernet, na sériových komunikačních linkách jej implementují protokoly SLIP a PPP.

Relativně nízké ceny pomalých vytáčených linek a rychlejších telefonních okruhů vedly k velkému rozšíření sériové komunikace protokolem IP, zejména pro zajištění konektivity k Internetu koncovým uživatelům. Hardware potřebný pro provoz protokolů SLIP nebo PPP je jednoduchý a široce dostupný. Potřebujeme pouze modem a sériový port vybavený portem FIFO.

Protokol SLIP je implementačně velmi jednoduchý a jistou dobu byl z obou zmíněných protokolů výrazně používanější. Dnes už většina uživatelů preferuje protokol PPP. Tento protokol nabízí celou řadu užitečných funkcí, které se zasloužily o jeho oblíbenost. O nejdůležitějších z nich se později zmíníme.

Jádro Linuxu obsahuje ovladače protokolu SLIP i PPP. Oba ovladače se už nějakou dobu používají a jsou stabilní a spolehlivé. V této a následující kapitole budeme hovořit o obou protokolech a o tom, jak je nakonfigurovat.

Obecné požadavky

Abyste mohli používat protokoly SLIP nebo PPP, bude třeba nastavit základní síťové funkce, které byly popsány v předchozích kapitolách. Přinejmenším musíte nastavit lokální rozhraní a povolit službu pro rozlišení názvů. Když se budete připojovat k Internetu, budete samozřejmě chtít používat systém DNS. Zde máte dvě možnosti: buď používat DNS přes sériovou linku a v souboru `/etc/resolv.conf` nastavit IP adresu DNS serveru vašeho poskytovatele, nebo si postupem popsaným v kapitole 6 nakonfigurovat caching-only DNS server.

Použití protokolu SLIP

Dial-up servery často nabízejí službu SLIP prostřednictvím speciálních uživatelských účtů. Po přihlášení k takovému účtu nejste vpuštěni do obecného uživatelského rozhraní; namísto toho je spuštěn program nebo skript, který povolí na serveru pro danou sériovou linku ovladač SLIP a nakonfiguruje patřičné síťové rozhraní. Totéž se musí provést na vaší straně spojení.

V některých operačních systémech je ovladač SLIP speciálním uživatelským programem; v operačním systému Linux je součástí jádra systému, takže je výrazně rychlejší. Je však nutné, aby byla sériová linka explicitně převedena do režimu SLIP. To se provede pomocí speciálního režimu linky, SLIPDISC. Je-li zařízení tty v normálním režimu (DISC0), probíhá přenos dat uživatelským procesem voláními `read(2)` a `write(2)` a ovladač SLIP nebude schopen zapisovat nebo číst ze zařízení tty. V režimu SLIPDISC jsou role obráceny: nyní nebude moci zapisovat nebo číst ze zaříze-

ní žádný uživatelský proces, ale všechna data přicházející na sériový port budou přímo předána ovladači SLIP.

Vlastní ovladač protokolu SLIP rozumí množství variací protokolu SLIP. Kromě běžného protokolu SLIP ovládá také protokol CSLIP, který u odcházejících IP paketů provádí takzvanou Van Jacobsonovu kompresi hlaviček (viz RFC 1144). To výrazně zvyšuje propustnost dat při interaktivní práci. Mimoto existují i šestibitové verze obou protokolů.

Jednoduchý způsob, jak přepnout sériovou linku do režimu SLIP, spočívá ve využití nástroje **slattach**. Předpokládejme, že máte modem na zařízení `/dev/ttyS3` a že jste se úspěšně přihlásili k serveru SLIP. Potom spustíte následující příkaz:

```
# slattach /dev/ttyS3 &
```

Tento příkaz přepne režim zařízení ttyS3 na SLIPDISC a připojí je k jednomu ze síťových rozhraní SLIP. Je-li to vaše první aktivní spojení pomocí protokolu SLIP, bude linka připojena k rozhraní sl0; další bude připojena k rozhraní sl1 a tak dále. Současné verze jader operačního systému podporují až 256 současných připojení pomocí protokolu SLIP.

Implicitní režim nastavený příkazem **slattach** bude CSLIP. K volbě některého jiného režimu lze použít parametr `-p`. Chcete-li používat standardní protokol SLIP (bez komprese), zadejte:

```
# slattach -p slip /dev/ttyS3 &
```

Další možné režimy jsou uvedeny v tabulce 7.1. Kromě toho existuje speciální pseudorežim `adaptive`, při němž jádro provádí automatickou detekci toho, jaký režim se používá na druhém konci linky.

Tabulka 7.1 – Režimy linky SLIP v Linuxu

Režim	Popis
`slip`	Klasický protokol SLIP.
`cslip`	Protokol SLIP s Van Jacobsonovou kompresí hlaviček.
`slip6`	Protokol SLIP s šestibitovým kódováním. Použitá kódovací metoda se podobá příkazu **uuencode** a převádí všechny datagramy na tisknutelné znaky. Tato konverze je užitečná, pokud nemáte sériovou linku se správně fungujícím osmým bitem.
`cslip6`	Protokol SLIP s Van Jacobsonovou kompresí hlaviček a šestibitovým kódováním.
`adaptive`	Nejedná se o samostatný režim linky, jádro se pokusí detekovat režim používaný vzdáleným počítačem a přejde do něj.

Musíte používat stejný režim jako vzdálená strana. Pokud připojený počítač používá řekněme režim CSLIP, vy jej musíte používat také. Pokud spojení nefunguje, jako první ověřte, že obě strany používají stejný komunikační režim. Nevíte-li, co má vzdálený počítač nastaveno, vyzkoušejte adaptivní režim. Jádru se možná podaří správně detekovat potřebný typ.

Příkaz **slattach** umožňuje nejen aktivovat protokol SLIP, ale i jiné sériové protokoly, například PPP nebo KISS (další z protokolů používaných v radiových sítích). Takovéto jeho použití nicméně není příliš obvyklé, protože podporu zmíněných protokolů zajišťují lepší nástroje. Podrobnosti naleznete na manuálové stránce příkazu `slattach(8)`.

Když linku přepnete do režimu SLIP, musíte nakonfigurovat její síťové rozhraní. K tomu použijeme standardní příkazy **ifconfig** a **route**. Předpokládejme, že jsme se z brány **vlager** připojili k serveru **cowslip**. Pak je nutné spustit následující sekvenci příkazů:

```
# ifconfig sl0 vlager-slip pointopoint cowslip
# route add cowslip
# route add default gw cowslip
```

První příkaz nastaví rozhraní jako point-to-point spojení s hostitelem **cowslip**, druhý a třetí příkaz přidají směrovací záznamy na hostitele **cowslip** a implicitní trasu přes **cowslip**.

U příkazu **ifconfig** je vhodné všimnout si dvou věcí: Parametr pointopoint specifikuje adresu vzdáleného konce linky, pro místní rozhraní SLIP používáme název vlager-slip.

Už jsme říkali, že rozhraní SLIP může pracovat se stejnou adresou jako ethernetové rozhraní brány **vlager**. Název **vlager-slip** pak bude pouhým aliasem IP adresy 172.16.1.1. Druhá možnost je přiřadit lince SLIP samostatnou adresu. Používá se to zejména v případě, že lokální síť používá neregistrované IP adresy tak, jak je tomu v síti pivovaru. O této problematice budeme podrobněji hovořit v další části.

Ve zbytku kapitoly budeme lokální SLIP rozhraní brány **vlager** označovat názvem **vlager-slip**.

Při rušení spojení pomocí protokolu SLIP musíte nejprve odstranit všechny trasy přes hostitele **cowslip**. K tomu slouží příkaz **route** s parametrem `del`. Potom je nutné odpojit rozhraní a poslat programu **slattach** signál HUP. Nakonec je třeba zavěsit modem pomocí terminálového programu:

```
# route del default
# route del cowslip
# ifconfig sl0 down
# kill -HUP 516
```

Hodnotu *516* je nutné nahradit příslušným identifikátorem procesu **slattach**, který chcete ukončit. (Identifikátor zjistíte například příkazem **ps ax**.)

Práce s privátními IP sítěmi

V kapitole 5 jsem se zmiňoval, že pivovar používá ethernetovou síť s neregistrovanými IP adresami, které jsou rezervovány pouze pro interní použití. Pakety z nebo na takovou síť se v Internetu nesměrují. Pokud bychom se přes bránu **vlager** připojili na počítač **cowslip** a brána by fungovala jako směrovač sítě pivovaru, počítače na této síti by se nemohly spojit s „opravdovými“ internetovými počítači, protože jejich pakety by byly prvním pořádným směrovačem v tichosti zahozeny.

Abychom tento problém vyřešili, nakonfigurujeme bránu **vlager** jako jakousi „startovací rampu“ internetových služeb. Vzhledem k vnějšímu světu se bude chovat jako normální počítač připojený protokolem SLIP s registrovanou IP adresou (kterou jí pravděpodobně přidělí poskytovatel přístupu provozující bránu **cowslip**). Kdokoliv se k bráně **vlager** přihlásí, bude moci používat textově orientované programy jako **ftp**, **telnet** nebo dokonce **lynx** a využívat tak Internet. Kdokoliv v síti pivovaru se tedy může telnetnout a přihlásit k bráně **vlager** a na ní pak spouštět internetové programy. Kvůli uživatelům služby WWW bychom například mohli na bráně provozovat takzvaný *proxy server*, který by předával požadavky uživatelů ze všech počítačů na odpovídající internetové hostitele.

Nutnost přihlásit se k bráně **vlager** ovšem práci s Internetem poněkud komplikuje. Kromě toho, že nám ale ušetřila papírování (a náklady) spojené s registrací celé IP sítě, zároveň nám funguje jako firewall. Firewally jsou vyhrazené počítače, které poskytují uživatelům lokální sítě omezený přístup k Internetu, aniž by zároveň vystavovaly hostitele na interní síti možnostem útoků z vnějšího světa. Konfigurace jednoduchého firewallu je popsána v kapitole 9. V kapitole 11 budeme hovořit o funkci zvané „IP maškaráda“, která představuje mocnou alternativu k proxy serverům.

Předpokládejme, že pivovar má pro svou SLIP linku přidělenu IP adresu 192.168.5.74[60]. Jediné co musíte udělat bude zajistit, aby ve výše popsané konfiguraci byla prostřednictvím souboru `/etc/hosts` tato adresa přidělena rozhraní **vlager-slip**. Samotný postup aktivace SLIP spojení se nijak nezmění.

Použití nástroje dip

Až sem to bylo poměrně jednoduché. Přesto však možná budete chtít výše uvedené kroky zautomatizovat natolik, aby bylo možné celý postup vyvolat jediným příkazem, který otevře sériové zařízení, zavolá modemem k poskytovateli, přihlásí se, přepne linku do režimu SLIP a nakonfiguruje síťové rozhraní. Přesně k tomu účelu slouží nástroj **dip**.

dip znamená *Dialup IP*. Původně jej napsal Fred van Kempen a postupně jej modifikovala celá řada lidí. Dnes existuje jedna verze, kterou používají prakticky všichni: Verze `dip337p-uri` je součástí většiny moderních distribucí Linuxu, kromě toho je k dispozici v FTP archivu **metalab.unc.edu**.

Nástroj **dip** je interpretem jednoduchého skriptového jazyka, který se vám stará o modem, nastavuje linku do režimu SLIP a konfiguruje různá rozhraní. Skriptovací jazyk je velmi mocný a bude vyhovovat většině konfigurací.

Aby mohl **dip** nakonfigurovat rozhraní SLIP, potřebuje práva superuživatele. Mohlo by to svádět k tomu povolit programu setuid root, aby jej mohli všichni uživatelé použít bez toho, že byste superuživatelská práva přidělili přímo jim. To je velmi nebezpečné, protože nastavením falešných rozhraní a implicitních tras je možné zcela rozvrátit provoz sítě. A co je ještě horší, vaši uživatelé tím získají možnost spojení s libovolným serverem SLIP a mohou tak síť vystavit nebezpečným útokům. Pokud tedy chcete umožnit svým uživatelům vytvářet spojení SLIP, napište pro každý plánovaný server SLIP malý wrapper, který teprve bude spouštět **dip** se skriptem pro vytvoření příslušného spojení. Dobře napsanému wrapperu je možné bez obav přidělit superuživatelská práva[61]. Alternativní a pružnější řešení je povolit oprávněným uživatelům superuživatelský přístup k programu **dip** pomocí nástroje jako je **sudo**.

Příklad skriptu

Předpokládejme, že se protokolem SLIP připojujeme k hostiteli **cowslip** a že jsme k vytvoření tohoto spojení napsali pro program **dip** skript nazvaný `cowslip.dip`. Program **dip** budeme spouštět s názvem skriptu jako s parametrem:

```
# dip cowslip.dip
```

60 Pozn. překladatele: Všimněte si, že adresa 192.168.5.74 je rovněž rezervovaná privátní adresa! To už ale není problém náš, ale problém poskytovatele připojení. Někteří poskytovatelé totiž, aby „neplýtvali“ svými reálnými IP adresami, přidělují klientům privátní adresy a nějakou svou hlavní branou pak zajišťují IP maškarádu. Pokud vás pohoršuje, že platíte a nedostali jste „opravdovou“ adresu, nemáte k tomu důvod. Při dobře nastavené maškarádě je pro vás převod adres zcela transparentní a zadarmo dostáváte výhodu dobré ochrany vašeho počítače před zlým okolním světem. Jste omezeni pouze v tom, že ze svého počítače nemůžete poskytovat vnějšímu světu služby WWW a jiných serverů.

61 Program **diplogin** musí být rovněž spouštěn s setuid root. Viz text na konci kapitoly.

```
DIP: Dialup IP Protocol Driver version 3.3.7 (12/13/93)
Written by Fred N. van Kempen, MicroWalt Corporation.
connected to cowslip.moo.com with addr 192.168.5.74
#
```

Samotný skript je uveden v příkladu 7.1.

Příklad 7.1 – Skript pro program dip.

```
# Příklad dip skriptu pro připojení ke cowslip-u
# Nastavení lokálního a vzdáleného jména a adresy
        get $local vlager-slip
        get $remote cowslip
        port ttyS3              # volba sériového portu
        speed 38400             # nastavení maximální rychlosti
        modem HAYES             # nastavení typu modemu
        reset                   # reset modemu a tty
        flush                   # smazání odezvy od modemu
# Příprava na vytáčení
        send ATQOV1E1X1\r
        wait OK 2
        if $errlvl != 0 goto error
        dial 41988
        if $errlvl != 0 goto error
        wait CONNECT 60
        if $errlvl != 0 goto error
# Teď jsme připojeni
        sleep 3
        send \r\n\r\n
        wait ogin: 10
        if $errlvl != 0 goto error
        send Svlager\n
        wait ssword: 5
        if $errlvl != 0 goto error
        send knockknock\n
        wait running 30
        if $errlvl != 0 goto error
# Jsme přihlášení, vzdálená strana spouští SLIP.
        print Connected to $remote with address $rmtip
        default                 # Tato linka bude implicitní trasou
        mode SLIP               # Taky přejdeme do režimu SLIP
# ošetření případných chyb
error:
        print SLIP to $remote failed.
```

Po spojení s hostitelem **cowslip** a povolení protokolu SLIP se proces **dip** odpojí od terminálu a poběží na pozadí. Potom můžete začít používat normální síťové služby po lince SLIP. Chcete-li spojení ukončit, vyvolejte nástroj **dip** s parametrem `-k`. Tento příkaz pošle procesu **dip** signál HUP, přičemž použije záznam identifikačního čísla procesu **dip**, který je uložen v souboru `/etc/dip.pid`:

```
# dip -k
```

Ve skriptovém jazyku nástroje **dip** znamenají klíčová slova uvedená symbolem dolaru názvy proměnných. Nástroj má předdefinovanou skupinu proměnných, které uvádíme níže. Například proměnné *$remote* a *$local* obsahují názvy vzdáleného a místního hostitele, kteří se účastní spojení pomocí protokolu SLIP.

První dva příkazy ve vzorovém skriptu jsou příkazy **get**, které programu **dip** umožňují nastavovat hodnoty proměnných. Zde nastavujeme název místního a vzdáleného počítače na **vlager**, respektive **cowslip**.

Dalších pět řádků nastavuje terminálovou linku a modem. Příkaz **reset** pošle modemu inicializační řetězec. Následující příkaz **flush** odstraní odpověď od modemu, aby mohla správně fungovat následující přihlašovací sekvence. Tato přihlašovací sekvence je poměrně přehledná: nejdříve se vytočí číslo 41988, což je telefonní číslo hostitele **cowslip** a přihlásíme se na účet **Svlager** s heslem **knockknock**. Příkaz **wait** způsobí čekání na řetězec uvedený jako první parametr, druhý parametr udává timeout, po kterém se čekání ukončí, pokud řetězec nedorazí. Příkazy **if** na různých místech přihlašovací procedury průběžně kontrolují, zda nedošlo k nějaké chybě.

Posledními příkazy spouštěnými po přihlášení jsou **default**, který nastaví linku SLIP jako implicitní trasu, a **mode**, který nastaví režim linky a nakonfiguruje její rozhraní.

Referenční příručka programu dip

V následujícím textu uvádíme popis většiny příkazů programu **dip**. Přehled všech dostupných příkazů získáte, když nástroj **dip** spustíte v testovacím režimu a zadáte příkaz **help**. Chcete-li zjistit syntaxi konkrétního příkazu, zadejte jej bez parametrů. (Nefunguje to samozřejmě pro příkazy, které parametry nepoužívají.) Následující příklad ilustruje použití nápovědy programu **dip**:

```
# dip -t
DIP: Dialup IP Protocol Driver version 3.3.7p-uri (25 Dec 96)
Written by Fred N. van Kempen, MicroWalt Corporation.
Debian version 3.3.7p-2 (debian).

DIP> help
DIP knows about the following commands:

        beep        bootp       break       chatkey     config
        databits    dec         default     dial        echo
        flush       get         goto        help        if
        inc         init        mode        modem       netmask
        onexit      parity      password    proxyarp    print
        psend       port        quit        reset       securidfixed
        securid     send        shell       skey        sleep
        speed       stopbits    term        timeout     wait

DIP> echo
Usage: echo on|off
DIP>
```

V průběhu následujícího textu ukazují příklady začínající výzvou `DIP>` jak daný příklad zadat v ladicím režimu a jaká bude odezva příkazu. Příklady bez této výzvy se vztahují k obsahu skriptů.

Příkazy modemu

Nástroj **dip** obsahuje značný počet příkazů pro konfiguraci sériové linky a modemu. Význam některých z nich je zřejmý, například příkazu **port** vybírajícího sériový port, nebo příkazů **speed**,

databits, **stopbits** a **parity**, které nastavují obecné parametry linky. Příkaz **modem** nastavuje typ modemu. V současné době je jediným podporovaným typem modemu typ HAYES (musí být zadáno velkými písmeny). Programu **dip** musíte typ modemu zadat, jinak by odmítl provést příkazy **dial** a **reset**. Příkaz **reset** posílá modemu inicializační řetězec; hodnota řetězce přitom závisí na typu modemu. Pro modemy kompatibilní se standardem HAYES je tímto řetězcem příkaz **ATZ**.

Příkaz **flush** slouží ke smazání všech odpovědí, které modem do této chvíle poslal. Bez něj by nemusela přihlašovací sekvence následující po inicializaci modemu fungovat správně, protože by načetla odezvu **OK**, vrácenou modemem po provedení inicializace.

Příkaz **init** nastavuje inicializační řetězec, který bude předán modemu před začátkem vytáčení. Implicitním řetězcem pro modemy kompatibilní se standardem HAYES je „ATE0 Q0 V1 X1“, který zapne zobrazování příkazů, dlouhé kódy výsledků a nastaví volání naslepo (bez detekce vytáčecího tónu). Moderní modemy mají poměrně vhodně zvolená standardní nastavení, takže tato inicializace nebývá nutná, i když na druhé straně ničemu neuškodí.

Příkaz **dial** odešle inicializační řetězec a vytočí číslo vzdáleného systému. Implicitním vytáčecím příkazem pro modemy kompatibilní se standardem HAYES je příkaz **ATD**.

Příkaz echo

Příkaz echo slouží jako ladící prostředek. Po zadání příkazu **echo on** bude **dip** vypisovat na konzolu všechno, co posílá na sériovou linku. Zobrazování lze vypnout příkazem **echo off**.

Nástroj **dip** také umožňuje dočasně opustit skriptový režim a vstoupit do terminálového režimu. V tomto režimu lze nástroj **dip** používat jako jakýkoliv jiný běžný terminálový program. Můžete v něm zapisovat na sériovou linku nebo z ní číst. Chcete-li tento režim opustit, stiskněte kombinaci kláves **Ctrl-]**.

Příkaz get

Příkazem **get** se v programu **dip** nastavují proměnné. Nejjednodušším způsobem je přiřadit proměnné konstantní hodnotu tak, jak to děláme ve skriptu `cowslip.dip`. Můžete ale také požádat uživatele, aby hodnotu zadal. K tomu slouží klíčové slovo **ask**, které je nutno uvést na místě hodnoty proměnné:

```
DIP> get $local ask
Enter the value for $local: _
```

Třetí metoda spočívá v získání hodnoty ze vzdáleného hostitele. Na první pohled to vypadá zvláštně, ale v některých případech je tento způsob velmi užitečný. Některé servery SLIP vám nedovolí při spojení pomocí protokolu SLIP používat vlastní IP adresu a místo toho vám při každém přihlášení přidělí nějakou adresu ze svého seznamu adres a zobrazí zprávu, která vás bude o přidělené adrese informovat. Pokud tato zpráva bude například „`Your address: 193.174.7.202`“, můžete si následujícím kódem tuto adresu vyzvednout:

```
# konec přihlášení
wait address: 10
get $locip remote
```

Příkaz print

Tento příkaz umožňuje vypsat text na konzolu, ze které byl nástroj **dip** spuštěn. V rámci příkazu print lze použít libovolné definované proměnné, například:

```
DIP> print Using port $port at speed $speed
```

```
Using port cua3 at speed 38400
```

Názvy proměnných

Nástroj **dip** rozumí pouze předdefinované skupině proměnných. Název proměnné vždy začíná symbolem dolaru a musí být psán malými písmeny.

Proměnné *$local* a *$locip* obsahují název místního hostitele a jeho IP adresu. Když proměnné *$local* přiřadíte kanonický název hostitele, pokusí se **dip** automaticky tento název převést na IP adresu a přiřadit ji proměnné *$locip*. Obdobně se **dip** chová při nastavení proměnné *$locip*: reverzním dotazem se pokusí zjistit název, který této adrese odpovídá a uloží jej do proměnné *$local*.

Proměnné *$remote* a *$rmtip* se chovají úplně stejně a obsahují název a IP adresu vzdáleného hostitele.

Proměnná *$mtu* obsahuje hodnotu MTU daného spojení.

Tyto proměnné jsou jedinými proměnnými, kterým mohou být přímo přidělovány hodnoty pomocí příkazu **get**. Hodnoty dalších proměnných se nastavují v důsledku volání stejnojmenných konfiguračních příkazů a obsahují použitou hodnotu nastavení – například *$modem*, *$port* nebo *$speed*.

Proměnná *$errlvl* umožňuje přistupovat k výsledku naposledy spuštěného příkazu. Návratová hodnota rovná nule znamená úspěšnou operaci, zatímco nenulová hodnota naznačuje chybnou operaci.

Příkazy if a goto

Příkaz **if** umožňuje podmíněné větvení, nejedná se o úplný příkaz *if* známý z různých programovacích jazyků. Jeho syntaxe je následující:

```
if var op number goto label
```

Celý podmíněný výraz je tvořen jednoduchým porovnáním hodnoty proměnné (specifikované parametrem *var*) s celočíselnou hodnotou (specifikovanou parametrem *number*). Operátor porovnání *op* může být ==, !=, >, <, >= nebo <=.

Příkaz **goto** zajistí, že skript bude pokračovat na řádce za návěštím definovaným hodnotou *label*. Návěští musí začínat na prvním znaku řádku a bezprostředně za ním musí následovat dvojtečka.

Příkazy send, wait a sleep

Tyto příkazy umožňují implementovat ve skriptech jednoduché dialogy. Příkaz **send** zapíše své parametry na sériovou linku. Nepodporuje použití proměnných, zato však rozumí všem kombinacím znaků s převráceným lomítkem, které pocházejí z jazyka C, jako je například \n pro nový řádek nebo \b pro backspace. Znak tildy (~) slouží jako zkratka pro sekvenci CR/LF.

Parametrem příkazu **wait** je slovo. Příkaz čte vstup ze sériové linky tak dlouho, dokud nedojde sekvence znaků odpovídající zadanému slovu. Slovo nesmí obsahovat mezeru. K příkazu wait můžete přidat i druhý nepovinný parametr, který bude udávat délku čekání v sekundách. Pokud očekávané slovo v zadané době nedojde, příkaz nastaví hodnotu proměnné *$errlvl* na 1. Příkaz se používá k detekci přihlašovací a jiných výzev.

Příkaz **sleep** slouží k nastavení časové prodlevy, například k vyčkání na dokončení přihlašovací procedury. Interval je zadáván v sekundách.

Příkazy mode a default

Tyto příkazy se používají k přepnutí sériové linky do režimu SLIP a ke konfiguraci rozhraní. Příkaz **mode** je posledním příkazem spuštěným v nástroji **dip** předtím, než se nástroj přepne do režimu démona.

Příkaz **mode** má jako parametr název protokolu. Nástroj **dip** v současné době akceptuje protokoly SLIP, CSLIP, SLIP6, CSLIP6, PPP a TERM. Bohužel v současné verzi není možné použití adaptivního protokolu SLIP.

Po přepnutí sériové linky do režimu SLIP se spustí příkaz **ifconfig**, který nakonfiguruje rozhraní jako dvoubodový spoj a příkaz **route**, kterým se nastaví trasa ke vzdálenému hostiteli.

Pokud skript navíc spustí před příkazem **mode** i příkaz **default**, nastaví se SLIP linka také jako implicitní trasa.

Spuštění v režimu serveru

Nastavení klienta s protokolem SLIP bylo poměrně obtížné. Nastavení hostitele aby fungoval jako SLIP server je mnohem jednodušší.

Existují dva způsoby konfigurace SLIP serveru. V obou případech potřebujete pro každého SLIP klienta jeden přihlašovací účet. Řekněme, že poskytujete SLIP linku Arthuru Dentovi na adrese **dent.beta.com**. Do souboru `passwd` můžete přidat následující řádek, kterým vytvoříte účet **dent**:

```
dent:*:501:60:Arthur Dent's SLIP account:/tmp:/usr/sbin/diplogin
```

Pomocí příkazu **passwd** pak nastavíte heslo tohoto účtu.

Jednou z možných realizací SLIP serveru je použití nástroje **dip** v režimu serveru. V tom případě se spouští příkazem **diplogin**. Jeho hlavním konfiguračním souborem bude soubor `/etc/diphosts`, kde uvádíte IP adresu přiřazenou klientovi poté, co se přihlásí. Alternativně můžete použít nástroj **sliplogin**, který je odvozen z balíku BSD a umožňuje mnohem pružnější konfigurační schéma, na jehož základě lze spouštět skripty kdykoliv se vzdálený hostitel připojí nebo odpojí.

Když se nyní uživatel **dent** přihlásí, spustí se nástroj **dip** jako server. Aby zjistil, zda je přihlášený uživatel oprávněn používat službu SLIP, vyhledá jeho jméno v souboru `/etc/diphosts`. Tento soubor definuje přístupová práva a parametry spojení každého uživatele služby SLIP. Obecný formát záznamu v souboru `/etc/diphosts` vypadá takto:

```
# /etc/diphosts
user:password:rem-addr:loc-addr:netmask:comments:protocol,MTU
#
```

Jednotlivé údaje jsou popsány v tabulce 7.2.

Tabulka 7.2 – Popis údajů v souboru /etc/diphosts

Údaj	Popis
`user`	Jméno uživatele k němuž se tento údaj vztahuje.
`password`	Tento údaj umožňuje dodatečnou ochranu dalším heslem v závislosti na připojení. Můžete zde zadat zakódované heslo (stejně jako v souboru /etc/passwd) a program **diplogin** pak uživatele požádá o zadání tohoto hesla předtím, než mu povolí SLIP přístup. Toto heslo se používá navíc kromě klasického přihlašovacího hesla, které bude muset uživatel zadat také.
`rem-addr`	Adresa přiřazená vzdálenému počítači. Může být zadána buď názvem, který bude poté přeložen, nebo přímo jako IP adresa v tečkové notaci.
`loc-addr`	Adresa přiřazená místnímu konci SLIP linky. Rovněž může být zadána názvem nebo číselnou hodnotou.
`netmask`	Síťová maska používaná pro potřeby směrování. Tento údaj je často chápán chybně. Nejedná se o masku pro samotnou linku SLIP, ale o masku, která bude společně s údajem rem-addr sloužit ke směrování na vzdálený systém. Musí být proto použita taková maska, jakou používá síť vzdáleného systému.
`comments`	Tento údaj může být libovolný text, sloužící k popisu údajů v souboru. Není k ničemu využíván.
`protocol`	Tento údaj definuje, jaký protokol nebo režim linky se má použít. Možné hodnoty jsou stejné jako hodnoty parametru –p příkazu **slattach**.
`MTU`	Povolené MTU dané linky. Tento údaj definuje největší délku datagramu, přenášenou linkou. Jakýkoliv datagram poslaný na linku a delší než MTU bude fragmentován na části kratší než MTU. Typicky se MTU nastavuje na obou koncích linky stejně.

Příklad záznamu pro uživatele **dent** by mohl vypadat takto:

```
dent::dent.beta.com:vbrew.com:255.255.255.0:Arthur Dent:CSLIP,296
```

Uživatel **dent** má povolen přístup bez dodatečné ochrany heslem. Bude mít přiřazenu adresu odpovídající stroji **dent.beta.com** se síťovou maskou 255.255.255.0. Jeho implicitní trasa povede na počítač **vbrew.com** a bude používat protokol CSLIP s MTU 296 bajtů.

Když se uživatel **dent** přihlásí, nástroj **diplogin** si o něm vytáhne informace ze souboru `diphosts`. Pokud není druhý údaj prázdný, vyzve ho k zadání „externího bezpečnostního hesla“. Řetězec zadaný uživatelem se pak zašifruje a porovná s heslem v souboru `diphosts`. Nebudou-li hesla souhlasit, bude přihlášení odmítnuto. Obsahuje-li heslo řetězec s/key a nástroj **dip** byl přeložen s podporou S/Key, dojde k autentikaci protokolem S/Key. Tento autentikační mechanismus je popsán v dokumentaci k balíku **dip**.

Po úspěšném přihlášení přepne **diplogin** linku do režimu CSLIP nebo SLIP a nastaví rozhraní a směrování. Spojení trvá až do doby, než se vzdálený uživatel odpojí a modem zavěsí linku. Poté **diplogin** přepne linku zpět do normálního režimu a ukončí se.

Nástroj **diplogin** vyžaduje práva superuživatele. Pokud program **dip** nespouštíte jako setuid root, musíte **diplogin** vytvořit jako samostatnou kopii programu **dip** a ne pouze jako odkaz. Pak můžete **diplogin** bezpečně spouštět setuid root, aniž by byl ovlivněn status programu **dip**.

KAPITOLA 8

Protokol PPP

Protokol PPP slouží stejně jako protokol SLIP k posílání datagramů po sériové lince, avšak odstraňuje spoustu nedostatků, kterými trpí protokol SLIP. V první řadě umožňuje i přenos jiných protokolů, nejen protokolu IP. Dále zajišťuje chybovou detekci na lince, zatímco protokol SLIP bez problémů přenáší i poškozené datagramy, pokud není poškozena přímo jejich hlavička. Navíc umožňuje komunikujícím stranám dohodnout na počátku parametry spojení, například IP adresy nebo maximální velikost datagramu a zajišťuje ověření totožnosti klienta. Vestavěný mechanismus dohody usnadňuje automatické navázání spojení, možnost autentikace klienta odstraňuje nutnost vytvářet speciální uživatelské účty, které potřebuje protokol SLIP. Pro každou z těchto funkcí používá PPP samostatný protokol. Jednotlivé stavební kameny protokolu PPP si nyní popíšeme. Tato kapitola nicméně není úplným popisem protokolu PPP. Pokud potřebujete vědět více, doporučujeme vám přečíst si definiční RFC a další zhruba desítku RFC, které s tímto protokolem souvisejí[62]. Nakladatelství O'Reilly navíc vydalo velmi podrobnou knihu, *Using & Managing PPP* Andrewa Suna.

Na nejnižší vrstvě protokolu PPP se nachází protokol HDLC, *High-Level Data Link Control Protocol*. Ten definuje ohraničení jednotlivých rámců protokolu PPP a poskytuje 16bitový kontrolní součet[63]. Na rozdíl od primitivnějšího zapouzdření v protokolu SLIP může rámec v protokolu PPP obsahovat pakety i jiných protokolů než IP, například protokolu IPX sítě Novell nebo protokolu Appletalk. Tato možnost se implementuje tak, že základní rámec HDLC je rozšířen o údaj definující typ paketu, který je rámcem přenášen.

Protokol LCP, *Link Control Protocol*, se používá nad protokolem HDLC a používá se k dohodnutí parametrů týkajících se datového spojení, jako jsou například *Maximum Receive Unit*, MRU, jež uvádí maximální velikost datagramu, kterou je komunikující strana ochotná přijímat.

Důležitým krokem ve fázi konfigurace spojení protokolem PPP je ověření totožnosti klienta. Ačkoliv není povinné, je u vytáčených linek prakticky nezbytné, aby se znemožnil libovolný přístup komukoliv. Volaný hostitel (server) typicky vyzve klienta k ověření tím, že prokáže znalost nějakého tajného klíče. Pokud klient nebude schopen tuto znalost prokázat, spojení se ukončí. U protokolu PPP funguje ověřování totožnosti oběma směry; to znamená, že i klient může požádat server, aby prokázal svou totožnost. Obě tyto ověřovací procedury jsou vzájemně nezávislé. Autentikace se provádí dvěma různými protokoly, *Password Authentication Protocol* (PAP) a *Challenge Handshake Authentication Protocol* (CHAP).

Každý síťový protokol, který je směrován po datové lince (například IP, AppleTalk a podobné) se konfiguruje dynamicky odpovídajícím protokolem *Network Control Protocol* (NCP)[64]. Například

62 Příslušné RFC dokumenty jsou uvedeny v seznamu literatury na konci knihy.

63 Protokol HDLC je podstatně obecnější, definuje jej International Standards Organisation (ISO) a je základní součástí i jiných protokolů, například X.25.

64 Pozn. překladatele: Neplést s NetWare Core Protocol, což je protokol, jímž v sítích Novell NetWare komunikují stanice a servery.

aby bylo možné po PPP lince poslat IP datagram, musí se obě strany nejprve dohodnout, jakou budou používat IP adresu. K tomu jim poslouží protokol *Internet Protocol Control Protocol* (IPCP) – jeden z protokolů množiny NCP.

Kromě standardního posílání IP datagramů po lince podporuje protokol PPP také Van Jacobsonovu kompresi hlaviček IP datagramů. Tato technika změnšuje velikost hlaviček až na tři bajty. Používá se i v protokolu CSLIP a je všeobecně známá pod názvem VJ komprese hlaviček. Použití této komprese může být rovněž sjednáno při spuštění protokolem IPCP.

Protokol PPP v Linuxu

V Linuxu je činnost protokolu PPP rozdělena na dvě části; na komponentu jádra, která obsluhuje nízkoúrovňové protokoly (HDLC, IPCP, IPXCP a podobně) a na démona **pppd**, který běží v uživatelském prostoru a obsluhuje protokoly vyšší úrovně, například PAP a CHAP. Současná implementace protokolu PPP na Linuxu obsahuje démona **pppd** a program **chat**, který automatizuje připojení ke vzdálenému systému.

Ovladač pro jádro napsal Michael Callahan a přepracoval jej Paul Mackerras. Démon **pppd** byl odvozen z volně šiřitelné implementace protokolu PPP[65] v počítačích Sun a 386BSD, jejímž autorem je Drew Perkins a další a nyní ji spravuje Paul Mackerras. Na platformu Linuxu ho převedl Al Longyear. Program **chat** napsal Karl Fox[66].

Protokol PPP je stejně jako protokol SLIP implementován pomocí speciálního režimu linky. Chcete-li používat nějakou sériovou linku jako linku s protokolem PPP, musíte nejprve obvyklým způsobem vytvořit spojení pomocí modemu a následně převést linku do režimu protokolu PPP. V tomto režimu budou všechna příchozí data předána ovladači protokolu PPP, který ověří platnost rámců protokolu HDLC (každý rámec HDLC je doplněn 16bitovým kontrolním součtem), rozbalí je a předá k dalšímu zpracování. Současná implementace dokáže přenášet protokol IP, s případnou Van Jacobsonovou kompresí hlaviček, a dále protokol IPX.

Ovladači jádra operačního systému pomáhá démon **pppd**, který realizuje inicializační a autentikační fázi, které musejí proběhnout před zahájením samotného datového přenosu. Chování démona **pppd** je možné doladit pomocí mnoha parametrů. Jelikož je ovladač protokolu PPP poměrně složitý, není možné popsat všechny tyto parametry v jediné kapitole. Tato kniha tak nepopisuje všechny vlastnosti démona **pppd**, poskytuje pouze stručný úvod. Chcete-li se o tomto problému dozvědět více, přečtěte si knihu *Using & Managing PPP*, manuálové stránky démona **pppd** a soubor README v distribuci zdrojových kódů démona. Zde najdete odpovědi na většinu otázek, o kterých v této kapitole nehovoříme. Užitečný je rovněž dokument PPP-HOWTO.

Asi nejlepší pomoci s nastavením protokolu PPP se vám dostane od někoho, kdo používá stejnou distribuci Linuxu jako vy. Problémy s konfigurací protokolu PPP jsou velmi běžné, takže můžete vyzkoušet diskusní skupiny uživatelů Linuxu ve vašem okolí, nebo linuxový IRC kanál. Pokud stále něco nebudete vědět, přečtěte si konferenci **comp.protocols.ppp**. Na tomto místě se potkává většina lidí, kteří na vývoji démona **pppd** pracují.

Spuštění démona pppd

Když se chcete připojit na Internet pomocí protokolu PPP, musíte nejdříve nastavit základní síťovou podporu, jako je lokální zpětnovazebné zařízení a resolver. O této problematice jsme mluvili v kapitolách 5 a 6. Jako jmenný server můžete v souboru `/etc/resolv.conf` nastavit server vašeho poskytovatele připojení, znamená to ale, že všechny DNS požadavky budou přenášeny vaší

65 Obecné otázky týkající se protokolu PPP vám zodpoví v konferenci Linux-net na adrese **vger.rutgers.edu**.

66 Karla můžete kontaktovat na adrese **karl@morningstar.com**.

sériovou linkou. Tato situace není optimální, protože čím jste blíže jmennému serveru, tím rychlejší bude vyhledávání. Alternativním řešením je nakonfigurovat na některém počítači na vaší síti caching-only server. Pak to znamená, že první dotaz na určité jméno bude poslán po sériové lince, všechny další dotazy však budou obslouženy přímo vaším lokálním serverem a odezva bude mnohem rychlejší. O této konfiguraci jsme mluvili v kapitole 6.

V úvodním příkladu navázání PPP spojení pomocí démona **pppd** budeme předpokládat, že jsme opět na hostiteli **vlager**. Nejprve zavoláme na server **c3po** a přihlásíme se na účet **ppp**. Server **c3po** aktivuje svůj ovladač protokolu PPP. Po ukončení komunikačního programu, kterým navážeme spojení, spustíme následující příkaz, přičemž místo zařízení `ttyS3` použijete to zařízení, jímž jsme se připojili:

```
# pppd /dev/ttyS3 38400 crtscts defaultroute
```

Tento příkaz přepne sériovou linku `ttyS3` do režimu PPP a naváže IP spojení se serverem **c3po**. Linka bude pracovat s přenosovou rychlostí 38400 b/s. Parametr **crtscts** zapíná hardwarový handshake na sériovém portu, což je pro rychlosti nad 9600 b/s nezbytné.

Démon **pppd** si ihned po spuštění dohodne se vzdáleným protějškem některé parametry linky prostřednictvím protokolu LCP. Při sjednávání parametrů budou obvykle fungovat implicitní hodnoty, takže se jimi nebudeme nyní zabývat. V rámci této dohody si oba konce linky vymění nebo vyžádají své IP adresy.

Pro tuto chvíli budeme také předpokládat, že server **c3po** od nás nebude vyžadovat žádný typ autentikace, čímž máme konfigurační fázi úspěšně za sebou.

Následně dohodne démon **pppd** se svým protějškem parametry IP spojení pomocí protokolu IPCP. Protože jsme démonu nezadali žádnou IP adresu, pokusí se prostřednictvím resolveru zjistit adresu lokálního počítače. Oba hostitelé si navzájem sdělí své IP adresy.

Tato implicitní nastavení jsou zpravidla dostačující. Dokonce i v případě, že je váš počítač připojen do sítě Ethernet, můžete použít stejnou IP adresu jak pro Ethernet, tak i pro rozhraní PPP. Nicméně démon nám umožňuje nastavit jinou IP adresu, popřípadě si adresu vyžádat od vzdáleného počítače. Tyto volby jsou popsány později v části *Nastavení IP konfigurace*.

Po provedení fáze nastavení protokolem IPCP připraví démon **pppd** síťovou vrstvu pro použití linky PPP. Nejprve nastaví síťové rozhraní PPP jako dvoubodové spojení s tím, že první PPP linka bude pojmenována `ppp0`, druhá `ppp1` a tak dále. Dále do směrovací tabulky přidá položku ukazující na hostitele na druhém konci spojení. Ve výše zmíněném příkladu nastaví démon **pppd** trasu na server **c3po** jako implicitní, protože jsme použili parametr defaultroute[67]. Tato volba zjednoduší směrování, protože způsobí, že všechny datagramy poslané jiným než lokálním hostitelům budou poslány na server **c3po** – což je jediná cesta, jak se s takovými hostiteli spojit. Démon **pppd** podporuje různá další směrovací schémata, budeme o nich podrobněji hovořit později.

Konfigurační soubory

Dříve než démon **pppd** analyzuje parametry na příkazovém řádku, projde několik souborů, zda neobsahují implicitní volby. Tyto soubory mohou obsahovat libovolné parametry příkazové řádky, které mohou být rozepsány na několika řádcích. Komentáře začínají symbolem #.

Prvním prohlíženým souborem je soubor `/etc/ppp/options`. Je rozumné pomocí tohoto souboru nastavit všeobecně platné volby, protože tak zabráníte uživatelům v nechtěném narušení bezpečnosti celého systému. Chcete-li například, aby démon **pppd** vyžadoval od svého protějšku ně-

67 Implicitní trasa se nastaví pouze v případě, že ještě žádná není nastavena.

jaký typ ověření totožnosti (buď pomocí protokolu PAP nebo pomocí protokolu CHAP), uveďte v tomto souboru volbu `auth`. Tuto volbu nemůže uživatel potlačit, takže nemůže navázat spojení pomocí protokolu PPP s žádným systémem, který není v ověřovací databázi. Některé volby nicméně lze potlačit, například nastavení připojovacího řetězce.

Druhým souborem čteným po souboru `/etc/ppp/options` je soubor `.ppprc`. Nachází se v domovském adresáři uživatele a umožňuje tak uživatelům nastavit vlastní standardní volby.

Příklad souboru `/etc/ppp/options` by mohl vypadat takto:

```
# Globální volby pro pppd běžící na vlager.vbrew.com
lock                          # používej zamykání zařízení dle UUCP
auth                          # požaduj autentikaci
usehostname                   # pro CHAP použij lokální jméno
domain vbrew.com              # naše doména
```

Klíčové slovo lock nastaví démona **pppd**, aby používal standardní metodu zamykání zařízení používanou v UUCP. Podle této konvence vytvoří každý proces, který přistupuje k sériovému zařízení (řekněme k zařízení `/dev/ttyS3`), soubor `LCK..ttyS3` ve speciálním adresáři zámkových souborů. Tím se dalším programům, jako je například **minicom** nebo **uucico**, signalizuje, že zařízení nemají používat, protože je používá protokol PPP.

Další tři volby se vztahují k autentikaci a tedy k bezpečnosti systému. Nastavení autentikace je ideální uvést v globálním konfiguračním souboru, protože jde o „privilegovaná“ nastavení a nelze je přepsat hodnotami v uživatelských souborech `~/.ppprc`.

Automatické vytáčení programem chat

Jednou z věcí, která se vám mohla zdát v předchozím příkladu nepohodlná, je nutnost manuálně navázat spojení dříve, než se spustí démon **pppd**. Na rozdíl od nástroje **dip** nemá démon **pppd** svůj vlastní skriptový jazyk, který by umožňoval vytočení a přihlášení ke vzdálenému systému. Místo toho spoléhá na nějaký externí program nebo skript příkazového interpretu, který za něj tento úkol provede. Příkaz, který to zajistí, lze démonu **pppd** předat pomocí volby `connect` na příkazovém řádku. Démon **pppd** přesměruje standardní vstup a výstup tohoto příkazu na sériovou linku.

Balík **pppd** obsahuje velmi jednoduchý program nazvaný **chat**, který je určen k takovéto automatizaci jednoduchých přihlašovacích sekvencí. Budeme o něm hovořit podrobněji.

Pokud používáte složitější přihlašovací postup, potřebujete něco výkonnějšího, než je **chat**. Užitečnou alternativou je například program **expect** Dona Libese. Obsahuje výkonný jazyk vycházející z jazyka Tcl, a byl navržen přesně pro tento druh aplikace. Pokud používáte přihlašovací mechanismus, který v sobě zahrnuje řekněme autentikaci s použitím nějakého „kalkulátorového“ generátoru klíče, program **expect** to dokáže. Protože existuje velké množství různých variant přihlašování, nebudeme se zde zabývat tím, jak vytvořit příslušné skripty. Poznamenáme jenom, že vytvořený skript můžete volat tak, že jeho název předáte programu **pppd** pomocí volby `connect`. Je také důležité upozornit vás, že po dobu běhu skriptu jsou standardní vstup a výstup přesměrovány na modem a nepracují na terminálu, na němž byl **pppd** spuštěn. Pokud potřebujete nějakou interakci s uživatelem, musíte ji ošetřit vytvořením samostatného virtuálního terminálu nebo nějakým jiným mechanismem.

Program **chat** umožňuje používat komunikační skripty ve stylu UUCP. Komunikační skript je v podstatě složen ze střídající se sekvence očekávaných řetězců, které přijdou ze vzdáleného systému a z odpovědí, které na ně posíláme. Následuje typický výpis části komunikačního skriptu:

```
ogin: b1ff ssword: s3k|<r1t
```

Tento skript říká programu **chat**, aby vyčkal až vzdálený systém pošle výzvu k přihlášení a potom mu poslal přihlašovací jméno **b1ff**. Čekáme pouze na řetězec `ogin:`, takže nebude vadit, když bude přihlašovací výzva začínat malým nebo velkým písmenem, nebo když dorazí ve zkomolené formě. Následující řetězec je opět očekávaný řetězec, který pozdrží program **chat**, dokud nepřijde výzva k zadání hesla, potom odešle naše heslo jako odpověď.

To je v podstatě vše o tom, jak komunikační skripty pracují. Kompletní skript sloužící k vytáčení serveru by musel samozřejmě obsahovat patřičné příkazy pro modem. Předpokládejme, že váš modem rozumí množině příkazů standardu Hayes a že telefonní číslo vytáčeného serveru je 318714. Kompletní volání programu **chat**, které by provedlo spojení se serverem **c3po,** by potom vypadalo následovně:

```
$ chat -v '' ATZ OK ATDT318714 CONNECT '' ogin: ppp word: GaGariN
```

Podle definice musí být první řetězec očekávaný řetězec, ale protože nám modem nic neřekne, dokud ho nepobídneme, necháme program **chat** „přeskočit" první řetězec tím, že mu zadáme jen prázný řetězec. Pak pokračujeme odesláním příkazu ATZ, což je inicializační řetězec pro modemy kompatibilní se standardem Hayes a následně budeme čekat na odpověď (OK). Následující řetězec pošle příkaz pro vytočení i s vytáčeným číslem a čeká na odezvu CONNECT. Dále následuje opět prázdný řetězec, protože teď nechceme nic posílat a čekáme na výzvu k přihlášení. Zbytek komunikačního skriptu funguje naprosto shodně s výše popsaným postupem. Celý popis je možná trochu matoucí, za chvíli ale uvidíme, že existuje způsob jak vytvořit skript pro program **chat** podstatně jednodušeji.

Parametr `-v` zajistí, že program **chat** bude veškeré aktivity zaznamenávat prostřednictvím služby **syslog**[68].

Zadávání komunikačního skriptu na příkazové řádce s sebou nese jistá rizika, protože si uživatelé mohou prohlédnout příkazovou řádku daného procesu pomocí příkazu **ps**. Tomuto problému se vyhnete, když umístíte komunikační skript do souboru jako `dial-c3po`. Pomocí parametru `-f` pak donutíte program **chat**, aby četl skript ze souboru a ne z příkazové řádky. Tato operace s sebou přináší další výhodu, protože sekvence dialogu nyní budou podstatně čitelnější. Budeme-li převádět náš příklad, soubor `dial-c3po` bude vypadat takto:

```
''      ATZ
OK      ATDT318714
CONNECT ''
ogin:   ppp
word:   GaGariN
```

Když vytváříme pro program **chat** skript tímto způsobem, zapisujeme sekvenci na jejíž přijetí čekáme na levou stranu řádku, sekvenci jíž odpovíme pak na pravou stranu řádku. V této podobě je vytáčecí skript podstatně čitelnější.

Celé spuštění démona **pppd** by mohlo nyní vypadat následovně:

```
# pppd connect "chat -f dial-c3po" /dev/ttyS3 38400 -detach \
        crtscts modem defaultroute
```

Kromě volby *connect* specifikující skript pro vytáčení jsme do příkazové řádky přidali ještě další dvě volby: *–detach* sdělí démonu **pppd**, aby se neodpojoval od konzoly a zůstal jako proces v po-

68 Pokud v souboru `syslog.conf` přesměrujete tyto zaznamenávané zprávy do souboru, ověřte, že soubor není veřejně čitelný. Program **chat** totiž zaznamenává celý komunikační skript včetně hesla.

zadí. Klíčové slovo *modem* sděluje démonu **pppd**, že na sériovém zařízení je připojen modem a aby proto provedl některé akce pro modem specifické, jako je například zavěšení linky před a po volání. Pokud toto klíčové slovo nepoužijete, nebude démon **pppd** sledovat signál DCD sériového portu a nepozná tak, pokud by vzdálená strana neočekávaně zavěsila.

Výše uvedené příklady byly poměrně jednoduché; program **chat** však umožňuje psát mnohem složitější komunikační skripty. Dovoluje například specifikovat řetězce, při jejichž přijetí se program ukončí s chybou. Typickými řetězci pro zastavení běhu skriptu jsou zprávy jako BUSY nebo NO CARRIER, které modem generuje v případě, že je volané číslo obsazené nebo vzdálená strana nezvedá telefon. Aby program rozpoznal tyto zprávy okamžitě a nečekal na vypršení časových intervalů, můžete je zadat na začátku skriptu pomocí klíčového slova ABORT:

```
$ chat -v ABORT BUSY ABORT 'NO CARRIER' '' ATZ OK ...
```

Podobným způsobem můžete ve skriptu změnit hodnotu čekací doby pomocí volby TIMEOUT. V některých případech je nutné podmíněné provedení určitých částí komunikačního skriptu: pokud například od vzdálené strany nepřijmete výzvu k přihlášení, budete chtít poslat příkaz BREAK nebo znak CR. Toho docílíte přidáním podskriptu k očekávanému řetězci. Ten se bude skládat ze sekvence posílaných a očekávaných řetězců, stejně jako tomu bylo v celém skriptu. Sekvence podskriptu je oddělena pomlčkami. Podskript se spouští tehdy, pokud očekávaný řetězec (k němuž je podskript připojen) nebude přijat v nastaveném časovém intervalu. V našem příkladu bychom mohli komunikační skript upravit takto:

```
ogin:-BREAK-ogin: ppp ssword: GaGariN
```

Když **chat** nepřijme v nastaveném intervalu výzvu k přihlášení, spustí podskript, který odešle příkaz BREAK a znovu čeká na přihlašovací výzvu. Když bude tentokrát výzva přijata, bude se pokračovat v přihlášení, pokud se v nastaveném intervalu nepřijme, ohlásí skript chybu.

Nastavení konfigurace protokolu IP

K nastavení parametrů protokolu IP v době navazování spojení slouží protokol IPCP. Typicky každá ze stran posílá paket IPCP Configuration Request, v němž specifikuje, které volby chce nastavit nestandardním způsobem a jak. Po přijetí požadavku vzdálená strana jednotlivé nároky posoudí a buď je schválí, nebo zamítne.

Démon **pppd** vám dává k dispozici velký prostor pro nastavování parametrů, které se budou při připojování domlouvat. Jednotlivé parametry se nastavují různými volbami příkazové řádky, o kterých budeme hovořit později.

Volba IP adres

Všechna IP rozhraní musí mít přidělenu IP adresu, proto i PPP zařízení má vždy nějakou IP adresu. Rodina protokolů PPP poskytuje mechanismus, který umožňuje automatické přiřazení IP adres PPP rozhraním. Tak je možné, že program PPP na jednom konci dvoubodového spoje přidělí vzdálenému rozhraní adresu, kterou bude používat, nebo mohou obě strany používat vlastní adresy.

Některé PPP servery obsluhující velký počet klientů přidělují IP adresy dynamicky: připojenému systému se adresa přidělí v okamžiku kdy zavolá a po odpojení se mu adresa odebere. Tím se počet potřebných IP adres omezí na počet současně možných připojení. Toto řešení je sice výhodné pro správce PPP serveru, obvykle však bývá méně výhodné pro připojující se klienty. V kapi-

tole 6 jsme hovořili o způsobu, jakým se IP adresy mapují na názvy hostitelů pomocí databází. Aby se klient mohl k vašemu systému připojit, musí znát jeho IP adresu nebo název. Pokud jste ale uživateli PPP služby, která vám přiděluje IP adresu dynamicky, těžko ji mohou vaši klienti znát, pokud nebudete mít implementován nějaký mechanismus, který změní záznamy DNS serveru poté, co je vám adresa přidělena. Takové systémy existují, nebudeme se jimi ale podrobněji zabývat, soustředíme se na příjemnější řešení, kdy používáte stejnou IP adresu pokaždé, když se k serveru připojíte[69].

V předchozím příkladu nám démon **pppd** vytočil server **c3po** a navázal s ním spojení. Nebyla provedena žádná opatření, která by některému z obou konců přidělila konkrétní IP adresu. Místo toho jsme využili standardního chování démona **pppd**, kdy se pokusí z názvu lokálního systému (v našem případě **vlager**) zjistit lokální IP adresu a tu použije na svém konci linky, adresu systému **c3po** si nechá sdělit od něj. Kromě tohoto řešení podporuje protokol PPP ještě další alternativy.

Chcete-li používat konkrétní adresy, zadáte démonu **pppd** následující parametr:

```
local_addr:remote_addr
```

Hodnoty *local_addr* a *remote_addr* mohou být zapsány buď ve formě tečkové notace, nebo jako názvy hostitelů[70]. Tento parametr způsobí, že se démon **pppd** pokusí použít první adresu jako svou vlastní a druhou adresu „vnutit“ protější straně. Pokud by vzdálená strana některou z adres odmítla, nedojde k navázání spojení[71].

Pokud se připojujete ke nějakému serveru a očekáváte, že vám adresu přidělí, měli byste zajistit, ať se váš démon **pppd** nepokouší „prosadit“ vlastní adresu. Dosáhnete toho pomocí volby `noipdefault` a parametr *local_addr* necháte prázdný. Uvedením parametr `noipdefault` zabráníte démonu v použití lokální adresy počítače pro linku PPP.

Pokud si chcete nastavit lokální adresu a je vám jedno, jakou adresu chce používat vzdálená strana, nevyplňujte údaj *remote_addr*. Pokud například chcete, aby **vlager** používal místo své adresy pro linku PPP adresu 130.83.4.27, uveďte na příkazovém řádku `130.83.4.27:`. Podobně chcete-li nastavit pouze vzdálenou adresu, neuveďte lokální adresu. Implicitně pak démon **pppd** použije adresu přidělenou vašemu hostiteli trvale.

Směrování přes linku PPP

Po nastavení síťového rozhraní démon **pppd** nastaví trasu ke vzdálenému systému pouze jako trasu k jedinému hostiteli. Představuje-li vzdálený hostitel celou lokální síť, budete se určitě chtít spojit i se systémy „za“ vaším partnerem na lince, takže musíte nastavit trasu na síť.

Už jsme si ukázali, že démona **pppd** je možné za pomoci volby `defaultroute` požádat, aby trasu přes vzdálený systém nastavil jako implicitní. Tato volba je velmi užitečná v případě, že se PPP server chová jako brána do Internetu.

69 Podrobnější informace o systémech dynamického mapování názvů na IP adresy můžete najít na *http://www.dynip.com/* a *http://www.justlinux.com/dynamic_dns.html*.

70 Použití názvů má v tomto případě dopad na autentikace protokolem CHAP. Podívejte se prosím do části *Autentikace protokolem PPP*, která následuje dále.

71 Parametry `ipcp-accept-local` a `ipcp-accept-remote` nařídí démonu **pppd** akceptovat lokální a vzdálenou adresu nabízené vzdáleným koncem i v případě, že v lokální konfiguraci nastavujeme vlastní adresy. Pokud tyto parametry nejsou uvedeny, odmítne démon **pppd** přijmout jakoukoliv adresu, kterou mu vzdálená strana nabízí.

Realizace obráceného případu, kdy se váš systém chová jako brána pro vzdálený systém, je také poměrně jednoduchá. Představme si například zaměstnance pivovaru, jehož domácí počítač se jmenuje **oneshot**. Předpokládejme dále, že jsme bránu **vlager** nastavili jako volaný PPP server. Pokud **vlager** nakonfigurujeme tak, aby zaměstnanci dynamicky přidělil IP adresu podsítě pivovaru, můžeme démonu **pppd** zadat parametr `proxyarp`, kterým pro systém **oneshot** nastavíme funkci ARP proxy. Tím zajistíme, že **oneshot** bude přímo dostupný všem počítačům ze sítí pivovaru i vinařství.

Ne vždy to ale jde tak jednoduše, jako v tomto případě. Například spojení dvou lokálních sítí typicky vyžaduje přidání konkrétního síťového směrování, protože obě sítě již mohou mít své vlastní implicitní trasy. Navíc pokud byste v obou sítích nastavili PPP linku jako implicitní trasu, vznikla by smyčka, ve které budou mezi oběma protějšky obíhat pakety určené pro neznámé lokace tak dlouho, dokud neskončí jejich životnost.

Jako příklad předpokládejme, že si pivovar otevírá pobočku v nějakém jiném městě. Tato pobočka bude mít svůj vlastní Ethernet, který bude používat síťovou IP adresu 172.16.3.0, což je podsíť 3 v síti pivovaru třídy B. Pobočka se chce spojit s hlavní sítí pivovaru protokolem PPP, aby si mohli aktualizovat databáze zákazníků. Hostitel **vlager** se opět chová jako brána, bude podporovat protokol PPP. Jeho partner na síti pobočky se jmenuje **vbourbon** a má IP adresu 172.16.3.1. Síť je znázorněna na obrázku A2 v příloze A.

Když se hostitel **vbourbon** spojí s hostitelem **vlager**, nastaví jako obvykle implicitní trasu na hostitele **vlager**. Nicméně na bráně **vlager** máme implicitně nastavenou dvoubodovou trasu pouze na hostitele **vbourbon** a musíme explicitně nastavit trasu na podsíť 3 přes bránu **vbourbon**. Můžeme to udělat ručně příkazem **route** po navázání PPP spojení, což ovšem není příliš praktické řešení. Naštěstí můžeme zajistit i automatické nastavení této trasy pomocí další funkce démona **pppd**, o níž jsme zatím nehovořili. Jde o příkaz **ip-up**. Je to skript nebo program umístěný v adresáři `/etc/ppp` a démon **pppd** jej automaticky spustí po nastavení PPP rozhraní. Pokud tuto funkci používáme, skript se spouští s následujícími parametry:

```
ip-up iface device speed local_addr remote_addr
```

Následující tabulka obsahuje význam jednotlivých parametrů. (V prvním sloupci uvádíme číslo, které příkazový interpret používá pro přístup k dané hodnotě.)

Parametr	Název	Funkce
`$1`	*iface*	Používáné síťové rozhraní, například `ppp0`.
`$2`	*device*	Cesta k použitému sériovému zařízení (pokud se používá stdin/stdout, pak `/dev/tty`).
`$3`	*speed*	Rychlost sériové linky v bitech za sekundu.
`$4`	*local_addr*	IP adresa našeho konce linky v tečkové notaci.
`$5`	*remote_addr*	IP adresa vzdáleného konce linky v tečkové notaci.

V našem případě by mohl skript **ip-up** obsahovat následující kód[72]:

```
#!/bin/sh
case $5 in
```

72 Pokud bychom chtěli po připojení vytvořit trasy i na další sítě, ošetřili bychom to na místě sekvence „...“ ve skriptu pomocí dalších příkazů `case`.

```
172.16.3.1)             # je to vbourbon
        route add -net 172.16.3.0 gw 172.16.3.1;;
...
esac
exit 0
```

Analogicky se po ukončení spojení volá skript `/etc/ppp/ip-down`, kterým můžeme zrušit nastavení provedená skriptem **ip-up**. Ve skriptu **ip-down** bychom tedy zadali příkaz **route**, kterým bychom odstranili trasu vytvořenou skriptem **ip-up**.

Nicméně směrovací schéma ještě není kompletní. Na obou hostitelích s protokolem PPP jsme nastavili příslušné položky směrovací tabulky, avšak žádný z ostatních hostitelů na obou sítích o nově vzniklém PPP spojení nic neví. Nevadí to v případě, že hostitelé na síti pivovaru budou mít nastavenu implicitní trasu na **vlager** a hostitelé na síti pobočky implicitní trasu na **vbourbon**. Pokud by to neplatilo, jedinou možností by bylo použití směrovacího démona jako je **gated**. Po vytvoření nového spojení na bráně **vlager** by démon všechny počítače na své síti o této trase vyrozuměl.

Nastavení parametrů linky

V předchozím textu jsme hovořili o protokolu LCP (Link Control Protocol), který se používá ke sjednání vlastností linky a k jejímu otestování.

Dvě nejdůležitější volby, jež mohou být sjednávány pomocí protokolu LCP, jsou *Asynchronous Control Character Map* a *Maximum Receive Unit*. Existuje i řada dalších voleb, které umožňuje protokol nastavit, jsou však příliš specializované na to, abychom se zde jimi mohli zabývat.

Nastavení Asynchronous Control Character Map, takzvaná *asynchronní mapa* nebo *async map*, se používá na asynchronních linkách, jako jsou například telefonní linky, k identifikaci řídicích znaků, které musí být nahrazeny escape sekvencí (tedy konkrétní sekvencí dvou jiných znaků). Měli byste se například vyvarovat použití znaků XON a XOFF, které jsou používány k softwarovému handshakingu, protože špatně nastavený modem by se mohl po přijetí znaku XOFF zablokovat. Dalším kandidátem jsou Ctrl-] (escape znak **telnetu**). Protokol PPP umožňuje nahrazovat libovolné znaky s ASCII kódy od 0 do 31 tím, že je specifikujete v asynchronní mapě.

Asynchronní mapa je bitová mapa o šířce 32 bitů zapsaná v šestnáctkové podobě. Nejméně významný bit odpovídá znaku s ASCII kódem 00 (NULL), nejvýznamnější bit znaku s kódem 31. Těchto 32 znaků představuje řídící znaky ASCII kódování. Pokud je bit odpovídající konkrétnímu znaku v mapě nastaven, znamená to, že znak má být před odesláním na linku nahrazován escape sekvencí.

Pokud chcete svému protějšku specifikovat, které znaky musí a nemusí nahrazovat, můžete mapu definovat parametrem `asyncmap` démona **pppd**. Pokud chcete například nahrazovat pouze znaky `^S` a `^Q` (ASCII kódy 17 a 19, používané často jako znaky XON a XOFF), použijete následující nastavení:

```
asyncmap 0x000A0000
```

Převod je velmi jednoduchý, pokud dokážete převádět dvojková čísla na šestnáctková. Představte si před sebou 32 bitů. Nejpravější odpovídá ASCII znaku 00 (NULL), nejlevější znaku ASCII 31. Všechny bity nahrazovaných znaků nastavte jako jedničkové, všechny bity nenahrazovaných znaků jako nulové. Pro převod tohoto řetězce na parametr pro program **pppd** prostě vezměte vždy čtveřici bitů a převeďte ji na šestnáctkovou hodnotu. Měli byste dostat osm šestnáctkových číslic. Spojte je dohromady, přidejte před ně „0x“ abyste naznačili, že jde o šestnáctkovou hodnotu, a máte to hotovo.

Implicitně je asynchronní mapa nastavena na hodnotu *0xffffffff*, to znamená, že se budou nahrazovat všechny řídicí znaky. Je to bezpečná implicitní volba, obvykle je to ale více, než potřebujete. Každý znak uvedený v asynchronní mapě se převádí na dva znaky přenášené po lince, takže mapování jde na úkor zvýšeného zatížení linky a odpovídajícího poklesu výkonu[73].

Ve většině případů vyhovuje asynchronní mapa `0x0` – neprovádí se žádné mapování.

Hodnota Maximum Receive Unit (MRU) říká protější straně, jakou maximální velikost HDLC rámců chceme přijímat. Trochu to připomíná hodnotu Maximum Transfer Unit (MTU), nicméně tato dvě čísla nemají téměř nic společného. Hodnota MTU je parametrem síťového zařízení jádra a definuje maximální velikost rámce, který umí příslušné rozhraní přenést. Hodnota MRU představuje více méně pouze doporučení pro vzdálenou stranu, aby nevytvářela rámce delší než tato hodnota, nicméně rozhraní stejně musí být schopno zpracovat rámce až do délky 1500 bajtů.

Volba hodnoty MRU proto není ani tak otázkou toho, co je daná linka schopna přenést, jako spíše otázkou nastavení, se kterým dosáhnete nejvyššího výkonu. Pokud hodláte po lince provozovat interaktivní aplikace, pak je dobré nastavit hodnotu MRU na nízkou hodnotu 296 bajtů, aby nějaký větší paket (například paket z relace FTP) nezpůsobil „zaseknutí" interaktivního přenosu. Chcete-li démonu **pppd** sdělit, že hodláte používat MRU o velikosti 296 bajtů, předáte mu parametr `mru 296`. Malé hodnoty MRU mají ovšem smysl pouze v případě, že používáte VJ kompresi hlaviček (která je standardně zapnuta). V opačném případě byste totiž značnou část přenosového pásma vyplýtvali tím, že byste přenášeli celé hlavičky IP datagramů.

Démon **pppd** rozumí i dalším volbám protokolu LCP, které nastavují celkové chování procesu sjednávání parametrů, jako je například maximální počet konfiguračních požadavků, které si mohou obě strany vyměnit, než se spojení ukončí. Dokud si nebudete absolutně jisti tím, co děláte, neměli byste tyto parametry měnit.

Kromě toho existují ještě dvě volby týkající se echa protokolu LCP. Protokol PPP definuje dvě zprávy, *Echo Request* a *Echo Response*. Tyto zprávy používá démon **pppd** ke kontrole, zda je linka stále aktivní. Funkci můžete povolit za parametrem `lcp-echo-interval`, za kterým následuje časový údaj v sekundách. Pokud nebudou v tomto intervalu ze vzdáleného hostitele přijaty žádné rámce, vygeneruje démon Echo Request a bude očekávat, že mu protějšek vrátí zprávu Echo Response. Jestliže se tak nestane, pak se spojení po určitém počtu odeslaných výzev ukončí. Tento počet je možné nastavit pomocí volby `lcp-echo-failure`. Implicitně je celá tato funkce vypnuta.

Obecné bezpečnostní úvahy

Špatně nakonfigurovaný démon protokolu PPP může mít z hlediska bezpečnosti zhoubný vliv na celý systém. V nejhorším případě to vypadá tak, že má každý uživatel možnost připojení do vaší sítě Ethernet (a to je velice špatné). V této stati si povíme o několika málo opatřeních, která by měla zabezpečit vaši konfiguraci protokolu PPP.

Ke konfiguraci síťových zařízení a směrovací tabulky jsou zapotřebí práva superuživatele. Typicky se tento problém řeší tak, že se **pppd** spouští jako setuid root. Démon **pppd** však umožňuje nastavit řadu vlastností významných z hlediska bezpečnosti.

73 Pozn. překladatele: Není to úplně zanedbatelné snížení – při rovnoměrném rozložení přenášených znaků může být přenosová rychlost snížena v nejhorším případě o více než 11 %.

Abyste se chránili před útoky provedenými změnou parametrů démona **pppd**, měli byste v souboru `/etc/ppp/options` specifikovat nejdůležitější nastavení, například ta, uvedená v dřívější části *Konfigurační soubory*. Některá z nich, například nastavení autentikace, uživatel nemůže potlačit, a proto poskytují rozumnou ochranu před zneužitím. Důležitým nastavením, které je nutné chránit, je parametr `connect`. Pokud máte v plánu povolit spuštění démona **pppd** pro připojení k Internetu i jiným uživatelům než je superuživatel, vždy byste měli v souboru `/etc/ppp/options` specifikovat volby `connect` a `noauth`. Pokud to neuděláte, uživatelé budou moci prostřednictvím démona **pppd** spustit v režimu superuživatele jakékoliv příkazy tím, že volbu `connect` uvedou na příkazovém řádku nebo ve svém osobním konfiguračním souboru.

Další rozumná věc je omezit použití démona **pppd** pouze na ty uživatele, kteří toto právo opravdu potřebují tak, že v souboru `/etc/group` založíte novou skupinu a do ní začleníte pouze takto privilegované uživatele. Skupinového vlastníka souboru **pppd** byste pak měli změnit na tuto skupinu a měli byste souboru odstranit právo všeobecného spuštění. Předpokládejme, že jste vytvořili skupinu *dialout*, pak stačí provést následující příkazy:

```
# chown root /usr/sbin/pppd
# chgrp dialout /usr/sbin/pppd
# chmod 4750 /usr/sbin/pppd
```

Samozřejmě se musíte chránit i vůči systémům, k nimž přistupujete. Abyste se chránili před hostiteli, kteří se vydávají za někoho jiného, měli byste při komunikaci s protějškem vždy používat nějaký druh ověření totožnosti. Kromě toho byste měli cizím hostitelům zakázat používání IP adres dle vlastního výběru a omezit jejich výběr jen na několik málo adres. V následující stati se budeme zabývat právě těmito tématy.

Autentikace protokolem PPP

Mechanismus PPP umožňuje, aby si kterákoliv z komunikujících stran vyžádala ověření totožnosti svého protějšku jedním ze dvou mechanismů – protokoly *Password Authentication Protocol* (PAP) nebo *Challenge Handshake Authentication Protocol* (CHAP). Po navázání spojení může kterýkoliv účastník požádat druhého o autentikaci, ať už jde o volaného nebo volajícího. V následujícím popisu budeme v případě potřeby používat obecné termíny „klient“ a „server“ k rozlišení systému, který si autentikaci vyžádal, a systému, který se autentikuje. Démon protokolu PPP si může autentikaci vyžádat tak, že odešle speciální požadavek protokolem LCP, ve kterém uvede požadovanou metodu autentikace.

Protokol PAP versus CHAP

Protokol PAP, používaný většinou internetových poskytovatelů, pracuje v podstatě stejně jako klasická přihlašovací procedura. Klient ověří svoji totožnost tak, že serveru pošle své uživatelské jméno a (volitelně zašifrované) heslo. Server tyto údaje porovná se svou databází „tajemství“[74]. Tato technika je však zranitelná odposlechem, kdy útočník sleduje data přenášená sériovou linkou, ale lze ji také obejít metodou pokus-omyl.

Protokol CHAP takové nedostatky nemá. Server pošle klientovi náhodně vygenerovaný řetězec s „výzvou“ a spolu s ním i svůj název hostitele. Klient na základě názvu hostitele vyhledá příslušnou tajnou informaci, zkombinuje ji s přijatou výzvou a zašifruje tento řetězec pomocí jednosměr-

74 Slovíčko „tajemství“ (secret) je termín, kterým protokol PPP označuje hesla. Pro „tajemství“ protokolu PPP neplatí stejná délková omezení jako pro přihlašovací hesla Linuxu.

né šifrovací funkce. Výsledek pak společně s názvem hostitele klienta pošle zpět serveru. Server pak provede stejné výpočty a dojde-li k témuž výsledku, povolí klientovi přístup.

Další vlastností protokolu CHAP je, že nevyžaduje po klientovi ověření jeho totožnosti jen při spuštění, ale posílá výzvy v pravidelných intervalech, aby se ujistil, že klienta nenahradil nějaký vetřelec, například přepnutím telefonních linek nebo tím, že chybně nakonfigurovaný modem neinformoval démona o ukončení původního volání a umožnil přijetí dalšího volání.

Démon **pppd** udržuje tajné klíče protokolů PAP a CHAP ve dvou samostatných souborech, pojmenovaných `/etc/ppp/pap-secrets` a `/etc/ppp/chap-secrets`. Uvedením vzdáleného hostitele v jednom z těchto souborů máte možnost určit, který protokol bude použit pro jeho i vaši autentikaci.

Démon **pppd** implicitně nevyžaduje po svém protějšku ověření totožnosti, ale je-li o to vzdálenou stranou požádán, svou totožnost prokáže. Protože je protokol CHAP mnohem silnější než protokol PAP, snaží se ho démon **pppd** používat, kdykoliv jen je to možné. Pokud ho protějšek nepodporuje nebo pokud démon **pppd** nemůže pro vzdálený systém nalézt v souboru `chap-secrets` tajný klíč, použije protokol PAP. Nenajde-li klíč potřebný pro autentikaci ani v souboru `pap-secrets`, odmítne se autentikovat a v důsledku toho se spojení ukončí.

Toto chování lze upravit několika způsoby. Použijete-li klíčové slovo `auth`, bude démon **pppd** požadovat po svém protějšku autentikaci. Bude souhlasit s použitím protokolu CHAP i PAP samozřejmě za předpokladu, že v příslušné databázi má uloženy potřebné informace. Existují i další možnosti jak zapnout nebo vypnout konkrétní autentifikační protokol, ty však nebudeme popisovat.

Pokud budou všechny systémy, se kterými máte spojení pomocí protokolu PPP, souhlasit s ověřením své totožnosti, měli byste vložit volbu `auth` do globálního souboru `/etc/ppp/options` a pro každý systém definovat v souboru `chap-secrets` příslušná hesla. Pokud některý systém nepodporuje protokol CHAP, vložte záznam o tomto systému do souboru `pap-secrets`. Tím zajistíte, že se k vám nebude moci připojit žádný cizí systém.

V dalším textu popíšeme soubory s tajnými klíči protokolu PPP, tedy soubory `pap-secrets` a `chap-secrets`. Nacházejí se v adresáři `/etc/ppp` a obsahují trojice klient, server a heslo, případně i seznam IP adres. Interpretace polí klienta a serveru je u protokolů CHAP a PAP odlišná a závisí také na tom, zda dokazujeme svoji totožnost našemu protějšku, nebo zda požadujeme, aby on prokázal svou totožnost nám.

Soubor hesel protokolu CHAP

Když musíte nějakému serveru dokázat svoji totožnost pomocí protokolu CHAP, vyhledá démon **pppd** v souboru `chap-secrets` položku, kde pole klienta odpovídá názvu lokálního systému a pole serveru názvu vzdáleného systému, který autentikaci požaduje. Když požadujete po protějšku, aby dokázal svoji totožnost, budou role obrácené: démon **pppd** vyhledá položku, u které se pole klienta shoduje s názvem vzdáleného hostitele (zasílá se v odpovědi protokolu CHAP) a pole serveru je shodné s názvem místního hostitele.

Následuje příklad souboru `chap-secrets` pro bránu **vlager**[75]:

75 Uvozovky nejsou součástí tajné informace, slouží pouze k zachování mezer, které v ní mohou být použity.

```
# Autentikační informace protokolu CHAP pro bránu vlager
#
# klient          server           heslo                  adresa
#-------------------------------------------------------------------------
vlager.vbrew.com  c3po.lucas.com   "Use The Source Luke" vlager.vbrew.com
c3po.lucas.com    vlager.vbrew.com "arttoo! arttoo!"     c3po.lucas.com
*                 vlager.vbrew.com "TuXdrinksVicBitter"  pub.vbrew.com
```

Když se brána **vlager** protokolem PPP spojí se systémem **c3po**, vyžadá si hostitel **c3po** autentikaci tím, že pošle výzvu protokolu CHAP. **pppd** na **vlageru** pak v souboru `chap-secrets` najde údaj, kde je klient **vlager.vbrew.com** a server **c3po.lucas.com**, takže najde první řádek, který vidíme v předchozím příkladu. Pak vytvoří odpověď z výzvy a hesla (`Use The Source Luke`) a pošle ji hostiteli **c3po**.

Démon **pppd** však také vytvoří autentikační výzvu pro **c3po**, která bude obsahovat jedinečný náhodně generovaný řetězec a plně kvalifikované jméno hostitele **vlager.vbrew.com**. Hostitel **c3po** vytvoří analogickým postupem odpověď a odešle ji zpět bráně **vlager**. Nyní démon **pppd** vyjme z odpovědi název klienta (**c3po.vbrew.com**) a vyhledá v souboru `chap-secrets` řádek, v němž je klientem **c3po** a serverem **vlager**. Této podmínce vyhovuje druhý řádek, takže démon **pppd** zkombinuje svou výzvu s tajnou informací `arttoo! arttoo!` a výsledek porovná s odpovědí, která přišla od hostitele **c3po**.

Volitelné čtvrté pole obsahuje seznam IP adres, které jsou přijatelné pro klienty uvedené v prvním poli. Adresy mohou být zadány v tečkové notaci nebo jako názvy hostitelů, které vyhledá resolver. Pokud by hostitel **c3po** během sjednávání spojení protokolem IPCP požadoval použití IP adresy, která není uvedena v tomto seznamu, bude požadavek zamítnut a spojení bude ukončeno. Hostitel **c3po** je tak omezen pouze na použití své vlastní IP adresy. Je-li pole s adresami prázdné, budou povoleny libovolné adresy; pokud je zde znak „-“, nemůže daný klient použít žádnou IP adresu.

Třetí řádek ve vzorovém souboru `chap-secrets` povoluje navázat spojení libovolnému hostiteli, protože hodnota * v poli klienta nebo serveru odpovídá libovolnému názvu hostitele. Jediným požadavkem je, že klient musí znát příslušnou tajnou informaci a musí použít adresu **pub.vbrew.com**. Položky se zástupnými znaky představující názvy hostitelů se mohou v souboru s tajnými informacemi objevit na kterémkoliv místě, protože démon **pppd** bude vždy používat ten řádek, který aktuální dvojici klient/server odpovídá nejlépe.

Za určitých okolností může démon **pppd** potřebovat pomoc při vytváření jmen. Jak jsme si již říkali, název vzdáleného hostitele dodá vždy protějšek ve výzvě nebo v odpovědi protokolu CHAP. Název lokálního hostitele bude implicitně zjištěn voláním funkce `gethostname(2)`. Pokud jste nastavili název systému jako nekvalifikovaný název hostitele, pak musíte démonu **pppd** poskytnout název domény pomocí volby `domain`:

```
# pppd ... domain vbrew.com
```

Tato volba připojí k názvu **vlager** i název domény pivovaru u všech aktivit, které se vztahují k ověřování totožnosti. Další volbami, které mění představu démona **pppd** o názvu místního hostitele, jsou volby `usehostname` a `name`. Když na příkazovou řádku zadáte místní IP adresu pomocí dvojice *`local:remote`* a parametr *`local`* bude obsahovat název a nikoliv IP adresu, bude použit tento název.

Soubor hesel protokolu PAP

Soubor s tajnými informacemi protokolu PAP je velmi podobný souboru, který využívá protokol CHAP. První dvě pole vždy obsahují jméno uživatele a název serveru; třetí pole obsahuje tajnou informaci protokolu PAP. Když vzdálený počítač prokazuje svou totožnost, použije démon **pppd** ten záznam, kde server odpovídá místnímu hostiteli a uživatel odpovídá uživatelskému jménu poslanému vzdáleným systémem. Pokud prokazujeme svou totožnost my, použije démon **pppd** ten řádek, kde název serveru odpovídá názvu vzdáleného systému a odešle na tomto řádku uvedené uživatelské jméno a heslo.

Příklad souboru hesel protokolu PAP může vypadat takto:

```
# /etc/ppp/pap-secrets
#
# uživatel    server     heslo         adresa
vlager-pap    c3po       cresspahl     vlager.vbrew.com
c3po          vlager     DonaldGNUth   c3po.lucas.com
```

První řádek používáme při své autentikace hostiteli **c3po**. Druhý řádek udává, jak se hostitel **c3po** autentikuje nám.

Název **vlager-pap** v prvním sloupci představuje jméno uživatele, které pošleme hostiteli **c3po**. Démon **pppd** implicitně odesílá jako uživatelské jméno název místního systému, parametrem `user` s uvedením jiného jména to však můžeme změnit.

Při výběru položky ze souboru `pap-secrets` musí démon **pppd** znát název vzdáleného hostitele. Protože neexistuje žádný způsob, jak by ho mohl zjistit, musíte mu ho předat na příkazovém řádku pomocí volby `remotename`, za níž následuje název hostitele vašeho protějšku. Abychom se pomocí výše uvedeného souboru mohli autentikovat hostiteli **c3po**, musíme na příkazovém řádku démona **pppd** doplnit následující volbu:

```
# pppd ... remotename c3po user vlager-pap
```

Ve čtvrtém poli (a ve všech následujících polích) můžete zadat, jaké adresy může daný hostitel používat, je to stejné jako při použití protokolu CHAP. Protějšek pak může používat pouze adresy z tohoto seznamu. Ve vzorovém příkladu požadujeme, aby hostitel **c3po** používal svou skutečnou IP adresu a žádnou jinou.

Protokol PAP představuje poměrně slabou metodu na ověření totožnosti, a proto se doporučuje, kdykoliv je to možné, používat místo něj protokol CHAP. Z toho důvodu zde nebudeme protokol PAP popisovat podrobněji. Další informace o jeho použití najdete na manuálové stránce

`pppd(8)`.

Ladění nastavení protokolu PPP

Démon **pppd** bude implicitně zapisovat veškeré varování a chybové zprávy prostřednictvím démona **syslog**. Do souboru `syslog.conf` musíte přidat údaj, který tyto zprávy přesměruje do souboru nebo přímo na konzolu, jinak by je démon **syslog** jednoduše zrušil. Následující položka způsobí, že budou všechny zprávy posílány do souboru `/var/log/ppp-log`:

```
daemon.*                /var/log/ppp-log
```

Jestliže nastavení démona PPP nefunguje správně, nahlédněte do tohoto souboru. Pokud vám záznamy v tomto souboru nenapoví, můžete parametrem `debug` zapnout výpis podrobnějších ladi-

cích informací. Tato volba nařídí démonu **pppd**, aby prostřednictvím démona **syslog** zaznamenával obsah všech přijatých a odeslaných řídících paketů.

Konečně nejdrastičtějším způsobem je povolení ladicích informací na úrovni jádra operačního systému. To je možné provést spuštěním démona **pppd** s volbou `kdebug`. Za touto volbou následuje číselný údaj, který je vytvořen součtem následujících hodnot: 1 pro obecné ladicí informace, 2 pro vytištění obsahu všech příchozích rámců protokolu HDLC a 4 pro výpis všech odchozích rámců protokolu HDLC. Abyste mohli zachytávat ladicí zprávy jádra operačního systému, musíte buď spustit démona **syslogd**, který čte soubor `/proc/kmsg`, nebo démona **klogd**. Oba démoni směrují ladicí informace jádra démonu **syslog**.

Složitější konfigurace protokolu PPP

Nejběžnější aplikací protokolu PPP je jeho využití k připojení telefonní linkou k nějaké síti, jako je například Internet. Ne všem však toto základní použití stačí. V této části budeme hovořit o některých složitějších konfiguracích protokolu PPP v Linuxu.

PPP server

Aby se démon **pppd** choval jako server, stačí nastavit sériové tty zařízení tak, aby po přijetí příchozích dat správně volalo démona **pppd**. Jednou z možností jak to udělat je vytvořit speciální účet, řekněme **ppp**, a jako přihlašovací skript mu přidělit skript nebo program, který spustí démona **pppd** se správnými parametry. Alternativou, chcete-li použít protokoly PAP nebo CHAP, je použít program **mgetty** pro obsluhu modemu a využít jeho funkce „/AutoPPP/“.

Chcete-li server vytvořit přihlašovací metodou, stačí do souboru `/etc/passwd` přidat následující řádek[76]:

```
ppp:x:500:200:Public PPP Account:/tmp:/etc/ppp/ppplogin
```

Pokud váš systém podporuje stínová hesla, musíte dále do souboru `/etc/shadow` přidat:

```
ppp:!:10913:0:99999:7:::
```

Samozřejmě použité hodnoty UID a GID závisí na tom, kdo chcete aby spojení vlastnil a jak je vytváříte. Pomocí příkazu **passwd** dále musíte pro tento účet vytvořit heslo.

Skript **ppplogin** by mohl vypadat takto:

```
#!/bin/sh
# ppplogin - skript spouštějící pppd po přihlášení
mesg n
stty -echo
exec pppd -detach silent modem crtscts
```

Příkaz **mesg** zabrání ostatním uživatelům na tento terminál zapisovat například příkazem **write**. Příkaz **stty** vypíná echo. Tento příkaz je nezbytný, jinak by totiž bylo vzdálenému systému zpět odesláno vše, co poslal on. Nejdůležitějším parametrem démona **pppd** je `-detach`, protože tím démonu zabráníme odpojit se od řízeného terminálu. Pokud bychom jej neuvedli, démon by se přepnul na pozadí a skript by se ukončil. To by vedlo k zavěšení linky a ukončení spojení. Parametr `silent` způsobí, že démon nejprve počká, co mu pošle vzdálený systém, a pak na to bude

76 Tuto operaci vám usnadní programy **useradd** nebo **adduser**, pokud je ovšem máte.

reagovat. Tato volba zabrání ukončení spojení kvůli vypršení časového limitu v případě, že spuštění klientského démona protokolu PPP trvá dlouho. Parametr `modem` zajistí, že démon bude řídit stavové linky modemu na sériovém portu. Tento parametr byste měli uvést vždy, když démon **pppd** komunikuje pomocí modemu. Parametr `crtscts` zapíná hardwarový handshaking.

Kromě této možnosti budete možná chtít použít i nějakou autentikační metodu, typicky uvedením parametru `auth` na příkazovém řádku démona **pppd** nebo v globálním konfiguračním souboru. Na manuálových stránkách naleznete rovněž popis toho, jak zapínat a vypínat konkrétní autentikační protokoly.

Pokud budete chtít použít program **mgetty**, stačí vám nastavit jej tak, aby podporoval to sériové zařízení, na němž máte připojen modem (viz část *Konfigurace démona mgetty* v kapitole 4). Dále v konfiguračním souboru nastavíte démona **pppd**, aby používal protokoly PAP a/nebo CHAP a konečně do souboru `/etc/mgetty/login.config` přidáte něco takového, jako:

```
# Konfigurace mgetty tak aby detekoval příchozí PPP volání
# a spouštěl démona pppd pro obsluhu spojení
/AutoPPP/ -      ppp   /usr/sbin/pppd auth -chap +pap login
```

První údaj představuje malé kouzlo umožňující detekovat, že příchozí volání je vedeno protokolem PPP. Velikost písmen v tomto řetězci nesmíte měnit, velká a malá písmena se rozlišují. Třetí sloupec udává uživatelské jméno, které se bude objevovat v příkazu **who**, když se někdo přihlásí. Zbytek řádku představuje příkaz pro spuštění démona. V našem případě nařizujeme autentikaci protokolem PAP, zakazujeme protokol CHAP a říkáme, že pro autentikaci uživatelů má být použit systémový soubor `passwd`. To je zřejmě nastavení, které bude ve většině případů vyhovovat. Jednotlivé parametry můžete uvést buď na příkazovém řádku, nebo v konfiguračním souboru démona **pppd**.

Následuje malý přehled operací, které byste měli provést, pokud chcete k vašemu počítači povolit vytáčený přístup protokolem PPP. Vždy si ověřte, že daný krok funguje a pak teprve přejděte k dalšímu:

1. Nakonfigurujte modem do režimu automatické odpovědi. U modemů kompatibilních se standardem Hayes to provedete příkazem jako `ATS0=3`. Pokud budete používat démona **mgetty**, není tento krok nutný.
2. Nastavte sériové zařízení nějakým **getty** příkazem tak, aby odpovídalo na příchozí volání. Typicky se používá démon **mgetty**.
3. Zvažte použitou autentikaci. Budete volajícího identifikovat protokolem PAP, CHAP, nebo systémovým přihlášením?
4. Výše popsaným postupem nastavte démona **pppd** jako server.
5. Rozmyslete si směrování. Musíte volajícímu poskytovat trasu k síti? Směrování můžete nastavit skriptem `ip-up`.

Vyžádané vytáčení

Jakmile se objeví nějaký IP provoz, který je zapotřebí přenést po telefonní lince, funkce *vyžádaného vytáčení* způsobí, že modem zavolá vzdálenému hostiteli a naváže s ním spojení. Vyžádané vytáčení je užitečné zejména v případech, kdy si nemůžete dovolit zůstat telefonní linkou trvale připojeni k poskytovateli. Například pokud platíte místní hovory časovým tarifem, může být levnější připojovat se pouze tehdy, když to potřebujete a odpojit se, jakmile Internet nepotřebujete[77].

77 Pozn. překladatele: Což u nás platí vždy.... V komunikačně nemonopolních Spojených státech existují telefonní společnosti, které například místní hovory neúčtují vůbec, nebo je zpoplatňují jednorázovým poplatkem. Škoda mluvit....

Tradiční řešení používalo příkaz **diald**, který sice fungoval dobře, jeho nastavení však bylo velmi komplikované. Verze 2.3.0 a vyšší démona **pppd** mají vestavěnou podporu vyžádaného vytáčení, která se konfiguruje velmi snadno. Aby tato funkce fungovala, musíte rovněž používat moderní jádro. Bude vyhovovat kterékoliv jádro od 2.0 výše.

Nastavení démona **pppd** pro podporu vyžádaného vytáčení obnáší pouze uvedení příslušných parametrů v jeho konfiguračním souboru nebo na příkazovém řádku. Následující tabulka shrnuje parametry, které se k vyžádanému vytáčení vztahují.

Parametr	Popis
`demand`	Tento parametr říká, že PPP linka má být použita v režimu vyžádaného vytáčení. Bude vytvořeno síťové zařízení, příkaz `connect` se však provede až když místní hostitel vygeneruje nějaký datagram. Tato volba je k nastavení vyžádaného vytáčení nezbytná.
`active-filter` *výraz*	Tato volba definuje, které pakety mají být chápány jako aktivní provoz. Jakýkoliv paket odpovídající zadanému výrazu vynuluje počitadlo neaktivity a způsobí tak, že démon **pppd** bude znovu čekat s ukončením spojení. Syntaxe nastavení filtru je odvozena od příkazu **tcpdump**. Implicitnímu nastavení vyhovují všechny datagramy.
`holdoff` *n*	Tento parametr nastavuje minimální čas v sekundách pro opětovné obnovení spojení od chvíle jeho neplánovaného ukončení. Když spojení vypadne a démon **pppd** je stále považuje za aktivní, bude obnoveno až po takto nastaveném čase. Toto nastavení nemá vliv na opětovné navázání spojení po jeho ukončení z důvodu neaktivity.
`idle` *n*	Je-li tento parametr nastaven, démon **pppd** ukončí spojení po uplynutí nastaveného počtu sekund neaktivity. Počitadlo se nuluje vždy když se přenese aktivní paket.

Jednoduchá konfigurace pro vyžádané vytáčení by tedy mohla vypadat takto:

```
demand
holdoff 60
idle 180
```

Toto nastavení povolí vyžádané vytáčení, po výpadku spojení je bude obnovovat za 60 sekund a v případě neaktivity ukončí spojení po 180 sekundách.

Trvalé vytáčení

Trvalé vytáčení je funkce, kterou budou používat ti, kteří jsou telefonní linkou připojeni k síti trvale. Mezi vyžádaným vytáčením a trvalým vytáčením je drobný rozdíl. Při použití trvalého vytáčení je spojení navázáno ihned po spuštění démona **pppd** a „trvalost“ se projeví v okamžiku, kdy se telefonní spojení neočekávaně přeruší. Tato funkce zajistí, že linka bude trvale funkční, protože v případě výpadku spojení bude spojení automaticky navázáno znovu.

Možná patříte mezi šťastlivce, kteří za telefonní připojení neplatí – možná voláte pouze místní hovory a váš operátor je poskytuje zdarma, nebo vám telefon platí zaměstnavatel. V těchto situacích je trvalé vytáčení velmi užitečné. Pokud za hovory platíte, musíte být trochu opatrnější. Platíte-li za hovory časovým tarifem, zřejmě vám trvalé vytáčení vyhovovat nebude, ledaže byste opravdu potřebovali být připojeni 24 hodin denně. Pokud platíte za hovory jednorázovou částkou a ne časovým tarifem, je třeba dávat pozor na to, aby modem trvale vytáčení neopakoval. Démon **pppd** nabízí parametry, které mohou pomoci řešení tohoto problému.

Pro aktivaci trvalého vytáčení musíte v některém konfiguračním souboru uvést parametr `persist`. Pouhým uvedením tohoto parametru zajistíte, že démon **pppd** bude automaticky navazo-

vat spojení příkazem `connect` znovu vždy, když linka vypadne. Pokud by vám vadilo okamžité opakované vytáčení (například pokud vzdálený server nebo modem potřebují nějaký čas na zotavení z výpadku), můžete použít parametr `holdoff` a specifikovat tak dobu, po kterou démon počká, než se pokusí o nové navázání spojení. Tento parametr sice nedokáže úplně vyřešit problém plateb za neustálé telefonování v případě závažnější chyby, může však jeho dopad poněkud redukovat.

Typická konfigurace trvalého vytváčení bude vypadat takto:

```
persist
holdoff 300
```

Čas pozdržení se udává v sekundách. V našem případě tak démon po výpadku spojení počká pět minut a pak se pokusí spojení obnovit.

Je dokonce možné kombinovat trvalé vytáčení s vyžádaným vytáčením a parametrem `idle` definovat dobu na ukončení spojení v případě neaktivity, nicméně tato kombinace není úplně typická. Na manuálových stránkách démona **pppd** je stručně popsána i tato konfigurace.

KAPITOLA 9

TCP/IP Firewall

Bezpečnost je pro společnosti i jednotlivce stále důležitější. Internet poskytuje mocné nástroje k šíření informací o sobě a k získávání informací o jiných, zároveň však své uživatele vystavuje nebezpečím, kterých by jinak byli ušetřeni. Mezi možná nebezpečí patří počítačová kriminalita, krádeže informací a poškozování dat.

Neautorizovaná a bezzásadová osoba, která získá přístup k nějakému systému, může uhodnout systémové heslo nebo využít chyb nebo nestandardního chování některých programů k tomu, aby získala normální uživatelský účet. Jakmile má možnost se k počítači přihlásit, může mít přístup k informacím, které lze zneužít – například obchodně citlivé informace jako marketingové plány, podrobnosti o nových projektech nebo databáze zákazníků. Poškození nebo modifikace takovýchto údajů může firmě způsobit značné škody.

Nejspolehlivější metoda jak těmto nehodám zabránit spočívá v tom, že neautorizovaným osobám nebude přístup k počítači umožněn. Zde vstupují do hry firewally.

> **UPOZORNĚNÍ:** Nastavit bezpečný firewall je umění. Předpokládá to dobré pochopení technologie, zároveň to ale vyžaduje stejně dobré pochopení filozofie, na níž firewally stojí. V tomto textu nebudeme hovořit o všem, co potřebujete, rozhodně vám doporučujeme podniknout vlastní pečlivý průzkum předtím, než se rozhodnete pro nějaké řešení firewallu – včetně řešení, která uvádíme zde.

Informací potřebných pro návrh a nastavení dobrého firewallu je tolik, že by vydaly na samostatnou knihu. Existuje proto logicky celá řada knih, ve kterých si můžete své vědomosti o firewallech rozšířit. Dvě z nich jsou:

Building Internet Firewalls D. Chapmana a E. Zwickyho (O'Reilly). Tato kniha popisuje jak navrhnout a nainstalovat firewall na systémech Unix, Linux a Windows NT a jak nakonfigurovat internetové služby, aby přes firewall fungovaly.

Firewalls and Internet Security W. Cheswicka a S. Bellovina (Addison Wesley). Tato kniha popisuje filozofie návrhu a implementace firewallů.

V této kapitole se zaměříme na technické podrobnosti specifické pro Linux. Později si uvedeme příklad konfigurace firewallu, která vám může posloužit jako základ pro vaši vlastní konfiguraci, nicméně – jako vždy, když je ve hře bezpečnost – nevěřte nikomu! Dvakrát zkontrolujte návrh, ujistěte se, že všemu rozumíte, a pak jej upravte podle svých potřeb. Abyste byli v bezpečí, musíte mít jistotu.

Metody útoků

Pro síťového administrátora je nutné aby chápal podstatu možných útoků na počítačovou bezpečnost. Stručně si popíšeme jednotlivé typy útoků, abyste mohli přesně pochopit, před čím vás může firewall na Linuxu chránit. Kromě toho byste si měli nastudovat i další materiály, abyste se mohli chránit i před dalšími typy útoků. Dále shrnujeme některé nejvýznamnější metody útoků a metody, jak se proti nim chránit.

Neautorizovaný přístup

Znamená to jednoduše tolik, že osoby, které by neměly být schopny používat váš systém se k němu mohou připojit a mohou jej použít. Například osoby mimo vaši firmu se mohou pokoušet o připojení k vašemu účetnímu počítači nebo internímu NFS serveru.

Existuje řada metod jak se proti těmto útokům bránit, nutné je však přesně definovat, kdo má k čemu mít přístup. Není pak problém zakázat síťový přístup všem kromě těch, kteří jej mít mají.

Využití známých slabin programů

Některé programy a síťové služby nebyly původně navrženy s důrazem na bezpečnost a jsou tak zákonitě zranitelné. Příkladem jsou vzdálené služby BSD (rlogin, rexec a další).

Nejlepší metodou ochrany proti těmto útokům je vypnutí všech zranitelných služeb nebo jejich náhrada jinými alternativami. Při použití Open Source programů je někdy možné některé slabiny odstranit úpravou zdrojového kódu.

Zablokování služby

Útok zablokování služby způsobí, že program nebo služba nebude fungovat nebo jej uživatelé nebudou moci použít. Tento typ útoku je možné provést na úrovni síťové vrstvy odesláním vhodně upravených vadných datagramů, které způsobí výpadek síťového spojení. Útok je rovněž možné provést na úrovni aplikační vrstvy, kdy pomocí vhodně zvolených příkazů dojde k přetížení služby nebo k jejímu selhání.

Útoky zablokováním služby je možné minimalizovat odfiltrováním podezřelého síťového provozu a podezřelých aplikačních příkazů a požadavků. Je dobré znát podrobnosti o používaných metodách a sledovat zprávy, které o nových typech útoků informují.

Spoofing

Tento typ útoků znamená, že počítač nebo aplikace se vydávají za někoho jiného. Typicky se útočník vydává za „přátelský“ systém tím, že falšuje IP adresy v datagramech. Například známá chyba v BSD službě rlogin umožňuje touto metodou předstírat připojení z jiného systému pomocí uhodnutí sekvenčních čísel v TCP paketech.

Jako ochranu proti těmto útokům ověřujte autenticitu datagramů a příkazů. Neprovádějte směrování datagramů s neplatnými zdrojovými adresami. Do mechanismů řízení spojení zaveďte nepředvídatelné prvky, využijte například náhodné volby sekvenčních čísel a přiřazovaných portů.

Odposlouchávání

Nejjednodušší typ útoku. Útočící systém je nastaven tak, aby přijímal data, která mu nepatří. Dobře napsaný odposlouchávací program může ze síťového provozu získat přihlašovací jména a hesla uživatelů. Na tento typ útoků jsou zejména náchylné vysílací sítě jako je Ethernet.

Jako obranu proti těmto útokům se vyhněte použití vysílaných technologií a přenášejte citlivá data v zašifrované podobě.

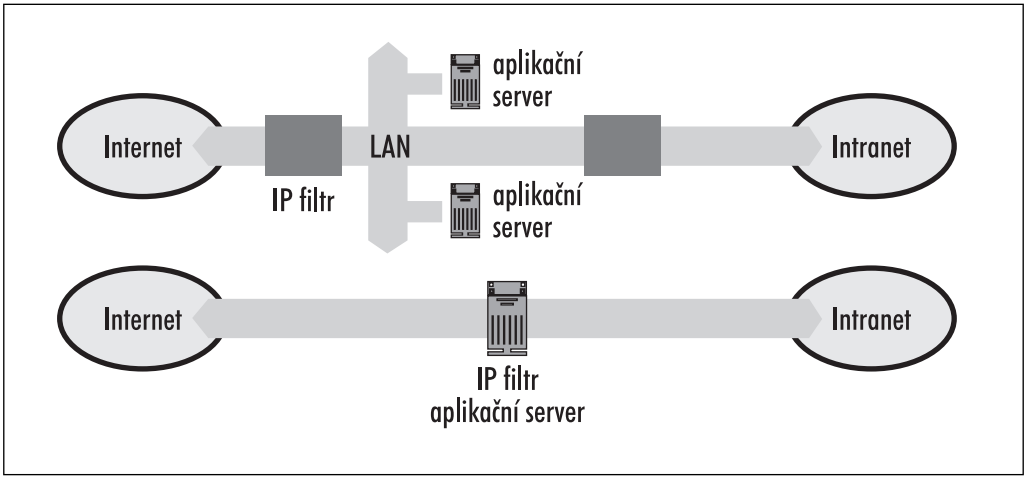
IP firewally jsou účinné jako ochrana před neautorizovaným přístupem, zablokováním služby na úrovni síťové vrstvy a spoofingem. Nejsou příliš účinné jako ochrana před využitím slabých míst v programech a proti odposlechu.

Co je to firewall?

Firewall je bezpečný a důvěryhodný počítač zapojený mezi privátní a veřejnou sítí[78]. Firewallový systém má nastavena pravidla udávající, jaký síťový provoz může propouštět a jaký má být zablokován nebo odmítnut. V některých velkých organizacích se firewally používají i uvnitř sítě jako ochrana citlivých oddělení organizace od ostatních zaměstnanců. Většina případů počítačové kriminality totiž pochází *zevnitř* organizace a nikoliv *zvenčí*.

Firewally je možné konstruovat různými metodami. Nejpokročilejší metoda používá několik samostatných systémů a rozděluje síť na různě zabezpečené úrovně. Dva počítače fungující jako filtry umožňují průchod pouze přesně definovaným typům provozu a mezi nimi jsou umístěny síťové servery jako například poštovní brána, WWW server a podobně. Takováto konfigurace může být velice bezpečná a umožňuje snadno nastavit kdo se může připojit zvenčí dovnitř a zevnitř ven. Tento typ ochrany se obvykle používá ve velkých společnostech.

Typičtěji je firewall tvořen jediným počítačem, který zajišťuje vše. Jedná se o poněkud méně bezpečné řešení, protože pokud bude v samotném firewallu chyba, která umožní neautorizovaný přístup k němu, může být narušena celá bezpečnost sítě. Nicméně tento typ firewallu je levnější a snáze spravovatelný než složitější výše popsané uspořádání. Obrázek 9.1 znázorňuje oba typy konfigurace.



Obrázek 9.1 – Hlavní metody návrhu firewallu

Jádro Linuxu obsahuje řadu vestavěných funkcí, které mu umožňují pracovat jako účinný IP firewall. Implementace síťových služeb obsahuje funkce, které umožňují řadou různých způsobů konfigurovat filtrování paketů, a zajišťují mechanismy, které umožní přesně nastavit, jaká pravidla chcete použít. Firewall založený na Linuxu je natolik flexibilní, že může být použit v obou konfi-

78 Termín *firewall* pochází z oblasti protipožární ochrany. Jedná se o bariéru z nehořlavého materiálu, umístěnou mezi potenciálním zdrojem požáru a jeho okolím.

guracích podle obrázku 9.1. Síťový software Linuxu obsahuje i další funkce, které se k firewallům vztahují – IP účtování (viz kapitola 10) a IP maškaráda (viz kapitola 11).

Co je to filtrování?

Filtrování je jednoduše mechanismus, který rozhoduje o tom, které IP datagramy mají být zpracovány normálně a které mají být zrušeny. *Zrušením* rozumíme, že datagram bude smazán a úplně ignorován, jako kdyby vůbec nebyl přijat. K rozhodování o tom, které datagramy zpracovávat a které zahazovat je možné použít celou řadu kritérií, například:

- Typ protokolu: TCP, UDP, ICMP a podobně
- Číslo portu (pro TCP/UDP)
- Typ datagramu: SYN/ACK, data, ICMP Echo Request a podobně
- Zdrojovou adresu datagramu – odkud pochází
- Cílovou adresu datagramu – kam má dojít

Důležité je chápat, že filtrování se odehrává na úrovni síťové vrstvy. Filtrování neví nic o aplikacích, které síťové spojení používají, stará se pouze o samotné spojení. Můžete například uživatelům zakázat přístup do vaší interní sítě na standardním portu telnetu, pokud ovšem používáte pouze filtrování, nemůžete jim zabránit použít telnet pro připojení k jinému portu, ke kterému toto připojení povolujete. Tomuto typu problémů můžete předejít pomocí proxy serverů pro všechny služby, které firewallem povolujete. Proxy server rozumí aplikaci, kterou obsluhuje a může proto zabránit zneužitím, například použít telnet pro průchod přes firewall prostřednitvím WWW portu. Pokud bude váš firewall podporovat funkci WWW proxy, veškerá připojení na port WWW bude obsluhovat proxy server a povolí průchod pouze legitimním HTTP požadavkům. Existuje celá řada proxy serverů. Některé z nich jsou dostupné zdarma, k dispozici je i řada komerčních produktů. Některé oblíbené popisuje dokument Firewall-HOWTO, my se jimi v tomto textu nebudeme zabývat.

Filtrační pravidla jsou sestavena z různých kombinací výše uvedených kritérií. Můžete například chtít, aby uživatelé na síti pivovaru neměli na Internetu přístup k ničemu jinému než ke službě WWW. Nakonfigurujete tedy firewall tak, aby povoloval průchod:

- datagramům se zdrojovou adresou sítě pivovaru, libovolnou cílovou adresou a cílovým portem 80 (služba WWW),
- datagramům s libovolnou zdrojovou adresou, zdrojovým portem 80 (služba WWW) a cílovou adresou kdekoliv v síti pivovaru

Všimněte si, že používáme dvě pravidla. Potřebujeme totiž, aby naše data prošla ven, ale zároveň aby data zvnějšku mohla projít dovnitř. Jak za chvíli uvidíme, Linux konstrukci takovýchto pravidel usnadňuje a umožňuje obě dvě zadat jediným příkazem.

Vytvoření firewallu na Linuxu

K vytvoření firewallu na Linuxu potřebujete sestavit jádro s podporou firewallu a dále potřebujete odpovídající konfigurační nástroj. V jádrech před verzí 2.2 je tímto nástrojem **ipfwadm**. V jádrech 2.2.*x* byla uvedena třetí generace IP firewallů, nazvaná *IP Chains*. Administrují se nástrojem **ipchains**, který je podobný nástroji **ipfwadm**. Jádra 2.3.15 a pozdější podporují čtvrtou generaci firewallů, pojmenovanou *netfilter*. Systém *netfilter* je systém mnoha tváří, poskytuje zpětnou kom-

patibilitu s nástroji **ipfwadm** i **ipchains** a nabízí i nový nástroj **iptables**. O rozdílech mezi těmito třemi systémy budeme hovořit za chvíli.

Konfigurace jádra pro firewall

Aby mohl Linux fungovat jako firewall, musí tuto funkci podporovat jádro. Nastavení této podpory je velmi jednoduché, po spuštění `make menuconfig` stačí zadat potřebné volby[79]. Jak se jádro konfiguruje jsme popisovali v kapitole 3. V jádrech 2.2 musíte zadat následující volby:

```
Networking options  --->
       [*] Network firewalls
       [*] TCP/IP networking
       [*] IP: firewalling
       [*] IP: firewall packet logging
```

V jádrech 2.4.0 a novějších volíte namísto toho následující nastavení:

```
Networking options  --->
     [*] Network packet filtering (replaces ipchains)
         IP: Netfilter Configuration  --->
              .
             <M> Userspace queueing via NETLINK (EXPERIMENTAL)
             <M> IP tables support (required for filtering/masq/NAT)
             <M>   limit match support
             <M>   MAC address match support
             <M>   netfilter MARK match support
             <M>   Multiple port match support
             <M>   TOS match support
             <M>   Connection state match support
             <M>   Unclean match support (EXPERIMENTAL)
             <M>   Owner match support (EXPERIMENTAL)
             <M>   Packet filtering
             <M>     REJECT target support
             <M>     MIRROR target support (EXPERIMENTAL)
              .
             <M>   Packet mangling
             <M>     TOS target support
             <M>     MARK target support
             <M>   LOG target support
             <M> ipchains (2.2-style) support
             <M> ipfwadm (2.0-style) support
```

Nástroj ipfwadm

Nástroj **ipfwadm** (IP Firewall Administration) slouží k vytváření pravidel firewallu v jádrech před verzí 2.2.0. Syntaxe příkazů je poněkud matoucí, protože příkazy dokáží provádět velmi komplikované operace, ukážeme si nicméně některé typické příklady ilustrující nejobvyklejší způsoby použití tohoto nástroje.

Nástroj **ipfwadm** bývá součástí většiny moderních distribucí Linuxu, i když se obvykle neinstaluje implicitně. Zřejmě budete muset nainstalovat nějaký balík, který tento nástroj obsahuje. Pokud

79 Funkce *firewall packet logging* je speciální funkce, která na zvolené zařízení zapisuje informace o datagramech, které vyhovují pravidlům firewallu, takže můžete sledovat provoz, který firewallem prochází.

ve vaší distribuci tento nástroj není, zdrojový balík můžete získat ze serveru *ftp.xos.nl* v `/pub/linux/ipfwadm` a můžete si program přeložit sami.

Nástroj ipchains

Stejně jako **ipfwadm**, i nástroj **ipchains** může při prvním setkání působit zmateně. Je stejně pružný jako nástroj **ipfwadm**, navíc má zjednodušenou syntaxi příkazů a nabízí mechanismus „zřetězení", který umožňuje spravovat více sad pravidel a spojovat je dohromady. Mechanismus zřetězování pravidel popíšeme v samostatné části ke konci kapitoly, protože pro většinu situací není tato funkce nutná.

Příkaz **ipchains** se objevuje ve většině distribucí Linuxu založených na jádře 2.2 a novějších. Pokud si jej chcete přeložit sami, najdete zdrojový kód na adrese *http://www.rustcorp.com/linux/ipchains/*. Součástí zdrojového balíku je i „maskovací" skript **ipfwadm-wrapper**, který se tváří jako program **ipfwadm**, provádí však převod parametrů a volá nástroj **ipchains**. Díky tomu je migrace ze starší verze firewallu poměrně pohodlná.

Nástroj iptables

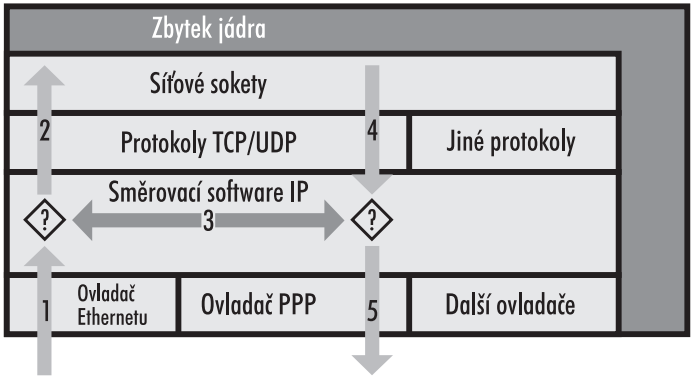
Syntaxe nástroje **iptables** se podobá syntaxi nástroje **ipchains**. Rozdíly vyplývají z různých vylepšení a z nového návrhu nástroje, který umožňuje jeho použití prostřednictvím sdílených knihoven. Stejně jako u nástroje **ipchains**, i pro nástroj **iptables** uvedeme stejné příklady, takže si budete moci porovnat syntaxi jednotlivých příkazů.

Nástroj **iptables** je součástí zdrojového balíku *netfilter* na adrese *http://www.samba.org/netfilter/*. Pravděpodobně se bude nacházet i v distribucích Linuxu založených na jádře 2.4.

O systému *netfilter* budeme hovořit v samostatné části úplně na konci této kapitoly.

Tři způsoby realizace filtrace

Uvědomme si, jakým způsobem linuxový počítač, respektive jakýkoliv počítač schopný směrovat IP pakety, zpracovává docházející datagramy. Základní kroky, znázorněné na obrázku 9.2, jsou:



Obrázek 9.2 – Fáze zpracování IP datagramu

- Přijetí IP datagramu (1).
- Datagram se zkontroluje, zda je určen pro proces na tomto počítači.
- Pokud datagram patří tomuto počítači, zpracuje se lokálně (2).
- Pokud datagram není určen tomuto počítači, prohlédne se směrovací tabulka a hledá se vhodná trasa. Datagram se pak předá správnému rozhraní nebo se zahodí, pokud pro něj není známá trasa (3).
- Datagramy od lokálních procesů se posílají směrovacímu systému, který je předá správnému rozhraní (4).
- Odcházející datagram se zkontroluje, zda je pro něj k dispozici trasa a pokud ne, zahodí se.
- Odeslání IP datagramu (5).

V našem diagramu reprezentuje trasa 1-3-5 směrování datagramu od počítače z lokální Ethernetové sítě na počítač dosažitelný linkou PPP. Trasy 1-2 a 4-5 představují příjem a odeslání datagramů síťovým procesem na našem počítači. Trasa 4-3-2 představuje tok dat lokálním zpětnovazebním rozhraním. Data typicky přicházejí a odcházejí různými síťovými rozhraními. Otazníky ve schématu označují místa, kdy musí IP vrstva provést rozhodnutí o směrování.

Firewall v jádře Linuxu dokáže filtrovat na různých místech tohoto procesu. Znamená to, že můžete filtrovat datagramy přicházející na váš počítač, datagramy, které váš počítač směruje, i datagramy, které jsou odesílány.

V nástrojích **ipfwadm** a **ipchains** odpovídají pravidla třídy Input trase 1 v diagramu, třída Forwarding odpovídá trase 3 a třída Output trase 5. Až budeme později hovořit o programu *netfilter*, uvidíme, že v něm se místa zpracování změnila a třída Input odpovídá trase 2 a třída Output trase 4. Má to zásadní dopad na způsob sestavování pravidel, nicméně základní principy zůstávají ve všech verzích firewallů stejné.

Celé uspořádání může na první pohled vypadat komplikovaně, zajišťuje však flexibilitu, která umožňuje vytvářet velmi složité a mocné konfigurace.

Původní IP firewall (jádra 2.0)

První generace podpory firewallu v Linuxu se objevila v jádrech série 1.1. Jednalo se o přenos BSD firewallu ipfw na platformu Linux, který provedl Alan Cox. Podpora v jádrech 2.0 pak byla druhá generace podpory a na jejím vylepšení se podíleli Jos Vos, Pauline Middelink a další.

Použití programu ipfwadm

Příkaz **ipfwadm** sloužil jako konfigurační nástroj druhé generace linuxových firewallů. Asi nejlepší způsob, jak si tento nástroj popsat, bude uvedení příkladu. Začneme tím, že zkusíme nastavit příklad, o kterém jsme se už zmínili.

Základní příklad

Řekněme, že máme nějakou organizaci se sítí a pro přístup k Internetu používáme firewall založený na Linuxu. Dále předpokládejme, že chceme našim uživatelům povolit přístup k webovým serverům na Internetu, nechceme však propouštět jakýkoliv jiný provoz.

Zavedeme tedy pravidla, která povolí předávání datagramů se zdrojovou adresou pocházející z naší sítě na cílový port 80, a dále předávání odpovídajících datagramů s odpovědí.

Předpokládejme, že naše síť používá 24bitovou síťovou masku (síť třída C) a adresu 172.16.1.0. Můžeme pak použít následující pravidla:

```
# ipfwadm -F -f
# ipfwadm -F -p deny
# ipfwadm -F -a accept -P tcp -S 172.16.1.0/24 -D 0/0 80
# ipfwadm -F -a accept -P tcp -S 0/0 80 -D 172.16.1.0/24
```

Parametr `-F` říká programu **ipfwadm**, že se jedná o pravidlo třídy Forwarding. První příkaz říká programu **ipfwadm**, aby zrušil všechna pravidla této třídy. Tím si zajistíme, že začneme přidávat naše pravidla za přesně definovaného stavu firewallu.

Druhé pravidlo definuje implicitní předávací politiku. Říkáme jádru, že nepovolujeme předávání IP datagramů. Nastavení implicitní politiky je velice důležité, protože říká jádru co má dělat, pokud přijme datagram, který není popsán žádným z pravidel. U většiny konfigurací firewallu nastavujeme implicitní politiku na zákaz předávání, čímž zajistíme, že firewall nepředá nic, co jsme mu explicitně nepřikázali předávat.

Třetí a čtvrtý řádek představují implementaci našich pravidel. Třetí pravidlo umožňuje definovaným datagramům opouštět naši síť, čtvrté pravidlo umožňuje příjem definovaných datagramů.

Podívejme se na jednotlivé parametry:

-F	Jedná se o předávací pravidlo.
-a accept	Pravidlo přidáváme s politikou „accept“, tedy povolujeme předávání datagramů, které pravidlu vyhovují.
-P tcp	Pravidlo platí pro datagramy protokolu TCP (nikoliv pro protokoly UDP a ICMP).
-S 172.16.1.0/24	Prvních 24 bitů zdrojové adresy musí odpovídat adrese 172.16.1.0.
-D 0/0 80	Nula bitů cílové adresy musí odpovídat adrese 0.0.0.0. Což je jenom „stručnější“ výraz, jak říct „cokoliv“. Číslo 80 představuje cílový port, tedy službu WWW. Místo čísla portu můžete použít i název služby definovaný v souboru `/etc/services`, takže parametr `-D 0/0 www` bude znamenat to samé.

Program **ipfwadm** pracuje se síťovou maskou ve tvaru, který není úplně obvyklý. Zápis */nn* znamená, kolik bitů adresy je významných, tedy velikost masky. Významné bity se počítají zleva doprava, některé typické příklady uvádí tabulka 9.1.

Tabulka 9.1 – Typické hodnoty síťové masky

Maska	Bitů
255.0.0.0	8
255.255.0.0	16
255.255.255.0	24
255.255.255.128	25
255.255.255.192	26
255.255.255.224	27
255.255.255.240	28
255.255.255.248	29
255.255.255.252	30

Už jsme se zmínili o tom, že program **ipfwadm** nabízí malý trik, který usnadňuje zadávání tohoto typu pravidel. Tímto trikem je parametr `-b`, který znamená obousměrné pravidlo.

Příznak obousměrného pravidla nám dovoluje sloučit dvě pravidla do jediného takto:

```
# ipfwadm -F -a accept -P tcp -S 172.16.1.0/24 -D 0/0 80 -b
```

Důležité vylepšení

Podívejme se na naše pravidla pečlivěji. Všimli jste si, že jsme stále ponechali jednu možnost, jak může někdo zvenku náš firewall překonat?

Naše pravidla povolují propustit zvenčí všechny datagramy ze zdrojového portu 80. Sem ovšem spadají i datagramy s nastaveným bitem SYN! Bit SYN definuje, že TCP datagram je žádost o spojení. Pokud má útočník privilegovaný přístup k nějakému počítači, může se přes náš firewall připojit k jakémukoliv počítači, stačí jenom, aby odesílal své datagramy z portu 80. To jsme ovšem rozhodně nechtěli.

Naštěstí je náš problém řešitelný. Příkaz **ipfwadm** obsahuje další příznak, který umožňuje vytvářet pravidla vztahující se na datagramy s nastaveným bitem SYN. Změníme proto náš příklad a nastavíme pravidla takto:

```
# ipfwadm -F -a deny -P tcp -S 0/0 80 -D 172.16.10.0/24 -y
# ipfwadm -F -a accept -P tcp -S 172.16.1.0/24 -D 0/0 80 -b
```

Parametr `-y` říká, že pravidlo platí pro datagramy s nastaveným příznakem SYN. Naše nové pravidlo tedy říká „zakaž všechny datagramy určené pro naši síť pocházející odkudkoliv z portu 80 s nastaveným příznakem SYN“ – jinak řečeno „zakaž všechny žádosti o otevření spojení přicházející z portu 80“.

Proč jsme toto nové pravidlo umístili *před* naše původní pravidlo? IP firewall funguje tak, že použije první vyhovující pravidlo. Datagramy, které chceme zakázat, vyhovují *oběma* pravidlům, proto musí být zakazující pravidlo uvedeno *před* pravidlem povolujícím.

Výpis pravidel

Po zadání pravidel můžeme následujícím příkazem požádat o jejich vypsání:

```
# ipfwadm -F -l
```

Toto pravidlo vypíše všechna nastavená předávací pravidla. Výstup bude vypadat takto nějak:

```
# ipfwadm -F -l
IP firewall forward rules, default policy: deny
type  prot source              destination         ports
deny  tcp  anywhere            172.16.10.0/24      www -> any
acc   tcp  172.16.1.0/24       anywhere            any -> www
```

Příkaz **ipfwadm** se pokusí přeložit čísla portů na názvy služeb pomocí souboru `/etc/services`.

Na výpisu ovšem nevidíme některé důležité podrobnosti. Nevidíme zde například efekt parametru `-y`. Příkaz **ipfwadm** dokáže generovat i podrobnější výstup po zadání parametru `-e`. Neukážeme si jej celý, protože je příliš dlouhý, než aby se vešel na stránku, nicméně tento výpis obsahuje i sloupec `opt` (options), který uvádí nastavení příznaku `-y` pro potlačení paketů SYN:

```
# ipfwadm -F -l -e
IP firewall forward rules, default policy: deny
 pkts bytes type  prot opt  tosa tosx ifname  ifaddress   source          ...
    0     0 deny  tcp  --y- 0xFF 0x00 any     any         anywhere        ...
    0     0 acc   tcp  b--- 0xFF 0x00 any     any         172.16.1.0/24   ...
```

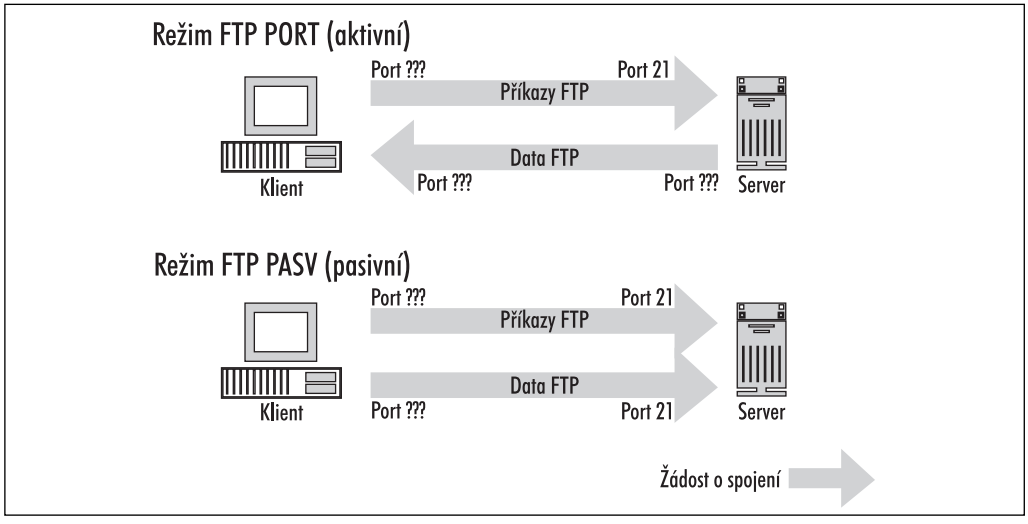
Složitější příklad

Předchozí příklad byl velmi jednoduchý. Ne všechny síťové služby se konfigurují tak snadno jako služba WWW, ve skutečnosti bývá konfigurace typického firewallu podstatně složitější. Podívejme se nyní na další typický příklad, na službu FTP. Chceme, aby se uživatelé naší interní sítě mohli přihlásit k FTP serverům na Internetu a číst a zapisovat soubory. Nechceme ale, aby se někdo z Internetu mohl přihlásit k našim interním FTP serverům.

Víme, že služba FTP používá dva TCP porty: port 20 (ftp-data) a port 21 (ftp), takže:

```
# ipfwadm -a deny -P tcp -S 0/0 20 -D 172.16.1.0/24 -y
# ipfwadm -a accept -P tcp -S 172.16.1.0/24 -D 0/0 20 -b
#
# ipfwadm -a deny -P tcp -S 0/0 21 -D 172.16.1.0/24 -y
# ipfwadm -a accept -P tcp -S 172.16.1.0/24 -D 0/0 21 -b
```

V pořádku? Ne nutně! Servery FTP mohou pracovat ve dvou různých režimech: pasivním a aktivním[80]. V pasivním režimu FTP server čeká na připojení od klienta. V aktivním režimu sám navazuje spojení s klientem. Standardně se obvykle používá aktivní režim. Rozdíl mezi oběma režimy ilustruje obrázek 9.3.



Obrázek 9.3 – Režimy práce FTP serveru

80 Aktivní režim se (poněkud neintuitivně) zapíná příkazem **PORT**. Pasivní režim se zapíná příkazem **PASV**.

Většina FTP serverů při práci v aktivním režimu navazuje spojení z portu 20, což nám poněkud usnadňuje situaci, bohužel se tak nechovají všechny[81].

Co to pro nás znamená? Podívejme se na pravidlo týkající se portu 20, tedy datového portu FTP. Tak jak máme toto pravidlo definováno předpokládá, že náš klient bude navazovat spojení se serverem. To bude platit v pasivním režimu. Bude ovšem velmi obtížné nakonfigurovat firewall pro podporu aktivního režimu, protože nemůžeme spolehlivě vědět, které porty budou použity. Pokud bychom povolili příchozí spojení na všech portech, vystavili bychom tak útokům všechny služby, které spojení přijímají.

Problém nejbezpečněji vyřešíme tak, že povolíme použití služby FTP pouze v pasivním režimu. Většina FTP serverů a řada FTP klientů tento režim podporují. I oblíbený klient **ncftp** podporuje pasivní režim, nicméně bude nutné jej pro tento režim nakonfigurovat. I řada webových prohlížečů, například Netscape, podporují pasivní režim, takže by neměl být problém najít software, který bude této podmínce vyhovovat. Celý problém můžete také sprovodit ze světa tím, že na firewall nainstalujete FTP proxy server, který bude přijímat spojení z interní sítě a bude navazovat spojení na Internet.

Při nastavování firewallu obvykle narazíte na celou řadu podobných problémů. Vždy musíte dávat velký pozor na to, jak služba přesně pracuje, abyste mohli zavést všechna potřebná pravidla. Úplná konfigurace firewallu může být velice složitá.

Přehled parametrů programu ipfwadm

Příkaz **ipfwadm** pracuje s celou řadou parametrů, které se vztahují ke konfiguraci IP firewallu. Obecná syntaxe je:

```
ipfwadm kategorie příkaz parametr [volby]
```

Podívejme se podrobněji na jednotlivé části.

Kategorie

Musí být uveden právě jeden z následujících parametrů. Tímto parametrem firewallu říkáme, který typ pravidel modifikujeme.

`-I` Vstupní pravidlo (třída Input)

`-O` Výstupní pravidlo (třída Output)

`-F` Předávací pravidlo (třída Forward)

Příkaz

Musí být uveden alespoň jeden z následujících parametrů. Bude se vztahovat pouze k pravidlům zvolené kategorie. Příkazy firewallu říkají, co má udělat.

`-a [politika]` Přidá nové pravidlo na konec seznamu pravidel.

`-i [politika]` Vloží pravidlo do seznamu pravidel.

`-d [politika]` Vymaže pravidlo ze seznamu pravidel.

`-p politika` Nastaví implicitní politiku.

`-l` Vypíše všechna pravidla.

`-f` Smaže všechna pravidla.

81 Nechová se tak například démon ProFTPd, přinejmenším ve starších verzích.

Možná nastavení politiky jsou:

`accept`	Umožní příjem, odeslání nebo předání vyhovujícího datagramu.
`deny`	Zablokuje příjem, odeslání nebo předání vyhovujícího datagramu.
`reject`	Zablokuje příjem, odeslání nebo předání vyhovujícího datagramu a zároveň odesílajícímu hostiteli odešle ICMP zprávu o chybě.

Parametr

Musí být uveden alespoň jeden z následujících parametrů. Pomocí parametrů se definuje, pro které datagramy má pravidlo platit.

`-P` *`protokol`*	Může specifikovat protokoly TCP, UDP nebo ICMP. Například `-P tcp`
`-S` *`adresa`*`[/`*`maska`*`] [`*`port`*`]`	Zdrojová adresa, pro níž pravidlo platí. Pokud neuvedete masku, předpokládá se maska „/32“. Můžete rovněž zadat port, jehož se pravidlo týká. Aby pravidlo fungovalo, musíte parametrem `-P` definovat protokol. Pokud neuvedete port nebo rozsah portů, vyhovují pravidlu všechny porty. Porty mohou být definovány svými názvy podle souboru `/etc/services`. V případě protokolu ICMP slouží údaj o portu ke specifikaci typu ICMP datagramu. Je možné definovat i rozsahy portů zadáním *`od_portu`*`:`*`po_port`*. Například: `-S 172.29.16.1/24 ftp:ftp_data`
`-D` *`adresa`*`[/`*`maska`*`] [`*`port`*`]`	Cílová adresa, pro níž pravidlo platí. Pro zadání cílové adresy platí stejná pravidla jako pro zadání zdrojové adresy. Například: `-D 172.29.16.1/24 smtp`
`-V` *`adresa`*	Udává adresu síťového rozhraní, na němž je paket přijímán (třída `-I`) nebo odesílán (třída `-O`). Tím můžeme vytvářet pravidla vztahující se k určitým síťovým rozhraním, například `-V 172.29.16.1`
`-W` *`název`*	Udává název síťového rozhraní. Parametr se chová stejně jako `-V`, pouze namísto adresy rozhraní zadáváte jeho název. Například: `-W ppp0`

Nepovinné volby

Následující volby mohou být občas velmi užitečné:

`-b` Obousměrné pravidlo. Tento příznak zahrnuje provoz oběma směry mezi zadaným cílem a zdrojem. Ušetří nám zadávání dvou pravidel – jednoho pro příchozí a druhého pro odchozí provoz.

`-o` Zapíná záznam vyhovujících datagramů do logu jádra. Všechny datagramy vyhovující nějakému pravidlu budou zaznamenány jako zpráva jádra. Tento parametr je užitečný k detekci pokusů o neautorizovaný přístup.

`-y` Používá se k zachycení žádostí o navázání spojení. Použití této volby způsobí, že pravidlu budou vyhovovat pouze TCP datagramy s žádostí o navázání spojení. Vyhovují mu tedy datagramy s nastaveným příznakem SYN a nenastaveným příznakem ACK. Je užitečná k odfiltrování požadavků o navázání TCP spojení, pro jiné protokoly se ignoruje.

`-k` Používá se k zachycení potvrzujících datagramů. Pravidlu vyhovují pouze TCP datagramy potvrzující přijetí žádosti o navázání spojení. Vyhovují mu pouze TCP datagramy s nasta-

veným příznakem ACK. Používá se k odfiltrování požadavků o navázání TCP spojení, pro jiné protokoly se ignoruje.

Typy ICMP datagramů

Konfigurační příkazy firewallu umožňují specifikovat typy ICMP datagramů. Na rozdíl od portů TCP a UDP neexistuje konfigurační soubor, který by definoval typy ICMP datagramů a jejich popis. Typy ICMP datagramů jsou definovány v dokumentu RFC-1700 Assigned Numbers. Kromě toho jsou typy ICMP datagramů uvedeny v hlavičkových souborech standardní knihovny C. Definuje je soubor `/usr/include/netinet/ip_icmp.h`, který je součástí standardního balíku GNU knihovny a používají jej programátoři při vytváření programů pracujících s protokolem ICMP. Uvádíme je v tabulce 9.2. Rozhraní programu **iptables** umožňuje definovat typy ICMP datagramů i názvem, proto ve druhém sloupci uvádíme i tyto názvy.

Tabulka 9.2 – Typy ICMP datagramů

Typ číslo	Název v iptables	Název typu
0	echo-reply	Echo Reply
3	destination-unreachable	Destination Unreachable
4	source-quench	Source Quench
5	redirect	Redirect
8	echo-request	Echo Request
11	time-exceeded	Time Exceeded
12	parameter-problem	Parameter Problem
13	timestamp-request	Timestamp Request
14	timestamp-reply	Timestamp Reply
15	nemá	Information Request
16	nemá	Information Reply
17	address-mask-request	Address Mask Request
18	address-mask-reply	Address Mask Reply

IP Firewall Chains (jádra 2.2)

Většina vlastností v Linuxu se vyvíjí tak, aby odpovídala požadavkům uživatelů. Ani firewally nejsou výjimkou. Klasická implementace IP firewallu ve většině případů vyhovuje, nicméně při konfiguraci složitějších prostředí může být komplikovaná a neefektivní. Z toho důvodu byla uvedena nová metoda konfigurace IP firewallu a byly uvedeny nové funkce. Tato nová metoda se označuje jako „IP Firewall Chains“ a poprvé byla obecně uvolněna v jádře 2.2.0.

Systém IP Firewall Chains vytvořili Paul Russel a Michael Neuling[82]. Dokumentaci k programu IP Firewall Chains vytvořil Paul v dokumentu IPCHAINS-HOWTO.

IP Firewall Chains vám umožňuje vytvářet třídy firewallových pravidel, do kterých pak můžete přidávat nebo odebírat počítače nebo sítě. Výhodou takového řetězení pravidel je, že mohou zvýšit výkon firewallu v případě, že obsahuje velké množství pravidel.

82 Paula můžete kontaktovat na adrese *Paul.Russel@rustcorp.com.au*.

IP Firewall Chains je podporováno v jádrech 2.2 a existuje i jako patch pro jádra 2.0.*. Zmíněný dokument HOWTO popisuje kde patch získat a uvádí také řadu užitečných informací o tom, jak efektivně pracovat s konfiguračním nástrojem **ipchains**.

Použití nástroje ipchains

Existují dvě možnosti jak nástroj **ipchains** použít. První způsob používá skript **ipfwadm-wrapper**, což je v podstatě náhrada programu **ipfwadm**, která transparentně spouští program **ipchains**. Pokud jej budete používat, nemusíte číst dále. Místo toho si znovu přečtěte předcházející popis programu **ipfwadm** a všude jej nahraďte názvem **ipfwadm-wrapper**. Toto řešení je plně funkční, není ale zaručeno, že skript bude dále vyvíjen a nemůžete také využít výhody, které technologie Firewall Chains nabízí.

Druhá možnost jak program **ipchains** použít spočívá v jeho volání s tím, že se naučíte novou syntaxi a upravíte pravidla tak, aby namísto staré syntaxe používala novou. Při troše opatrnosti můžete také zjistit, že při převodu starých pravidel na nová je můžete částečně optimalizovat. Syntaxe programu **ipchains** je jednodušší než u programu **ipfwadm**, takže toto řešení doporučujeme.

Program **ipfwadm** konfiguroval firewall pomocí tří tříd pravidel. Pomocí Firewall Chains můžete vytvářet libovolný počet tříd, které budou navzájem propojeny, nicméně vždy budou přítomny tři základní třídy. Tyto standardní třídy přesně odpovídají třídám programu **ipfwadm**, jsou však pojmenovány `input`, `forward` a `output`.

Podívejme se nejprve na obecnou syntaxi příkazu **ipchains** a pak si ukážeme, jak **ipchains** použít namísto **ipfwadm** aniž bychom se zabývali pokročilejšími možnostmi tohoto programu. Budeme se držet našich původních příkladů a přepíšeme je pro novou syntaxi.

Syntaxe příkazu ipchains

Syntaxe příkazu **ipchains** je velmi jednoduchá. Podívejme se na její nejdůležitější rysy. Obecně se **ipchains** spouští takto:

```
ipchains příkaz specifikace_pravidla volby
```

Příkazy

Existuje celá řada způsobů, jak programem **ipchains** manipulovat s pravidly a s třídami pravidel. Podívejme se na ty, které se vztahují k IP firewallům.

`-A` *třída*	Přidává na konec třídy jedno nebo více pravidel. Pokud je jako zdroj nebo cíl uveden název počítače a ten má přiřazeno více IP adres, budou přidána pravidla pro všechny adresy[83].
`-I` *třída č_prav*	Přidá jedno nebo více pravidel na začátek třídy. Opět je-li zadán název a tomu odpovídá více adres, budou přidána pravidla pro všechny adresy.
`-D` *třída*	Vymaže ze třídy jedno nebo více pravidel, která odpovídají následující specifikaci.

83 Pozn. překladatele: Zatím jsme žili v představě, že jedné IP adrese může sice odpovídat více názvů (jeden kanonický a libovolný počet aliasů), nicméně že určitý název se vždy převede pouze na jednu jedinou adresu. Jak vidíme, neplatí to vždy. DNS povoluje přiřadit jednomu názvu více IP adres a toto uspořádání se používá k distribuci zátěže na více počítačů.

Například názvu *www.seznam.cz* odpovídá *** IP adres (a tento název je tedy obsluhován *** počítači). Vždy když se bavíte se serverem Seznam, budete náhodně připojeni k jednomu z počítačů, na nichž server běží. Tento mechanismus distribuce zátěže se označuje jako *DNS round-robin*.

`-D` *třída č_prav* Vymaže ze třídy pravidlo na pozici *č_prav*. Pravidla jsou číslována od jedničky.

`-R` *třída č_prav* Nahradí pravidlo na pozici *č_prav* novým pravidlem.

`-C` *třída* Porovná datagram popsaný následující specifikací s pravidly dané třídy. Příkaz vrátí zprávu o tom, jak bude datagram pravidly této třídy zpracován. Tato volba je velmi užitečná k testování konfigurace firewallu a budeme se jí později věnovat podrobněji.

`-L` [*třída*] Vypíše pravidla dané třídy, případně všech tříd, pokud není třída zadána.

`-F` [*třída*] Vymaže pravidla dané třídy, případně všech tříd, není-li třída zadána.

`-Z` [*třída*] Vynuluje počitadla datagramů a bajtů pro všechna pravidla dané třídy nebo všech tříd, není-li třída specifikována.

`-N` *třída* Vytvoří novou třídu se zadaným názvem. Třída zadaného jména nesmí existovat. Tímto příkazem se vytvářejí uživatelem definované třídy.

`-X` [*třída*] Vymaže třídu definovanou uživatelem, případně všechny uživatelem definované třídy, není-li název třídy uveden. Aby příkaz uspěl, na třídu nesmějí existovat odkazy z žádných jiných tříd.

`-P` *třída politika* Nastavuje implicitní politiku dané třídy. Platné politiky jsou `ACCEPT`, `DENY`, `REJECT`, `REDIR` a `RETURN`. `ACCEPT`, `DENY` a `REJECT` mají stejný význam jako v klasické implementaci firewallu. Pravidlo `REDIR` říká, že datagram má být transparentně přesměrován na nějaký port firewallu. Hodnota `RETURN` způsobí návrat do třídy, z níž byla volána třída s tímto pravidlem a zpracování pokračuje pravidlem následujícím za volajícím pravidlem.

Specifikace pravidel

Pomocí celé řady parametrů programu **ipchains** je možné vytvářet pravidla udávající, které typy paketů jsou pravidlem zpracovány. Pokud ve specifikaci pravidla není některý z parametrů uveden, předpokládá se jeho standardní hodnota.

`-p [!]`*protokol* Udává protokol, který pravidlu vyhovuje. Platné názvy protokolů jsou `tcp`, `udp`, `icmp` a `all`. Pokud chcete definovat jiný protokol, můžete uvést jeho číslo. Hodnotou `4` můžete například definovat zapouzdřující protokol `ipip`. Je-li uveden `!`, pravidlo je negováno a budou mu vyhovovat všechny protokoly kromě zadaného. Není-li parametr uveden, předpokládá se hodnota `all`.

`-s [!]`*adresa*`[/`*maska*`] [[!]`*port*`]`

Udává zdrojovou adresu a port datagramu, který pravidlu vyhovuje. Adresa může být zadána názvem počítače, názvem sítě nebo IP adresou. Nepovinný údaj *maska* představuje síťovou masku a může být zadán buď v tradiční podobě (například /255.255.255.0), nebo v moderní podobě (například /24). Nepovinný údaj *port* definuje TCP nebo UDP port nebo typ ICMP datagramu. Port můžete specifikovat pouze v případě, že jste parametrem `-p` definovali jeden z protokolů `tcp`, `udp` nebo `icmp`. Může být zadán i rozsah portů zadáním spodní a horní meze oddělených dvojtečkou. Například hodnota `20:25` znamená porty 20 (včetně) až 25 (včetně). Opět je možné pomocí `!` pravidlo negovat.

`-d [!]adresa[/maska] [[!]port]`

Udává cílovou adresu a port datagramu, který pravidlu vyhovuje. Parametr se používá stejně jako `-s`.

`-j cíl` Definuje akci, která se má provést, jestliže datagram pravidlu vyhovuje. Tento parametr můžete chápat jako příkaz „běž na“. Platné cílové hodnoty jsou `ACCEPT`, `DENY`, `REJECT`, `REDIR` a `RETURN`. Jejich význam jsme vysvětlili dříve. Kromě toho můžete uvést název uživatelské třídy pravidel a zpracování datagramu bude pokračovat v této třídě. Pokud parametr není uveden, nestane se s datagramem vůbec nic, a dojde pouze k inkrementaci počitadel bajtů a datagramů.

`-i [!]název_rozhraní`

Definuje rozhraní na němž je datagram přijat nebo odesílán. Symbol `!` opět neguje význam pravidla. Pokud název rozhraní končí symbolem +, vyhovují všechna rozhraní začínající zadaným řetězcem. Například `-i ppp+` definuje všechna rozhraní PPP, `-i ! eth+` definuje všechna rozhraní kromě ethernetových rozhraní.

`[!]-f` Udává, že pravidlo platí pro všechno kromě prvního fragmentu fragmentovaného datagramu.

Volby

Následující volby příkazu **ipchains** jsou obecné. Některé z nich slouží k nastavení specifických nuancí programu **ipchains**.

`-b` Příkaz vygeneruje dvě pravidla. Jedno pravidlo bude odpovídat zadaným parametrům, druhé bude odpovídat těmto parametrům v opačném pořadí.

`-v` Zapne „výřečný“ výstup programu **ipchains**. Program bude sdělovat více informací.

`-n` Způsobí, že **ipchains** bude vypisovat IP adresy a porty jako čísla a nebude se pokoušet o jejich převod na názvy.

`-l` Zapne logování datagramů. Každý datagram vyhovující pravidlu bude jádrem zaznamenán prostřednictvím funkce `printk()`, kterou obvykle obsluhuje program **sysklogd** a zapisuje do souboru. Tímto způsobem můžete zachytit netypické datagramy.

`-o[max_vel]` Datagram vyhovující danému pravidlu se zkopíruje na zařízení „síťové linky“ v uživatelském prostoru. Parametr *max_vel* definuje maximální počet bajtů, které se z každého datagramu kopírují. Tato volba je užitečná zejména pro programátory, v budoucnosti ji však možná využijí i různé programy.

`-m značka` Datagramy vyhovující pravidlu budou *označeny* hodnotou. Značka je 32bitové číslo bez znaménka. V současných implementacích pro označení není použití, v budoucích verzích však může značka rozhodovat o tom, jak má být datagram zpracován dalšími programy – například směrovacím systémem. Pokud hodnota značky začíná symboly + nebo `-`, přičte se nebo odečte ke stávající hodnotě značky.

`-t` *andmaska xormaska*

Umožňuje manipulovat s bity typu služby v hlavičce IP datagramů, které pravidlu vyhovují. Bity typu služby se používají v inteligentních směrovačích ke zvýšení priority datagramů. Směrovací software Linuxu podporuje tyto prioritní operace. Hodnoty *andmaska* a *xormaska* definují bitové masky, kterými budou bity typu služby v datagramu logicky ANDovány a XORovány. Jedná se o pokročilou funkci, která je popsána v dokumentu IPCHAINS-HOWTO.

`-x` Všechna čísla ve výstupu programu **ipchains** budou uvedena přesně, bez zaokrouhlování.

`-y` Pravidlo bude platit pro TCP datagramy s nastaveným bitem SYN a nenastavenými bity ACK a FIN. Slouží k filtrování požadavků na navázání spojení.

Zpět k základnímu příkladu

Opět předpokládejme, že máme nějakou organizaci se sítí a pro přístup k Internetu používáme firewall založený na Linuxu. Našim uživatelům chceme povolit přístup k webovým serverům na Internetu, nechceme však propouštět jakýkoliv jiný provoz.

Síť používá 24bitovou síťovou masku (síť třída C) a adresu 172.16.1.0. Můžeme pak použít následující pravidla:

```
# ipchains -F forward
# ipchains -P forward DENY
# ipchains -A forward -s 0/0 80 -d 172.16.1.0/24 -p tcp -y -j DENY
# ipchains -A forward -s 172.16.1.0/24 -d 0/0 80 -p tcp -b -j ACCEPT
```

První příkaz vymaže všechna pravidla třídy `forward`, druhý příkaz nastaví implicitní politiku této třídy na `deny`. Třetí a čtvrté pravidlo definují požadované filtrování. Čtvrté pravidlo propouští tam a zpět datagramy mezi námi a webovými servery, třetí pravidlo zakazuje navázat spojení z venkovního portu 80.

Pokud budeme chtít přidat pravidla povolující našim uživatelům použití služby FTP v pasivním režimu, bude to vypadat takto:

```
# ipchains -A forward -s 0/0 20 -d 172.16.1.0/24 -p tcp -y -j DENY
# ipchains -A forward -s 172.16.1.0/24 -d 0/0 20 -p tcp -b -j ACCEPT
# ipchains -A forward -s 0/0 21 -d 172.16.1.0/24 -p tcp -y -j DENY
# ipchains -A forward -s 172.16.1.0/24 -d 0/0 21 -p tcp -b -j ACCEPT
```

Výpis pravidel programu ipchains

K vypsání pravidel definovaných programem **ipchains** použijeme parametr `-L`. Stejně jako u programu **ipfwadm** můžeme pomocí dalších parametrů definovat podrobnost výpisu. V nejjednodušším případě vytvoří **ipchains** asi takovýto výstup:

```
# ipchains -L -n
Chain input (policy ACCEPT):
Chain forward (policy DENY):
target     prot opt     source               destination          ports
DENY       tcp  -y----  0.0.0.0/0            172.16.1.0/24        80 ->   *
ACCEPT     tcp  ------  172.16.1.0/24        0.0.0.0/0            * ->   80
ACCEPT     tcp  ------  0.0.0.0/0            172.16.1.0/24        80 ->   *
```

```
ACCEPT     tcp  ------  172.16.1.0/24       0.0.0.0/0           * ->   20
ACCEPT     tcp  ------  0.0.0.0/0           172.16.1.0/24       20 ->   *
ACCEPT     tcp  ------  172.16.1.0/24       0.0.0.0/0           * ->   21
ACCEPT     tcp  ------  0.0.0.0/0           172.16.1.0/24       21 ->   *

Chain output (policy ACCEPT):
```

Pokud neuvedeme název třídy, vypíše program pravidla ve všech třídách. Parametr `-n` programu říká, aby se nepokoušel o převod adres a čísel portů na názvy. Z výpisu by mělo být jasné, co nám program říká.

Podrobnější výpis vynucený parametrem `-v` uvádí větší množství podrobností. Ve výstupu jsou uvedeny hodnoty počitadel datagramů a paketů, masky pro typ služby, názvy rozhraní, značky a maximální velikost.

Každé pravidlo programu **ipchains** obsahuje počitadlo datagramů a počitadlo bajtů. Tímto mechanismem se implementuje IP účtování, o kterém hovoříme v kapitole 10. Implicitně se hodnoty těchto počitadel vypisují v zaokrouhleném tvaru a pomocí symbolů `K` a `M` se uvádějí tisíce a miliony[84]. Pokud uvedeme parametr `-x`, budou hodnoty počitadel vypsány v úplném tvaru bez zaokrouhlení.

Použití tříd

Teď už víme, jak pomocí programu **ipchains** nahradit program **ipfwadm** s tím, že získáváme poněkud jednodušší syntaxi a některé funkce navíc. Je načase seznámit se s tím, kdy a proč používat uživatelem definované třídy pravidel. Kromě toho si povíme něco o skriptech, které se s balíkem **ipchains** distribuují.

Uživatelem definované třídy

Tři třídy pravidel v klasickém firewallu představovaly mechanismus pro vytváření konfigurací, které byly snadno pochopitelné a udržovatelné na menších sítích s jednoduchými požadavky na firewall. Jakmile potřebujeme složitější konfiguraci, vynoří se celá řada problémů. Větší sítě typicky vyžadují použití většího množství pravidel než jsme doposud viděli, zároveň narůstají nároky na komplikovanější pravidla, která pokrývají různé specifické situace. S rostoucím počtem pravidel klesá výkon firewallu, protože každý datagram je nutné testovat proti velkému množství podmínek, navíc se komplikuje správa firewallu. Není totiž možné atomicky zapnout nebo vypnout celou skupinu pravidel a v době změny konfigurace firewallu tak síť může být vystavena útokům.

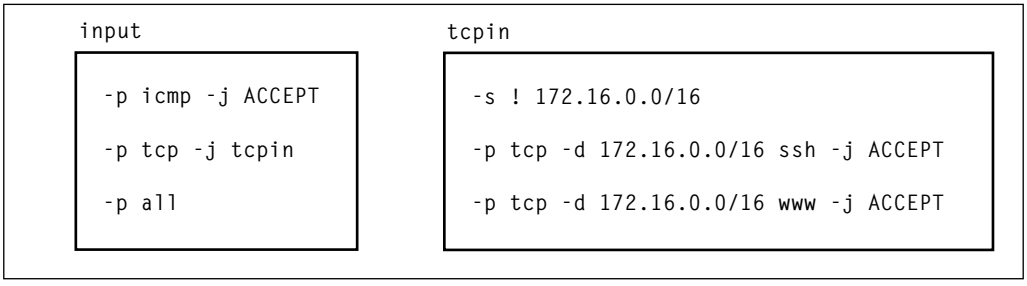
Návrh systému Firewall Chains řeší oba tyto problémy, protože umožňuje správci firewallu vytvářet libovolný počet tříd pravidel, která se pak propojují do tří vestavěných tříd. Pomocí parametru `-N` programu **ipchains** můžeme vytvořit novou třídu pravidel, jejíž název bude dlouhý maximálně osm znaků (je rozumné v názvech tříd používat pouze malá písmena). Pomocí parametru `-j` definujeme akci, která se má provést, když datagram pravidlu vyhovuje. Tímto parametrem můžeme definovat, že pokud datagram vyhovuje zadanému pravidlu, může být podroben dalšímu testování proti pravidlům uživatelské třídy. Budeme to ilustrovat na příkladu.

Podívejme se na následující příkazy **ipchains**:

84 Pozn. překladatele: `K` je samozřejmě 1024 a `M` samozřejmě 1048576.

```
ipchains -P input DENY
ipchains -N tcpin
ipchains -A tcpin -s ! 172.16.0.0/16
ipchains -A tcpin -p tcp -d 172.16.0.0/16 ssh -j ACCEPT
ipchains -A tcpin -p tcp -d 172.16.0.0/16 www -j ACCEPT
ipchains -A input -p tcp -j tcpin
ipchains -A input -p all
```

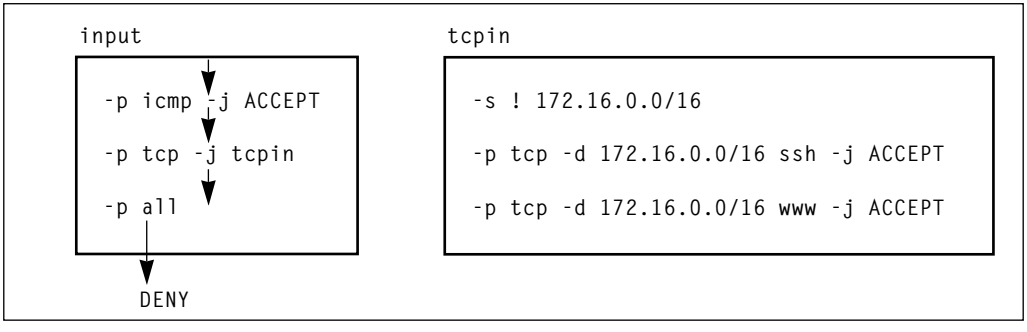
Implicitní politiku třídy `input` jsme nastavili na `deny`. Druhým příkazem vytváříme uživatelskou třídu nazvanou `tcpin`. Třetí příkaz přidává do třídy `tcpin` pravidlo, kterému vyhovuje jakýkoliv datagram, který byl zvenku předán naší síti, tomuto pravidlu není přidělena žádná akce. Toto pravidlo je zavedeno pro potřeby účtování a bude podrobněji vysvětleno v kapitole 10. Dalším dvěma pravidlům vyhovují datagramy určené naší lokální síti z portů `ssh` nebo `www`, datagramy vyhovující těmto pravidlům budou přijaty. Další pravidlo je to místo, kde začínají kouzla s třídami. Způsobí, že firewall bude všechny datagramy protokolu TCP testovat proti uživatelské třídě `tcpip`. Konečně jsme do třídy `input` přidali další pravidlo, kterému budou vyhovovat všechny datagramy. Jedná se o další účtovací pravidlo. Skupiny takto vzniklých pravidel znázorňuje obrázek 9.4.



Obrázek 9.4 – Jednoduché skupiny pravidel

Třídy `input` a `tcpin` obsahují námi vestavěná pravidla. Zpracování datagramu začíná vždy v některé z vestavěných tříd. Ukážeme si jakým způsobem třídy fungují na příkladech zpracování různých typů datagramů.

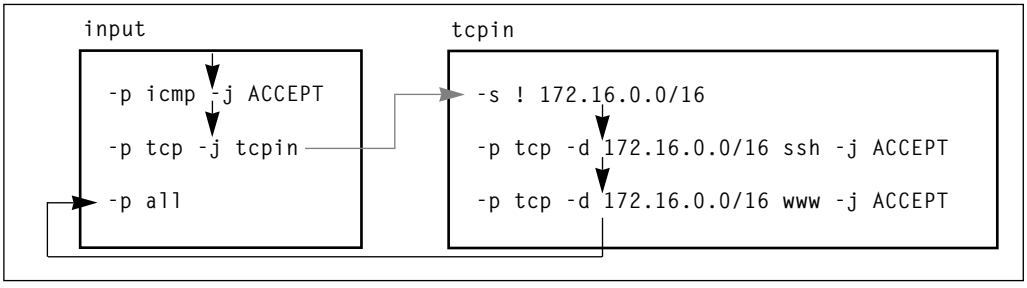
Podívejme se nejprve co se bude dít, když bude přijat UDP datagram určený pro některého našeho hostitele. Jeho zpracování jednotlivými pravidly demonstruje obrázek 9.5.



Obrázek 9.5 – Sekvence testovaných pravidel při zpracování UDP datagramu

Datagram je přijat třídou `input` a projde oběma prvními pravidly, protože ta se vztahují k protokolům ICMP a TCP. Vyhovuje třetímu pravidlu třídy, to však nespecifikuje žádnou akci, takže sice dojde k inkrementaci počitadel bajtů a datagramů, nic dalšího se ale nestane. Datagram dojde na konec třídy `input`, kde narazí na její implicitní pravidlo a je zrušen.

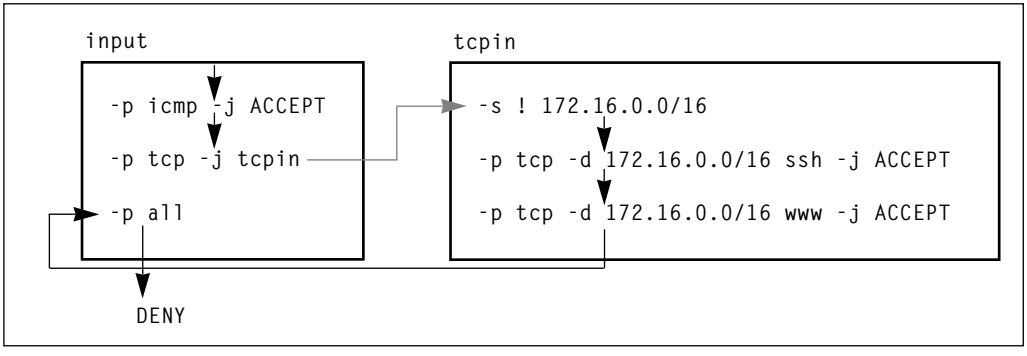
Abychom viděli chování uživatelem definované třídy, podívejme se nyní co se stane, když přijmeme TCP datagram určený pro port `ssh`. Sekvence jeho zpracování je znázorněna na obrázku 9.6.



Obrázek 9.6 – Sekvence zpracování TCP datagramu pro port `ssh`

V tomto případě datagram vyhovuje druhému pravidlu třídy `input`, které specifikuje cíl `tcpin`. Jeli jako cíl datagramu specifikována uživatelem definovaná třída, dojde k tomu, že datagram bude testován pravidly této třídy, takže zpracování pokračuje prvním pravidlem třídy `tcpin`. Prvnímu pravidlu vyhovuje jakýkoliv datagram pocházející z vnější sítě a pravidlo nespecifikuje žádnou akci. Jedná se opět o účtovací pravidlo a zpracování datagramu pokračuje druhým pravidlem ve třídě. Tomuto pravidlu datagram vyhovuje a pravidlo specifikuje akci `ACCEPT`. K dalšímu zpracování datagramu nedochází a datagram je přijat.

Nakonec se podívejme na to, co se stane, když datagram dojde na konec uživatelem definované třídy. Ukážeme si to na zpracování TCP datagramu určeného pro jiný než jeden ze dvou povolených portů. Zpracování takového datagramu demonstruje obrázek 9.7.



Obrázek 9.7 – Sekvence zpracování TCP datagramu pro port `telnet`

Uživatelem definované třídy nemají implicitní politiku. Jakmile jsou otestována všechna pravidla a datagram žádnému nevyhovuje, chová se třída tak, jako kdyby obsahovala implicitní pravidlo `RETURN`. Pokud vám takové chování nevyhovuje, musíte na konci třídy definovat pravidlo, kterému budou vyhovovat všechny datagramy a které provede vámi požadovanou implicitní akci. V našem příkladu dojde při zpracování datagramu třídou `tcpin` k pokračování zpracování v nadřaze-

né třídě pravidlem následujícím za tím, kterým byla uživatelská třída volána. Datagram nakonec dorazí na konec třídy `input`, která má definovánu implicitní politiku a datagram bude zrušen.

Tento příklad byl velmi jednoduchý, demonstroval však smysl použití tříd. Praktické použití tříd bude mnohem složitější. Poněkud komplikovanější příklad je vytvořen následující sekvencí příkazů.

```
#
# Implicitní předávací politika REJECT
ipchains -P forward REJECT
#
# vytvoření uživatelských tříd
ipchains -N sshin
ipchains -N sshout
ipchains -N wwwin
ipchains -N wwwout
#
# Odmítáme spojení špatným směrem
ipchains -A wwwin -p tcp -s 172.16.0.0/16 -y -j REJECT
ipchains -A wwwout -p tcp -d 172.16.0.0/16 -y -j REJECT
ipchains -A sshin -p tcp -s 172.16.0.0/16 -y -j REJECT
ipchains -A sshout -p tcp -d 172.16.0.0/16 -y -j REJECT
#
# Odmítáve vše na konci uživatelské třídy
ipchains -A sshin -j REJECT
ipchains -A sshout -j REJECT
ipchains -A wwwin -j REJECT
ipchains -A wwwout -j REJECT
#
# předáváme služby www a ssh odpovídajícím třídám
ipchains -A forward -p tcp -d 172.16.0.0/16 ssh -b -j sshin
ipchains -A forward -p tcp -s 172.16.0.0/16 -d 0/0 ssh -b -j sshout
ipchains -A forward -p tcp -d 172.16.0.0/16 www -b -j wwwin
ipchains -A forward -p tcp -s 172.16.0.0/16 -d 0/0 www -b -j wwwout
#
# Pravidla povolující hostitele vkládáme na 2. pozici našich tříd
ipchains -I wwwin 2 -d 172.16.1.2 -b -j ACCEPT
ipchains -I wwwout 2 -s 172.16.1.0/24 -b -j ACCEPT
ipchains -I sshin 2 -d 172.16.1.4 -b -j ACCEPT
ipchains -I sshout 2 -s 172.16.1.4 -b -j ACCEPT
ipchains -I sshout 2 -s 172.16.1.6 -b -j ACCEPT
#
```

V tomto příkladu jsme pomocí uživatelsky definovaných tříd zjednodušili správu konfigurace firewallu a zároveň zlepšili efektivitu zpracování datagramů v porovnání s využitím pouze vestavěných tříd.

V příkladu vytváříme uživatelské třídy pro oba směry toku datagramů služeb `www` a `ssh`. Do třídy `wwwout` umístíme pravidla definující počítače, které mají povoleno vytvářet odchozí připojení na webové servery, ve třídě `sshin` definujeme počítače, kterým povolujeme připojit se k nám službou **ssh**. Předpokládali jsme, že potřebujeme individuálně nastavovat, které počítače v naší síti mohou vytvářet a přijímat `www` a `ssh` spojení. Zjednodušení je zřejmé, protože uživatelem definovaná třída nám umožňuje sdružovat logicky spolu související pravidla a nemusíme mít všechna

pravidla smíchaná dohromady. Dochází i ke zlepšení výkonu firewallu, protože pro jednotlivé datagramy se v průměru zkrátil počet pravidel, proti kterým jsou testovány. Účinnost tohoto mechanismu se zvýší s rostoucím počtem pravidel. Pokud bychom nepoužili uživatelské třídy, v nejhorším případě bude přijatý datagram testován proti všem pravidlům v seznamu. Budeme-li předpokládat, že každému pravidlu vyhovuje stejný počet datagramů z balíku všech přijímaných, i pak v průměru porovnáváme každý datagram proti polovině pravidel. Pomocí uživatelských tříd se vyhneme testování proti skupinám podrobných pravidel jednoduše tím, že datagram nevyhovuje jednoduchému pravidlu v základní třídě, které by jej do podřízené třídy „poslalo".

Podpůrné skripty programu ipchains

Softwarový balík **ipchains** je dodáván se třemi podpůrnými skripty. O prvním z nich jsme už v krátkosti hovořili, další dva slouží k jednoduchému uložení a obnovení konfigurace firewallu.

Skript **ipfwadm-wrapper** emuluje řádkovou syntaxi programu **ipfwadm**, pravidla firewallu však vytváří voláním programu **ipchains**. Tento skript představuje možnost jak firewall snadno nakonfigurovat pomocí původních pravidel a mohou jej využít ti, kteří se nechtějí učit novou syntaxi programu **ipchains**. Skript **ipfwadm-wrapper** se ve dvou věcech chová odlišně od programu **ipfwadm**: Příkaz **ipchains** nepodporuje definici rozhraní jeho IP adresou, proto skript **ipfwadm-wrapper** sice akceptuje parametr `-v`, pokouší se jej ale převést na parametr `-w` programu **ipchains** tím, že hledá název rozhraní, kterému je daná adresa přiřazena. Vždy když použijete parametr `-v`, skript **ipfwadm-wrapper** vás na tuto skutečnost upozorní. Druhým rozdílem je to, že nejsou korektně převedena pravidla pro účtování fragmentů.

Skripty **ipchains-save** a **ipchains-restore** výrazně usnadňují vytváření a úpravy konfigurace firewallu. Příkaz **ipchains-save** přečte stávající konfiguraci firewallu a ve zjednodušeném tvaru ji zapíše na standardní výstup. Příkaz **ipchains-restore** pak čte data v tomto formátu a danými pravidly nastaví firewall. Výhoda použití těchto skriptů proti přímé modifikaci konfiguračního skriptu firewallu a jeho testování spočívá v tom, že konfiguraci můžete jednou dynamicky vytvořit a pak ji uložit. Kdykoliv v budoucnu ji můžete obnovit, změnit a případně znovu uložit.

Skripty se používají asi takovýmto způsobem. Příkaz

```
ipchains-save >/var/state/ipchains/firewall.state
```

uloží stávající konfiguraci firewallu. K jejímu obnovení typicky v době startu systému zadáte:

```
ipchains-restore </var/state/ipchains/firewall.state
```

Skript **ipchains-restore** testuje, zda už neexistuje některá z uživatelských tříd definovaných ve vstupních datech. Spustíte-li skript s parametrem `-f`, nejprve smaže pravidla již existující třídy a pak ji naplní pravidly podle vstupních dat. Bez uvedení tohoto parametru se vás skript zeptá, zda má konfiguraci dané třídy přeskočit, nebo zda má původní pravidla smazat a nahradit je novými.

Netfilter a IP Tables (jádra 2.4)

Při vývoji programu Firewall Chains došel Paul Russel k závěru, že firewally by měly být méně komplikované, takže se soustředil na zjednodušení způsobů, jimiž jsou datagramy ve firewallovém kódu jádra zpracovávány a navrhl nový mechanismus filtrování, který je jednak jednodušší a jednak podstatně pružnější. Tento mechanismus nazval *netfilter*.

V době vzniku tohoto textu nebyl návrh mechanismu *netfilter* ještě stabilizován. Doufáme, že omluvíte jakékoliv nesrovnalosti v popisu tohoto mechanismu a jeho konfiguračních nástrojů, ke kterým dojde v důsledku změn provedených v programu po vzniku tohoto textu. Považujeme *netfilter* za natolik významnou techniku, že jej zde chceme stručně popsat, i když jsme si vědomi, že některé z našich popisů budou spekulativní. Pokud narazíte na nějaké nesrovnalosti, přesný a aktuální popis všech vlastnostní systému *netfilter* by měl být k nalezení v příslušném dokumentu HOWTO.

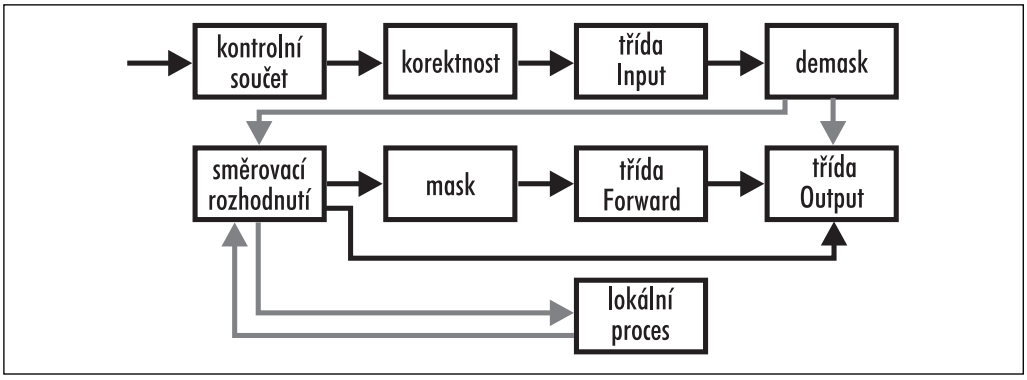
Co je špatné na technice IP Chains? Významně zjednodušila efektivitu a správu firewallových pravidel. Nicméně mechanismus zpracování datagramů je stále příliš složitý zejména ve spojení s dalšími technologiemi souvisejícími s firewally, například s technologií IP maškarády (viz kapitola 11) nebo s jinými technikami překladu adresu. Jedním z důvodů těchto komplikací je skutečnost, že IP maškaráda a technologie NAT (Network Address Translation) byly vyvinuty nezávisle na firewallu a až později byly do jeho kódu integrovány. Výhodnější řešení je tyto techniky od počátku zahrnout do celkového návrhu firewallu. Pokud chce někdo v současném uspořádání zasáhnout do mechanismu zpracování datagramů v jádře, obtížně se hledá místo, kam vlastní kód zařadit a budou nutné přímé modifikace kódu jádra.

Kromě toho existují i další problémy. Konkrétně třída „input“ popisuje vstup síťové IP vrstvy jako celku. Tato třída zahrnuje jak datagramy, které jsou určeny danému počítači, tak i datagramy, které jím mají být směrovány. Je to trošku proti zdravému rozumu, protože se tím částečně překrývají funkce třídy input a forward, která zpracovává pouze směrované datagramy, nicméně nastupuje vždy až po třídě input. Pokud chcete jiným způsobem ošetřovat datagramy určené lokálními hostiteli a jinak datagramy jím směrované, musíte vytvářet složitá pravidla, která budou zpracovávat jeden nebo druhý typ datagramů. Stejný problém se týká i třídy output.

Tyto komplikace se nevyhnutelně přenesly i na práci správce systému, protože se projevují v tom, jak je třeba vytvářet pravidla pro firewall. Navíc jakékoliv rozšíření filtrovacích funkcí vyžaduje přímé zásahy do jádra, protože v něm jsou filtrační politiky implementovány a neexistuje k nim žádné transparentní rozhraní. Systém *netfilter* řeší jak složitost, tak neobratnost předchozích řešení tím, že v jádře implementuje obecnou platformu, která zjednodušuje proces zpracování datagramů a zároveň umožňuje modifikovat filtrační politiky bez nutnosti modifikovat jádro.

Podívejme se nyní na dvě klíčové změny. Obrázek 9.8 ilustruje, jakým způsobem jsou datagramy zpracovávány mechanismem IP Chains, na obrázku 9.9 vidíme, jak je zpracovává implementace *netfilter*. Hlavní rozdíly spočívají v odstranění maškarády z hlavního kódu a ve změně umístění tříd input a output. Zároveň s těmito změnami byl vyvinut nový konfigurační nástroj, nazvaný **iptables**.

V implementaci IP Chains se třída input vztahuje na všechny datagramy přijaté počítačem, bez ohledu na to, zda jsou určeny pro něj, nebo zda je má pouze směrovat. V implementaci *netfilter* se třída input vztahuje pouze na datagramy určené lokálnímu hostiteli, třída forward pak pouze na datagramy určené jinému hostiteli. Analogicky se v implementaci IP Chains vztahuje třída output na všechny odesílané datagramy bez ohledu na to, zda je odesílá lokální systém, nebo zda je směruje jinému hostiteli. V implementaci *netfilter* se třída output vztahuje pouze na datagramy generované lokálním systémem a netýká se datagramů směrovaných jinému hostiteli. Jen samotná tato změna přináší výrazné zjednodušení řady firewallových konfigurací.

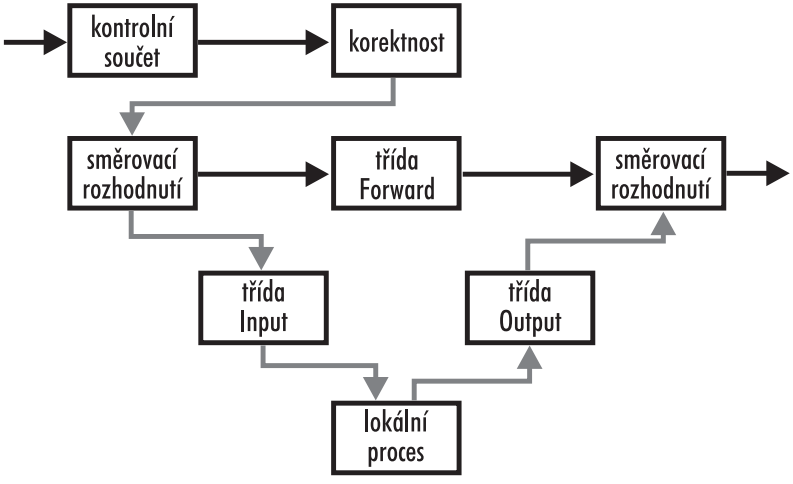


Obrázek 9.8 – Zpracování datagramů v implementaci IP Chains

Komponenty „demask“ a „mask“ na obrázku 9.8 jsou samostatné komponenty jádra, které odpovídají za zpracování příchozích a odchozích datagramů maškarády. V modulech implementace *netfilter* byly navrženy znovu.

Představme si případ, kdy jsou implicitní politiky tříd input, output i forward nastaveny na `deny`. V implementaci IP Chains pak budeme potřebovat šest pravidel, aby mohl nějaký typ datagramu projít firewallem: vždy dvě do tříd input, output a forward (jedno z pravidel bude pokrývat odchozí směr, druhé pak příchozí směr). Jistě si dokážete představit, jak se to může komplikovat, když budeme chtít rozlišovat služby, které mohou být směrovány a služby, které se mohou připojit k lokálnímu hostiteli, směrovat se však nebudou. Implementace IP Chains umožňuje tyto konfigurace poněkud zjednodušit pomocí uživatelských tříd, nicméně návrh firewallu stále nebude zřejmý a bude vyžadovat značnou míru zkušeností.

Implementace *netfilter* s programem **iptables** úplně odstraňuje tyto komplikace. Propouštíme-li nějakou službu přes firewall, nicméně jí nedovolujeme dosáhnout lokálního počítače, budou nám stačit dvě pravidla: obě ve třídě forward, jedno pro směr „tam“ a druhé pro směr „zpět“. Jedná se o intuitivní řešení firewallu a návrh konfigurace firewallu se tak výrazně usnadní.



Obrázek 9.9 – Zpracování datagramu v implementaci netfilter

Podrobný popis provedených změn obsahuje dokument PACKET-FILTERING-HOWTO, takže my se zde zaměříme na praktičtější aspekty.

Zpětná kompatibilita s ipfwadm a s ipchains

Významnou vlastností implementace *netfilter* je to, že dokáže emulovat rozhraní programů **ipfwadm** a **ipchains**. Tato emulace tak poněkud usnadňuje přechod na novou generaci firewallu.

Dva moduly jádra, `ipfwadm.o` a `ipchains.o` zajišťují zpětnou kompatibilitu s programy **ipfwadm** a **ipchains**. Můžete mít nahrán pouze jeden z těchto modulů a to jen tehdy, není-li nahrán modul `ip_tables.o`. Po nahrání příslušného modulu se *netfilter* chová úplně stejně jako předchozí implementace firewallu.

Emulace rozhraní programu **ipchains** se zapne následujícími příkazy:

```
rmmod ip_tables
modprobe ipchains
ipchains ...
```

Použití programu iptables

Program **iptables** slouží k nastavení filtračních pravidel mechanismu *netfilter*. Jeho syntaxe je přímo odvozena od programu **ipchains**, liší se však v jednom podstatném rysu: je *rozšiřitelná*. Znamená to, že funkce programu je možné doplnit bez nutnosti jeho nového přeložení. Tento trik je realizován pomocí sdílených knihoven. Pro program existuje několik standardních rozšíření, o kterých budeme za chvíli mluvit.

Než budete moci program **iptables** použít, musíte nahrát modul jádra *netfilter*, který zajišťuje podporu tohoto programu. Nejjednodušší způsob spočívá v následujícím volání programu **modprobe**:

```
modprobe ip_tables
```

Příkaz **iptables** slouží jak k nastavení filtrace, tak i k nastavení překladu adres. Zajišťují to dvě tabulky pravidel, nazvané *filter* a *nat*. Tabulka *filter* se nahrává automaticky, pokud neuvedete parametr `-t`. Kromě toho systém obsahuje pět vestavěných tříd. Třídy `INPUT` a `FORWARD` se týkají tabulky *filter*, třídy `PREROUTING` a `POSTROUTING` tabulky *nat* a třída `OUTPUT` obou tabulek. V této kapitole budeme hovořit pouze o tabulce *filter*. Tabulku *nat* popisujeme v kapitole 11.

Obecná syntaxe většiny příkazů **iptables** je:

```
iptables příkaz specifikace_pravidel rozšíření
```

Nejprve se seznámíme s některými parametry a pak si ukážeme příklady.

Příkazy

Existuje řada způsobů, jak v programu **iptables** manipulovat s pravidly a třídami pravidel. K nastavení firewallů se vztahují následující příkazy:

`-A` *třída* — Přidává na konec třídy jedno nebo více pravidel. Pokud je jako zdroj nebo cíl uveden název počítače a ten má přiřazeno více IP adres, budou přidána pravidla pro všechny adresy.

`-I` *třída č_prav* — Přidá jedno nebo více pravidel na začátek třídy. Opět je-li zadán název a tomu odpovídá více adres, budou přidána pravidla pro všechny adresy.

`-D třída` Vymaže ze třídy jedno nebo více pravidel, která odpovídají následující specifikaci.

`-D třída č_prav` Vymaže ze třídy pravidlo na pozici *č_prav*. Pravidla jsou číslována od jedničky.

`-R třída č_prav` Nahradí pravidlo na pozici *č_prav* novým pravidlem.

`-C třída` Porovná datagram popsaný následující specifikací s pravidly dané třídy. Příkaz vrátí zprávu o tom, jak bude datagram pravidly této třídy zpracován. Tato volba je velmi užitečná k testování konfigurace firewallu a budeme se jí později věnovat podrobněji.

`-L [třída]` Vypíše pravidla dané třídy, případně všech tříd, pokud není třída zadána.

`-F [třída]` Vymaže pravidla dané třídy, případně všech tříd, není-li třída zadána.

`-Z [třída]` Vynuluje počitadla datagramů a bajtů pro všechna pravidla dané třídy nebo všech tříd, není-li třída specifikována.

`-N třída` Vytvoří novou třídu se zadaným názvem. Třída zadaného jména nesmí existovat. Tímto příkazem se vytvářejí uživatelem definované třídy.

`-X [třída]` Vymaže třídu definovanou uživatelem, případně všechny uživatelem definované třídy, není-li název třídy uveden. Aby příkaz uspěl, na třídu nesmějí existovat odkazy z žádných jiných tříd.

`-P třída politika` Nastavuje implicitní politiku dané třídy. Platné politiky jsou `ACCEPT`, `DROP`, `QUEUE` a `RETURN`. `ACCEPT` povolí průchod datagramu. `DROP` způsobí zrušení datagramu. `QUEUE` způsobí předání datagramu do uživatelského prostoru k dalšímu zpracování. `RETURN` způsobí návrat do nadřízené třídy pravidel.

Specifikace pravidel

Program **iptables** má celou řadu parametrů sloužících k definici pravidel. Kdykoliv se zadává pravidlo, používají se všechny následující parametry. Pokud některý z parametrů není zadán, použije se jeho implicitní hodnota.

`-p [!]protokol` Udává protokol, který pravidlu vyhovuje. Platné názvy protokolů jsou `tcp`, `udp`, `icmp` nebo číselná hodnota protokolu, pokud ji znáte[85]. Hodnotou `4` můžete například definovat zapouzdřující protokol `ipip`. Je-li uveden `!`, pravidlo je negováno a budou mu vyhovovat všechny protokoly kromě zadaného. Není-li parametr uveden, budou pravidlu vyhovovat všechny protokoly.

`-s [!]adresa[/maska]`

Udává zdrojovou adresu datagramu, který pravidlu vyhovuje. Adresa může být zadána názvem počítače, názvem sítě nebo IP adresou. Nepovinný údaj *maska* představuje síťovou masku a může být zadán buď v tradiční podobě (například /255.255.255.0) nebo v moderní podobě (například /24).

`-d [!]adresa[/maska]`

Udává cílovou adresu datagramu, který pravidlu vyhovuje. Parametr se používá stejně jako `-s`.

`-j cíl` Definuje akci, která se má provést, jestliže datagram pravidlu vyhovuje. Tento parametr můžete chápat jako příkaz „běž na“. Platné cílové hodno-

85 Názvy a čísla protokolů naleznete v souboru `/etc/protocols`.

ty jsou `ACCEPT`, `DROP`, `QUEUE` a `RETURN`. Jejich význam jsme vysvětlili dříve. Kromě toho můžete uvést název uživatelské třídy pravidel a zpracování datagramu bude pokračovat v této třídě. Pokud parametr není uveden, nestane se s datagramem vůbec nic, a dojde pouze k inkrementaci počitadel bajtů a datagramů.

`-i [!]`*`název_rozhraní`*

Definuje rozhraní, na němž je datagram přijat. Symbol `!` opět neguje význam pravidla. Pokud název rozhraní končí symbolem +, vyhovují všechna rozhraní začínající zadaným řetězcem. Například `-i ppp+` definuje všechna rozhraní PPP, `-i ! eth+` definuje všechna rozhraní kromě ethernetových rozhraní.

`-o [!]`*`název_rozhraní`*

Definuje rozhraní, na němž je datagram odesílán. Použití je stejné jako u parametru `-i`.

`[!]-f` Udává, že pravidlo platí pro všechno kromě prvního fragmentu fragmentovaného datagramu.

Volby

Následující volby příkazu **iptables** jsou obecné. Některé z nich slouží k nastavení specifických nuancí systému *netfilter*.

`-v` Zapne „výřečný“ výstup programu **iptables**. Program bude sdělovat více informací.

`-n` Způsobí, že **iptables** bude vypisovat IP adresy a porty jako čísla a nebude se pokoušet o jejich převod na názvy.

`-x` Všechna čísla ve výstupu programu **iptaibles** budou uvedena přesně, bez zaokrouhlování.

`--line-numbers` Při výpisu pravidel jednotlivých tříd budou řádky číslovány. Čísla řádků odpovídají pořadí pravidla ve třídě.

Rozšíření

Už jsme se zmínili, že nástroj **iptables** je rozšiřitelný pomocí modulů sdílených knihoven. Existuje několik standardních rozšíření, která implementují funkce programu **ipchains**. Abyste mohli rozšíření použít, musíte jeho název sdělit programu **iptables** parametrem `-m` *`název`*. Následující seznam obsahuje parametry `-m` a `-p`, které nastavují kontext rozšíření, společně s parametry, které rozšíření poskytuje.

Rozšíření TCP: `-m tcp -p tcp`

`-sport [!] [port[:port]]`

Definuje zdrojový port datagramu, z nějž musí datagram pocházet, aby pravidlu vyhovoval. Je možné specifikovat i rozsahy portů zadáním spodní a horní meze oddělených dvojtečkou. Například hodnota `20:25` znamená porty od 20 (včetně) po 25 (včetně). Znakem `!` je možné význam pravidla negovat.

`-dport [!] [port[:port]]`

Definuje cílový port datagramu, na nějž musí být datagram určen, aby pravidlu vyhovoval. Použití parametru je stejné jako u `-sport`.

`-tcp-flags [!] maska comp`

Pravidlu budou vyhovovat ty TCP datagramy, jejichž příznaky odpovídají podmínkám specifikovaným parametry `maska` a `comp`. `maska` je čárkami oddělený seznam příznaků, které mají být do testování zahrnuty, `comp` je čárkami oddělený seznam testovaných příznaků, které musí být nastaveny. Platnými příznaky jsou *SYN*, *ACK*, *FIN*, *RST*, *URG*, *PSH*, *ALL* a *NONE*. Jedná se o složitější nastavení, popis významu jednotlivých příznaků najdete v nějakém kvalitním popisu protokolu TCP, například v dokumentu RFC 793. Symbolem `!` je možné význam pravidla negovat.

`[!] -syn` Udává, že pravidlu budou vyhovovat pouze datagramy s nastaveným příznakem *SYN* a nenastavenými příznaky *ACK* a *FIN*. Tyto datagramy slouží k navázání TCP spojení a pravidlo tak slouží k ošetření otevírání spojení. Tento příznak je zkratkou zápisu: `-tcp-flags SYN,RST,ACK SYN` Při použití symbolu negace budou pravidlu vyhovovat datagramy s nenastavenými příznaky *SYN* a *ACK*.

Rozšíření UDP: `-m udp -p udp`

`-sport [!] [port[:port]]`

Definuje zdrojový port datagramu, z nějž musí datagram pocházet aby pravidlu vyhovoval. Je možné specifikovat i rozsahy portů zadáním spodní a horní meze oddělených dvojtečkou. Například hodnota `20:25` znamená porty od 20 (včetně) po 25 (včetně). Znakem `!` je možné význam pravidla negovat.

`-dport [!] [port[:port]]`

Definuje cílový port datagramu, na nějž musí být datagram určen, aby pravidlu vyhovoval. Použití parametru je stejné jako u `-sport`.

Rozšíření ICMP: `-m icmp -p icmp`

`-icmp-type [!] typ`

Udává typ ICMP zprávy, která bude pravidlu vyhovovat. Typ je možné zadat číslem nebo názvem. Některé platné názvy jsou: `echo-request`, `echo-reply`, `source-quench`, `time-exceeded`, `destination-unreachable`, `network-unreachable`, `host-unreachable`, `protocol-unreachable` a `port-unreachable`.

Rozšíření MAC: `-m mac`

`-mac-source [!] adresa`

Udává ethernetovou adresu hostitele který datagram vyslal. Tento parametr má význam pouze ve třídách input a forward, protože u všech datagramů ve třídě output jsme odesílatelem my.

Znovu přepracovaný jednoduchý příklad

Pokud budeme chtít náš známý příklad implementovat pomocí *netfilter*, stačí nahrát modul `ip-chains.o` a tvářit se, že máme nainstalován program **ipchains**. Nicméně si ukážeme jeho přímou implementaci programem **iptables**.

Znovu předpokládáme, že máme síť a chceme uživatelům povolit www přístup na Internet a nechceme povolit jakýkoliv jiný provoz.

Máme 24bitovou síťovou masku (třída C) a síťovou adresu 172.16.1.0. Programem **iptables** zavedeme následující pravidla:

```
# modprobe ip_tables
# iptables -F FORWARD
# iptables -P FORWARD DROP
# iptables -A FORWARD -m tcp -p tcp -s 0/0 --sport 80 -d 172.16.1.0/24 /
    --syn -j DROP
# iptables -A FORWARD -m tcp -p tcp -s 172.16.1.0/24 --sport /
    80 -d 0/0 -j ACCEPT
# iptables -A FORWARD -m tcp -p tcp -d 172.16.1.0/24 --dport 80 -s 0/0 -j /
    ACCEPT
```

V tomto příkladu budou příkazy **iptables** interpretovány úplně stejně jako příkazy **ipchains**. Musíme mít nahrán modul `ip_tables.o`. Všimněte si, že program **iptables** nepodporuje parametr `-b`, takže musíme zadávat samostatná pravidla pro oba směry.

Manipulace s bity typu služby

Bity typu služby (TOS) jsou čtyři bitové příznaky v hlavičce IP datagramu. Pokud je některý z těchto bitů nastaven, mohou směrovače zpracovávat datagramy odlišně od zpracování datagramů bez těchto příznaků. Každý z bitů má svůj vlastní význam a v jednom datagramu může být nastaven pouze jeden z nich, kombinace příznaků nejsou povoleny. Bity se označují jako bity typu služby, protože umožňují, aby odesílající aplikace síti sdělila typ služby, kterou vyžaduje.

Existují následující třídy síťových služeb:

Minimum delay

Používá se, pokud nejdůležitějším kritériem je čas přenosu datagramu od zdroje k cíli, tedy pokud se požaduje co nejmenší zpoždění. Poskytovatel připojení může současně používat řekněme satelitní a optickou linku. Data přenášená satelitní linkou procházejí obecně delší trasou a jejich zpoždění je větší než pozemní spojení mezi dvěma stejnými místy. Směrovače mohou tedy zajistit, že datagramy s tímto příznakem budou přenášeny optickou linkou.

Maximum throughput

Používá se, pokud je důležitý objem dat přenášený v jednotlivých časových intervalech. Existuje celá řada síťových aplikací, pro něž není kritické zpoždění přenosu, je však pro ně důležitá dostatečná propustnost sítě – například přenosy on-line audia nebo videa. Poskytovatel pak bude tato data směrovat přes vysoce propustnou linku, byť s větším zpožděním, například přes satelitní linku.

Maximum reliability

Používá se, pokud je důležité, aby data v mezích možností dorazila k cíli bez nutnosti jejich opakovaného odesílání. Protokol IP může být přenášen přes řadu přenosových médií. Protokoly SLIP a PPP jsou například typickými protokoly datové linky, nejsou nicméně tak spolehlivé jako pře-

nos přes nějakou síť, například síť X.25. Poskytovatel může mít k dispozici záložní spoj s vysokou spolehlivostí a při uvedení tohoto příznaku bude data posílat tímto spojem.

Minimum cost

Používá se, je-li nutné minimalizovat přenosové náklady. Pronájem pásma na satelitní transoceánské lince je obecně levnější než pronájem pásma na optickém kabelu se stejným cílem a pokud poskytovatel používá oba spoje, může přenášená data zpoplatňovat různě podle toho, který spoj byl použit. V našem případě může nastavení tohoto bitu způsobit směrování datagramu přes levnější satelitní linku.

Nastavení TOS bitů pomocí ipfwadm a ipchains

Programy **ipfwadm** a **ipchains** pracují s TOS bity prakticky stejně. V obou můžete nastavit pravidla, kterým budou vyhovovat datagramy s určitými příznaky. Pomocí parametru `-t` pak můžete příznaky měnit.

Změny se definují dvěma bitovými maskami. První maska se logicky ANDuje s příznaky v datagramu, druhá se s nimi logicky XORuje. Pokud vám to připadá komplikované, hned si ukážeme, jak jednotlivé příznaky nastavovat.

Bitové masky se definují osmibitovou šestnáctkovou hodnotou. Jak **ipfwadm**, tak **ipchains** používají stejnou syntaxi:

```
-t andmaska xormaska
```

Pokud chcete určitý typ služby nastavit, můžete naštěstí vždy použít stejné masky, aniž by bylo nutné nad tím příliš přemýšlet. Doporučené použití a odpovídající masky jsou uvedeny v tabulce 9.3.

Tabulka 9.3 – Doporučené nastavení TOS bitů

TOS	maska AND	maska XOR	doporučené použití
Minimum Delay	0x01	0x10	ftp, telnet, ssh
Maximum Throughput	0x01	0x08	ftp-data, www
Maximum Reliability	0x01	0x04	snmp, dns
Minimum Cost	0x01	0x02	nntp, smtp

Nastavení TOS bitů pomocí iptables

Nástroj **iptables** umožňuje definovat pravidla, kterým budou vyhovovat pouze datagramy s určitým nastavením TOS bitů pomocí parametru `-m tos` a nastavovat TOS bity datagramům vyhovujícím danému pravidlu pomocí parametru `-j akce`. TOS bity je možné nastavovat pouze ve třídách output a forward. Testování a nastavování probíhá nezávisle. Můžete vytvářet libovolné skupiny pravidel. Můžete například vytvořit pravidlo, které zruší všechny datagramy s určitým nastavením TOS, nebo můžete vytvořit pravidlo, které nastaví TOS u datagramů pocházejících od určitého počítače. Ve většině případů budete používat pravidla, která kombinují jak testování, tak nastavení TOS bitů stejně, jako to bylo u programů **ipfwadm** a **ipchains**.

Namísto složité dvoumaskové konfigurace programů **ipfwadm** a **ipchains** používá program **iptables** jednodušší řešení, které přímo říká, které TOS bity musí vyhovovat nebo které TOS bity

mají být nastaveny. Navíc je nemusíte specifikovat jejich šestnáctkovými hodnotami a můžete použít názvy uvedené v následující tabulce.

Obecná syntaxe pravidla, kterému vyhovují datagramy s určitým nastavením TOS je:

```
-m tos --tos název [další_parametry] -j akce
```

Obecná syntaxe pravidla pro nastavení TOS bitů je:

```
[další_parametry] -j TOS --set název
```

Tyto příkazy se obvykle používají současně, pokud to však potřebujete, můžete je použít i nezávisle.

Název	Šestnáctkově
Normal-Service	0x00
Minimize-Cost	0x02
Maximize-Reliability	0x04
Maximize-Throughput	0x08
Minimize-Delay	0x10

Testování konfigurace firewallu

Po navržení požadované konfigurace firewallu je nutné ověřit, že se konfigurace opravdu chová tak, jak jste chtěli. Jedna možnost je použít testovací počítač vně vaší sítě a vyzkoušet, jestli se vám nepodaří přes firewall proniknout. Tato metoda je ovšem velmi pracná a zdlouhavá a je omezena na testování pouze z těch adres, které jste schopni použít.

Implementace firewallu na Linuxu nabízí rychlejší a snadnější metodu. Umožňuje vám ručně navrhnout testy a spustit je proti konfiguraci firewallu stejně, jako byste posílali skutečné datagramy. Tato metoda testování je podporována všemi generacemi firewallů na Linuxu, tedy **ipfwadm**, **ipchains** a **iptables**. Samotný test se provádí pomocí příkazu *check*.

Obecný postup testu je následující:

1. Navrhněte a vytvořte firewall pomocí programu **ipfwadm**, **ipchains** nebo **iptables**.
2. Navrhněte sérii testů, které prověří, zda firewall funguje tak, jak potřebujete. U těchto testů můžete používat libovolné zdrojové a cílové adresy, takže zvolte různé kombinace adres tak, aby některé byly povoleny a jiné zakázány. Pokud povolujete nebo zakazujete adresy z nějakého rozsahu, je rozumné otestovat adresy na hranicích rozsahu – jednu adresu právě uvnitř a jednu adresu právě vně. Tím si ověříte, že máte hranice nastaveny správně, protože běžná chyba spočívá ve špatném nastavení síťové masky. Pokud filtrujete na základě protokolů nebo čísel portů, měli byste ověřit všechny významné kombinace všech parametrů. Pokud například chcete, aby za určitých okolností byly propouštěny pouze TCP datagramy, ověřte si, že UDP datagramy firewallem neprojdou.
3. Vytvořte pravidla pro programy **ipfwadm**, **ipchains** nebo **iptables**, která implementují jednotlivé testy. Obvykle se vyplatí zapsat všechna pravidla ve skriptu, takže budete moci testy spouštět opakovaně, pokud objevíte nějaké chyby v konfiguraci. Testy používají prakticky stejnou syntaxi jako pravidla firewallu, jednotlivé parametry však mají poněkud odlišný význam. Například parametr definující zdrojovou adresu v pravidle určuje, jaká musí být zdrojová adresa datagramu, aby pravidlu vyhovoval. Zdrojový adresa v testovacím pra-

vidle naproti tomu určuje, jakou zdrojovou adresu bude mít vygenerovaný testovací datagram. V programu **ipfwadm** musíte použít přepínač `-c` udávající, že pravidlo je testovacím pravidlem. V programech **ipchains** a **iptables** slouží ke stejnému účelu parametr `-C`. Ve všech pravidlech musíte definovat zdrojovou adresu, cílovou adresu, protokol a rozhraní. Další parametry, například číslo portu nebo nastavení TOS bitů, jsou nepovinné.

4. Proveďte každý testovací příkaz a sledujte výstup. Výsledkem každého testovacího příkazu je jediné slovo sdělující, jaký je konečný cíl datagramu poté, co jej firewall zpracuje – tedy, co se s tímto datagramem stane. V programech **ipchains** a **iptables** jsou samozřejmě testovány i uživatelem definované třídy pravidel.
5. Porovnejte výsledky testů s požadovanými výsledky. Pokud narazíte na nějaké nesrovnalosti, budete muset analyzovat navržená pravidla a zjistit, kde jste udělali chybu. Pokud jste jednotlivé testovací příkazy zadali do skriptu, budete moci po opravě konfigurace firewallu snadno spustit test znovu. Bývá rozumné úplně vymazat nastavená pravidla a vytvářet je od počátku, nikoliv se pokoušet o jejich dynamické úpravy. Tím máte zajištěno, že testovaná konfigurace firewallu opravdu odpovídá té, kterou nastavujete v jeho konfiguračních skriptech.

Podívejme se v krátkosti na to, jak by vypadalo testování naší základní firewallové konfigurace pomocí programu **ipchains**. Připomeňme si, že lokální síť měla adresu 172.16.1.0 se síťovou maskou 255.255.255.0 a povolovali jsme pouze připojení z naší sítě na webové servery. Žádná jiná data by neměla firewallem projít. Začněme datagramem o němž víme, že by měl projít – spojení z naší sítě na nějaký webový server:

```
# ipchains -C forward -p tcp -s 172.16.1.0 1025 -d 44.136.8.2 80 -i eth0
accepted
```

Všimněte si, které parametry a jak jsme zadávali. Výstup příkazu nám říká, že datagram byl přijat k předání, což je to, co jsme očekávali.

Teď vyzkoušíme další test, tentokrát se zdrojovou adresou, která nepatří do naší sítě. Tento datagram by neměl projít:

```
# ipchains -C forward -p tcp -s 172.16.2.0 1025 -d 44.136.8.2 80 -i eth0
denied
```

Teď vyzkoušíme další testy, sice se stejnými vlastnostmi jako v prvním případě, ale s jinými protokoly. Tyto datagramy by projít neměly:

```
# ipchains -C forward -p udp -s 172.16.1.0 1025 -d 44.136.8.2 80 -i eth0
denied
# ipchains -C forward -p icmp -s 172.16.1.0 1025 -d 44.136.8.2 80 -i eth0
denied
```

Ještě vyzkoušíme jiný cílový port a opět předpokládáme, že datagram neprojde:

```
# ipchains -C forward -p tcp -s 172.16.1.0 1025 -d 44.136.8.2 23 -i eth0
denied
```

Návrh série vyčerpávajících testů je poměrně zdlouhavá záležitost. I když se může jednat o téměř stejně náročnou operaci jako byl samotný návrh konfigurace firewallu, je to jediná metoda jak ověřit, že návrh funguje tak, jak jste očekávali.

Příklad konfigurace firewallu

Představili jsme si základy konfigurace firewallu. Nyní se podíváme na to, jak by mohla vypadat skutečná konfigurace firewallu.

Konfigurace v následujícím příkladu byla navržena tak, aby ji bylo možné snadno rozšířit a upravit. Celý příklad předkládáme ve třech verzích. První z nich je implementována pomocí příkazu **ipfwadm** (nebo skriptem **ipfwadm-wrapper**), druhá používá **ipchains** a třetí **iptables**. V příkladu nevyužíváme uživatelem definované třídy, nicméně můžete si porovnat rozdíly a podobnosti mezi novou a starou syntaxí nástrojů pro konfiguraci firewallu.

```
#!/bin/bash
##############################################################################
# VERZE PRO IPFWADM
# Příklad konfigurace firewallu na jednom počítači, který sám
# neposkytuje žádné služby.
##############################################################################

# UŽIVATELEM KONFIGUROVATELNÁ ČÁST

# Název a umístění nástroje ipfwadm. Pro jádra 2.2.* zadejte ipfwadm-wrapper
IPFWADM=ipfwadm

# Cesta k souboru ipfwadm
PATH="/sbin"

# Interní adresový prostor naší sítě a jemu příslušné síťové zařízení
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# Vnější adresy a odpovídající síťové zařízení
ANYADDR="0/0"
ANYDEV="eth1"

# TCP služby které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
TCPIN="smtp www"
TCPOUT="smtp www ftp ftp-data irc"

# UDP služby které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
UDPIN="domain"
UDPOUT="domain"

# ICMP služby které chceme povolit.
# "" znamená všechny typy služeb
# Čísla typů viz /usr/include/netinet/ip_icmp.
# Mezerami oddělovaný seznam.
ICMPIN="0 3 11"
ICMPOUT="8 3 11"
```

```
# Logování. Odstraněním komentáře se zapne logování datagramů,
# které firewall nepropustil.
# LOGGING=1

# KONEC UŽIVATELEM KONFIGUROVATELNÉ ČÁSTI
################################################################################
# Vyprázdnění pravidel třídy Incoming
$IPFWADM -I -f

# Implicitně zakazujeme přístup třídou Incoming
$IPFWADM -I -p deny

# SPOOFING
# Nechceme zvenčí přijímat žádné datagramy se zdrojovou adresou
# z naší sítě, takže je blokujeme
$IPFWADM -I -a deny -S $OURNET -W $ANYDEV

# SMURF
# Zakazujeme ICMP na naši vysílací adresu, ochrana před "Smurf" útoky
$IPFWADM -I -a deny -P icmp -W $ANYDEV -D $OURBCAST

# TCP
# Přijímáme všechny TCP datagramy patřící existujícím spojením (tedy
# s nastaveným bitem ACK) pro všechny povolené TCP porty.
# To by mělo obsáhnout více než 95% TCP paketů.
$IPFWADM -I -a accept -P tcp -D $OURNET $TCPIN -k -b

# TCP - PŘÍCHOZÍ SPOJENÍ
# Požadavky na TCP spojení zvenčí povolíme pouze na vybraných portech.
$IPFWADM -I -a accept -P tcp -W $ANYDEV -D $OURNET $TCPIN -y

# TCP - ODCHOZÍ SPOJENÍ
# Povolíme všechna odchozí spojení na povolených portech.
$IPFWADM -I -a accept -P tcp -W $OURDEV -D $ANYADDR $TCPOUT -y

# UDP - PŘÍCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPFWADM -I -a accept -P udp -W $ANYDEV -D $OURNET $UDPIN

# UDP - ODCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPFWADM -I -a accept -P udp -W $OURDEV -D $ANYADDR $UDPOUT

# ICMP - PŘÍCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPFWADM -I -a accept -P icmp -W $ANYDEV -D $OURNET $UDPIN

# ICMP - ODCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPFWADM -I -a accept -P icmp -W $OURDEV -D $ANYADDR $UDPOUT

# IMPLICITNÍ a LOGOVÁNÍ
# Všechny zbývající datagramy projdou k implicitnímu pravidlu a
```

```
# budou zahozeny. Pokud byla nastavena proměnná LOGGING, budou
# zaznamenány.
#
if [ "$LOGGING" ]
then
        # Logovat zahozené TCP
        $IPFWADM -I -a reject -P tcp -o

        # Logovat zahozené UDP
        $IPFWADM -I -a reject -P udp -o

        # Logovat zahozené ICMP
        $IPFWADM -I -a reject -P icmp -o
fi
#
# konec
```

Nyní budeme stejný příklad implementovat pomocí programu **ipchains**.

```
#!/bin/bash
################################################################################
# VERZE PRO IPCHAINS
# Příklad konfigurace firewallu na jednom počítači, který sám
# neposkytuje žádné služby.
################################################################################

# UŽIVATELEM KONFIGUROVATELNÁ ČÁST

# Název a umístění nástroje iptables.
IPCHAINS=ipchains

# Cesta k souboru ipchains
PATH="/sbin"

# Interní adresový prostor naší sítě a jemu příslušné síťové zařízení
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"

# Vnější adresy a odpovídající síťové zařízení
ANYADDR="0/0"
ANYDEV="eth1"

# TCP služby které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
TCPIN="smtp www"
TCPOUT="smtp www ftp ftp-data irc"

# UDP služby které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
UDPIN="domain"
UDPOUT="domain"
```

```
# ICMP služby které chceme povolit.
# "" znamená všechny typy služeb
# Čísla typů viz /usr/include/netinet/ip_icmp.
# Mezerami oddělovaný seznam.
ICMPIN="0 3 11"
ICMPOUT="8 3 11"

# Logování. Odstraněním komentáře se zapne logování datagramů,
# které firewall nepropustil.
# LOGGING=1

# KONEC UŽIVATELEM KONFIGUROVATELNÉ ČÁSTI
################################################################################
# Vyprázdnění pravidel třídy Input
$IPCHAINS -F input

# Implicitně zakazujeme přístup třídou Input
$IPCHAINS -P input deny

# SPOOFING
# Nechceme zvenčí přijímat žádné datagramy se zdrojovou adresou
# z naší sítě, takže je blokujeme
$IPCHAINS -A input -s $OURNET -i $ANYDEV -j deny

# SMURF
# Zakazujeme ICMP na naši vysílací adresu, ochrana před "Smurf" útoky
$IPCHAINS -A input -p icmp -w $ANYDEV -d $OURBCAST -j deny

# Chceme přijímat fragmenty, v ipchains to musíme explicitně povolit.
$IPCHAINS -A input -f -j accept

# TCP
# Přijímáme všechny TCP datagramy patřící existujícím spojením (tedy
# s nastaveným bitem ACK) pro všechny povolené TCP porty.
# To by mělo obsáhnout více než 95% TCP paketů.
$IPCHAINS -A input -p tcp -d $OURNET $TCPIN ! -y -b -j accept

# TCP - PŘÍCHOZÍ SPOJENÍ
# Požadavky na TCP spojení zvenčí povolíme pouze na povolených portech.
$IPCHAINS -A input -p tcp -i $ANYDEV -d $OURNET $TCPIN -y -j accept

# TCP - ODCHOZÍ SPOJENÍ
# Povolíme všechna odchozí spojení na povolených portech.
$IPCHAINS -A input -p tcp -i $OURDEV -d $ANYADDR $TCPOUT -y -j accept

# UDP - PŘÍCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPCHAINS -A input -p udp -i $ANYDEV -d $OURNET $UDPIN -j accept

# UDP - ODCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPCHAINS -A input -p udp -i $OURDEV -d $ANYADDR $UDPOUT -j accept
```

```
# ICMP - PŘÍCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPCHAINS -A input -p icmp -w $ANYDEV -d $OURNET $UDPIN -j accept

# ICMP - ODCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPCHAINS -A input -p icmp -i $OURDEV -d $ANYADDR $UDPOUT -j accept

# IMPLICITNÍ a LOGOVÁNÍ
# Všechny zbývající datagramy projdou k implicitnímu pravidlu a
# budou zahozeny. Pokud byla nastavena proměnná LOGGING, budou
# zaznamenány.
#
if [ "$LOGGING" ]
then
      # Logovat zahozené TCP
        $IPCHAINS -A input -p tcp -l -j reject

        # Logovat zahozené UDP
        $IPCHAINS -A input -p udp -l -j reject

        # Logovat zahozené ICMP
        $IPCHAINS -A input -p icmp -l -j reject
fi
#
# konec.
```

V příkladu používajícím program **iptables** pracujeme s pravidly třídy `FORWARD`, protože třída `INPUT` má v této implementaci jiný význam. Vedlejším efektem je, že žádná z pravidel nechrání samotný firewall. Pokud bychom chtěli přesně replikovat příklad s programem **ipchains**, museli bychom všechna pravidla zkopírovat i do třídy `INPUT`. Kvůli jednoduchosti místo toho zahazujeme všechny datagramy přijaté z vnější sítě.

```
#!/bin/bash
##############################################################################
# VERZE PRO IPTABLES
# Příklad konfigurace firewallu na jednom počítači, který sám
# neposkytuje žádné služby.
##############################################################################

# UŽIVATELEM KONFIGUROVATELNÁ ČÁST

# Název a umístění nástroje ipchains.
IPTABLES=iptables

# Cesta k souboru ipchains.
PATH="/sbin"

# Interní adresový prostor naší sítě a jemu příslušné síťové zařízení
OURNET="172.29.16.0/24"
OURBCAST="172.29.16.255"
OURDEV="eth0"
```

```
# Vnější adresy a odpovídající síťové zařízení
ANYADDR="0/0"
ANYDEV="eth1"

# TCP služby které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
TCPIN="smtp,www"
TCPOUT="smtp,www,ftp,ftp-data,irc"

# UDP služby které chceme povolit.
# "" znamená všechny porty.
# Mezerami oddělovaný seznam.
UDPIN="domain"
UDPOUT="domain"

# ICMP služby které chceme povolit.
# "" znamená všechny typy služeb
# Čísla typů viz /usr/include/netinet/ip_icmp.
# Mezerami oddělovaný seznam.
ICMPIN="0,3,11"
ICMPOUT="8,3,11"

# Logování. Odstraněním komentáře se zapne logování datagramů,
# které firewall nepropustil.
# LOGGING=1

# KONEC UŽIVATELEM KONFIGUROVATELNÉ ČÁSTI
################################################################################
# Vyprázdnění pravidel třídy Forward
$IPTABLES -F FORWARD

# Implicitně zakazujeme přístup třídou Forward
$IPTABLES -P FORWARD deny

# Zahození všech datagramů pro tento počítač zvenčí.
$IPTABLES -A INPUT -i $ANYDEV -j DROP

# SPOOFING
# Nechceme zvenčí přijímat žádné datagramy se zdrojovou adresou
# z naší sítě, takže je blokujeme
$IPTABLES -A FORWARD -s $OURNET -i $ANYDEV -j DROP

# SMURF
# Zakazujeme ICMP na naši vysílací adresu, ochrana před "Smurf" útoky
$IPTABLES -A FORWARD -m multiport -p icmp -i $ANYDEV -d $OURNET -j DENY

# Chceme přijímat fragmenty, v iptables to musíme explicitně povolit.
$IPTABLES -A FORWARD -f -j ACCEPT

# TCP
# Přijímáme všechny TCP datagramy patřící existujícím spojením (tedy
```

```
# s nastaveným bitem ACK) pro všechny povolené TCP porty.
# To by mělo obsáhnout více než 95% TCP paketů.
$IPTABLES -A FORWARD -m multiport -p tcp -d $OURNET --dports $TCPIN /
    ! --tcp-flags SYN,ACK ACK -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p tcp -s $OURNET --sports $TCPIN /
    ! --tcp-flags SYN,ACK ACK -j ACCEPT


# TCP - PŘÍCHOZÍ SPOJENÍ
# Požadavky na TCP spojení zvenčí povolíme pouze na povolených portech.
$IPTABLES -A FORWARD -m multiport -p tcp -i $ANYDEV -d $OURNET $TCPIN /
    --syn -j ACCEPT

# TCP - ODCHOZÍ SPOJENÍ
# Povolíme všechna odchozí spojení na povolených portech.
$IPTABLES -A FORWARD -m multiport -p tcp -i $OURDEV -d $ANYADDR /
    --dports $TCPOUT --syn -j ACCEPT

# UDP - PŘÍCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -d $OURNET /
    --dports $UDPIN -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p udp -i $ANYDEV -s $OURNET /
    --sports $UDPIN -j ACCEPT

# UDP - ODCHOZÍ
# Povolíme UDP datagramy na povolených portech.
$IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -d $ANYADDR /
    --dports $UDPOUT -j ACCEPT
$IPTABLES -A FORWARD -m multiport -p udp -i $OURDEV -s $ANYADDR /
    --sports $UDPOUT -j ACCEPT

# ICMP - PŘÍCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPTABLES -A FORWARD -m multiport -p icmp -i $ANYDEV -d $OURNET /
    --dports $ICMPIN -j ACCEPT

# ICMP - ODCHOZÍ
# Povolíme ICMP datagramy na povolených portech.
$IPTABLES -A FORWARD -m multiport -p icmp -i $OURDEV -d $ANYADDR /
    --dports $ICMPOUT -j ACCEPT

# IMPLICITNÍ a LOGOVÁNÍ
# Všechny zbývající datagramy projdou k implicitnímu pravidlu a
# budou zahozeny. Pokud byla nastavena proměnná LOGGING, budou
# zaznamenány.
#
if [ "$LOGGING" ]
then
        # Logovat zahozené TCP
        $IPTABLES -A FORWARD -m tcp -p tcp -j LOG
      # Logovat zahozené UDP
        $IPTABLES -A FORWARD -m udp -p udp -j LOG
```

```
        # Logovat zahozené ICMP
        $IPTABLES -A FORWARD -m udp -p icmp -j LOG
fi
#
# konec.
```

V řadě jednoduchých situací budete moci tento příklad použít s tím, že pouze upravíte údaje v části označené jako „uživatelem konfigurovatelná“, kde specifikujete, které protokoly a typy datagramů chcete propouštět dovnitř a ven. Ve složitějších případech budete muset modifikovat i spodní část příkladu. Nezapomeňte, že se jedná pouze o příklad, takže jej pečlivě prostudujte, zda opravdu dělá to, co vy potřebujete.

IP účtování

V současném světě komerčních internetových služeb je stále důležitější vědět, kolik dat přijímáte a odesíláte svým síťovým připojením. Pokud jste poskytovatel internetového připojení a účtujete podle objemu přenesených dat, je pro vás tato znalost nezbytná. Pokud jste zákazník poskytovatele, který podle objemu dat účtuje, určitě vás bude zajímat, nakolik jsou údaje uváděné poskytovatelem správné.

Kromě toho má účtování i jiné využití, nemusí jít jen o peníze a poplatky. Pokud provozujete nějaký server, který nabízí řadu různých síťových služeb, jistě vás bude zajímat, nakolik jsou jednotlivé služby zatěžovány. Pomocí těchto informací se usnadňují rozhodnutí jako zda aktualizovat hardware, zda rozdělit služby na více serverů a podobně.

Jádro Linuxu nabízí funkci, která umožňuje shromažďovat všechny typy užitečných informací o síťovém provozu, který jádro vidí. Tato funkce se označuje jako *IP účtování*.

Konfigurace jádra pro účtování

Účtovací služby jsou úzce spjaty s firewally. Místa, kde se účtovací data shromažďují, odpovídají těm místům, kde probíhá filtrace firewallem: směr do a z vašeho počítače a software zajišťující směrování. Pokud jste nečetli kapitolu věnovanou firewallům, bude dobré to napravit, protože zde budeme vycházet z některých znalostí, které jsme se dozvěděli v kapitole 9.

Než budete účtování aktivovat, musíte si ověřit, zda je jádro pro tuto službu nakonfigurováno. Podívejte se, zda existuje soubor `/proc/net/ip_acct`. Pokud ano, jádro účtování podporuje. Pokud ne, musíte sestavit nové jádro a v jádrech 2.0 a 2.2 odpovědět „Y“ na následující dotazy:

```
Networking options  --->
        [*] Network firewalls
        [*] TCP/IP networking
         ...
        [*] IP: accounting
```

V jádrech 2.4 pak na tyto otázky:

```
Networking options  --->
    [*] Network packet filtering (replaces ipchains)
```

Konfigurace IP účtování

Protože je účtování blízce spjato s funkcemi firewallu, konfiguruje se i stejným nástrojem, tedy programy **ipfwadm**, **ipchains** nebo **iptables** podle verze jádra. Syntaxe příkazu se velmi podo-

bá klasickým pravidlům firewallu, takže o ní nebudeme příliš hovořit a místo toho se zaměříme na to, co můžete pomocí této funkce o síťovém provozu zjistit.

Obecná syntaxe příkazu **ipfwadm** pro konfiguraci IP účování je:

```
# ipfwadm -A [směr] [příkaz] [parametry]
```

Nový je parametr *směr*. Může mít hodnoty `in`, `out` nebo `both`. Směry jsou chápány z pohledu účtovacího počítače, tedy `in` znamená data přicházející ze sítě do tohoto počítače, `out` data odcházející z tohoto počítače na síť a `both` je jednoduše součet obou těchto směrů.

Obecná syntaxe příkazů **ipchains** a **iptables** je:

```
# ipchains -A třída specifikace_pravidla
# iptables -A třída specifikace_pravidla
```

Příkazy **ipchains** a **iptables** umožňují definovat směr způsobem, který je konzistentnější se samotnými pravidly firewallu. V těchto programech sice nemůžete specifikovat pravidla provádějící účtování pro oba směry současně, na druhé straně můžete specifikovat pravidla pro směrovací třídu, což ve starších verzích nebylo možné. Rozdíly uvidíme v příkladech o něco později.

Příkazy jsou stejné jako pravidla pro konfiguraci firewallu, pouze se zde nespecifikuje akce, která se má s datagramem provést. Účtovací pravidla je možné známými způsoby přidávat, vkládat, mazat a vypisovat. V programech **ipchains** a **iptables** jsou účtovacími pravidly všechna platná pravidla, pravidla u nichž není specifikována akce parametrem `-j` pak fungují jako čistě účtovací.

Parametry specifikující pravidla jsou pro účtování stejná jako pro samotný firewall. Pomocí nich přesně definujeme typy síťového provozu, které chceme účtovat.

Účtování podle adresy

Podívejme se nyní na příklad ilustrující, jak IP účtování používat.

Představme si směrovač, který spojuje dvě oddělení virtuálního pivovaru. Směrovač má dvě ethernetová rozhraní, `eth0` a `eth1`, která jsou připojena ke zmíněným dvěma oddělením, a rozhraní `ppp0`, které nás vysokorychlostní sériovou linkou připojuje k síti univerzity Groucho Marx.

Dále si představme, že pro potřeby proplácení potřebujeme znát objemy dat generované jednotlivými odděleními na sériové lince a pro potřeby správy potřebujeme znát objem dat přenášený mezi odděleními.

Následující tabulka shrnuje adresy rozhraní, které budeme v příkladu používat:

Rozhraní	Adresa	Maska
`eth0`	172.16.3.0	255.255.255.0
`eth1`	172.16.4.0	255.255.255.0

Abychom mohli odpovědět na otázku „Kolik provozu generují jednotlivá rozdělení na lince PPP?“, zavedeme následující pravidla:

```
# ipfwadm -A both -a -W ppp0 -S 172.16.3.0/24 -b
# ipfwadm -A both -a -W ppp0 -S 172.16.4.0/24 -b
```

nebo

```
# ipchains -A input -i ppp0 -d 172.16.3.0/24
# ipchains -A output -i ppp0 -s 172.16.3.0/24
# ipchains -A input -i ppp0 -d 172.16.4.0/24
# ipchains -A output -i ppp0 -s 172.16.4.0/24
```

a s programem **iptables**

```
# iptables -A FORWARD -i ppp0 -d 172.16.3.0/24
# iptables -A FORWARD -o ppp0 -s 172.16.3.0/24
# iptables -A FORWARD -i ppp0 -d 172.16.4.0/24
# iptables -A FORWARD -o ppp0 -s 172.16.4.0/24
```

První polovina pravidel v každé skupině říká „počítej data přenášená oběma směry přes rozhraní `ppp0` s zdrojem nebo cílem (připomeňme si funkci příznaku `-b` v programech **ipfwadm** a **ipchains**) `172.16.3.0./24`“. Druhá polovina pravidel dělá to samé, ovšem pro druhou ethernetovou síť.

Abychom mohli odpovědět na druhou otázku, „Kolik dat se přenáší mezi odděleními?“, potřebujeme následující pravidla:

```
# ipfwadm -A both -a -S 172.16.3.0/24 -D 172.16.4.0/24 -b
```

nebo

```
# ipchains -A forward -s 172.16.3.0/24 -d 172.16.4.0/24 -b
```

nebo

```
# iptables -A FORWARD -s 172.16.3.0/24 -d 172.16.4.0/24
# iptables -A FORWARD -s 172.16.4.0/24 -d 172.16.3.0/24
```

Tato pravidla počítají datagramy se zdrojovou adresou v jednom oddělení a s cílovou adresou v druhém a naopak.

Účtování podle portu služby

Předpokládejme nyní, že chceme získat přesnější představu o tom, jaký typ provozu se přenáší přes PPP linku. Může nás například zajímat, nakolik je linka vytížena službami FTP, SMTP a WWW.

Skript s pravidly, která tyto informace získají, může vypadat takto:

```
#!/bin/sh
# Statistiky FTP, SMTP a WWW přenosů přes PPP linku pro ipfwadm
#
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 ftp ftp-data
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 smtp
ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 www
```

nebo

```
#!/bin/sh
# Statistiky FTP, SMTP a WWW přenosů přes PPP linku pro ipchains
#
ipchains -A input -i ppp0 -p tcp -s 0/0 ftp-data:ftp
```

```
ipchains -A output -i ppp0 -p tcp -d 0/0 ftp-data:ftp
ipchains -A input -i ppp0 -p tcp -s 0/0 smtp
ipchains -A output -i ppp0 -p tcp -d 0/0 smtp
ipchains -A input -i ppp0 -p tcp -s 0/0 www
ipchains -A output -i ppp0 -p tcp -d 0/0 www
```

nebo

```
#!/bin/sh
# Statistiky FTP, SMTP a WWW přenosů přes PPP linku pro iptables
#
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport ftp-data:ftp
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport ftp-data:ftp
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport smtp
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport smtp
iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport www
iptables -A FORWARD -o ppp0 -m tcp -p tcp --dport www
```

V této konfiguraci najdeme řadu zajímavých momentů. Nejprve si všimněte, že jsme specifikovali protokol. Protože v pravidlech definujeme čísla portů, musíme uvést i protokol, jelikož protokoly TCP a UDP používají samostatné porty. Všechny zmíněné služby používají protokol TCP, proto jsme specifikovali jej. Dále jsme v jednom příkazu definovali dvě služby – `ftp` a `ftp-data`. Program **ipfwadm** dovoluje definovat jeden port, rozsah portů nebo seznam portů. Příkaz **ipchains** pak dovoluje definovat jeden port nebo rozsah portů a této možnosti jsme také využili. Zápis „`ftp-data:ftp`“ znamená „porty od ftp-data (20) po ftp (21)“ a tímto zápisem v programech **ipchains** i **iptables** specifikujeme skupinu portů. Pokud je v účtovacím pravidle specifikováno více portů, budou se do celkových statistik započítávat data přijatá nebo odeslaná kterýmkoliv z těchto portů. Protože víme, že služba FTP používá dva porty, jeden pro přenos příkazů a druhý pro přenos dat, uvedli jsme je oba dva, abychom získali celkové statistiky FTP přenosů. Dále jsme definovali zdrojovou adresu „`0/0`“, což je speciální notace označující všechny adresy – v programech **ipfwadm** a **ipchains** je totiž při zadání portů nutné zadat i adresu.

Podívejme se nyní na tento příklad blíže a zkusme získat o využití rozhraní přesnější představu. Řekněme, že služby WWW, FTP a SMTP představují „základní“ provoz, všechno ostatní je „nezákladní“ provoz. Pokud nás bude zajímat poměr mezi základním a nezákladním provozem, můžeme definovat například takováto pravidla:

```
# ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 ftp ftp-data smtp www
# ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 1:19 22:24 26:79 81:32767
```

Podíváte-li se do souboru `/etc/services`, zjistíte, že druhé pravidlo pokrývá všechny porty kromě portů služeb WWW, FTP a SMTP.

Jak to budeme implementovat pomocí programů **ipchains** a **iptables**, které dovolují definovat port pouze jediným parametrem? V takovém případě můžeme s výhodou využít uživatelsky definované třídy stejně, jako jsme to dělali při konfiguraci samotného firewallu. Podívejte se na následující řešení:

```
# ipchains -N a-essent
# ipchains -N a-noness
# ipchains -A a-essent -j ACCEPT
# ipchains -A a-noness -j ACCEPT
# ipchains -A forward -i ppp0 -p tcp -s 0/0 ftp-data:ftp -j a-essent
```

```
# ipchains -A forward -i ppp0 -p tcp -s 0/0 smtp -j a-essent
# ipchains -A forward -i ppp0 -p tcp -s 0/0 www -j a-essent
# ipchains -A forward -j a-noness
```

Vytvořili jsme dvě uživatelské třídy, `a-essent`, kde shromažďujeme účtovací data základních služeb, a `a-noness`, kde shromažďujeme data ostatních služeb. Pak jsme do směrovací třídy přidali pravidla, která pokrývají základní služby a přeskakují do třídy `a-essent`, v níž se nachází jediné pravidlo povolující všechny datagramy a zároveň shromažďující jejich statistiky. Poslední pravidlo skáče do třídy `a-noness`, kde máme opět jediné pravidlo, povolující a počítající všechny datagramy. Pravidlo směřující do třídy `a-noness` nebude uplatněno na žádnou ze základních služeb, protože ty už byly povoleny ve své samostatné třídě. Statistiky základních a ostatních služeb tak budeme mít k dispozici v pravidlech v námi definovaných třídách. Toto řešení je pouze jedno z možných, jsou i jiná. Implementace stejného chování programem **iptables** bude vypadat takto:

```
# iptables -N a-essent
# iptables -N a-noness
# iptables -A a-essent -j ACCEPT
# iptables -A a-noness -j ACCEPT
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport ftp-data:ftp -j a-essent
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport smtp -j a-essent
# iptables -A FORWARD -i ppp0 -m tcp -p tcp --sport www -j a-essent
# iptables -A FORWARD -j a-noness
```

Všechno vypadá poměrně jednoduše. Bohužel při účtování podle typů služeb narážíme na jeden malý, nicméně nezanedbatelný problém. Jistě si vzpomente, když jsme v souvislosti se sítěmi TCP/IP hovořili o hodnotě MTU. Hodnota MTU definuje velikost největšího možného datagramu, který bude síťovým zařízením přenesen. Pokud směrovač přijme datagram delší než je MTU rozhraní, přes něž má být odeslán, provede směrovač trik zvaný *fragmentace*. Směrovač rozdělí velký datagram na části kratší než MTU daného rozhraní a odešle tyto části. Pro jednotlivé části vygeneruje směrovač nové hlavičky a pomocí nich bude přijímající počítač schopen data zrekonstruovat. Bohužel procesem fragmentace se u všech kromě prvního fragmentu ztrácí číslo portu. Znamená to, že IP účtování bude schopné správně zaznamenávat pouze první fragmenty a nefragmentované datagramy. Program **ipfwadm** používá malý trik, takže i když nebudeme vědět, které službě jednotlivé fragmenty patří, stále je budeme schopni počítat. První verze účtovacích programů na Linuxu přiřazovaly fragmentům falešné číslo portu 0xFFFF, a to jsme mohli počítat. K zachycení druhého a dalších fragmentů proto použijeme následující pravidlo:

```
# ipfwadm -A both -a -W ppp0 -P tcp -S 0/0 0xFFFF
```

Implementace použitá v IP Chains je sice trochu propracovanější, nicméně výsledek je prakticky stejný. S programem **ipchains** použijeme následující příkaz:

```
# ipchains -A forward -i ppp0 -p tcp -f
```

A s **iptables**:

```
# iptables -A FORWARD -i ppp0 -m tcp -p tcp -f
```

Ani zde se nedozvíme původní port přenášených dat, nicméně budeme alespoň schopni říct, kolik dat je fragmentovaných a budeme schopni určit, kolik provozu tato data tvoří.

V jádrech 2.2 můžete při sestavování jádra použít parametr, který celý problém vyřeší za předpokladu, že daný počítač představuje jediný přístupový uzel do sítě. Pokud při sestavování jádra za-

dáte volbu `IP: always defragment`, linuxový směrovač všechny fragmentované datagramy před dalším směrováním a odesláním zrekonstruuje. Tato operace se provádí předtím, než datagramy zpracovává firewall a účtovací systém, takže se k nim nikdy žádné fragmenty nedostanou. V jádrech 2.2 musíte přeložit a nahrát modul `forward-fragment` z balíku *netfilter*.[86]

Účtování ICMP datagramů

Protokol ICMP nepoužívá čísla portů a proto se podrobnosti o statistikách tohoto protokolu zjišťují obtížněji. Protokol ICMP používá různé typy datagramů. Řada z nich je neškodných a běžných, jiné se objevují jen za speciálních okolností. Občas se znudění útočníci pokoušejí zablokovat přístup k nějakému systému tím, že na něj posílají obrovské množství ICMP datagramů. Tento postup se běžně označuje jako *ping flooding*. I když IP účtování s tím nemůže nic udělat (ovšem firewall do jisté míry může[87]), můžeme přinejmenším vytvořit účtovací pravidla z nichž se dozvíme, že se někdo o takovýto útok pokouší.

Protokol ICMP nepoužívá porty tak jako protokoly TCP a UDP. Namísto toho pracuje s různými typy ICMP zpráv. Můžeme vytvářet pravidla pro účtování jednotlivých typů těchto zpráv. V programu **ipfwadm** to uděláme tak, že místo čísla portu specifikujeme číslo typu ICMP zprávy. Typy ICMP jsme uvedli v části *Typy ICMP datagramů* v kapitole 9.

Účtovací pravidla zjišťující statistiky dat příkazu **ping** budou vypadat takto:

```
# ipfwadm -A both -a -P icmp -S 0/0 8
# ipfwadm -A both -a -P icmp -S 0/0 0
# ipfwadm -A both -a -P icmp -S 0/0 0xff
```

nebo pro **ipchains**

```
# ipchains -A forward -p icmp -s 0/0 8
# ipchains -A forward -p icmp -s 0/0 0
# ipchains -A forward -p icmp -s 0/0 -f
```

nebo pro **iptables**

```
# iptables -A FORWARD -m icmp -p icmp --sports echo-request
# iptables -A FORWARD -m icmp -p icmp --sports echo-reply
# iptables -A FORWARD -m icmp -p icmp -f
```

První pravidlo počítá zprávy „ICMP Echo Request“ (žádosti programu **ping**), druhé pravidlo zprávy „ICMP Echo Reply“ (odpovědi na tyto žádosti). Třetí pravidlo počítá statistiky ICMP fragmentů. Jedná se o podobný trik, o kterém jsme hovořili v souvislosti s fragmenty TCP a UDP.

Pokud v pravidlech specifikujete zdrojové a/nebo cílové adresy, můžete získávat i informace o tom, odkud pakety pocházejí – například zda zevnitř nebo vně vaší sítě. Jakmile zjistíte odkud datagram pochází, můžete na firewallu zavést pravidla, která je zablokují, nebo můžete podnik-

86 Pozn. překladatele: Jak bylo řečeno, tento přístup lze použít pouze v případě, že jste k vnějšímu světu připojeni jediným uzlem. Pokud máte přípojných bodů více, není zaručeno, že všechny fragmenty jednoho datagramu projdou stejným uzlem – bude je sestavovat až cílový počítač a směrovač tuto funkci nemůže plnit, protože prostě nemusí všechny fragmenty obdržet.

87 Pozn. překladatele: Většinou nepomůže ani firewall. Předpokládejme typickou konfiguraci, kdy jste k Internetu připojeni relativně pomalou linkou přes nějaký směrovač či firewall a další rozvod v interní síti je řešen relativně rychlými linkami. Pokud vás někdo zahltí ICMP datagramy, firewall je sice může všechny zahodit takže neproniknou dále do interní sítě, nicméně internetová linka zůstává plně vytížena a nebudete ji moci použít. Pak vám musí pomoci někdo na druhém konci vaší linky (tedy zřejmě poskytovatel) a zablokovat datagramy na *svém* firewallu, aby se na vaši linku vůbec nedostaly.

nout jiné akce, například kontaktovat správce vzdálené sítě s žádostí o řešení problému, případně můžete zvolit nějaký radikálnější postup v případě, že se jedná o napadení vaší sítě.

Účtování podle protokolu

Nyní si představme, že nás zajímá, jaký provoz připadá na protokoly TCP, UDP a ICMP. Můžeme to zjistit následujícími pravidly:

```
# ipfwadm -A both -a -W ppp0 -P tcp -D 0/0
# ipfwadm -A both -a -W ppp0 -P udp -D 0/0
# ipfwadm -A both -a -W ppp0 -P icmp -D 0/0
```

nebo

```
# ipchains -A forward -i ppp0 -p tcp -d 0/0
# ipchains -A forward -i ppp0 -p udp -d 0/0
# ipchains -A forward -i ppp0 -p icmp -d 0/0
```

nebo

```
# iptables -A FORWARD -i ppp0 -m tcp -p tcp
# iptables -A FORWARD -o ppp0 -m tcp -p tcp
# iptables -A FORWARD -i ppp0 -m udp -p udp
# iptables -A FORWARD -o ppp0 -m udp -p udp
# iptables -A FORWARD -i ppp0 -m icmp -p icmp
# iptables -A FORWARD -o ppp0 -m icmp -p icmp
```

Při zavedení těchto pravidel budou vyhodnocovány všechny datagramy přicházející PPP linkou a budou se zvlášť počítat statistiky pro TCP, UDP a ICMP datagramy. V programu **iptables** sledujeme příchozí a odchozí provoz samostatně, protože program neumí pracovat s obousměrnými pravidly.

Vyhodnocení výsledků účtování

Je sice hezké, že dokážeme statistiky přenosů zjišťovat, jak se na ně ale podíváme? Zajímají-li nás nashromážděná data a odpovídající pravidla, použijeme příkaz pro vypsání pravidel firewallu. Ve výstupu u každého pravidla jsou uvedeny hodnoty počitadel paketů a bajtů.

V příkazech **ipfwadm**, **ipchains** a **iptables** se práce s účtovacími daty liší, takže o nich budeme hovořit samostatně.

Výpis dat programem ipfwadm

Nejjednodušší způsob zjištění účtovacích dat programem **ipfwadm** bude vypadat takto:

```
# ipfwadm -A -l
IP accounting rules
pkts  bytes dir prot source              destination          ports
 9833 2345K i/o all  172.16.3.0/24       anywhere             n/a
56527   33M i/o all  172.16.4.0/24       anywhere             n/a
```

Takto se dozvíme počet paketů odeslaných v jednotlivých směrech. Pokud použijeme rozšířený výpis pomocí parametru `-e` (neuvádíme jej zde, protože se nevejde na šířku stránky), uvidíme ta-

ké zadané parametry a názvy rozhraní. Význam většiny údajů ve výstupu by měl být zřejmý, nicméně následující údaje si možná zaslouží vysvětlení:

dir	Směr platnosti pravidla. Možné hodnoty jsou `in`, `out` a `both`.
prot	Protokol, pro nějž pravidlo platí.
opt	Parametry, které jsme zadali v pravidlu.
ifname	Název rozhraní, pro nějž pravidlo platí.
ifaddress	Adresa rozhraní, pro níž pravidlo platí.

Program **ipfwadm** vypisuje standardně počitadla ve zkráceném tvaru, zaokrouhleně na nejbližší tisíce (K) nebo milióny (M). Přesné hodnoty statistik můžeme zjistit pomocí následujících parametrů:

```
# ipfwadm -A -l -e -x
```

Výpis dat programem ipchains

Program **ipchains** nevypisuje účtovací dat, pokud si o ně neřekneme parametrem `-v`. Nejjednodušší způsob vypsání těchto dat proto vypadá

```
# ipchains -L -v
```

Stejně jako v programu **ipfwadm** i zde můžeme přepínačem `-x` požádat o rozšířený výpis, kdy budou hodnoty uvedeny v příslušných jednotkách:

```
# ipchains -L -v -x
```

Výpis dat programem iptables

Program **iptables** se chová velmi podobně jako program **ipchains**. I zde musíme uvést parametr `-v` aby došlo k zobrazení účtovacích dat:

```
# iptables -L -v
```

Rovněž zde můžeme použít přepínač `-x` pro zobrazení rozšířeného výpisu.

Nulování počitadel

Pokud necháte účtování běžet příliš dlouho, počitadla přetečou. Dojde-li k přetečení, těžko zjistíte, jaké hodnoty vlastně byly napočítány. Abyste těmto problémům předešli, měli byste data číst pravidelně, zaznamenávat je a nulovat počitadla, takže budete získávat data periodicky pro nastavené intervaly.

V programech **ipfwadm** a **ipchains** můžete nulování zajistit velmi jednoduše:

```
# ipfwadm -A -z
```

nebo

```
# ipchains -Z
```

nebo

```
# iptables -Z
```

Můžete dokonce zkombinovat vypsání dat a vynulování počitadel do jednoho příkazu, takže budete mít zajištěno, že nedojde k žádné ztrátě informací mezi těmito dvěma operacemi:

```
# ipfwadm -A -l -z
```

nebo

```
# ipchains -L -Z
```

nebo

```
# iptables -L -Z -v
```

Tyto příkazy vypíší statistiky počitadel a ihned počitadla vynulují. Pokud budete chtít data shromažďovat pravidelně, uvedete zřejmě tyto příkazy ve skriptu, který výstup zaznamená a někam uloží, přičemž skript budete spouštět periodicky pomocí příkazu **cron**.

Vymazání pravidel

Poslední možný užitečný příkaz umožňuje zrušit všechna nastavená účtovací pravidla. Je to užitečné zejména pokud budete chtít zásadně změnit způsob účtování, aniž byste museli restartovat účtovací počítač.

Parametr `-f` příkazu **ipfwadm** vymaže všechna pravidla zadaného typu. Program **ipchains** má přepínač `-F`, který dělá to samé:

```
# ipfwadm -A -f
```

nebo

```
# ipchains -F
```

nebo

```
# iptables -F
```

Těmito příkazy se odstraní všechna nadefinovaná účtovací pravidla, takže je nemusíte odstraňovat jedno po druhém. Vymazání pravidel v programu **ipchains** nezpůsobí odstranění uživatelem definovaných tříd, budou pouze vymazána pravidla v těchto třídách.

Pasivní shromažďování účtovacích dat

Poslední zajímavý trik: Pokud máte počítač připojený k Ethernetu, můžete pomocí účtovacích pravidel zaznamenávat všechna data na daném segmentu, ne jenom data vysílaná a přijímaná účtovacím počítačem. Počítač bude pasivně přijímat všechna data na segmentu a zaznamená je.

Nejprve musíte na počítači vypnout předávání IP datagramů, aby se nepokoušel směrovat všechna data, která přijme[88]. V jádrech 2.0.36 a 2.2 se to provede příkazem

```
# echo 0 >/proc/sys/net/ipv4/ip_forward
```

88 Nemůžete to samozřejmě udělat, pokud počítač pracuje jako směrovač. Vypnete-li v takovém případě předávání datagramů, počítač přestane směrovat. Uvedený postup se hodí pouze pro systém s jediným fyzickým rozhraním.

Pak příkazem **ifconfig** přepnete ethernetové rozhraní do promiskuitního režimu. Nyní můžete zavést účtovací pravidla zaznamenávající provoz na celém segmentu, aniž by se přímo týkala účtovacího počítače.

KAPITOLA 11

IP maškaráda a překlad síťových adres

Nemusíte mít nijak dobrou paměť, abyste si pamatovali doby, kdy si propojení více počítačů v síti mohly dovolit pouze velké organizace. Současné síťové technologie jsou natolik levné, že došlo hned ke dvěma věcem. Za prvé jsou lokální sítě dnes zcela běžné, dokonce i u domácích uživatelů. Řada uživatelů Linuxu používá dva nebo více počítačů propojených nějakým Ethernetem. Za druhé jsou síťové prostředky, zejména IP adresy, velice vzácným artiklem a zatímco kdysi se přidělovaly zdarma, dnes se běžně kupují a prodávají.

Většina uživatelů s lokální sítí a připojením k Internetu bude zřejmě chtít, aby byl Internet přístupný ze všech počítačů sítě. Směrovací pravidla protokolu IP jsou ovšem velice přísná. Tradiční řešení by vyžadovalo získat celou síť IP adres, pravděpodobně síť třídy C, pak jednotlivým počítačům na síti přidělit adresy z této oblasti a konečně celou síť nějakým směrovačem připojit k Internetu.

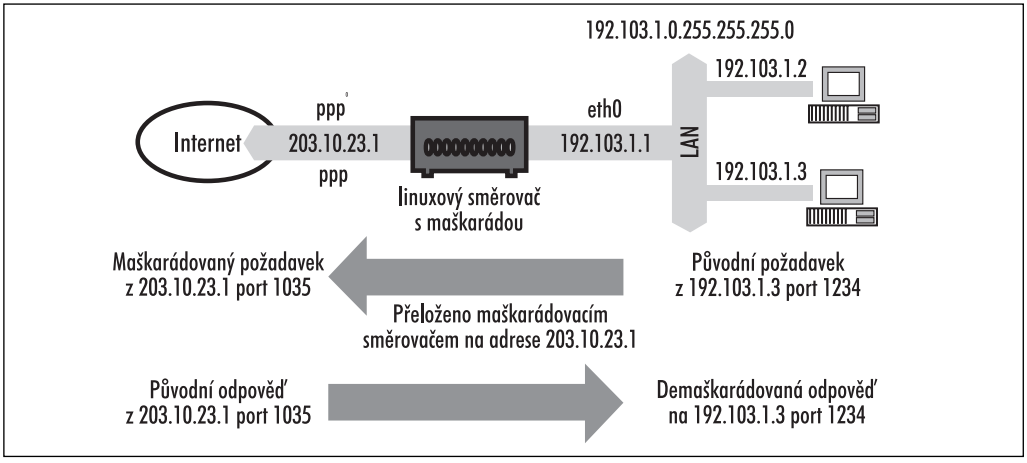
V komerčním prostředí Internetu je ale takové řešení nesmírně drahé. Budete muset platit za přidělení sítě adres. Dále budete zřejmě platit poskytovateli připojení za nastavení tras na vaši síť tak, aby byla z Internetu dostupná. Pro velké společnosti může být toto řešení dostupné, u domácích instalací se však zřejmě finančně nevyplatí.

Naštěstí Linux nabízí řešení tohoto problému. Toto řešení je tvořeno skupinou síťových funkcí, označovaných jako *Překlad síťových adres* (Network Address Translation, NAT). NAT je proces, který mění síťové adresy v hlavičkách datagramů v době jejich přenosu. Na první pohled to zní zvláštně, za chvíli však uvidíme, že tím vyřešíme před chvílí zmiňovaný problém. IP maškaráda je jedna z technologií NAT, která umožňuje všem hostitelům na síti přistupovat k Internetu prostřednictvím jediné IP adresy.

IP maškaráda dovoluje, aby všechny počítače na síti používaly privátní (rezervované) IP adresy a linuxový směrovač bude provádět inteligentní překlady adres a portů v čase přenosu. Když přijme datagram z počítače v lokální síti, zjistí, o jaký typ datagramu jde (tedy TCP, UDP, IMCP a podobně) a změní datagram tak, aby to vypadalo, že jej odeslal samotný směrovač (a zároveň si zapamatuje, že to udělal). Pak pošle datagram na Internet prostřednictvím své jediné platné IP adresy. Když konečný adresát tento datagram přijme, domnívá se, že pochází přímo od směrovače a pošle mu odpověď. Když směrovač implementující maškarádu takovýto datagram přijme, podívá se do tabulek maškarádovaných spojení a zjistí, zda daný datagram patří nějakému počítači v lokální síti a po-

kud ano, opět provede úpravu hlaviček datagramu na původní hodnoty a odešle datagram na lokální síť.

Jednoduchý příklad je znázorněn na obrázku 11.1.



Obrázek 11.1 – Typická konfigurace maškarády

Máme malou ethernetovou síť používající jednu z rezervovaných síťových adres. Síť používá k přístupu na Internet linuxový směrovač s maškarádou. Jedna ze stanic na naší síti (192.168.1.3) chce navázat spojení se vzdáleným hostitelem 209.1.106.178 na portu 8888. Stanice směruje svůj požadavek na směrovač s maškarádou, který jej vyhodnotí jako požadavek, jenž je nutné ošetřit maškarádou. Přijme datagram, alokuje nějaký volný port (1035), uvede svou vlastní IP adresu a port z původního požadavku a pošle datagram cílovému hostiteli. Ten se domnívá, že přijal požadavek přímo od maškarádovacího počítače a vygeneruje odpověď. Maškarádovací počítač po přijetí odpovědi najde v tabulce maškarádových spojení příslušnou asociaci a provede opačnou substituci než u původního požadavku. Odpověď pak pošle původnímu odesílateli.

Lokální počítač se domnívá, že se baví přímo se vzdáleným hostitelem. Vzdálený hostitel pro změnu o lokálním hostiteli vůbec nic neví a domnívá se, že se baví přímo s maškarádovacím hostitelem. Pouze maškarádovací hostitel ví, kdo se s kým doopravdy baví a na jakých portech a zajišťuje překlady adres a portů potřebné pro průběh spojení.

Může to sice vypadat trošku komplikovaně a možná to i tak je, nicméně to funguje a celá služba se velmi snadno konfiguruje. Nelamte si proto hlavu s tím, pokud vám nějaké detaily unikají.

Vedlejší efekty, výhody a nevýhody

IP maškaráda s sebou přináší různé vedlejší efekty, z nichž některé jsou užitečné a jiné mohou být naopak nepříjemné.

Žádný z hostitelů na interní síti za maškarádovacím počítačem není přímo viditelný, tím pádem vám stačí pouze jediná platná a směrovatelná IP adresa a pomocí ní mohou s Internetem komunikovat všechny počítače na interní síti. Má to i nevýhodu – žádný z hostitelů na lokální síti není z Internetu viditelný a není tak možné se s nimi z Internetu spojit. Jediným viditelným počítačem na maškarádované síti je samotný počítač maškarády. To je významné při provozu služeb jako je

pošta nebo FTP. Musíte rozhodnout, které služby bude zajišťovat přímo maškarádovací počítač a které služby bude ošetřovat prostřednictvím proxy serveru nebo jiným speciálním způsobem.

Dále, protože žádný z maškarádovaných hostitelů není viditelný, jsou relativně chráněny před útoky zvenčí – tím se zjednoduší případně úplně vyloučí nutnost konfigurace firewallu. Nicméně na to nelze příliš spoléhat. Celá síť bude zabezpečena jenom tak, jak je zabezpečen počítač maškarády, takže pokud vám na bezpečnosti záleží, i tak byste měli použít firewall.

Dále může mít maškaráda určitý dopad na výkon sítě. U typické maškarády to bude pravděpodobně stěží postřehnutelné. Pokud pracujete s velkým počtem aktivních maškarádovaných spojení, můžete zjistit, že zpracování datagramů na maškarádovacím počítači má dopad na propustnost sítě. V porovnání s klasickým směrováním musí maškaráda provést s každým jedním datagramem poměrně velký počet operací. Počítač 386SX16 pro maškarádu na telefonní lince bude zřejmě stačit, ovšem pokud jej budete chtít použít jako hlavní směrovač na firemní síti připojené vysokorychlostní linkou, zřejmě bude jeho výkon nedostatečný.

A konečně, některé síťové služby prostě nebudou s maškarádou fungovat, alespoň ne bez určité pomoci. Typicky se jedná o služby, které závisejí na navazování příchozích spojení, například některé typy přímých komunikačních kanálů, některé funkce v IRC nebo jisté typy video a audio služeb. Pro některé z těchto služeb existují speciální moduly jádra, které jejich provoz umožňují – budeme o nich za chvíli mluvit. U jiných zjistíte, že žádná podpora neexistuje, takže musíte být opatrní. Maškarádu nelze použít úplně vždy.

Konfigurace jádra pro maškarádu

Abyste mohli maškarádu použít, musíte mít jádro přeloženo s její podporou. Při konfiguraci jádra 2.2 musíte zvolit následující volby:

```
Networking options  --->
        [*] Network firewalls
        [*] TCP/IP networking
        [*] IP: firewalling
        [*] IP: masquerading
        --- Protocol-specific masquerading support will be built as modules.
        [*] IP: ipautofw masq support
        [*] IP: ICMP masquerading
```

Některé typy maškarády jsou dostupné pouze jako moduly jádra. Znamená to, že při sestavování jádra musíte kromě klasického „`make zImage`“ zadat i příkaz „`make modules`“.

Jádra série 2.4 neposkytují podporu maškarády jako parametr při sestavování jádra. Místo toho musíte zvolit podporu filtrace paketů:

```
Networking options  --->
    [M] Network packet filtering (replaces ipchains)
```

V jádrech série 2.2 existuje řada pomocných modulů, které se vytvářejí při sestavování jádra. Některé protokoly začínají komunikaci odchozím spojením na určitém portu a pak očekávají příchozí spojení na jiném. Za normálních okolností nelze takové protokoly pod maškarádou použít, protože neexistuje metoda, jak bez podrobné analýzy samotného obsahu datagramů asociovat druhý požadavek s prvním. Přesně to dělají podpůrné moduly – prohlédnou si tělo datagramu a umožňují maškarádu i pro protokoly, u nichž by jinak nebyla možná. Podporovány jsou následující protokoly:

Modul	Protokol
ip_masq_ftp	FTP
ip_masq_irc	IRC
ip_masq_raudio	RealAudio
ip_masq_cuseeme	CU-See-Me
ip_masq_vdolive	VDO Live
ip_masq_quake	IdSoftware Quake

K zavedení podpory zmíněných protokolů musíte uvedené moduly nahrát ručně příkazem **insmod**. Tyto moduly nelze nahrát démonem **kerneld**. Každý z modulů používá jako parametr číslo portu, na kterém má poslouchat. Pro modul protokolu RealAudio™ můžete zadat[89]:

```
# insmod ip_masq_raudio.o ports=7070,7071,7072
```

Čísla zadávaných portů závisejí na konkrétním protokolu. Další podrobnosti o modulech maškarády a o jejich použití nabízí dokument IP masquerade mini-HOWTO Ambrose Au[90].

Balík *netfilter* obsahuje moduly, které plní podobnou funkci. Pokud budete například chtít zajistit podporu spojení FTP relací, použijete moduly `ip_conntrack_ftp` a `ip_nat_ftp.o`.

Konfigurace maškarády

Pokud jste četli kapitoly o firewallech a účtování, zřejmě vás nepřekvapí, že i maškaráda se nastavuje příkazy **ipfwadm**, **ipchains** a **iptables**.

Pravidla maškarády představují velmi speciální třídu filtračních pravidel. Maškarádě mohou podléhat pouze datagramy přicházející na jednom rozhraní a odcházející jiným rozhraním. Pravidla maškarády se velmi podobají předávacím pravidlům firewallu, obsahují však speciální parametr, který jádru říká, že datagram má projít maškarádou. Příkaz **ipfwadm** používá parametr `-m`, **ipchains** pak `-j MASQ` a **iptables** `-j MASQUERADE`, která definují, že na datagram má být uplatněna maškaráda.

Podívejme se na příklad. Student katedry počítačů na univerzitě Groucho Marx má doma několik počítačů propojených ethernetovou sítí. Pro tuto síť používá IP adresy z jedné z rezervovaných oblastí. Bydlí společně s dalšími studenty a všichni chtějí používat Internet. Protože studenti nikdy nebyli při penězích, nemohou si dovolit pevné připojení k Internetu a používají místo toho vytáčenou PPP linku. Všichni chtějí na svých počítačích používat služby jako IRC, WWW a FTP – řešením je maškaráda.

Nejprve musí nakonfigurovat linuxový počítač pro podporu vytáčené linky a jako směrovač pro lokální síť. IP adresa, kterou počítač dostane v okamžiku připojení, není důležitá. Směrovač je nastaven s maškarádou a pro lokální síť používá IP adresy z rezervované oblasti 192.168.1.0. Dále je nutné zajistit, aby všechny počítače na lokální síti měly nastavenu implicitní trasu směřující na tento směrovač.

Následující příkazy **ipfwadm** stačí k zavedení maškarády v této konfiguraci sítě:

89 RealAudio je ochranná známka společnosti Progressive Networks Corporation.

90 Ambrose můžete kontaktovat na adrese *ambrose@writeme.com*.

```
# ipfwadm -F -p deny
# ipfwadm -F -a accept -m -S 192.168.1.0/24 -D 0/0
```

nebo s **ipchains**

```
# ipchains -P forward -j deny
# ipchains -A forward -s 192.168.1.0/24 -d 0/0 -j MASQ
```

anebo s **iptables**

```
# iptables -t nat -P POSTROUTING DROP
# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Kdykoliv se nyní některý z počítačů na lokální síti pokusí o spojení se službou na nějakém vzdáleném systému, odesílané datagramy budou zpracovány maškarádou. První pravidlo vždy zakazuje přímé směrování jakýchkoliv jiných datagramů a poněkud tak zvyšuje bezpečnost konfigurace.

K vypsání vytvořených pravidel maškarády můžete použít parametr `-l` příkazu **ipfwadm**, o kterém jsme hovořili v kapitole věnované firewallům.

K vypsání výše vytvořených pravidel bychom mohli použít příkaz

```
# ipfwadm -F -l -e
```

Měli bychom dostat asi takovýto výsledek:

```
# ipfwadm -F -l -e
IP firewall forward rules, default policy: deny
 pkts bytes type  prot opt  tosa tosx ifname  ifaddress  …
    0     0 acc/m all  ---- 0xFF 0x00 any     any        …
```

Parametr „`/m`“ ve výpisu říká, že se jedná o pravidlo maškarády.

K výpisu pravidel maškarády v programu **ipchains** použijete parametr `-L`. Při zadání výše vytvořených pravidel dostaneme něco takového:

```
# ipchains -L
Chain input (policy ACCEPT):
Chain forward (policy DENY):
target     prot opt     source               destination          ports
MASQ       all  ------  192.168.1.0/24       anywhere             n/a

Chain output (policy ACCEPT):
```

Jakákoliv pravidla vztahující se k maškarádě jsou uvedena s operací `MASQ`.

Konečně s příkazem **iptables** použijeme:

```
# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy DROP)
target     prot opt source               destination
MASQUERADE  all  --  anywhere             anywhere           MASQUERADE

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

I zde jsou pravidla vztahující se k maškarádě uvedena operací `MASQUERADE`.

Nastavení časovacích parametrů IP maškarády

Při navázání každého spojení si software implementující maškarádu vytvoří v paměti asociaci mezi počítači, pro něž bylo spojení vytvořeno. Tyto asociace můžete v kterémkoliv okamžiku vidět v souboru `/proc/net/ip_masquerade`. Po určité době nečinnosti budou asociace zrušeny.

Časový limit asociace je možné nastavit příkazem **ipfwadm**. Obecná syntaxe je

```
ipfwadm -M -s <tcp> <tcpfin> <udp>
```

S příkazem **ipchains** je syntaxe

```
ipchains -M -S <tcp> <tcpfin> <udp>
```

Implementace v programu **iptables** používá podstatně delší časové limity a nedovoluje je změnit.

Jednotlivé hodnoty jsou časovače s nastavením v sekundách. Jejich význam uvádí následující tabulka:

Parametr	Popis
`tcp`	Timeout TCP relace. Jak dlouho může být TCP spojení neaktivní, než dojde k odstranění asociace.
`tcpfin`	Timeout TCP po příznaku FIN. Jak dlouho zůstane asociace zachována po ukončení TCP spojení.
`udp`	Timeout UDP relace. Jak dlouho může být UDP spojení neaktivní, než dojde k odstranění asociace.

Obsluha dotazů na jmenné servery

Obsluha dotazů na jmenné servery od počítačů na síti s maškarádou je vždy problematická. V tomto prostředí přicházejí v úvahu dvě řešení. Jedna možnost je nakonfigurovat všechny počítače tak, aby používaly stejný DNS server jako počítač s maškarádou a pak nechat maškarádu, ať se postará o zbytek. Druhá možnost je na nějakém počítači rozběhnout jmenný server v konfiguraci caching-only a nechat všechny stanice používat tento jmenný server. I když je to poněkud složitější řešení, bude pravděpodobně výhodnější, protože se tak redukuje DNS provoz na internetové lince a pro většinu dotazů přijde odpověď rychleji, protože budou obslouženy z vyrovnávací paměti lokálního serveru. Nevýhodou je pouze to, že celá konfigurace je poněkud složitější. Konfiguraci jmenného serveru popisujeme v kapitole 6, v části *Konfigurace typu „caching-only“*.

Další informace o překladu síťových adres

Software *netfilter* umožňuje řadu různých řešení překladu síťových adres, IP maškaráda je pouze jedním z nich.

Je například možné vytvořit taková pravidla překladu, že se budou překládat pouze určité adresy nebo rozsahy adres a ostatní zůstanou beze změny, nebo překládat všechny adresy na skupinu adres a ne pouze na jedinou tak, jak to dělá maškaráda. Pomocí příkazu **iptables** můžete vytvořit překladová pravidla, která budou mapovat cokoliv na cokoliv, s využitím kombinací všech standardních atributů, například zdrojových adres, cílových adres, typů protokolů, čísel portů a podobně.

O překladu zdrojové adresy datagramu se v dokumentaci k balíku *netfilter* hovoří jako o „Source NAT“ nebo `SNAT`. Překlad cílové adresy se označuje jako „Destination NAT“, `DNAT`. Překlady TCP a UDP portů se označují termínem `REDIRECT`. Výrazy `SNAT`, `DNAT` a `REDIRECT` představují operace, které můžete v pravidlech programu **iptables** definovat příznakem `-j` a vytvářet tak různá složitá a kombinovaná pravidla.

Téma překladu adres a jeho použití by dokázalo zabrat celou samostatnou kapitolu[91]. Bohužel zde nemáme prostor, abychom se tomuto tématu mohli věnovat podrobněji. Další informace naleznete v dokumentu IPTABLES-HOWTO, kde se mimo jiné hovoří i o problematice překladu síťových adres.

91 Ne-li dokonce celou knihu.

Důležité síťové aplikace

Po úspěšném nastavení protokolu IP a resolveru musíte začít věnovat pozornost službám, které hodláte na síti poskytovat. Tato kapitola se zabývá konfiguracemi několika jednoduchých síťových aplikací, včetně serveru **inetd** a programů z rodiny **rlogin**. Krátce se také zmíníme o rozhraní Remote Procedure Call, na kterém jsou založeny služby Network File System (NFS) a Network Information System (NIS). Konfigurace systémů NFS a NIS je nicméně o dost složitější, takže o ní budeme hovořit v samostatných kapitolách, stejně jako o elektronické poště a o síťových news.

Samozřejmě v této knize nemůžeme probrat všechny síťové aplikace. Budete-li chtít nainstalovat nějakou z aplikací, která zde není popsána, například programy **talk**, **gopher** nebo **http**, můžete příslušné informace nalézt na odpovídajících manuálových stránkách.

Superserver inetd

Programy, které poskytují služby po síti, se označují jako *démony*. Démon je program, který otevře určitý port, typicky známý port určité služby, a čeká na příchozí spojení. Pokud k takovému spojení dojde, vytvoří proces potomka, jenž toto spojení obslouží a rodičovský proces bude stále pokračovat v odposlouchávání dalších požadavků. Toto řešení sice funguje dobře, má ale několik nevýhod. V paměti musí být trvale přítomna alespoň jedna instance každého démona pro každou provozovanou službu. Kromě toho se v každém démonu opakuje ta část kódu, která zajišťuje obsluhu a poslouchání na portu.

Řešením této neefektivity je na většině unixových systémů speciální síťový démon, který můžeme chápat jako superserver. Tento démon otevírá sockety pro řadu síťových služeb a na všech poslouchá. Když se na některém z portů objeví příchozí spojení, superserver je přijme a spustí server pro konkrétní port, kterému pak předá socket k obsluze. Superserver pak pokračuje v poslouchání na portech[92].

Nejběžnější superserver se jmenuje **inetd**, Internet Daemon. Je spouštěn při zavádění systému a seznam služeb, které má spravovat, získává z konfiguračního souboru `/etc/inetd.conf`. Kromě výše zmíněných volaných serverů existuje i řada triviálních služeb, které provádí démon **inetd** sám. Tyto služby se nazývají *interní služby*. Jednou z nich jsou služba **chargen**, která generuje řetězec znaků nebo služba **daytime**, která vrací systémový čas.

92 Pozn. překladatele: Toto řešení má však nevýhodu v režii spojené se spuštěním serveru služby démonem **inted**. Proto se u hodně zatížených služeb, kde se požaduje co nejrychlejší odezva, dává přednost tomu, že běží přímo server dané služby a nepoužívá se pro ni nepřímé spouštění superserverem. Typicky tak trvale běží démon **httpd**, server služby WWW, jehož spouštění prostřednictvím superdémona by bylo pomalé.

Záznamy v konfiguračním soubory jsou uloženy na samostatných řádcích ve formátu:

```
služba typ protokol čekání uživatel server příkazový_řádek
```

Jednotlivá pole mají následující význam:

služba — Určuje název služby. Název služby musí být převoditelný na číslo portu prostřednictvím souboru `/etc/services`. Tento soubor bude popsán později v části *Soubory services a protocols*.

typ — Určuje typ socketu. Může mít hodnotu stream (u protokolů s navazováním spojení) nebo dgram (pro datagramové služby). Služby založené na protokolu TCP proto vždy používají hodnotu stream, služby založené na protokolu UDP dgram.

protokol — Určuje název přenosového protokolu, který bude daná služba používat. Musí to být korektní název protokolu, který se nachází v souboru `protocols`, o kterém budeme hovořit později.

čekání — Tato volba se vztahuje pouze na sokety typu dgram. Může mít hodnotu wait nebo nowait. Je-li zadána volba wait, spustí démon **inetd** pro daný port v jednom okamžiku vždy pouze jeden server. V opačném případě bude po spuštění serveru okamžitě pokračovat v odposlouchávání portu. Tato nastavení se liší podle typu serveru. Jednovláknové servery, které přečtou příchozí datagramy, odpoví na ně a ukončí se by měly být spouštěny jako wait. Patří sem většina RPC služeb. Opačným typem jsou vícevláknové servery, které umožňují současný běh více instancí serveru. U těchto serverů se používá hodnota nowait. Sockety typu stream by měly vždy používat volbu nowait.

uživatel — Tato volba určuje identifikační číslo uživatele, pod kterým bude daný proces spouštěn. Tímto uživatelem je často uživatel **root**, avšak některé služby mohou používat i jiné účty. Zde je velmi vhodné uplatňovat princip nejnižších práv, což znamená, že byste neměli příkaz spouštět na účtu s vyššími právy, pokud je spouštěný program pro svou správnou funkci nevyžaduje. Například server news NNTP běží s právy uživatele **news**, zatímco služby, které by mohly představovat bezpečnostní rizika (jako je služba **tftp** nebo **finger**) jsou často spouštěny s právy uživatele **nobody**.

server — Určuje název plné cesty ke spouštěnému programu serveru. Vnitřní služby jsou označeny klíčovým slovem internal.

příkazový_řádek — Tato volba určuje příkazovou řádku, která bude předána danému serveru. Začíná názvem spouštěného serveru a za ním mohou následovat předávané parametry. Pokud chcete použít TCP wrapper, musíte zde specifikovat úplnou cestu k serveru. Jinak zadáváte pouze název serveru tak jak se objevuje v seznamu procesů. O TCP wrapperech budeme zanedlouho hovořit. Toto pole je u interních služeb prázdné.

Vzorový soubor `/etc/inetd.conf` je uveden v příkladu 12.1. Služba **finger** je zakomentována a není tedy dostupná. Tato služba se často z bezpečnostních důvodů vypíná, protože útočníkům umožňuje zjistit uživatelská jména a další informace o uživatelích vašeho systému.

Příklad 12.1 – Příklad souboru /etc/inetd.conf

```
#
# služby démona inetd
ftp      stream tcp nowait root  /usr/sbin/ftpd    in.ftpd -l
telnet   stream tcp nowait root  /usr/sbin/telnetd in.telnetd -b/etc/issue
#finger  stream tcp nowait bin   /usr/sbin/fingerd in.fingerd
#tftp    dgram  udp wait  nobody /usr/sbin/tftpd   in.tftpd
#tftp    dgram  udp wait  nobody /usr/sbin/tftpd   in.tftpd /boot/diskless
#login   stream tcp nowait root  /usr/sbin/rlogind in.rlogind
#shell   stream tcp nowait root  /usr/sbin/rshd    in.rshd
#exec    stream tcp nowait root  /usr/sbin/rexecd  in.rexecd
#
# interní služby
#
daytime  stream tcp nowait root internal
daytime  dgram  udp nowait root internal
time     stream tcp nowait root internal
time     dgram  udp nowait root internal
echo     stream tcp nowait root internal
echo     dgram  udp nowait root internal
discard  stream tcp nowait root internal
discard  dgram  udp nowait root internal
chargen  stream tcp nowait root internal
chargen  dgram  udp nowait root internal
```

Služba **tftp** je rovněž zakomentována. Tato služba implementuje protokol *Trivial File Transfer Protocol*, který umožňuje z vašeho systému přenést libovolný veřejně čitelný soubor, aniž by byla prováděna jakákoliv kontrola pomocí hesla a podobně. To je nepříjemné zejména u souboru `/etc/passwd` zejména v případě, že nepoužíváte stínový soubor hesel.

Protokol TFTP obecně používají bezdiskoví klienti a X terminály ke stahování startovacího kódu ze zaváděcích serverů. Pokud potřebujete spouštět službu **tftpd** z tohoto důvodu, ujistěte se, že jste omezili její rozsah pouze na ty adresáře, ze kterých budou klienti získávat soubory. To provedete tak, že na příkazovém řádku démona **tftpd** uvedete dostupné adresáře. Vidíme to na druhém řádku výpisu.

Řízení přístupu wrapperem tcpd

Protože přístup k počítačům prostřednictvím sítě přináší mnoho bezpečnostních rizik, síťové aplikace jsou proti různým typům útoků odolné. Některé aplikace ovšem mohou obsahovat chyby (nejdrastičtěji to demonstroval internetový červ RTM) nebo nemusí umět rozlišovat mezi bezpečnými hostiteli, od kterých budou přijímány požadavky na konkrétní službu, a nebezpečnými hostiteli, jejichž požadavky by měly zamítnout. Již jsme se krátce zmínili o službách **finger** a **tftp**. Tyto služby budete typicky chtít povolit pouze „důvěryhodným hostitelům“, což ale nelze pomocí standardního nastavení, při kterém superserver **inetd** poskytuje konkrétní služby buď všem klientům, nebo nikomu.

Pro řízení přístupu podle hostitele se často používá démon **tcpd**, takzvaný *wrapper* (česky asi „obal“)[93]. Tento démon se volá u chráněných nebo sledovaných služeb namísto příslušného serve-

93 Napsal jej Wietse Venema, *wietse@wzv.win.tue.nl*.

ru služby. Démon zkontroluje, zda požadavek přichází od oprávněného hostitele a pouze v takovém případě pak spustí skutečný server služby. Tento postup nelze použít u služeb založených na protokolu UDP.

Chcete-li například „obalit" démona **finger**, musíte v souboru `inetd.conf` změnit odpovídající řádek následujícím způsobem Z:

```
# démon finger bez ochrany
finger    stream tcp nowait bin    /usr/sbin/fingerd in.fingerd
```

na:

```
# démon finger s ochranou
finger  stream  tcp     nowait  root    /usr/sbin/tcpd   in.fingerd
```

Pokud nepřidáte žádné řízení přístupu, bude se tato úprava klientovi jevit stejně, jako obvyklé nastavení služby **finger** s tou výjimkou, že veškeré požadavky budou zapisovány službou **syslog**.

Řízení přístupu je implementováno za pomoci souborů `/etc/hosts.allow` a `/etc/hosts.deny`. Tyto soubory obsahují položky, které povolují a zakazují přístup k určitým službám podle žádajícího hostitele. Když nástroj **tcpd** vyřizuje požadavek na službu **finger** od klienta **biff.foobar.com**, hledá v souborech `hosts.allow` a `hosts.deny` (v tomto pořadí) položku odpovídající jak požadované službě, tak i hostiteli klienta. Pokud je odpovídající položka nalezena v souboru `hosts.allow`, bude přístup povolen a soubor `hosts.deny` se už neprohlíží. Pokud bude odpovídající položka nalezena v souboru `hosts.deny`, bude požadavek odmítnut a spojení se ukončí. Jestliže se ani v jednom souboru odpovídající položka nenajde, bude požadavek přijat.

Položky v souborech pro řízení přístupu vypadají následovně:

seznam_služeb: *seznam_hostitelů* [:*příkazy_shellu*]

Pole *seznam_služeb* obsahuje seznam názvů služeb podle souboru `/etc/services` nebo klíčové slovo `ALL`. Chcete-li zadat všechny služby kromě služeb **finger** a **tftp**, můžete zapsat `ALL EXCEPT finger, tftp`.

Pole *seznam_hostitelů* obsahuje seznam názvů hostitelů nebo jejich IP adres, případně klíčová slova `ALL`, `LOCAL`, `UNKNOWN` nebo `PARANOID`. Klíčovému slovu `ALL` vyhovují všichni hostitelé, klíčovému slovu `LOCAL` vyhovují pouze ty názvy hostitelů, kteří neobsahují tečku[94], `UNKNOWN` znamená hostitele u nichž se nepodařil převod názvu nebo adresy a klíčovému slovu `PARANOID` hostitelé, u nichž nesouhlasí zpětný převod IP adresy na název[95]. Název začínající tečkou znamená všechny hostitele, jejichž doména je shodná s uvedeným názvem. Například názvu **.foobar.com** bude vyhovovat třeba hostitel **biff.foobar.com**, ne však hostitel **nurks.fredsville.com**. Zadání končící tečkou znamená všechy hostitele, kteří začínají danou IP adresou, takže například zadání **172.16.** bude vyhovovat hostitel 172.16.32.1, ne však hostitel 172.15.9.1. Hodnota ve tvaru *n.n.n.n*/*m.m.m.m* je chápána jako IP adresa se síťovou maskou, takže předchozí příklad bychom mohli definovat také jako 172.16.0.0/255.255.0.0. Konečně údaj začínající lomítkem specifikuje název souboru, v němž jsou uvedena jména nebo adresy vyhovujících hostitelů. Takže údaj */var/access/trustedhosts* způsobí, že program **tcpd** načte daný soubor a prohlédne jej, zda některý ze záznamů v souboru nevyhovuje právě připojenému hostiteli.

---

94 Přičemž název bez tečky mají typicky pouze lokální hostitelé uvedení v souboru `/etc/hosts`.

95 I když samo klíčové slovo implikuje poněkud extrémní význam, volba PARANOID je vhodná standardní volba, protože chrání před zákeřnými hostiteli, kteří se vydávají za někoho jiného. Ne všechny verze programu **tcpd** tuto volbu podporují, pokud ji vaše verze neumí, musíte si přeložit novější verzi.

Chcete-li zakázat přístup ke službám **finger** a **tftp** všem hostitelům s výjimkou lokálních hostitelů, vložte do souboru `/etc/hosts.deny` následující řádek a soubor `/etc/hosts.allow` nechte prázdný:

```
in.tftpd, in.fingerd:  ALL EXCEPT LOCAL, .vaše.doména
```

Nepovinný údaj *příkazy_shellu* může obsahovat příkaz, jenž bude vykonán při splnění dané položky. Je to užitečné k nastavení „pastí“, které mohou odhalit potenciální útočníky:

```
in.ftpd: ALL EXCEPT LOCAL, .vbrew.com : \
      echo "request from %d@%h: >> /var/log/finger.log; \
      if [ %h != "vlager.vbrew.com:" ]; then \
          finger -l @%h >> /var/log/finger.log \
      fi
```

Nástroj tcpd nahradí parametry `%h` a `%d` skutečným názvem klienta a skutečným názvem služby. Další podrobnosti naleznete na manuálových stránkách `hosts_access(5)`.

Soubory services a protocols

Čísla portů, na kterých jsou nabízeny „standardní“ služby, jsou definována v RFC Assigned Numbers. Aby mohly servery nebo klienti převádět názvy služeb na tato čísla, musí být alespoň část z tohoto seznamu uložena na každém hostiteli; ukládá se v souboru `/etc/services`. Položky v souboru mají následující syntaxi:

```
služba port/protokol [aliasy]
```

Pole *služba* definuje název služby, *port* definuje port, na kterém je služba nabízena, a pole *protokol* definuje typ používaného transportního protokolu. Tento údaj je obecně buď `tcp` nebo `udp`. Stejnou službu je možné nabízet oběma protokoly, naopak lze na stejném portu různých protokolů nabízet různé služby. Pole *aliasy* umožňuje zadat alternativní názvy stejné služby.

Soubor `services`, který je dodáván společně se síťovým softwarem, nebudete muset obvykle měnit. Přesto zde uvádíme malý výpis z tohoto souboru:

Příklad 12.2 – Příklad souboru /etc/services

```
# Soubor služeb:
#
# známé služby
echo            7/tcp                   # Echo
echo            7/udp                   #
discard         9/tcp   sink null       # Discard
discard         9/udp   sink null       #
daytime        13/tcp                   # Daytime
daytime        13/udp                   #
chargen        19/tcp   ttytst source   # Character Generator
chargen        19/udp   ttytst source   #
ftp-data       20/tcp                   # File Transfer Protocol (Data)
ftp            21/tcp                   # File Transfer Protocol (Control)
telnet         23/tcp                   # Virtual Terminal Protocol
smtp           25/tcp                   # Simple Mail Transfer Protocol
nntp          119/tcp   readnews        # Network News Transfer Protocol
```

```
#
# unixové služby
exec          512/tcp                  # BSD rexecd
biff          512/udp  comsat          # notifikace pošty
login         513/tcp                  # vzdálený login
who           513/udp  whod            # vzdálené who a uptime
shell         514/tcp  cmd             # vzdálený příkaz, bez hesla
syslog        514/udp                  # vzdálené logování
printer       515/tcp  spooler         # vzdálený tiskový spooling
route         520/udp  router routed   # směrovací protokol RIP
```

Všimněte si, že například služba **echo** je poskytována na portu 7 jak protokolem TCP, tak i protokolem UDP, zatímco port 512 používají dvě odlišné služby: vzdálené spouštění (**rexec**) pomocí protokolu TCP a démon **COMSAT** (zajišťující oznamování došlé pošty, viz *xbiff(1x)*) protokolem UDP.

Podobně jako u názvů služeb potřebuje síťová knihovna i nějaký způsob, jakým by mohla přeložit názvy protokolů – například těch protokolů, které jsou použity v souboru services – na čísla protokolů, kterým by rozuměly IP vrstvy ostatních hostitelů. To se provede vyhledáním daného názvu v souboru `/etc/protocols`. Ten obsahuje na každém řádku jednu položku obsahující název protokolu a přidělené číslo. Provádění změn v tomto souboru je ještě méně pravděpodobné, než zasahování do souboru `/etc/services`. Příklad tohoto souboru vidíte na výpisu 12.3.

Příklad 12.3 – Příklad souboru /etc/protocols

```
#
# Internetové protokoly (IP)
#
ip      0       IP              # internet protocol, pseudo protocol number
icmp    1       ICMP            # internet control message protocol
igmp    2       IGMP            # internet group multicast protocol
tcp     6       TCP             # transmission control protocol
udp     17      UDP             # user datagram protocol
raw     255     RAW             # RAW IP interface
```

Vzdálené volání procedur

Balík RPC (*Remote Procedure Call*) poskytuje obecný mechanismus pro aplikace typu klient-server. Balík RPC vyvinula firma Sun Microsystems a jde o sbírku nástrojů a knihoven funkcí. Důležitou aplikací postavenou na balíku RPC je systém NIS (Network Information System, viz kapitola 13) a NFS (Network File System, viz kapitola 14).

Server RPC se skládá ze sady procedur, které si může klient volat tím způsobem, že pošle serveru RPC požadavek společně s parametry procedury. Server spustí místo klienta požadovanou proceduru a existuje-li návratová hodnota, pošle ji zpět klientovi. Aby mohl být balík RPC strojově nezávislý, všechna data předávaná mezi klientem a serverem se konvertují do formátu *External Data Representation* (XDR). K přenosu dat ve formátu XDR používá knihovna RPC standardní TCP

a UDP sockety. Firma Sun uvolnila balík RPC jako Public Domain, celý balík je popsán v kolekci několika RFC dokumentů.

Někdy způsobí zdokonalení RPC aplikace nekompatibilní změny v rozhraní volání procedur. Samozřejmě, že prostá výměna serveru může způsobit havárii všech aplikací, které stále očekávají původní chování. Proto mají programy RPC přiděleny číslo své verze, obvykle se začíná hodnotou 1 a při každé nové verzi rozhraní RPC je tato hodnota zvýšena. Server může často současně nabízet několik verzí RPC; klient ve svém požadavku specifikuje číslo verze, kterou chce použít.

Vlastní síťová komunikace probíhající mezi servery RPC a jejich klienty je zvláštní. Server RPC nabízí jednu nebo více skupin procedur; každá množina procedur se nazývá *program* a je jednoznačně určena takzvaným *číslem programu*. Seznam definující přiřazení názvů služeb k číslům programů je obvykle uložen v souboru `/etc/rpc`, jehož část ukazuje následující výpis.

Příklad 12.4 – Příklad souboru /etc/rpc

```
#
# /etc/rpc - různé služby založené na RPC
#
portmapper  100000    portmap sunrpc
rstatd      100001    rstat rstat_svc rup perfmeter
rusersd     100002    rusers
nfs         100003    nfsprog
ypserv      100004    ypprog
mountd      100005    mount showmount
ypbind      100007
walld       100008    rwall shutdown
yppasswss   100009    yppasswd
bootparam   100026
ypupdated   100028    ypupdate
```

U sítí na bázi protokolu TCP/IP čelili autoři balíku RPC problému, jak sdružit čísla programů s obecnými síťovými službami. Zvolili takovou variantu, kdy každý server poskytuje pro každý program a pro každou verzi programu jak port pro protokol TCP, tak i port pro protokol UDP. Obecně budou aplikace při posílání dat používat protokol UDP a protokol TCP budou používat pouze v případě, že se posílaná data nevejdou do jediného datagramu protokolu UDP.

Samozřejmě, že klienti musí mít možnost zjistit port, na který je namapováno dané číslo programu. V tomto případě by bylo použití konfiguračního souboru příliš nepružné, jelikož aplikace RPC nepoužívají vyhrazené porty, nemůžeme mít tedy žádnou jistotu, že port původně určený pro použití naší databázovou aplikací nebude obsazen nějakým jiným procesem. Proto aplikace RPC vyberou některý z dostupných portů a zaregistrují ho pomocí speciálního démona, takzvaného *mapovače portů* (*portmapper*). Tento démon se chová jako zprostředkovatel mezi všemi servery RPC, které jsou spuštěny na daném počítači: klient, který chce kontaktovat službu s daným číslem programu, se nejdříve bude dotazovat mapovače portů na hostiteli serveru, ten mu pak vrátí čísla portů TCP a UDP, na kterých je daná služba dosažitelná.

Tato metoda má konkrétní nevýhodu v tom, že zavádí jedno slabé místo celého systému, podobně jako to činí démon **inetd** u standardních Berkeley služeb. Avšak tento případ je ještě horší, protože když selže démon mapující porty, ztratí se veškeré informace o portech RPC; to obvykle způsobí, že budete muset manuálně znovu nastartovat všechny servery RPC nebo dokonce celý počítač.

V Linuxu se program na mapování portů nazývá `/sbin/portmap` nebo `/usr/sbin/rpc.portmap`. Kromě toho, že je třeba zajistit, aby se mapovač automaticky spouštěl při startu systému, není třeba žádná další konfigurace.

Konfigurace vzdáleného přihlašování a spouštění

Často je velmi užitečné spustit příkaz na vzdáleném počítači a vstupy a výstupy tohoto programu zapisovat nebo číst prostřednictvím síťového spojení.

Tradičně se pro spouštění programů na vzdáleném hostiteli používaly příkazy **rlogin**, **rsh** a **rcp**. Příklad příkazu **rlogin** jsme viděli v kapitole 1 v části pojmenované *Úvod do sítí TCP/IP*. O bezpečnostních problémech souvisejících s tímto příkazem jsme hovořili v téže kapitole v části *Bezpečnost systému*, kde jsme rovněž doporučili nahradit tento příkaz příkazem **ssh**. Balík **ssh** obsahuje náhrady zmíněných programů, programy **slogin**, **ssh** a **scp**.

Všechny tyto příkazy vyvolají na vzdáleném hostiteli příkazový interpret a umožní uživateli spouštět programy. Samozřejmě, že klient musí mít na hostiteli, na kterém hodlá spouštět příkazy, založený účet. Všechny tyto příkazy totiž provádějí ověření totožnosti. *r*-příkazy ověřují totožnost prostou výměnou jména a hesla bez šifrování, takže kdokoliv, kdo odposlouchává síťový provoz, může heslo zjistit. Rodina příkazů **ssh** používá vyšší úroveň zabezpečení: využívá šifrování s veřejným klíčem, při němž je možné provést autentikaci, aniž by po síti byla přenášena jakákoliv snadno zjistitelná data.

V některých případech je možné autentikaci ještě více potlačit. Pokud se například často přihlašujete k jiným počítačům na vaší lokální síti, asi nebudete chtít při každém přihlášení znovu zapisovat heslo. Takové řešení bylo možné už i u *r*-příkazů, balík **ssh** je ještě více zjednodušuje. Nicméně z bezpečnostního hlediska se stále nejedná o příliš vhodné řešení, protože jakmile dojde k narušení jednoho počítače na síti, bude útočník moci získat přístup i k účtům na všech ostatních počítačích. Jedná se však o řešení oblíbené a uživateli hojně používané.

Zaměříme se nyní na to, jak odstranit příkazy rodiny *r* a jak pracovat s balíkem **ssh**.

Vypnutí r-příkazů

Začneme tím, že příkazy rodiny *r*, pokud jsou nainstalovány, odstraníme. Nejjednodušší způsob jak tyto příkazy vypnout spočívá v zakomentování (nebo smazání) příslušných řádků v souboru `/etc/inetd.conf`. Odpovídající řádky budou vypadat asi takto:

```
# Shell, login, exec a talk jsou BSD protokoly.
shell    stream  tcp     nowait  root  /usr/sbin/tcpd /usr/sbin/in.rshd
login    stream  tcp     nowait  root  /usr/sbin/tcpd /usr/sbin/in.rlogind
exec     stream  tcp     nowait  root  /usr/sbin/tcpd /usr/sbin/in.rexecd
```

Řádky můžete zakomentovat tak, že na jejich začátku zapíšete #, nebo je můžete úplně smazat. Aby se změny projevily, musíte restartovat démona **inetd**. Ideálně bude rozumné odpovídající programy také smazat.

Instalace a konfigurace ssh

OpenSSH je volně distribuovaná verze rodiny ssh programů, mutace pro Linux je dostupná na adrese *http://violet.ibs.com.au/openssh* a je součástí většiny moderních distribucí Linuxu[96]. Nebude-

96 Balík OpenSSH byl vyvinut v rámci projektu OpenBSD a je pěkným příkladem výborného zdarma dostupného software.

me zde vysvětlovat, jak balík přeložit; dobrý popis je přímo součástí distribuce. Pokud se vám podaří získat již přeloženou verzi, s výhodou ji můžete nainstalovat.

Relace programu **ssh** zahrnuje dvě strany. Jednou z nich je klient **ssh**, který musí být nakonfigurován a spuštěn na lokálním počítači, druhým je démon **ssh**, který běží na vzdáleném hostiteli.

Démon ssh

Démon **sshd** je program, který naslouchá síťovým spojením od klientů **ssh**, zajišťuje autentikaci a spouští požadované příkazy. Má jeden hlavní konfigurační soubor `/etc/ssh/sshd_config` a dále speciální soubor obsahující klíč používaný pro autentikaci a šifrování přenášených dat. Každý server a každý klient má svůj vlastní klíč.

Náhodný klíč vygeneruje nástroj **ssh-keygen**. Typicky se použije jednou v době instalace k vygenerování klíče, který se pak uloží do souboru `/etc/ssh/ssh_host_key`. Klíče mohou mít délku 512 bitů a větší. Standardně generuje program **ssh-keygen** klíč o délce 1024 bitů a toto nastavení většina uživatelů používá. Náhodný klíč vygenerujete následujícím příkazem:

```
# ssh-keygen -f /etc/ssh/ssh_host_key
```

Budete požádáni o zadání hesla. Klíč na serveru však nesmí být chráněn heslem, takže jen stiskněte Enter a heslo nezadávejte. Výstup programu bude vypadat nějak takto:

```
Generating RSA keys:  ......ooooo0.............................ooooo0
Key generation complete.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /etc/ssh/ssh_host_key
Your public key has been saved in /etc/ssh/ssh_host_key.pub
The key fingerprint is:
1024 3a:14:78:8e:5a:a3:6b:bc:b0:69:10:23:b7:d8:56:82 root@moria
```

Zjistíte, že byly vygenerovány dva soubory. Jeden z nich obsahuje privátní klíč, který musíte udržovat v tajnosti a který je uložen v souboru `/etc/ssh/ssh_host_key`. Druhý je takzvaný veřejný klíč, který sdílíte s ostatními. Ten se nachází v souboru `/etc/ssh/ssh_host_key.pub`.

Jakmile máte vytvořeny klíče, můžete se pustit do nastavení konfiguračního souboru. Balík **ssh** je velmi mocný a konfigurační soubor proto obsahuje spoustu voleb. Uvedeme si jednoduchý příklad, od kterého můžete začít, o dalších možnostech a podrobnostech se dozvíte v dokumentaci k balíku **ssh**. Následující výpis představuje bezpečný a minimální konfigurační soubor démona **sshd**. Další konfigurační možnosti jsou podrobně popsány na manuálové stránce **sshd(8)**.

```
# /etc/ssh/sshd_config
#

# IP adresa na níž posloucháme na spojení. 0.0.0.0 znamená všechny
# lokální adresy.
ListenAddress 0.0.0.0

# TCP port na kterém posloucháme. Standard je 22.
Port 22

# Název souboru s privátním klíčem.
HostKey /etc/ssh/ssh_host_key
```

```
# Délka klíče v bitech.
ServerKeyBits 1024

# Může se přes ssh přihlásit root?
PermitRootLogin no

# Má démon ssh před povolením přihlášení ověřit, zda jsou domovský
# adresář uživatele a jeho přístupová práva bezpečná?
StrictModes yes

# Povolujeme starou autentikační metodu souborů ~/.rhosts a
# /etc/hosts.equiv?
RhostsAuthentication no
# Povolujeme autentikaci mechanismem RSA?
RSAAuthentication yes
# Povolujeme autentikaci heslem?
PasswordAuthentication yes

# Povolujeme /etc/hosts.equiv v kombinaci s RSA autentikací
# hostitele?
RhostsRSAAuthentication no
# Ignorujeme soubory ~/.rhosts?
IgnoreRhosts yes
# Povolujeme přihlášení k účtům s prázdnými hesly?
PermitEmptyPasswords no
```

K zajištění bezpečnosti systému je nutné správně nastavit přístupová práva ke konfiguračnímu souboru. Použijeme k tomu následující příkazy:

```
# chown -R root:root /etc/ssh
# chmod 755 /etc/ssh
# chmod 600 /etc/ssh/ssh_host_key
# chmod 644 /etc/ssh/ssh_host_key.pub
# chmod 644 /etc/ssh/sshd_config
```

Posledním krokem je spuštění démona **sshd**. Obvykle pro něj vytvoříte `rc` soubor nebo jej přidáte do nějakého existujícího, takže se bude spouštět automaticky při startu systému. Démon běží samostatně a neuvádí se v souboru `/etc/inetd.conf`. Démon musí být spuštěn pod právy uživatele root. Syntaxe spuštění je velmi prostá:

```
/usr/sbin/sshd
```

Po svém spuštění se démon automaticky přepne do pozadí. Nyní může váš počítač přijímat **ssh** spojení.

Klient ssh

Klientských programů **ssh** je několik: **slogin**, **scp** a **ssh**. Všechny čtou stejný konfigurační soubor, pojmenovaný obvykle `/etc/ssh/ssh_config`. Kromě toho mohou číst konfigurační soubory z adresáře `.ssh` v domovském adresáři uživatele, který je spouští. Nejdůležitějším z těchto souborů je `.ssh/config`, který může obsahovat nastavení přepisující standardní hodnoty v souboru `/etc/ssh/ssh_config` a soubory `.ssh/identity` a `.ssh/identity.pub`, které obsahují privát-

ní a veřejný klíč uživatele. Dalšími důležitými jsou `.ssh/known_hosts` a `.ssh/authorized_keys`, budeme o nich hovořit později v části *Práce s ssh*. Nejprve se podívejme na globální konfigurační soubor a soubory s uživatelskými klíči.

Soubor `/etc/ssh/ssh_config` se velmi podobá konfiguračnímu souboru serveru. I v něm je možné použít celou řadu voleb, jeho minimální podobu uvádí příklad 12.5. Podrobnosti o zbylých konfiguračních volbách jsou popsány na manuálové stránce **sshd(8)**. Můžete přidávat sekce odpovídající určitým hostitelům nebo skupinám hostitelů. Parametrem volby `Host` může být buď přesná specifikace názvu hostitele, nebo můžete použít zástupné znaky a specifikovat více hostitelů tak, jako to děláme i my v našem příkladu. Například zápisem `Host *.vbrew.com` můžete vytvořit údaj vztahující se ke všem hostitelům v doméně `vbrew.com`.

Příklad 12.5 – Příklad konfiguračního souboru klienta ssh

```
# /etc/ssh/ssh_config

# Standardní nastavení při připojení ke vzdálenému hostiteli
Host *
  # Komprimovat data relace?
  Compression yes
  # .. s jakou úrovní? (1 - rychlé/slabé, 9 - pomalé/účinné)
  CompressionLevel 6

  # Přepnutí na rsh pokud selže bezpečné spojení?
  FallBackToRsh no

  # Posílat keep-alive zprávy? Užitečné při použití maškarády
  KeepAlive yes

  # Zkoušet autentikaci RSA?
  RSAAuthentication yes
  # Zkoušet autentikaci RSA v kombinaci s .rhosts?
  RhostsRSAAuthentication yes
```

Když jsme hovořili o konfiguraci serveru, říkali jsme, že každý hostitel a každý uživatel mají svůj klíč. Klíč uživatele je uložen v souboru `~/.ssh/identity`. Klíč vygenerujete příkazem **ssh-keygen** stejně jako jsme generovali klíč pro server, v tomto případě však nemusíte zadávat název souboru. Program **ssh-keygen** standardně ukládá klíč na správné místo, nicméně se vás stejně zeptá, pokud byste chtěli toto umístění změnit. Někdy je užitečné mít více identifikačních souborů a tímto způsobem je právě můžete vytvořit. Stejně jako předtím vás program **ssh-keygen** požádá o zadání hesla. Hesla představují další úroveň zabezpečení a rozhodně je doporučujeme. Při zadávání hesla se toto heslo nebude zobrazovat na obrazovce.

> **UPOZORNĚNÍ:** Pokud heslo zapomenete, neexistuje metoda jak je zjistit. Volte heslo tak, abyste si je mohli snadno zapamatovat, na druhé straně ovšem tak, aby se nedalo jednoduše uhodnout – vaše jméno není dobrá volba. Aby bylo heslo účinné, mělo by být dlouhé 10 až 30 znaků a nemělo by být tvořeno obyčejným textem. Použijte v něm i různé speciální znaky. Pokud heslo zapomenete, budete muset vygenerovat nový klíč.

Všichni uživatelé by měli jednou spustit program **ssh-keygen**, aby došlo k vygenerování jejich klíčů. Program **ssh-keygen** vytvoří uživatelům adresáře `~/.ssh/`, nastaví jim správná přístupová prá-

va a vygeneruje privátní a veřejný klíč v souborech `.ssh/identity` a `.ssh/identity.pub`. Příklad použití programu vypadá takto:

```
$ ssh-keygen
Generating RSA keys:  .......ooooo0.............................
Key generation complete.
Enter file in which to save the key (/home/maggie/.ssh/identity):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/maggie/.ssh/identity.
Your public key has been saved in /home/maggie/.ssh/identity.pub.
The key fingerprint is:
1024 85:49:53:f4:8a:d6:d9:05:d0:1f:23:c4:d7:2a:11:67 maggie@moria
$
```

Nyní můžete **ssh** používat.

Práce s ssh

Nyní bychom měli mít příkaz **ssh** a související programy nainstalovány a připraveny ke spuštění. Podívejme se nyní krátce na to, jak je spustit.

Nejprve se zkusíme přihlásit ke vzdálenému hostiteli. Použijeme program **slogin** prakticky stejně, jako jsme v příkladu kdysi dříve popisovali použití programu **rlogin**. Při prvním připojení k hostiteli si program **ssh** zjistí veřejný klíč hostitele a požádá vás o potvrzení identity hostitele tím, že vám nabídne zkrácenou verzi jeho veřejného klíče, takzvaný **fingerprint** (otisk).

Administrátor vzdáleného systému by vám měl poskytnout otisk veřejného klíče svého hostitele, který si můžete uložit do souboru `.ssh/known_hosts`. Pokud vám administrátor tento otisk neposkytl, můžete se sice k hostiteli připojit, budete však upozorněni na to, že hostitel má vám neznámý veřejný klíč a budete dotázáni, zda si přejete tento klíč akceptovat. Pokud jste si jisti, že nikdo nezmanipuloval záznamy v DNS databázi a že se opravdu bavíte s tím hostitelem, se kterým chcete, pak klíč akceptujte. Klíč se automaticky uloží do souboru `.ssh/known_hosts` a při příštím připojení se už vás systém nebude na nic ptát. Pokud by se kdykoliv v budoucnu změnil veřejný klíč, kterým se vám vzdálený hostitel prokazuje, budete na to upozorněni, protože se může jednat o bezpečnostní ohrožení.

První přihlášení ke vzdálenému hostiteli bude vypadat nějak takto:

```
$ slogin vchianti.vbrew.com
The authenticity of host 'vchianti.vbrew.com' can't be established.
Key fingerprint is 1024 7b:d4:a8:28:c5:19:52:53:3a:fe:8d:95:dd:14:93:f5.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'vchianti.vbrew.com,172.16.2.3' to the list of/
    known hosts.
maggie@vchianti.vbrew.com's password:
Last login: Tue Feb  1 23:28:58 2000 from vstout.vbrew.com
$
```

Budete vyzváni k zadání hesla, kdy musíte zadat vaše heslo pro vzdálený systém, nikoliv lokální heslo. Heslo se při zadávání nezobrazuje.

Pokud neuvedete žádné další parametry, příkaz **slogin** vás na vzdálený systém přihlásí pod stejným uživatelským jménem jaké používáte na lokálním systému. Toto chování můžete změnit pa-

rametrem `-l`, za kterým zadáte přihlašovací jméno, které se má použít na vzdáleném systému. Přesně to jsme dělali v příkladu dříve v této knize.

Soubory z a na vzdáleného hostitele můžete kopírovat příkazem **scp**. Jeho syntaxe je podobná běžnému příkazu **cp** s tím rozdílem, že před názvem souboru můžete zadat i jméno hostitele. Takovéto zadání bude interpretováno jako cesta k souboru na vzdáleném hostiteli. Následující příklad ilustruje použití příkazu **scp** ke zkopírování lokálního souboru `/tmp/fred` do `/home/maggie` na hostiteli *chianti.vbrew.com*:

```
$ scp /tmp/fred vchianti.vbrew.com:/home/maggie/
maggie@vchianti.vbrew.com's password:
fred                  100% |*****************************| 50165   00:01 ETA
```

I zde budete požádáni o zadání hesla. Příkaz **scp** zobrazuje užitečný ukazatel průběhu operace. Soubory ze vzdáleného hostitele na lokální systém můžete kopírovat stejně snadno, stačí prostě zadat název vzdáleného hostitele a cestu na tomto hostiteli jako zdrojový parametr a jako cílový parametr cestu na místním systému. Je dokonce možné kopírovat soubory z jednoho vzdáleného hostitele na jiný vzdálený systém, tato operace se ale obvykle nepoužívá, protože veškerá data budou přenášena přes váš počítač.

Příkazy na vzdáleném hostiteli můžete spouštět pomocí příkazu **ssh**. Jeho syntaxe je opět velice jednoduchá. Následující příkaz ukazuje, jak může uživatel `maggie` získat výpis kořenového adresáře na hostiteli *vchianti.vbrew.com*:

```
$ ssh vchianti.vbrew.com ls -CF /
maggie@vchianti.vbrew.com's password:
bin/    console@  dos/     home/    lost+found/  pub@   tmp/  vmlinuz@
boot/   dev/      etc/     initrd/  mnt/         root/  usr/  vmlinuz.old@
cdrom/  disk/     floppy/  lib/     proc/        sbin/  var/
```

Příkaz **ssh** můžete uvést v rouře a přesměrovat na něj vstupy nebo výstupy stejně jako u jakéhokoliv jiného příkazu, pouze s tím rozdílem, že vstupy a výstupy budou směrovány na vzdáleného hostitele prostřednictvím **ssh** spojení. Následující příklad ukazuje, jak můžeme tuto funkci použít společně s programem **tar** ke zkopírování celé adresářové struktury ze vzdáleného systému na lokální:

```
$ ssh vchianti.vbrew.com "tar cf - /etc/" | tar xvf -
maggie@vchianti.vbrew.com's password:
etc/GNUstep
etc/Muttrc
etc/Net
etc/X11
etc/adduser.conf
..
..
```

Spouštěný příkaz jsme uzavřeli do uvozovek, aby bylo zřejmé, co jsou parametry příkazu **ssh** a co platí pro lokální příkazový interpret. Tento příklad spustí na vzdáleném hostiteli příkaz **tar** a archivuje jím strukturu celého adresáře `/etc/`, výstup příkazu se zapisuje na standardní výstup. Odtud jej rourou předáváme další instanci programu **tar**, která běží na lokálním systému a rozbaluje data přijatá ze standardního vstupu.

I zde budeme požádáni o zadání hesla. Teď už by mělo být zřejmé, proč jsme se zmiňovali o možnosti nakonfigurovat **ssh** tak, aby při každém volání heslo nepožadoval. Ukážeme si nyní, jak na-

konfigurovat **ssh** tak, aby při spojení se vzdáleným systémem *vchianti.vbrew.com* nežádal o zadání hesla. Už jsme se zmínili o souboru `.ssh/authorized_keys`, nyní jej použijeme. Soubor `.ssh/authorized_keys` obsahuje *veřejné* klíče všech vzdálených uživatelských účtů, ze kterých budeme chtít přihlašovat bez hesla. Automatické přihlášení můžete nastavit tak, že zkopírujete obsah souboru `.ssh/identity.pub` ze *vzdáleného* účtu do lokálního souboru `.ssh/authorized_keys`. Je nezbytně nutné, aby přístupová práva souboru `authorized_keys` dovolovala čtení a zápis do tohoto souboru jenom vám, v opačném případě by mohl kdokoliv použít jeho obsah a přihlašovat se ze vzdálených systémů na váš účet. Abyste zajistili správné nastavení práv, spusťte:

```
$ chmod 600 ~/.ssh/authorized_keys
```

Veřejný klíč je *jeden* dlouhý řádek textu. Pokud jej do lokálního souboru duplikujete operací kopírování a vložení, odstraňte eventuální znaky konce řádku, které se touto operací mohly vytvořit. Soubor `.ssh/authorized_keys` může obsahovat řadu klíčů, každý na samostatném řádku.

Nástroje balíku **ssh** jsou velmi mocné a nabízí celou řadu dalších užitečných funkcí a voleb, které by vás mohly zajímat. Další informace naleznete na manuálových stránkách a v dokumentaci, která se s tímto balíkem dodává.

KAPITOLA 13

Network Information System

Provozujete-li lokální síť, snažíte se obvykle poskytnout uživatelům takové prostředí, ve kterém by síť byla transparentní. Jedním z důležitých kroků, které jsou k tomu zapotřebí, je zajištění synchronizace důležitých dat, jako jsou informace o účtech uživatelů mezi všemi hostiteli. To uživatelům umožní přecházet od počítače k počítači, aniž by si museli pamatovat různá hesla a kopírovat soubory z jednoho systému na druhý. Centrálně uložená data nemusejí být replikována, pokud existuje nějaký pohodlný způsob, jak k nim ze sítě přistupovat. Centrálním uložením důležitých administrativních dat můžete zajistit jejich konzistenci, umožnit uživatelům pružně přecházet od počítače k počítači a usnadnit život správci systému, který bude udržovat pouze jednu centralizovanou kopii těchto dat.

Již dříve jsme se seznámili s důležitým příkladem takovéhoto řešení, které se používá na Internetu – se službou Domain Name System (DNS). DNS ovšem poskytuje pouze omezené množství informací, nejdůležitější z nich je mapování mezi názvy hostitelů a jejich IP adresami. Pro jiné typy informací žádná taková speciální služba neexistuje. Kromě toho, pokud spravujete pouze malou lokální síť, která nemá žádné spojení s Internetem, nebude se mnoho administrátorů namáhat s nastavováním systému DNS.

Tato situace vedla k tomu, že firma Sun vyvinula systém NIS, *Network Information System*. Systém NIS poskytuje prostředky pro obecný přístup k databázím, které lze využít například k šíření informací v souborech `passwd` nebo mezi všemi hostiteli ve vaší síti. To umožňuje, aby síť vypadala jako jediný systém se stejnými účty na všech hostitelích. Podobným způsobem můžete používat službu NIS k šíření informací o hostitelích, které jsou uloženy v souboru `/etc/hosts`, na všechny počítače v síti.

Služba NIS je založena na balíku RPC a skládá se ze serveru, z klientské knihovny a z několika administrativních nástrojů. Původně se systém NIS jmenoval *Yellow pages* (YP, žluté stránky), a tento název se doposud neformálně používá k označení této služby. Bohužel je název „Yellow Pages“ chráněn ochrannou známkou společnosti British Telecom, která požadovala po firmě Sun, aby tento název přestala používat. I s odstupem času však zůstaly některé názvy zakořeněny a zkratka YP se proto stále používá jako prefix názvů většiny příkazů, které se vztahují k systému NIS, například **ypserv** nebo **ypbind**.

Dnes je systém NIS dostupný snad ve všech verzích Unixu a existují dokonce i jeho volné implementace. Jednu z těchto implementací obsahuje balík BSD Net-2. Je odvozena z veřejně dostupné implementace, kterou poskytla firma Sun. Již dlouhou dobu je kód klientské knihovny této verze umístěn v linuxové knihovně *libc*, administrativní nástroje převedl na Linux Swen Thümmler[97]. Referenční verze serveru systému NIS zatím implementována není.

Peter Eriksson vytvořil novou implementaci pojmenovanou NYS[98]. Podporuje jak standardní systém NIS, tak i mnohem dokonalejší systém NIS+ od firmy Sun. Balík NYS neposkytuje pouze administrativní nástroje a server, ale přidává také řadu nových knihovních funkcí, které musíte přeložit do knihovny `libc`. Funkce obsahují mimo jiné nové konfigurační schéma pro rozlišování názvů hostitelů, které nahrazuje současné schéma používající soubor `host.conf`.

GNU knihovna libc, známá jako `libc6` obsahuje aktualizovanou verzi podpory tradičního NIS, kterou vyvinul Thorsten Kukuk[99]. Podporuje všechny knihovní funkce balíku NYS i rozšířené konfigurační schéma tohoto balíku. Stále budete potřebovat konfigurační nástroje a server, nicméně použití GNU knihovny `libc` vám usnadní práci s úpravami a překladem této knihovny.

V této kapitole se budeme zaměřovat spíše na podporu v GNU knihovně `libc` než na zbývající dvě implementace. Pokud budete chtít některou z nich použít, informace v této kapitole vám mohou, ale nemusejí stačit. Další informace naleznete v dokumentu NIS-HOWTO nebo v nějaké knize, například *Managing NFS and NIS* Hala Sterna (O'Reilly).

Seznámení se systémem NIS

Systém NIS uchovává databázové informace v souborech označovaných jako *mapy*, které obsahují dvojice klíč-hodnota. Příkladem takové dvojice je jméno uživatele a jeho zašifrované heslo. Mapy jsou uloženy na centrálním hostiteli, na němž běží server služby NIS a klienti mohou tyto informace číst pomocí různých volání RPC. Poměrně často bývají mapy uloženy v souborech DBM[100].

Vlastní mapy jsou obvykle generovány z hlavních textových souborů, jako jsou například soubory `/etc/hosts` nebo `/etc/passwd`. Pro některé soubory se vytvoří několik map, pro každý typ vyhledávacího klíče vždy samostatná mapa. Například v souboru `hosts` můžete hledat jak název hostitele, tak i IP adresu. Z tohoto souboru budou proto odvozeny dvě mapy, které se budou jmenovat `hosts.byname` a `hosts.byaddr`. Tabulka 10.1 uvádí běžně se vyskytující mapy společně se soubory, ze kterých byly vygenerovány.

Tabulka 13.1 – Hlavní mapy NIS a odpovídající soubory

Hlavní soubor	Mapa(y)	Popis
`/etc/hosts`	`hosts.byname, hosts.byaddr`	Mapování IP adres na jména hostitelů
`/etc/networks`	`networks.byname, networks.byaddr`	Mapování síťových IP adres na jména sítí
`/etc/passwd`	`passwd.byname, passwd.byuid`	Mapování hesel na uživatelská jména
`/etc/group`	`group.byname, group.bygid`	Mapuje identifikátory skupin na jejich názvy
`/etc/services`	`services.byname, services.bynumber`	Mapuje popis služeb na názvy služeb
`/etc/rpc`	`rpc.byname, rpc.bynumber`	Mapuje čísla Sun RPC služeb na jejich názvy
`/etc/protocols`	`protocols.byname, protocols.bynumber`	Mapuje čísla protokolů na jejich názvy
`/usr/lib/aliases`	`mail.aliases`	Mapuje poštovní aliasy na jejich názvy

97 Swena můžete kontaktovat na adrese *swen@uni-paderborn.de*. Klient NIS je dostupný jako soubor `yp-linux.tar.gz` na systému *metalab.unc.edu* v adresáři `system/Network`.

98 Petra můžete kontaktovat na adrese *pen@lysator.liu.se*. Aktuální verze systému NYS je 1.2.8.

99 Thorstena můžete kontaktovat na adrese *kukuk@suse.de*.

100 DBM je jednoduchá knihovna pro správu databázových informací, která používá ke zrychlení vyhledávání hashovací techniky. Volně dostupná implementace DBM existuje jako GNU projekt `gdbm`, který je součástí většiny distribucí Linuxu.

Možná se setkáte s tím, že některé balíky systému NIS nebo některé další programy podporují i jiné soubory a mapy. Tyto soubory a mapy typicky obsahují informace o aplikacích, které nejsou v této knize probírány, například mapa `bootparams`, kterou používá server **bootparamd** společnosti Sun.

Pro některé mapy se často používají *přezdívky*, které jsou kratší a z tohoto důvodu se lépe píší. Přezdívkám rozumí pouze nástroje **ypcat** a **ypmatch**, které slouží ke kontrole konfigurace NIS. Úplný seznam přezdívek, kterým nástroje rozumějí získáte takto:

```
$ ypcat -x
Use "passwd" for "passwd.byname"
Use "group" for "group.byname"
Use "networks" for "networks.byaddr"
Use "hosts" for "hosts.byaddr"
Use "protocols" for "protocols.bynumber"
Use "services" for "services.byname"
Use "aliases" for "mail.aliases"
Use "ethers" for "ethers.byname"
```

Server NIS se tradičně nazývá **ypserv**. Pro potřeby průměrné sítě zpravidla postačuje jediný server; na rozsáhlých sítích bude vhodné provozovat více serverů na různých počítačích a segmentech sítě, takže se rozloží zatížení serverů a směrovačů. Tyto servery jsou vzájemně synchronizovány tak, že jeden z nich je nastaven jako *master server*, ostatní jsou *slave servery*. Mapy se generují pouze na hlavním serveru a odtud se distribuují na podřízené servery.

Doposud jsme termín „síť“ používali hodně neurčitě. Služba NIS používá speciální termín označující skupinu hostitelů, kteří sdílejí službou NIS stejná konfigurační data – jde o takzvanou *doménu NIS*. Bohužel domény systému NIS nemají nic společného s doménami systému DNS, s nimiž jsme se již setkali. Abychom se vyhnuli nejasnostem, budeme vždy uvádět, kterou doménu máme na mysli.

Domény systému NIS plní čistě administrativní funkce. Ty jsou, kromě sdílení hesel mezi všemi počítači příslušné domény, většinou pro uživatele neviditelné. Proto má název přidělený doméně NIS význam pouze pro správce sítě. Zpravidla bude fungovat jakýkoliv název, který bude odlišný od všech názvů domén systému NIS vyskytujících se ve vaší lokální síti. Například správci ve virtuálním pivovaru mohou vytvořit dvě domény systému NIS, jednu z nich pro vlastní pivovar a druhou pro vinařství, a přiřadí jim názvy **brewery** a **winery**. Dalším poměrně běžným schématem je použití stejného názvu jak pro doménu systému DNS, tak i pro doménu systému NIS.

K nastavení a zobrazení názvu domény systému NIS na vašem hostiteli můžete použít příkaz **domainname**. Když tento příkaz voláte bez parametrů, zobrazí název aktuální domény systému NIS; budete-li chtít nastavit název domény, přihlašte se jako superuživatel a napište:

```
# domainname brewery
```

Domény NIS určují, kterého serveru se budou aplikace dotazovat. Například program **login** na hostiteli ve vinařství by se měl samozřejmě na přihlašovací informace dotazovat NIS serveru vinařství (nebo jednoho z nich, pokud by jich bylo více); zatímco aplikace na hostiteli v pivovaru může komunikovat pouze se serverem v pivovaru.

Teď ještě zbývá vyřešit jednu záhadu, konkrétně jak klient zjistí, se kterým serverem se má spojit. Nejjednodušší by bylo použít konfigurační soubor, v němž bude uveden hostitel, na kterém se server má hledat. Avšak tento přístup je poměrně nepružný, protože neumožňuje klientům používat různé servery (samozřejmě z téže domény) v závislosti na jejich dostupnosti. Proto spoléhají tradiční imple-

mentace systému NIS na speciálního démona nazývaného **ypbind**, který detekuje vhodný server NIS ve své doméně. Dříve než se bude aplikace NIS serveru na něco dotazovat, zjistí se nejprve u démona **ypbind**, který server má použít.

Démon **ypbind** vyhledává servery za pomoci vysílání do místní sítě IP; první, kdo odpoví, je považován za potenciálně nejrychlejší server, a proto bude použit pro všechny následující dotazy systému NIS. Po uplynutí určité doby nebo dojde-li k přerušení spojení se serverem, začne démon **ypbind** znovu hledat aktivní servery.

Dynamická vazba na servery je užitečná pouze v případě, že na síti existuje více serverů. Kromě toho může představovat i bezpečnostní problém. Démon **ypbind** slepě uvěří každému, kdo mu odpoví, což může být jak řádný server NIS, tak i zlomyslný vetřelec. Tato vlastnost je zvláště nepříjemná, spravujete-li pomocí systému NIS své databáze s hesly. Abyste tomu předešli, Linux umožňuje nastavit program **ypbind** tak, aby servery buď hledal dynamicky, nebo aby je načítal z konfiguračního souboru.

Systém NIS versus systém NIS+

Systémy NIS a NIS+ toho mají společného jen o málo více, než jen název a účel. Systém NIS+ je strukturován zcela odlišným způsobem. Místo přímého jmenného prostoru s oddělenými doménami systému NIS používá hierarchický prostor s názvy, který je podobný jmennému prostoru systému DNS. Místo map jsou používány takzvané *tabulky*, které jsou složeny z řádků a sloupců. Každý řádek reprezentuje objekt databáze systému NIS+, sloupce pak obsahují vlastnosti objektů, které systém NIS+ zná a o něž se stará. Každá tabulka příslušné domény systému NIS+ v sobě zahrnuje i tabulky svých rodičovských domén. Mimoto může položka v tabulce obsahovat odkaz na další tabulku. Tyto vlastnosti umožňují strukturovat informace mnoha způsoby.

NIS+ navíc podporuje zabezpečené a šifrované RPC, čímž se řeší bezpečnostní problémy prosté služby NIS.

Tradiční systém NIS má číslo verze balíku RPC rovno 2, zatímco systém NIS+ používá verzi 3. V době vzniku tohoto textu neexistuje dobře fungující implementace NIS+ pro Linux, takže o tomto systému zde hovořit nebudeme.

NIS na straně klienta

Pokud se vyznáte v psaní nebo přenosu síťových aplikací, pak jste si jistě všimli, že většina výše uvedených map systému NIS odpovídá funkcím, které jsou obsaženy v knihovně C. Chcete-li například získat informace ze souboru `passwd`, budete k tomuto účelu běžně používat funkce `getpwnam` a `getpwuid`, které vrací informace o účtu sdružené s příslušným jménem uživatele, respektive s číselným identifikátorem uživatele. Za normálních okolností provedou tyto funkce požadované vyhledání ve standardním souboru, tedy například v souboru `/etc/passwd`.

Avšak implementace těchto funkcí v systému NIS toto chování upraví a vloží do nich volání RPC procedury, takže vyhledání provede server NIS. Toto chování bude pro aplikaci zcela transparentní. Funkce mohou chápat data poskytovaná systémem NIS jako připojená ke standardnímu souboru `passwd`, takže pro aplikace budou dostupné informace z obou těchto zdrojů, nebo může standardní soubor touto mapou nahradit, takže data uvedená ve standardním souboru se ignorují a používají se pouze údaje systému NIS.

V tradičních implementacích systému NIS existovaly konvence týkající se toho, které mapy se připojovaly ke standardním souborům a které se používaly jako náhrada standardních souborů. Některé z těchto map, například mapy ze skupiny `passwd`, vyžadovaly úpravy souboru `passwd`, kte-

ré mohly při nesprávném postupu dát vzniknout bezpečnostním dírám. Aby se tomu zabránilo, používá balík NYS obecné konfigurační schéma, které určuje, zda bude konkrétní množina klientských funkcí používat standardní soubory, systém NIS nebo systém NIS+ a v jakém pořadí. O tomto schématu budeme ještě hovořit.

Provozování serveru NIS

Po přehršli teoretických technických informacích přišel čas věnovat se skutečné konfigurační práci. Budeme hovořit o konfiguraci serveru NIS. Pokud už ve vaší síti běží nějaký server NIS, nebudete zřejmě konfigurovat další; v takovém případě můžete další text klidně přeskočit.

Při experimentování se serverem se nezapomeňte ujistit, že jste mu nepřiřadili název domény systému NIS, který je již ve vaší síti používán. To by totiž mohlo narušit všechny síťové služby a rozzlobit spoustu lidí.

Existují dvě možné konfigurace serveru NIS: master nebo slave. Konfigurace slave vytváří záložní systém pro případ, že selže master server. My budeme hovořit pouze o konfiguraci typu master. Pokud byste chtěli konfigurovat slave server, dozvíte se podrobnosti v dokumentaci k serveru.

V současné době jsou na Linuxu volně dostupné dva servery NIS: jeden je obsažen v balíku `yps` Tobiase Rebera, druhý v balíku `ypserv` Petera Erikssona. Nezáleží na tom, který si vyberete.

Po nainstalování programu serveru (**ypserv**) do adresáře `/usr/sbin` byste měli vytvořit adresář, v němž budou uloženy soubory map, které bude váš server distribuovat. Budete-li používat NIS doménu s názvem **brewery,** měly by být mapy umístěny v adresáři `/var/yp/brewery`. Server zjistí, zda spravuje konkrétní doménu systému NIS tak, že ověří přítomnost adresáře s mapami. Pokud chcete zrušit obsluhu nějaké domény NIS, musíte adresář s mapami smazat.

Mapy jsou zpravidla kvůli rychlejšímu vyhledávání uloženy v souborech DBM. Vytvářejí se z hlavních souborů programem **makedbm** (pro server od Tobiase) nebo **dbmload** (pro server od Petera).

Převod hlavního souboru do formátu zpracovatelného programem **dbmload** obvykle vyžaduje podivné zásahy pomocí programů **awk** nebo **sed**, které se poněkud obtížně zapisují i pamatují. Proto obsahuje balík `ypserv` Petera Erikssona program pro vytvoření tohoto souboru (nazvaný `ypMakefile`), který všechnu potřebnou práci udělá za vás. Měli byste ho nainstalovat jako soubor `Makefile` do adresáře s mapami a upravit ho tak, aby obsahoval vámi distribuované mapy. Na začátku souboru naleznete položku *all*, ve které jsou uvedeny služby nabízené programem **ypserv**. Tento řádek vypadá přibližně následovně:

```
all: ethers hosts networks protocols rpc services passwd group netid
```

Pokud například nechcete vytvořit mapy řekněme `ethers.byname` a `ethers.byaddr`, odstraňte z výše uvedené položky volbu *ethers*. Chcete-li si své nastavení otestovat, můžete vytvořit pouze jednu či dvě mapy, například mapy `services.*`.

Po úpravě souboru `Makefile` zadejte v adresáři s mapami příkaz `make`. Tento příkaz požadované mapy automaticky vygeneruje a nainstaluje. Je třeba zajistit, aby se při každé změně hlavních souborů aktualizovaly i soubory s mapami, v opačném případě by síť provedené změny nezjistila.

V části *Nastavení NIS klienta pomocí GNU libc* vysvětlíme, jak nakonfigurovat klienstký kód systému NIS. Pokud vaše nastavení nefunguje, pak je třeba zjistit, zda na váš server vůbec přicházejí nějaké požadavky. Pokud při spouštění serveru **ypserv** zadáte parametr `--debug`, bude na konzolu vypisovat informace o všech docházejících dotazech a odesílaných odpovědích. Takto možná zjistíte, v čem je problém. Tobiasův server žádnou takovou volbou nedisponuje.

Bezpečnost serveru NIS

Systém NIS míval vážnou bezpečnostní slabinu: nechával soubor s hesly volně čitelný pro kohokoliv na Internetu, což umožňovalo řadu různých útoků. Jakmile útočník zná název vaší NIS domény a adresu jejího serveru, může poslat žádost o mapu `passwd.byname` a dostane tak celou databázi zašifrovaných hesel. S rychlým luštícím programem jako je **crack** a s dobrým slovníkem nebude žádný problém uhodnout alespoň část uživatelských hesel.

Proto byla zavedena volba *securenets*. To omezuje přístup k NIS serveru pouze na určité počítače podle jejich IP adresy nebo síťové adresy. Poslední verze serveru **ypserv** implementuje tuto funkci dvěma způsoby. První z nich je založen na konfiguračním souboru `/etc/ypserv.securenets` a druhý používá známé soubory `/etc/hosts.allow` a `/etc/hosts.deny`, o kterých jsme hovořili v kapitole 12. Omezení přístupu ke službě NIS na počítače pouze ze sítě pivovaru je možné provést přidáním následujícího řádku do souboru `hosts.allow`:

```
ypserv: 172.16.2.
```

Tím se povolí přístup k NIS serveru hostitelům na síti 172.16.2.0. K zakázání přístupu všem ostatním hostitelům je nutné přidat do souboru `hosts.deny` následující záznam:

```
ypserv: ALL
```

IP adresy nejsou jediná možnost jak v souborech `hosts.allow` a `hosts.deny` specifikovat počítače a sítě. Podrobnosti o dalších možnostech naleznete na manuálových stránkách `hosts_access(5)`. Upozorňujeme ale na to, že v položkách pro server **ypserv** *nelze* používat názvy hostitelů a sítí. Pokud zadáte název, server se jej pokusí přeložit na adresu – a resolver okamžitě volá **ypserv**, takže skončíme v nekonečné smyčce.

Zabezpečení můžete nastavit také pomocí souboru `/etc/ypserv.securenets`. Tento soubor musíte vytvořit, přičemž jeho struktura je velmi jednoduchá. Každý řádek definuje jednoho hostitele nebo síť, kteří mají k NIS serveru přístup. Adresy neuvedené v tomto souboru budou mít přístup zakázán. Řádky začínající znakem # jsou chápány jako komentář. Příklad souboru `/etc/ypserv.securenets` vidíte na výpisu 13.1.

Příklad 13.1 – Příklad souboru ypserv.securenets

```
# povolujeme připojení z lokálního hostitele - nezbytné
host 127.0.0.1
# stejné jako 255.255.255.255 127.0.0.1
#
# povolujeme připojení ze sítě pivovaru
255.255.255.0   172.16.1.0
#
```

Prvním údajem na řádku je síťová maska nebo klíčové slovo `host`, které má význam speciální masky 255.255.255.255. Druhým údajem je IP adresa, na níž se maska vztahuje.

Konečně třetí možností je použít zabezpečený mapovač portů. Zabezpečený mapovač (`portmap-5.0`) rovněž používá soubory `hosts.allow` a `hosts.deny`, nabízí však tuto funkci obecně všem RPC serverům, nejen serveru **ypserv**[101]. Neměli byste ovšem současně používat více zabezpečo-

101 Zabezpečený mapovač portů můžete získat anonymním FTP na počítači *ftp.win.tue.nl* v adresáři `/pub/security`.

vacích mechanismů, protože se tak zbytečně zvyšuje režie související s kontrolou oprávněnosti požadavku.

Nastavení NIS klienta pomocí GNU libc

Nyní budeme hovořit o tom, jak nakonfigurovat klienta systému NIS pomocí GNU knihovny libc.

Prvním krokem je, že klientskému balíku sdělíte, jaký server služby NIS má používat. Již jsme se zmínili, že automatické nastavení serveru zajišťuje program **ypbind**. Standardně se chová tak, že se na lokální síti dotazuje, zda na ní není server. Pokud konfigurujete počítač, který se bude často přesouvat mezi různými doménami, například notebook, nechte soubor `/etc/yp.conf` prázdný a program **ypbind** vám automaticky nalezne nejbližší server podle toho, kde se nacházíte.

Bezpečnější řešení je nadefinovat jméno serveru v souboru `/etc/yp.conf`. Velmi jednoduchý soubor pro počítač v síti vinařství by tam mohl mít nastaveno

```
# yp.conf - konfigurační soubor pro systém NIS
#
ypserver vbardolino
```

Příkaz `ypserver` říká vašemu hostiteli, že má jako NIS server domény použít ten server, jehož jméno je uvedeno za tímto příkazem. V našem případě jsme tedy nastavili server `vbardolino`. IP adresa odpovídající tomuto počítači musí být samozřejmě definována v souboru `hosts`, v opačném případě byste museli server definovat přímo jeho IP adresou.

Ve výše uvedeném příkladu sdělujeme programu **ypbind**, aby použil uvedený server pokaždé, bez ohledu na to, v jaké doméně pracujeme. Pokud počítač často přemisťujete mezi různými doménami systému NIS, budete možná chtít mít v souboru `yp.conf` informace o několika doménách. V souboru `yp.conf` můžete mít informace o serverech pro různé domény systému NIS, které specifikujete příkazem `domain`. Například pro laptop může výše uvedený soubor vypadat takto:

```
# yp.conf - konfigurační soubor pro systém NIS
#
domain minery server vbardolino
domain brewery server vstout
```

Takto upravený soubor `yp.conf` vám umožní přenášet laptop mezi oběma doménami. Při spouštění systému pouze stačí příkazem **domainname** nastavit doménu, ve které se nacházíme.

Existuje ještě třetí možnost nastavení. Ta se používá v případech, kdy pro určitou doménu neznáte její NIS server, v jiné doméně však chcete použít pevně daný server. Řekněme, že v doméně vinařství budeme chtít použít určitý daný server, v doméně pivovaru budeme chtít nechat server detekovat automaticky. Soubor `yp.conf` pak můžeme modifikovat takto:

```
# yp.conf - konfigurační soubor pro systém NIS
#
domain minery server vbardolino
domain brewery broadcast
```

Klíčové slovo `broadcast` nařizuje programu **ypbind**, aby server v dané doméně nalezl automaticky.

Po vytvoření tohoto základního konfiguračního souboru a nastavení přístupových práv tak, aby byl soubor veřejně přístupný, nastává čas pro otestování, zda systém funguje. Zvolte si mapu o níž víte, že ji server nabízí, například `hosts.byname`, a pokuste se ji získat pomocí nástroje **ypcat**:

```
# ypcat hosts.byname
172.16.2.2      vbeaujolais.vbrew.com    vbeaujolais
172.16.2.3      vbardolino.vbrew.com     vbardolino
172.16.1.1      vlager.vbrew.com         vlager
172.16.2.1      vlager.vbrew.com         vlager
172.16.1.2      vstout.vbrew.com         vstout
172.16.1.3      vale.vbrew.com           vale
172.16.2.4      vchianti.vbrew.com       vchianti
```

Získaný výstup by měl vypadat podobně jako výše uvedený výpis. Pokud místo tohoto obdržíte chybovou zprávu „Can't bind to server which serves domain" nebo nějakou podobnou zprávu, pak buď vámi zadaný název domény systému NIS nemá v souboru `yp.conf` definovaný odpovídající server, nebo je server z nějakého důvodu nedostupný. V druhém případě se ujistěte, že příkaz **ping** směrovaný na příslušného hostitele vrátí přijatelné výsledky, a že daný hostitel má skutečně spuštěn server NIS. Přítomnost serveru NIS na daném hostiteli můžete ověřit pomocí příkazu **rpcinfo**, který by měl vrátit následující výstup:

```
# rpcinfo -u serverhost ypserv
program 100004 version 1 ready and waiting
program 100004 version 2 ready and waiting
```

Výběr vhodných map

Jakmile server funguje a je dosažitelný, musíte rozhodnout, které konfigurační soubory nahradíte mapami systému NIS a ke kterým konfiguračním souborům mapy systému NIS připojíte. Mapy systému NIS budete zpravidla používat u funkcí, které vyhledávají hostitele a hesla. Funkce vyhledávající hostitele mají význam zejména tehdy, pokud nemáte spuštěnu službu BIND. Funkce vyhledávající hesla povolují všem uživatelům přihlášení ke svému účtu z libovolného systému v příslušné doméně NIS; tato funkce obvykle vyžaduje sdílení centrálního adresáře `/home` mezi všemi hostiteli pomocí služby NFS. Dále si popíšeme jak vytvořit sdílenou mapu hesel.

Ostatní mapy, jako například mapa `service.byname`, nejsou až tak užitečné, ale mohou vám ušetřit něco práce. Mapa `service.byname` je užitečná zejména v případě, kdy instalujete nějaké síťové aplikace, které používají název služby, jenž není uveden ve standardním souboru `services`.

Obecně budete asi chtít zachovat určitou možnost volby, zda má vyhledávací funkce používat lokální soubory, kdy se má dotazovat serveru NIS a kdy jiného serveru, například DNS. GNU knihovna libc umožňuje nastavit pořadí, ve kterém daná funkce jednotlivé služby použije. Toto pořadí je řízeno konfiguračním souborem `/etc/nsswitch.conf` (*Name Service Switch*), který se ovšem neomezuje jen na služby jmen. Pro každou vyhledávací funkci, která je podporována balíkem GNU libc, obsahuje konfigurační soubor řádek s výčtem služeb, které se budou používat.

Správné pořadí služeb závisí na typu dat, která jednotlivé služby nabízejí. Je málo pravděpodobné, že by mapa `services.byname` obsahovala položky, které se liší od položek uvedených v místním souboru `services`; tato mapa jich jen může obsahovat o něco více. Proto bude rozumné prohledávat nejprve lokální soubor a teprve pokud službu nenalezneme, obrátit se na server NIS. Naopak informace o názvech hostitelů se mění poměrně často, takže server DNS nebo NIS bude mít

určitě čerstvější informace než lokální soubor `hosts`, který se použije pouze jako záloha v případě, že žádný server nebude dostupný. V tomto případě tedy budeme lokální soubor prohledávat až jako poslední.

Následující příklad ukazuje způsob jak funkce `gethostbyname` a `gethostbyaddr` nastavit tak, aby nejprve použily služby NIS a DNS a až poté lokální soubor `hosts`. Funkce resolveru budou zkoušet jednotlivé služby v uvedeném pořadí, je-li vyhledávání úspěšné, vrátí výsledek, v opačném případě se bude zkoušet další služba.

```
# jednoduchý příklad /etc/nsswitch.conf
#
hosts:      nis dns files
services:   files nis
```

Dále následuje kompletní seznam služeb, které mohou být použity jako položky konfiguračního souboru `nsswitch.conf`. Jaké mapy, soubory, servery a objekty budou ve skutečnosti dotazovány závisí na názvu položky. Napravo od dvojtečky se může objevit:

nis	Použije NIS server nastavené domény. Umístění dotazovaného serveru je nastaveno v souboru `yp.conf`, který jsme si popsali dříve. Pro položku hosts budou dotazovány mapy `hosts.byname` a `hosts.byaddr`.
nisplus nebo nis+	Použije se NIS+ server nastavené domény. Umístění serveru se zjistí v souboru `/etc/nis.conf`.
dns	Použije se jmenný server systému DNS. Tento typ služby je užitečný pouze ve spojitosti s položkou hosts. Dotazované jmenné servery jsou převzaty ze standardního souboru `resolv.conf`.
files	Použije se místní soubor, pro položku hosts je to například soubor `/etc/hosts`.
compat	Zapnutí kompatibility se staršími formáty souborů. Tento přepínač se používá pokud NYS nebo glibc 2.x provádějí vyhledávání službami NIS nebo NIS+. Tyto verze standardně nedokáží interpretovat starší záznamy systému NIS v souborech `passwd` a `group`, tato volba povolí práci i se starším formátem.
db	Informace se vyhledají v souborech typu DBM, které jsou umístěny v adresáři `/var/db`. Název použitého souboru bude stejný jako název odpovídající mapy systému NIS.

Podpora NIS v GNU knihovně libc umožňuje v souboru `nsswitch.conf` použít následující databáze: aliases, ethers.group, hosts, netgroup, networks, passwd, protocols, publickey, rpc, services a shadow. Časem budou pravděpodobně přidány další.

Příklad 13.2 ukazuje o něco složitější příklad, který používá novou vlastnost konfiguračního souboru `nsswitch.conf`: klíčové slovo *[NOTFOUND=return]*. Toto klíčové slovo na řádku *hosts* způsobí, že pokud se název hostitele nepodaří najít v systémech NIS a DNS, nebude se už hledat v lokálním souboru `hosts`. Lokální soubor se bude prohledávat *pouze* v případě, že se z nějakého důvodu nezdaří volání DNS a NIS serveru. Soubor tak bude používán pouze v době spouštění systému a jako záloha pro případ, že servery nebudou dostupné.

Příklad 13.2 – Příklad souboru nsswitch.conf

```
# /etc/nsswitch.conf
#
hosts:      nis dns [NOTFOUND=return] files
networks:   nis [NOTFOUND=return] files
services:   files nis
protocols:  files nis
rpc:        files nis
```

GNU knihovna libc nabízí ještě další funkce, které jsou popsány na manuálové stránce `nsswitch`.

Použití map passwd a group

Jedním z hlavních nasazení systému NIS je synchronizace informací o uživatelích a účtech mezi všemi hostiteli v rámci příslušné domény systému NIS. Pak jednotliví hostitelé používají malý lokální soubor `/etc/passwd`, ke kterému se připojují informace z map systému NIS, které obsahují data od ostatních hostitelů. Bohužel pouhé povolení služby NIS v souboru `nsswitch.conf` pro tuto databázi nestačí.

Spoléháte-li se na informace s hesly šířené pomocí systému NIS, musíte se nejprve ujistit, že číselný identifikátor libovolného uživatele v lokálním souboru `passwd` se shoduje s identifikátorem používaným v databázi NIS. Konzistence uživatelských identifikátorů je důležitá i v jiných aplikacích, například pro připojování síťových NFS svazků.

Pokud se některé z identifikátorů v souboru `/etc/passwd` nebo `/etc/group` liší od hodnot v mapách, musíte upravit vlastnictví všech souborů, které se vztahují k danému uživateli. Nejprve změňte identifikátory uživatelů a skupin v souborech `passwd` a `groups`, pak najděte všechny soubory patřící změněným uživatelům a opravte jejich vlastnictví. Řekněme, že účet **news** měl uživatelský identifikátor 9 a účet **okir** měl identifikátor 103. Změnili jsme identifikátory na nové hodnoty a jako **root** nyní spustíme následující příkazy:

```
# find / -uid   9 -print >/tmp/uid.9
# find / -uid 103 -print >/tmp/uid.103
# cat /tmp/uid.9   | xargs chown news
# cat /tmp/uid.103 | xargs chown okir
```

Je důležité, abyste tyto příkazy spustili s novým souborem `passwd` a abyste si zjistili názvy všech souborů dříve, než začnete měnit vlastnictví kteréhokoliv z nich. K aktualizaci vlastnictví souborů skupinami použijte obdobnou sekvenci příkazů.

Jakmile provedete tyto změny, budou ve vašem systému souhlasit identifikátory uživatelů a skupin s identifikátory používanými systémem NIS. Další krok je přidat do souboru `nsswitch.conf` řádky, které povolí vyhledávání informací o uživatelích a skupinách pomocí systému NIS:

```
# /etc/nsswitch.conf - passwd a group
passwd: nis files
group:  nis files
```

Tato nastavení se projeví, když program **login** a příbuzné programy zjišťují informace o uživateli. Když se chce uživatel přihlásit, **login** nejprve prohledává mapy NIS a teprve pokud v nich uživatele nenajde, podívá se do lokálního souboru. Z lokálních souborů typicky odstraníte záznamy

většiny uživatelů a ponecháte v nich pouze záznamy pro uživatele **root** a obecné účty, jako například účet **mail**. To proto, že některé důležité systémové úkoly mohou vyžadovat mapování uživatelských identifikátorů na jména uživatelů a naopak. Například administrativní úkoly **cron** mohou spustit příkaz `su`, aby dočasně přešly na účet **news** nebo může podsystém protokolu UUCP poslat stavovou poštu. Pokud místní soubor `passwd` neobsahuje položky pro účty **news** a **uucp**, pak tyto úkoly neproběhnou při výpadku systému NIS správně.

Konečně pokud používáte starší implementaci NIS (která je v implementacích NYS a glibc pro soubory `passwd` a `groups` podporována pomocí přepínače `compat`), musíte do těchto souborů přidat speciální položky. Tyto položky definují místo, kam má NIS vkládat do souborů své databázové informace. Záznamy je možné přidat kamkoliv, typicky se přidávají na konec souboru. Do souboru `/etc/passwd` je třeba přidat záznam

```
+::::::
```

a do souboru `/etc/groups` záznam

```
+:::
```

S glibc 2.x i NIS můžete přepsat hodnoty v uživatelských záznamech přijaté ze serveru tím, že vytvoříte záznam, kde bude před uživatelským jménem uveden symbol „+“; a můžete rovněž určité uživatele vyloučit tak, že před jejich uživatelským jménem uvedete symbol „–“. Podívejme se na následující příklad:

```
+stuart::::::/bin/jacl
-jedd::::::
```

První z nich způsobí, že pro uživatele **stuart** bude použito naše nastavení přihlašovacího shellu namísto hodnoty poskytnuté serverem, druhý způsobí, že uživatel **jedd** se k danému počítači nebude moci přihlásit.

Vynořují se zde dva problémy: popsané nastavení bude fungovat pouze s přihlašovacími systémy, které neopoužívají stínová hesla. Problematika použití stínových hesel společně se systémem NIS bude popsána v další části. Druhým problémem je to, že k souboru `passwd` nepřistupují pouze přihlašovací příkazy – podívejme se například na příkaz **ls**, který používá většina lidí takřka pořád. Kdykoliv provádíte nějaké dlouhé výpisy, zobrazí příkaz **ls** symbolické názvy vlastníků souboru (uživatelů a skupin); to znamená, že kdykoliv příkaz **ls** narazí na nějaké uid nebo gid, bude se muset dotazovat serveru NIS. Dotaz systému NIS se zpracovává déle než odpovídající vyhledávání v lokálním souboru. Zřejmě zjistíte, že použití systému NIS se soubory `passwd` a `groups` povede ke znatelnému zpomalení programů, které s těmito soubory hodně pracují.

Ani tím však potíže nekončí. Představte si co se stane, když si chce uživatel změnit své heslo. Obvykle spustí příkaz **passwd**, který přijme nové heslo a aktualizuje místní soubor `passwd`. Tento postup ovšem nelze použít, protože soubor `passwd` není lokálně dostupný. Aby se uživatel musel při každé změně hesla přihlašovat k serveru NIS, to rovněž není řešení. Proto systém NIS poskytuje náhradu k příkazu **passwd**, a to příkaz **yppasswd**, který mění heslo v systému NIS. Při změně hesla na serveru příkaz prostřednictvím volání RPC kontaktuje démona **yppasswdd** na serveru a pošle mu informace o novém hesle. Typicky budete příkaz **yppasswd** instalovat přes klasickou verzi například takto:

```
# cd /bin
# mv passwd passwd.old
# ln yppasswd passwd
```

Zároveň musíte na serveru nainstalovat démona **rpc.yppasswdd** a spustit ho z některého síťového skriptu. Tím před uživateli skryjete úskalí spojená se změnou hesla.

NIS a stínová hesla

Použití systému NIS společně se stínovými hesly je poněkud problematické. Nejprve špatná zpráva: použitím systému NIS se ruší výhody stínového souboru hesel. Systém stínových hesel byl vytvořen proto, aby uživatelům bez práva superuživatele zabránil v přístupu k zašifrovaným heslům. Pokud budete systémem NIS sdílet data v souboru `shadow`, nezbytně tak zpřístupňujete stínová hesla komukoliv, kdo může na síti poslouchat odpovědi serveru NIS. V prostředí NIS je proto rozhodně lepší donutit uživatele používat „dobrá“ hesla a nesnažit se hesla skrýt. Pokud se nicméně pro toto řešení rozhodnete, zkusíme si ukázat, jak to lze provést.

V libc5 neexistuje možnost, jak systémem NIS sdílet soubor `shadow`. Jediná metoda distribuce hesel a uživatelských informací je použití standardních map `passwd.*`. Pokud máte stínová hesla nainstalována, nejrozumnější bude ze souboru `/etc/shadow` vygenerovat odpovídající soubor `passwd` nástrojem jako je **pwunconv** a pak z tohoto souboru vytvořit mapy NIS.

Samozřejmě existují i různé postranní cesty jak současně se systémem NIS využít i stínová hesla. Jedno takové řešení je například na každém hostiteli v síti nainstalovat soubor `/etc/shadow`, zatímco ostatní uživatelské informace bude distribuovat NIS. Toto řešení je ovšem velmi brutální a připravuje nás o hlavní výhodu systému NIS – totiž o usnadnění administrace.

Podpora NIS v GNU knihovně libc (libc6) obsahuje i podporu stínové databáze hesel. Nenabízí sice žádné řešení toho, že stínová hesla nyní budou volně přístupná, nicméně usnadní použití NIS v situacích, kdy chcete sdílet databázi stínových hesel. Abyste ji mohli sdílet, musíte vytvořit mapu `shadow.byname` a do souboru `/etc/nsswitch.conf` přidat následující řádky:

```
# Podpora stínových hesel
shadow:         compat
```

Pokud se systémem NIS používáte stínová hesla, měli byste zajistit jejich zabezpečení omezením přístupu k databázím NIS. Podívejte se do části *Bezpečnost serveru NIS* dříve v této kapitole.

KAPITOLA 14

Network File System

Network File System (NFS) je pravděpodobně nejznámnější síťovou službou, která je založena na RPC. Umožňuje přistupovat k souborům na vzdálených hostitelích stejně, jako kdyby šlo o místní soubory. Toto chování je možné díky kombinaci speciálních funkcí jádra a démonů uživatelského prostoru na straně klienta a serveru NFS na straně serveru. Tento přístup k souborům je pro klienta zcela transparentní a funguje na nejrůznějších serverových a klientských architekturách.

Systém NFS nabízí řadu výhod:

- Data, k nimž přistupují všichni uživatelé, mohou být uchovávána na centrálním hostiteli a klienti si tento adresář připojí při zavádění systému. Například všechny uživatelské adresáře můžete uchovávat na jediném hostiteli a z něj si všichni připojí adresář `/home`. Je-li tento systém nainstalován společně se systémem NIS, potom se uživatelé mohou přihlásit k libovolnému systému a přitom mohou stále pracovat s jednou sadou souborů.
- Data, která zabírají velké množství diskového prostoru, mohou být uchovávána na jediném hostiteli. Například všechny soubory a programy vztahující se k balíkům LaTeX a METAFONT mohou být uloženy a spravovány na jednom místě.
- Administrativní data mohou být uchovávána na jediném hostiteli. Již není třeba používat příkaz **rcp** k nainstalování stejného souboru na dvacet různých počítačů.

Nastavení klienta a serveru systému NFS není těžké, v této kapitole se dozvíme, jak to udělat.

Systém NFS je z velké části dílem Ricka Sladkeyho, který napsal jak kód NFS v jádře, tak i velkou část serveru NFS[102]. Server NFS byl odvozen z NFS serveru *unfsd* pro uživatelský prostor, který původně vytvořil Mark Shand, a z Harrisova NFS serveru *hnfs*, jehož autorem je Donald Becker.

Podívejme se nyní, jak systém NFS funguje. Klient může požádat o připojení adresáře ze vzdáleného hostitele ke svému místnímu adresáři naprosto stejným způsobem, jako při připojování fyzického zařízení. Nicméně syntaxe používaná k zadání vzdáleného adresáře je odlišná. Chcete-li například připojit adresář `/home` z hostitele **vlager** k adresáři `/users` na hostiteli **vale**, správce na hostiteli **vale** spustí následující příkaz[103]:

```
# mount -t nfs vlager:/home /users
```

Příkaz **mount** se pokusí pomocí procedur RPC spojit s připojovacím démonem **rpc.mountd** na hostiteli **vlager**. Server zkontroluje, zda má hostitel **vale** povoleno připojit požadovaný adresář a pokud ano, vrátí mu souborový handle. Tento handle pak bude použit ve všech následujících souborových požadavcích na adresář `/users`.

102 Ricka můžete kontaktovat na adrese *jrs@world.std.com*.

103 Po pravdě řečeno můžete vynechat parametr `-t nfs`, protože příkaz **mount** z dvojtečky pozná, že se jedná o adresář přístupný systémem NFS.

Při přístupu k souboru pomocí systému NFS volá jádro pomocí RPC démona **rpc.nfsd** (tedy démona NFS) na serveru. V tomto volání se jako parametry předávají handle souboru, název souboru a identifikátor uživatele a skupiny. Tyto hodnoty slouží k určení přístupových práv k požadovanému souboru. Aby se zabránilo neautorizovaným uživatelům ve čtení nebo úpravě souborů, musí být identifikátory uživatelů a skupin na obou hostitelích totožné.

Ve většině unixových implementací je systém NFS jak pro klienta, tak i pro server implementován pomocí démonů jádra, které se z uživatelského prostoru spouštějí při zavádění systému. Jsou to démony *NFS Daemon* (**rpc.nfsd**) na serveru a *Block I/O Daemon* (**biod**) na klientovi. Aby se zvýšila propustnost, provádí démon **biod** asynchronní vstupně-výstupní operace s využitím dopředného čtení a opožděného zápisu; navíc bývá obvykle současně spuštěno několik démonů **rpc.nfsd**.

Implementace systému NFS v Linuxu se nepatrně liší tím, že kód serveru běží zcela v uživatelském prostoru, takže současné spuštění více instancí je komplikované. Současná implementace démona **rpc.nfsd** nabízí experimentální funkci, která (s jistými omezeními) umožňuje spuštění více serverů. V jádrech verze 2.2 vyvinul Olaf Kirch podporu NFS serveru přímo v jádře. Její výkon je výrazně vyšší než u stávajících řešení v uživatelském prostoru. Budeme o ní hovořit později.

Příprava systému NFS

Před použitím systému NFS jako klienta i jako serveru je nutné ověřit, že jádro má vestavěnu podporu NFS. Novější jádra mají pro tento účel jednoduché rozhraní v souborovém systému `proc`, rozhraní `/proc/filesystems`, které můžete jednoduše zobrazit příkazem **cat**:

```
$ cat /proc/filesystems
        minix
        ext2
        msdos
nodev   proc
nodev   nfs
```

Pokud v tomto seznamu chybí souborový systém nfs, pak je nutné přeložit jádro operačního systému s podporou systému NFS, případně nahrát příslušný modul jádra, pokud máte podporu NFS přeloženu jako modul. Konfigurace síťových voleb jádra operačního systému je popsána v části *Konfigurace jádra* ve třetí kapitole.

Připojení svazku systému NFS

Svazky systému NFS se připojují podobně jako běžné souborové systémy. Spustíte příkaz **mount** s následující syntaxí[184]:

```
# mount -t nfs svazek_nfs lokální_adresář volby
```

Parametr *svazek_nfs* se zadává ve tvaru *vzdálený_hostitel:vzdálený_adresář*. Tato notace je specifická pro systém NFS, takže je možné vynechat volbu `-t nfs`.

Při připojování svazku NFS je možné rovněž zadat řadu různých voleb. Tyto volby mohou buď následovat na příkazové řádce za přepínačem `-o`, nebo je lze pro daný svazek uvést v příslušné položce v souboru `/etc/fstab`. V obou případech se více voleb navzájem odděluje čárkami a vol-

184 Říkáme „svazky“ místo „souborové systémy“, protože se nejedná o normální souborové systémy.

by nemohou obsahovat mezery. Volby zadané na příkazové řádce vždy potlačí volby, které jsou uvedeny v souboru `fstab`.

Příklad položky v souboru `fstab` může vypadat takto:

```
# svazek              připojovací bod   typ   volby
news:/var/spool/news  /var/spool/news   nfs   timeo=14,intr
```

Pak lze tento svazek připojit příkazem

```
# mount news:/var/spool/news
```

Pokud by v souboru `fstab` nebyla uvedena zmíněná nastavení, bylo by použití příkazu **mount** složitější. Předpokládejme například, že chcete připojit domovský adresář uživatele z počítače **moonshot**, který používá pro operace čtení a zápisu implicitní velikost bloku 4 KB. Velikost bloku můžete zvýšit pomocí následujícího příkazu, takže dosáhnete lepšího výkonu:

```
# mount moonshot:/home /home -o rsize=8192,wsize=8192
```

Úplný a podrobný popis všech voleb je uveden na manuálové stránce `nfs(5)`. Následující seznam představuje některé často používané volby.

rsize=n a *wsize=n* — Tyto volby určují velikosti datagramu, které budou používat klienti systému NFS u požadavků na čtení, respektive zápis. Jejich implicitní hodnota závisí na verzi jádra, typicky bývá 1024 bajtů.

timeo=n — Tato volba nastavuje čas (v desetinách sekundy), po který bude klient čekat na dokončení požadavku. Implicitní hodnota je 7 (tedy 0,7 sekundy). Co se bude dít po vypršení tohoto limitu záleží na tom, zda uvedete volby *hard* nebo *soft*.

hard — Explicitně označí tento svazek jako pevně připojený svazek. Standardní nastavení. Pokud dojde k velkému timeoutu, server to oznámí na konzolu a bude čekat neomezeně dál.

soft — Svazek je připojen volně (opak pevného připojení). Pokud při souborové operaci dojde k velkému timeoutu, bude ohlášena chyba vstupně-výstupní operace.

intr — Tato volba povoluje přerušit NFS volání pomocí signálů. Je užitečná, když potřebujete přerušit operaci pokud server neodpovídá.

Kromě voleb *rsize* a *wsize* všechny výše uvedené volby nastavují chování klienta pro případ, kdy server není dočasně dostupný. Tyto volby spolu souvisí následovně: kdykoliv klient pošle požadavek na server NFS, očekává, že bude operace dokončena v daném časovém limitu (ten udává volba *timeo*). Pokud v tomto intervalu nepřijde žádné potvrzení, nastane takzvaný *malý timeout*, a operace se spustí znovu s dvojnásobnou hodnotou timeoutu. Po dosažení maximálního timeoutu o délce 60 sekund dojde k *velkému timeoutu*.

Implicitně po výskytu velkého timeoutu klient vypíše zprávu na konzolu a celý proces se bude znovu opakovat, tentokrát ovšem s timeoutem rovným dvojnásobku hodnoty z předchozího pokusu. Teoreticky by tento postup mohl trvat nekonečně dlouhou dobu. Svazky, které se tvrdohlavě pokoušejí opakovat operaci tak dlouho, dokud nebude server opět dostupný, se nazývají *pevně připojené* svazky. Opačný typ svazků představují takzvané *volně připojené* svazky, které při výskytu velkého timeoutu generují vstupně-výstupní chybu. Protože se používá opožděný zápis, nebude ta-

to chybová zpráva předána volajícímu procesu dříve, než proces znovu zavolá funkci `write`, takže program nemá nikdy jistotu, zda zápis na volně připojený svazek proběhl správně.

Zda daný svazek připojit pevně nebo volně je částečně otázkou vkusu, ale také otázkou typu informací, které jsou na svazku uloženy. Pokud si například pomocí systému NFS připojíte svazek s X programy, určitě nebudete chtít aby se vaše relace chovala divně jen proto, že někdo zablokoval síť například současným spuštěním sedmi kopií Dooma, nebo krátkodobým vytáhnutím ethernetové zástrčky. Když takový svazek připojíte pevně, budete mít jistotu, že váš počítač bude čekat tak dlouho, dokud se znovu neobnoví spojení s NFS serverem. Na druhé straně nekritická data, například svazek news nebo archivy FTP, mohou být připojena volně, takže pokud bude vzdálený počítač dočasně nedostupný nebo vypnutý, nedojde k zablokování vaší relace. Pokud máte nestabilní spojení se serverem, nebo se připojujete přes nějaký hodně zatížený směrovač, můžete buď zvýšit počáteční hodnotu timeoutu (pomocí volby *timeo*), nebo můžete dané svazky připojit pevně. Implicitně se NFS svazky připojují pevně.

Pevné připojení může způsobit potíže, protože standardně jsou souborové operace nepřerušitelné. Pokud se tedy nějaký proces pokusí o zápis na nedosažitelný server, aplikace se zasekne a uživatel nebude moci operaci přerušit. Použijete-li současně s pevným připojením volbu *intr*, jakýkoliv procesem přijatý signál přeruší NFS volání a uživatel také bude moci toto volání zrušit a pokračovat v práci (samozřejmě aniž by se mu podařilo data uložit).

Démon **rpc.mountd** typicky nějakým způsobem udržuje informace o tom, které adresáře jsou připojeny kterým hostitelem. Tyto informace lze zobrazit za pomoci programu **showmount**, který je rovněž součástí balíku serveru systému NFS:

```
# showmount -e moonshot
Export list for localhost:
/home <anon clnt>

# showmount -d moonshot
Directories on localhost:
/home

# showmount -a moonshot
All mount points on localhost:
localhost:/home
```

Démony systému NFS

Pokud chcete službu NFS poskytovat ostatním hostitelům, musíte mít na svém počítači spuštěny démony **rpc.nfsd** a **rpc.mountd**. Protože se jedná o programy založené na balíku RPC, nejsou spravovány superserverem **inetd**, ale spouštějí se při zavádění systému a samy se registrují pomocí mapovače portů. Proto je můžete spustit až po spuštění démona **rpc.portmap**. Typicky v některém ze spouštěcích skriptů sítě zadáte něco podobného jako:

```
if [ -x /usr/sbin/rpc.mountd ]; then
        /usr/sbin/rpc.mountd; echo -n " mountd"
fi
if [ -x /usr/sbin/rpc.nfsd ]; then
        /usr/sbin/rpc.nfsd; echo -n " nfsd"
fi
```

Informace o vlastnictví, které poskytuje démon systému NFS svým klientům, obvykle obsahují pouze číselné identifikátory uživatelů a skupin. Pokud klient i server sdružují s těmito číselnými identifikátory shodné názvy uživatelů a skupin, říkáme, že sdílejí stejný uid/gid prostor. K tomu dochází například tehdy, pokud pomocí systému NIS distribuujete soubor `passwd` mezi všemi počítači na síti.

V některých případech se však tato čísla neshodují. Abyste nemuseli aktualizovat přiřazení uid a gid na straně klienta tak, aby odpovídalo serveru, můžete použít démona **rpc.ugidd**, který je schopen tyto nesrovnalosti řešit. Pomocí volby *map_daemon*, o které budeme hovořit dále, můžete démonu **rpc.nfsd** nařídit, aby ve spolupráci s démonem **rpc.ugidd** na straně klienta mapoval uid/gid prostor serveru na uid/gid prostor klienta. Démon **rpc.ugidd** se bohužel nenachází na všech moderních distribucích Linuxu, takže pokud jej potřebujete a vaše distribuce jej neobsahuje, budete si jej muset sami přeložit.

Démon **rpc.ugidd** je server založený na balíku RPC a spouští se ze síťových skriptů stejně jako démony **rpc.nfsd** a **rpc.mountd**:

```
if [ -x /usr/sbin/rpc.ugidd ]; then
        /usr/sbin/rpc.ugidd; echo -n " ugidd"
fi
```

Soubor exports

Nyní se podíváme na to, jak nakonfigurovat NFS server. Zejména se budeme zabývat tím, jak serveru řekneme, které souborové systémy mají být dostupné pro připojení a jak nastavit různé parametry, které řídí přístupová práva klientů k těmto souborovým systémům. Řízení přístupu k souborům na serveru si samozřejmě zajišťuje sám server. Soubor `/etc/exports` definuje souborové systémy, které bude server nabízet k připojení a použití.

Implicitně **rpc.mountd** nepovoluje připojení žádného adresáře, což je poměrně rozumný přístup. Chcete-li jednomu nebo více hostitelům povolit připojení adresáře pomocí systému NFS, musí být tento adresář *exportován*, což znamená, že musí být uveden v souboru `exports`. Vzorový soubor `exports` by mohl vypadat následovně:

```
# soubor exports pro vlager
/home             vale(rw) vstout(rw) vlight(rw)
/usr/X11R6        vale(ro) vstout(ro) vlight(ro)
/usr/TeX          vale(ro) vstout(ro) vlight(ro)
/                 vale(rw,no_root_squash)
/home/ftp         (ro)
```

Každý řádek definuje adresář a hostitele, kteří si ho mohou připojit. Název hostitele je obvykle zadán plně kvalifikovaným doménovým jménem, ale kromě toho může obsahovat i zástupné znaky * a ?, které fungují stejným způsobem, jako v příkazovém interpretu bash. Například adrese **lab*.foo.com** vyhovuje jak adresa **lab01.foo.com**, tak i adresa **laboratory.foo.com**. Hostitele je možné zadat také uvedením rozsahu IP adres ve formátu *adresa/maska*. Pokud není zadán žádný název hostitele, jako tomu bylo ve výše uvedeném příkladu s adresářem `/home/ftp`, potom si bude moci tento adresář připojit libovolný hostitel.

Při ověřování klienta pomocí souboru `exports` vyhledá démon **rpc.mountd** název hostitele klienta pomocí volání `gethostbyaddr`. V systému DNS vrátí toto volání kanonický název hostitele klienta, takže je nutné zajistit, aby se v souboru `exports` nepoužívaly žádné aliasy. V prostředí

NIS se vrací první nalezený název v databázi. Bez systémů NIS a DNS odpovídá vrácený název první vyhovující položce v souboru `hosts`.

Za názvem hostitele může následovat nepovinný seznam příznaků oddělených čárkami a uzavřený do kulatých závorek. Některé z možných příznaků jsou:

secure	Tento příznak vyžaduje, aby připojení bylo uskutečněno z rezervovaného portu, tedy z portu s číslem menším než 1024. Tato volba je zapnuta implicitně.
insecure	Tato volba vypíná příznak *secure*.
ro	Hierarchie souborového systému se připojí v režimu pouze pro čtení. Tato volba je implicitně zapnutá.
rw	Hierarchie souborového systému se připojí v režimu pro zápis i čtení
root_squash	Tato bezpečnostní vlastnost odebere superuživateli na vzdáleném počítači speciální práva na serveru tím, že požadavky s uid 0 na klientu mapuje na uid 65534 (tedy –2) na serveru. Toto uid by mělo patřit uživateli **nobody**.
no_root_squash	Nebude se přemapovávat klientské uid 0. Tato volba je implicitně zapnuta, takže superuživatel na klientském systému má práva superuživatele i na exportované adresáře.
link_relative	Tato volba převede absolutní symbolické odkazy (tedy takové odkazy, které začínají lomítkem) na relativní odkazy. Tato volba má smysl pouze v případě, kdy je připojen celý souborový systém hostitele, v opačném případě by se mohlo stát, že některé odkazy nebudou ukazovat nikam, anebo v horším případě na soubory, které by rozhodně neměly být vidět. Tato volba je implicitně zapnutá.
link_absolute	Zanechá všechny symbolické odkazy v původním stavu (toto je normální chování serverů NFS, které jsou dodávány firmou Sun).
map_identity	Tato volba říká serveru, že klient používá stejná uid a gid jako server. Tato volba je implicitně zapnutá.
map_daemon	Tato volba sdělí serveru NFS, že klient a server nesdílí společný uid/gid prostor. Démon **rpc.nfsd** pak pomocí dotazů na klientského démona **rpc.ugidd** vytvoří seznam mapující identifikátory uživatele id mezi klientem a serverem.
map_static	Tato volba umožňuje zadat soubor, který obsahuje statickou mapu uid a gid. Například zadáním `map_static=/etc/nfs/vlight.map` bude jako uid/gid mapa použit soubor `/etc/nfs/vlight.map`. Syntaxe souboru s mapou je popsána v manuálové stránce `exports`.
map_nis	Tato volba zajistí, že mapování uid a gid bude provádět server NIS.
anonuid a *anongid*	Tyto volby umožňují nastavit uid a gid anonymního účtu. Jedná se o užitečné nastavení pokud nějaký svazek exportujete veřejně.

Chyba zjištěná při analýze souboru `exports` bude při startu démonu **rpc.nfsd** nebo **rpc.mountd** zapsána prostřednictvím démona **syslogd** s úrovní *notice*.

Všimněte si, že názvy hostitelů jsou získány z IP adresy klienta pomocí zpětného mapování, takže je nutné mít správně nakonfigurovaný resolver. Pokud používáte službu BIND a zakládáte si na bezpečnosti, měli byste v souboru `host.conf` povolit kontrolu spoofingu. O této problematice hovoříme v kapitole 6.

Podpora serveru NFSv2 v jádře

Klasický NFS server běžící v uživatelském prostoru funguje spolehlivě, při vyšším zatížení má ale problémy s výkonem. Je to dáno zejména režií spojenou s použitím rozhraní pro systémová volání a také tím, že se dělí o čas s dalšími, často méně významnými procesy v uživatelském prostoru.

Jádro 2.2.0 obsahuje experimentální NFS server vyvinutý Olafem Kirchem a dále upravený H. J. Lu, G. Allan Morrisem a Trondem Myklebustem. Podpora NFS v jádře s sebou přináší podstatné zvýšení výkonu serveru.

V současné distribuci tento balík naleznete v připravené podobě. Pokud ne, najdete jej na adrese *http://csua.berkeley.edu/~gam3/knfsd/*. Abyste mohli nástroje serveru použít, musíte jádro přeložit se zahrnutím démona NFS. Zda jádro podporu NFS obsahuje zjistíte podle toho, zda existuje soubor `/proc/sys/sunrpc/nfsd_debug`. Pokud neexistuje, budete zřejmě muset nahrát modul **nfsd** pomocí nástroje **modprobe**.

Démon NFS v jádře používá standardní konfigurační soubor `/etc/exports`. Balík obsahuje vlastní verze démonů **rpc.mountd** a **rpc.nfsd**, které spouštíte prakticky stejně jako jejich protějšky v uživatelském prostoru.

Podpora serveru NFSv3 v jádře

V současné době je nejčastěji používanou vrzí NFS verze 2. Technologie nicméně pokročila a odhalila slabiny, které je možné řešit pouze přepracováním celého protokolu. NFS verze 3 podporuje větší soubory a souborové systémy, má podstatně zvýšenou bezpečnost a nabízí řadu výkonových zlepšení, která budou užitečná většině uživatelů.

Experimentální server NFSv3 vyvíjejí Olaf Kirch a Trond Myklebust. Je obsažen ve vývojovém jádře 2.3 a existuje i patch pro jádro 2.2. Ten je postaven na démonu NFSv2 jádra.

Patche jsou dostupné na domovské stránce NFS serveru pro jádro na adrese *http://csua.berkeley.edu/~gam3/knfsd/*.

KAPITOLA 15

IPX a souborový systém NCP

Ještě dlouho předtím, než se Microsoft dozvěděl něco o sítích a dokonce ještě předtím, než se o Internetu vědělo i mimo akademické kruhy, v různých firmách se používalo sdílení souborů a tiskáren prostřednictvím souborových a tiskových serverů založených na operačním systému Novell NetWare a jemu odpovídajících protokolů[105]. V řadě těchto prostředí dodnes existují sítě založené na těchto protokolech a je nutné integrovat jejich podporu s podporou novější rodiny TCP/IP.

Linux podporuje nejen protokoly TCP/IP ale i rodinu protokolů používaných operačním systémem NetWare společnosti Novell Corporation. Tyto protokoly jsou vzdálení bratranci protokolů TCP/IP a i když poskytují podobné skupiny služeb, v řadě aspektů se liší a jsou samozřejmě zcela nekompatibilní.

Linux obsahuje jak komerční, tak i volně dostupný software zajišťující integraci s produkty Novell.

V této kapitole si uvedeme stručný popis protokolů společnosti Novell, zaměříme se však hlavně na to, jak nastavit a používat volně dostupný software zajišťující spolupráci s produkty Novell.

Xerox, Novell a historie

Nejprve se podívejme na to, odkud se zmiňované protokoly vzaly a jak vypadají. Koncem sedmdesátých let společnost Xerox Corporation vyvinula a publikovala otevřený standard pojmenovaný Xerox Network Specification (XNS). Specifikace XNS popisovala skupinu protokolů určených obecně pro síťová prostředí se silnou orientací na použití na lokálních počítačových sítích. Specifikace obsahovala dva základní protokoly: Internet Datagram Protocol (IDP), který zajišťoval nespolehlivý přenos datagramů z hostitele na hostitele bez navazování spojení a protokol Sequenced Packed Protocol (SPP), což byla modifikovaná verze IDP zajišťující spolehlivý přenos s navázáním spojení. Datagramy na síti XNS byly adresovány individuálně. Adresační schéma používalo kombinaci 4bajtové síťové adresy IDP (která byla jednoznačně přiřazena každému síťovému segmentu) a 6bajtové adresy uzlu (tedy adresy síťové karty). Směrovače byla zařízení, která přepínala datagramy mezi dvěma nebo více samostatnými IDP sítěmi. IDP nepracoval s podsítěmi, každá nová skupina počítačů vyžadovala přidělení nové síťové adresy. Síťové adresy se volily tak, aby byly na celé propojené síti jednoznačné. V některých případech administrátoři vyvinuli konvence, kdy každý bajt síťové adresy kódoval nějakou informaci, například geografické umístění, takže sí-

105 Novell a NetWare jsou ochranné známky společnosti Novell Corporation.

ťové adresy byly alokovány určitým systematickým způsobem – nebyl to nicméně požadavek samotného protokolu.

Společnost Novell Corporation vyvinula na základě systému XNS vlastní síťový systém. Novell provedl drobné úpravy protokolů IDP a SPP a přejmenoval je na IPX (Internet Packet eXchange) a SPX (Sequenced Packet eXchange). Dále Novell doplnil vlastní protokoly, například NetWare Core Protocol (NCP), který s využitím protokolu IPX umožňoval sdílení souborů a tiskáren, nebo Service Advertisement Protocol (SAP), který umožňoval systémům na síti zjistit, jaké služby který hostitel nabízí.

Tabulka 15.1 představuje mapování protokolových rodin XNS, Novell a TCP/IP podle funkce jednotlivých protokolů. Vztahy mezi protokoly jsou pouze přibližné, nicméně měly by usnadnit pochopení dalšího výkladu v tom, že zhruba popisují zaměření jednotlivých protokolů.

Tabulka 15.1 – Vztahy protokolů rodin XNS, Novell a TCP/IP

XNS	Novell	TCP/IP	Funkce
IDP	IPX	UDP/IP	Nespolehlivý přenos bez navázání spojení
SPP	SPX	TCP	Spolehlivý přenos s navázáním spojení
	NCP	NFS	Souborové služby
	RIP	RIP	Výměna směrovacích informací
	SAP		Výměna informací o dostupných službách

IPX a Linux

První podporu protokolu IPX pro jádro Linuxu vyvinul Alan Cox v roce 1985[106]. V první verzi se tato podpora dala stěží použít pro víc než jen směrování IPX datagramů. Od té doby doplnila řada lidí, mezi nimi zejména Greg Page, mnohem rozsáhlejší podporu[107]. Greg vyvinul konfigurační nástroje, které budeme v této kapitole používat ke konfiguraci rozhraní. Volker Lendecke vyvinul podporu souborového systému NCP, která umožňuje na Linuxu připojovat svazky ze serveru Novell NetWare[108]. Vyvinul rovněž nástroje, které umožňují tisknout z Linuxu a na Linux. Ales Dryak a Martin Stover nezávisle vyvinuli démony serveru NCP, které umožňují klientům Novell NetWare připojovat adresáře linuxového počítače stejně jako to umožňuje démon NFS klientům protokolu NFS. Společnost Caldera Systems Inc. nabízí komerční plně licencovaný NetWare klient a server, který podporuje poslední standardy Novellu včetně podpory NetWare Directory Services[109].

V současné době tedy Linux nabízí široké spektrum služeb, které umožňuje Linux integrovat do stávajících sítí Novell.

106 Alana můžete kontaktovat na adrese *alan@lxorguk.ukuu.org.uk*.

107 Grega můžete kontaktovat na adrese *gpage@sovereign.org*.

108 Volkera můžete kontaktovat na adrese *lendecke@namu01.gwdg.org*.

109 Informace o tomto systému můžete najít na adrese *http://www.caldera.com*.

Caldera

I když nebudeme v této kapitole o systému Caldera nijak podrobně hovořit, musíme se o něm zmínit. Tento systém vyvinul Ray Noorda, bývalý pracovník společnosti Novell. Caldera NetWare je komerční produkt plně podporovaný společností Caldera. Tato společnost jej nabízí jako komponentu své vlastní distribuce Linuxu, pojmenovanou Caldera OpenLinux. Caldera je ideální metoda uvedení Linuxu do prostředí, která vyžadují jak komerční podporu, tak i schopnost integrace do stávajících nebo nových sítí Novell.

Podpora NetWare Caldera je plně licencována společností Novell, takže poskytuje vysokou jistotu, že produkty obou společností budou vzájemně spolupracovat. Jedinými dvěma výjimkami je čistý IP klientský režim a server NDS, kteréžto funkce nebyly v době vzniku tohoto textu k dispozici. Existuje i skupina správních nástrojů, které umožňují nejen spravovat NetWarové stroje na Linuxu, ale i samotné servery NetWare, a umožňují pro potřeby správy použít výhod mocných skriptovacích jazyků Linuxu. Další informace o produktu Caldera najdete na jeho webové stránce.

Podpora NDS

Společně s verzí NetWare 4 uvedl Novell novou funkci nazvanou NetWare Directory Service (NDS). Specifikace NDS ovšem není k dispozici bez přistoupení na smlouvu o zachování mlčenlivosti, což je omezení, které brání vývoji otevřené podpory. Jakousi podporu NDS obsahuje pouze verze 2.2.0 a vyšší balíku `ncpfs`, o kterém budeme hovořit dále. Tato podpora byla vyvinuta na základně zpětného rozboru protokolu NDS. Podpora zřejmě funguje, oficiálně je ale stále považována za experimentální. Se servery NetWare 4 můžete použít i nástroje bez podpory NDS za předpokladu, že server pracuje v režimu emulace bindery.

Caldera obsahuje plnou podporu NDS, protože jejich implementace je licencována Novellem. Tato implementace ovšem není volně dostupná. Nebudete tedy mít přístup ke zdrojovému kódu a nebudete moci software volně kopírovat a dále distribuovat.

Konfigurace jádra pro IPX a NCPFS

Konfigurace jádra pro podporu IPX a souborového systému NCP neobnáší nic jiného než vybrání správných voleb při sestavování jádra. Stejně jako u mnoha jiných částí jádra je možné i komponenty IPX a NCP buď vestavět přímo do jádra, nebo je přeložit jako moduly a nahrát příkazem **insmod** v době, kdy je budete potřebovat.

Aby Linux podporoval a směroval protokol IPX, musíte při sestavování jádra zvolit následující volby:

```
General setup  --->
    [*] Networking support

Networking options  --->
    <*> The IPX protocol

Network device support  --->
    [*] Ethernet (10 or 100Mbit)
        ... a odpovídající ovladače Ethernetu
```

Podpora souborového systému NCP, aby bylo možné v Linuxu připojit svazky Novell NetWare, vyžaduje následující volby:

```
Filesystems  --->
    [*] /proc filesystem support
    <*> NCP filesystem support (to mount NetWare volumes)
```

Po přeložení a instalaci nového jádra budete moci využít podpory protokolu IPX.

Konfigurace IPX rozhraní

Stejně jako u protokolu TCP/IP musíte před použitím rozhraní protokolu IPX nejprve nakonfigurovat. Protokol IPX má své specifické konfigurační požadavky, takže logicky existuje samostatná skupina konfiguračních nástrojů. Nyní si ukážeme, jak pomocí těchto nástrojů nakonfigurovat IPX rozhraní a trasy.

Podpora IPX síťovými zařízeními

Protokol IPX předpokládá, že jakákoliv skupina hostitelů, kteří mohou jeden druhému poslat IPX datagram bez směrování, patří do stejné IPX sítě. Všichni hostitelé na jednom segmentu Ethernetu budou patřit do stejné sítě IPX. Podobně (ale už méně intuitivně), dva hostitelé propojení sériovou linkou PPP musí patřit k IPX síti, kterou tvoří samotná sériová linka. V prostředí Ethernetu může být pro přenos IPX datagramů použito několik různých typů rámců. Typy rámců reprezentují různé protokoly Ethernetu a používají různé způsoby přenosu více protokolů po stejné ethernetové síti. Nejběžnějšími typy rámců jsou `802.2` a `Ethernet_II`. O typech rámců budeme dále hovořit v další části.

Protokol IPX v současné době v Linuxu podporují ethernetová zařízení a PPP zařízení. Aby bylo možné zařízení nakonfigurovat pro podporu IPX, musí být toto zařízení aktivní. Typicky konfigurujete ethernetové zařízení s podporou jak IP tak IPX, takže zařízení už existuje, ovšem pokud používáte čistě IPX síť, musíte příkazem **ifconfig** změnit status ethernetového zařízení takto:

```
# ifconfig eth0 up
```

Konfigurační nástroje IPX rozhraní

Greg Page vytvořil skupinu konfiguračních nástrojů pro IPX rozhraní, které se ve většině moderních distribucí dodávají už přeložené a kromě toho je můžete získat ve zdrojovém tvaru anonymním FTP přístupem ze systému *ftp://metalab.unc.edu/* v adresáři `/pub/Linux/system/filesystems/ncpfs/ipx.tgz`.

Typicky se nástroje IPX spouštějí při startu systému prostřednictvím RC skriptu. Pokud jste si dodávaný balík nástrojů nainstalovali, možná jej vámi používaná distribuce spouští automaticky.

Příkaz ipx_configure

Každé IPX rozhraní musí vědět, ke které IPX síti patří a který typ rámce má používat. Každý hostitel podporující IPX musí mít alespoň jedno rozhraní, jehož prostřednictvím se na něj budou ostatní hostitelé odkazovat, takzvané *primární* rozhraní. Podpora IPX v jádře Linuxu nabízí prostředky pro automatické nastavení těchto parametrů, zmíněná automatická podpora se zapíná a vypíná příkazem **ipx_configure**.

Bez dalších parametrů vypíše příkaz **ipx_configure** platné nastavení příznaků automatické konfigurace:

```
# ipx_configure
Auto Primary Select is OFF
Auto Interface Create is OFF
```

Příznaky Auto Primary i Auto Interface jsou standardně vypnuty. K jejich nastavení a aktivaci automatické konfigurace použijete následující příkaz:

```
# ipx_configure --auto_interface=on --auto_primary=on
```

Nastavení parametru `--auto_primary` na hodnotu `on` způsobí, že jádro automaticky zajistí, aby alespoň jedno z aktivních rozhraní hostitele fungovalo jako primární rozhraní.

Nastavení parametru `--auto_interface` na hodnotu `on` způsobí, že ovladač IPX v jádře bude poslouchat všechny rámce přijaté aktivními síťovými rozhraními a pokusí se zjistit adresu IPX sítě a používaný typ rámce.

Mechanismus automatické detekce funguje dobře na správně spravovaných sítích. Někdy si správcové sítě šetří práci a porušují pravidla, pak nemusí autodetekční kód fungovat správně. Nejběžnější taková situace nastává, pokud je jedna IPX síť nastavena tak, že po stejném Ethernetu komunikuje několika typy rámců. Technicky se jedná o neplatnou konfiguraci, protože hostitel s rámcem *802.2* nemůže přímo komunikovat s hostitelem s rámcem *Ethernet_II,* a tedy se nemohou nacházet na stejné IPX síti. Podpora IPX v Linuxu poslouchá na segmentu IPX datagramy, které se po něm přenášejí. Z nich se pokusí zjistit jaké síťové adresy se používají a jaké jim odpovídají typy rámců. Pokud se stejná síťová adresa používá pro více typů rámců nebo pro více rozhraní, linuxový kód to detekuje jako kolizi síťových adres a nebude schopen zjistit správný typ rámce. Že k tomu došlo poznáte podle následujícího hlášení:

```
IPX: Network number collision 0x3901ab00
eth0 etherII and eth0 802.3
```

Pokud takovýto problém zaznamenáte, vypněte funkci automatické detekce a nastavte rozhraní ručně pomocí příkazu **ipx_interface**, který je popsán v následující části.

Příkaz ipx_interface

Příkaz **ipx_interface** se používá k ručnímu přidávání, upravování a rušení IPX podpory u existujících síťových zařízení. Příkaz **ipx_interface** použijete v případě, že před chvílí popsaná automatická detekce nefunguje správně, nebo pokud na ni nechcete spoléhat. Příkaz **ipx_interface** vám umožňuje nastavit síťovou adresu IPX, status primárního rozhraní a typ rámce, který síť používá. Pokud budete vytvářet více IPX rozhraní, musíte pro každé použít jeden příkaz **ipx_interface**.

Syntaxe příkazu pro přidání IPX rozhraní je velmi jednoduchá a nejlépe se ukáže na příkladu. Přidejme IPX rozhraní k existujícímu ethernetovému zařízení:

```
# ipx_interface add -p eth0 etherII 0x32a10103
```

Jednotlivé parametry mají následující význam:

`-p`	Tento parametr říká, že rozhraní má fungovat jako primární. Jde o nepovinný parametr.
`eth0`	Název síťového zařízení, které doplňujeme o podporu IPX.
`etherII`	Tento parametr definuje typ použitého rámce, v našem případě Ethernet II. Kromě toho je možné zadat hodnoty `802.2`, `802.3` a `SNAP`.
`0x32a10103`	Adresa IPX sítě, které bude rozhraní náležet.

Následující příkaz odstraňuje z rozhraní podporu protokolu IPX:

```
# ipx_interface del eth0 etherII
```

A konečně, chcete-li zobrazit současné nastavení rozhraní, použijete příkaz:

```
# ipx_interface check eth0 etherII
```

Příkaz **ipx_interface** je podrobně popsán na své manuálové stránce.

Konfigurace IPX směrovače

Z krátkého popisu protokolů používaných v prostředí IPX si možná vzpomínáte, že IPX je směrovatelný protokol, který k šíření směrovacích informací využívá protokolu RIP (Routing Information Protocol). IPX verze protokolu RIP je dost podobná jeho IP verzi. Pracují prakticky stejným způsobem: směrovače periodicky vysílají obsah svých směrovacích tabulek a ostatní směrovače je přijímají a integrují do informací, které mají. Hostitelé potřebují vědět pouze to, jaká je jejich lokální síť a informace pro všechny jiné cíle odesílají přes svůj lokální směrovač. Za přenos a předání těchto datagramů na další skok na trase už odpovídá směrovač.

V prostředí IPX musí být po síti oznamován ještě další typ informací. Protokol SAP (Service Advertisement Protocol) přenáší informace o službách, které jednotliví hostitelé v síti poskytují. Právě díky protokolu SAP může uživatel získat seznam tiskových a souborových serverů na síti. Protokol SAP funguje tak, že hostitelé nabízející určité služby v pravidelných intervalech vysílají do sítě jejich seznam. Směrovače na IPX síti tyto informace shromažďují a společně se směrovacími informacemi je rozšiřují v dalších částech sítě. Směrovač protokolu IPX tedy musí šířit informace protokolů RIP i SAP.

Stejně jako pro protokol IP existuje na Linuxu i pro protokol IPX démon, **ipxd**, který zajišťuje úkony související se směrováním. A stejně jako v protokolu IP, samotné předávání datagramů mezi rozhraními protokolu IPX zajišťuje přímo jádro, které k tomu používá informace uložené ve směrovací tabulce IPX. Démon **ipxd** udržuje platné informace tím, že poslouchá na všech aktivních síťových rozhraních a v případě potřeby provede aktualizaci údajů ve směrovací tabulce. Kromě toho démon **ipxd** odpovídá na dotazy přímo připojených hostitelů, kteří požadují směrovací informace.

Příkaz **ipxd** je v některých distribucích k dispozici přímo přeložený, ve zdrojové podobě je k dispozici na adrese *http://metalab.unc.edu/* v adresáři `/pub/Linux/system/filesystems/ncpfs/ipxripd-x.xx.tgz`.

Démon **ipxd** nevyžaduje žádnou konfiguraci. Po svém spuštění démon automaticky udržuje trasy mezi nakonfigurovanými IPX zařízeními. Základem je, že před spuštěním démona **ipxd** musíte mít IPX zařízení správně nakonfigurována příkazem **ipx_interface**. Automatická detekce sice může fungovat, pokud ale poskytujete směrování, je lepší se na ni nespoléhat a nastavit síťová rozhraní ručně. Ušetříte si tím spoustu starostí s problémy ve směrování. Každých 30 sekund detekuje démon **ipxd** lokálně připojené IPX sítě a automaticky je spravuje. Díky tomu je možné provozovat síť i na rozhraních, která nemusejí být trvale aktivní, například na PPP rozhraních.

Démon **ipxd** se obvykle spouští při startu systému ze startovacích `rc` scriptů například takto:

```
# /usr/sbin/ipxd
```

Znak `&` není zapotřebí, protože **ipxd** se přepne na pozadí automaticky. Démon **ipxd** je nejužitečnější na počítačích, které fungují jako IPX směrovače, nicméně s výhodou se používá i na počíta-

čích na segmentech, které obsahují více IPX směrovačů. Zadáte-li parametr `-p`, bude démon **ipxd** pracovat pasivně, bude shromažďovat směrovací informace ze segmentu a aktualizovat směrovací tabulky, nebude však žádné směrovací informace vysílat. Díky tomu může hostitel mít aktuální směrovací tabulky trvale bez nutnosti ptát se směrovače pokaždé, když chce kontaktovat vzdáleného hostitele.

Statické směrování IPX příkazem ipx_route

V některých případech můžeme potřebovat určitou trasu definovat pevně. Stejně jako to jde u protokolu IP, je to možné i u protokolu IPX. Příkaz **ipx_route** zapíše trasu do směrovací tabulky IPX aniž by ji musel zjistit směrovací démon **ipxd**. Syntaxe příkazu je velmi jednoduchá (protože protokol IPX nepodporuje podsítě) a vypadá takto:

```
# ipx_route add 203a41bc 31a10103 00002a02b102
```

Tímto příkazem přidáme trasu na vzdálenou síť *203a41bc* přes směrovač na naší lokální síti *31a10103* s číslem uzlu *00002a02b102*.

Adresu uzlu směrovače můžete poněkud pracně zjistit příkazem **tcpdump** s parametrem `-e`, který vypisuje hlavičky linkové vrstvy a v nich hledat data od směrovače. Pokud je směrovač linuxový stroj, dá se jeho adresa zjistit daleko jednodušeji příkazem **ifconfig**.

Pomocí příkazu **ipx_route** můžete trasu vymazat takto:

```
# ipx_route del 203a41bc
```

Seznam tras, které jádro zná, můžete zjistit v souboru `/proc/net/ipx_route`. Naše směrovací tabulka vypadá zatím takto:

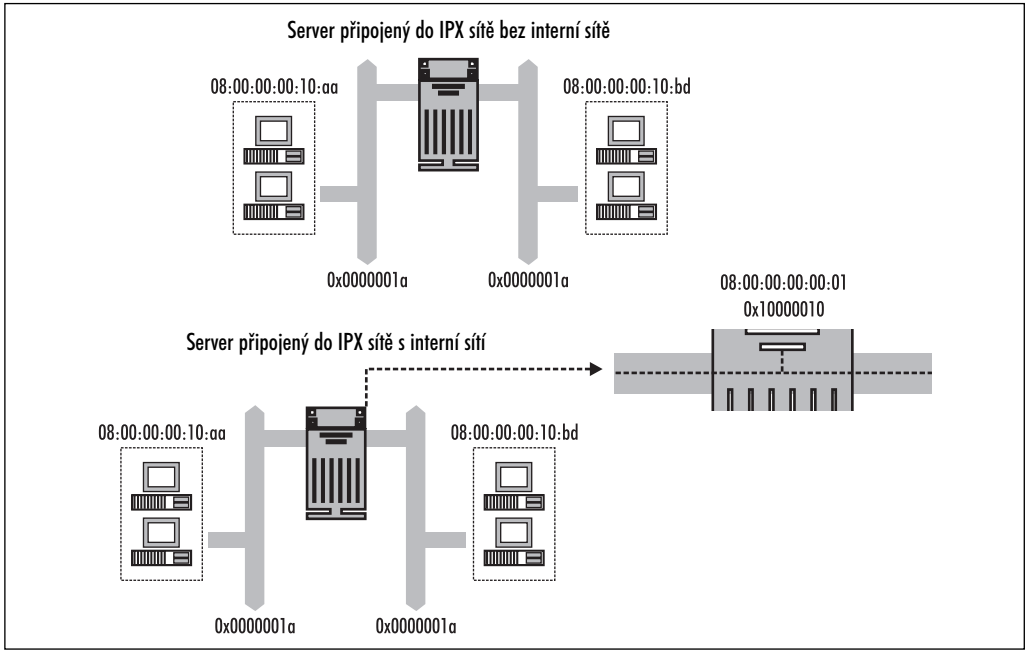
```
# cat ipx_route
Network    Router_Net   Router_Node
203A41BC   31A10103     00002a02b102
31A10103   Directly     Connected
```

Trasa na síť *31A10103* byla vytvořena automaticky při nakonfigurování IPX rozhraní. Tímto způsobem bude v souboru `/proc/net/ipx_route` uvedeno každé lokální rozhraní. Je jasné, že pokud má počítač fungovat jako směrovač, potřebuje přinejmenším dvě rozhraní.

Interní IPX sítě a směrování

Hostitelé s více než jedním IPX rozhraním používají pro každé z nich jednoznačnou kombinaci síťové adresy a adresy uzlu. Chcete-li se k takovému hostiteli připojit, můžete použít kteroukoliv z těchto kombinací. Když protokol SAP inzeruje službu, uvádí u nabízení služby také síťovou adresu a adresu uzlu. Pokud má hostitel více rozhraní, jedno z nich musí být zvoleno jako to, u nějž se služba bude inzerovat – to je právě smysl nastavení primárního rozhraní, o kterém jsme hovořili dříve. To ovšem znamená problém: trasa k tomuto rozhraní nemusí být vždy optimální a pokud by došlo k výpadku, který by odpojil daný síťový segment od ostatních, služba bude nedosažitelná i v případě, že *existují jiné* trasy k funkčním rozhraním. Tyto trasy ovšem ostatní hostitelé nikdy neznají, protože se neinzerují, a jádro nemá možnost dozvědět se, že má vybrat jiné primární rozhraní. Aby se tomuto problému předešlo, existuje speciální mechanismus, který umožňuje, aby byl hostitel pro potřeby protokolu SAP znám pod jedinou kombinací síť/uzel, která je nezávislá na trase. Tím se náš problém vyřeší, protože tato nová adresa je dosažitelná přes všechna síťová rozhraní a protokol SAP ji bude inzerovat. Abychom ilustrovali celý problém a jeho řešení, na obrázku

15.1 vidíme server připojený ke dvěma IPX sítím. V prvním případě nepoužíváme interní síť, ve druhém ano. Hostitel v prvním uspořádání použije jedno ze svých dvou rozhraní, řekněme rozhraní *0000001a:0800000010aa* jako primární a to bude inzerovat protokolem SAP. Pro hostitele na síti *0000001a* je to optimální, bohužel to však znamená, že hostitelé na síti *0000002c* budou své požadavky směrovat přes první zmíněnou síť i přesto, že server má na jejich síť přímé připojení jiným rozhraním.



Obrázek 15.1 – Interní IPX síť

Celý problém se vyřeší, pokud serveru vytvoříme virtuální síť s virtuální síťovou adresou, která představuje čistě softwarovou konstrukci. Tuto síť si můžeme nejlépe představit jako umístěnou *uvnitř* hostitele. Informace protokolem SAP se pak inzerují pouze pro tuto virtuální kombinaci síťové a uzlové adresy. Tato virtuální síť se označuje jako *interní síť*. Jak se ale ostatní hostitelé dozvědí, jak se s touto interní sítí spojit? Vzdálení hostitelé provádějí směrování na interní síť prostřednictvím síťového rozhraní, které je k jejich síti přímo připojeno. Znamená to, že pro hostitele s více síťovými rozhraními budete mít ve směrovací tabulce záznam s trasou na jeho interní síť. Tato trasa bude odpovídat optimální trase v daném okamžiku a pokud dojde k nějakému výpadku, protokol RIP zajistí její aktualizaci na (eventuální) náhradní trasu. Na obrázku 15.1 jsme nastavili interní síť s adresou *0x10000010* a s adresou uzlu *00:00:00:00:00:01*. Tato adresa bude představovat naše primární rozhraní a bude inzerována protokolem SAP. Směrování povede tuto síť jako dosažitelnou *oběma* ethernetovými rozhraními, takže jednotliví hostitelé budou vždy volit optimální trasu.

K vytvoření interní sítě použijeme příkaz **ipx_internal_net**, který je součástí IPX balíku Grega Page. Jeho použití je opět jednoduché:

```
# ipx_internal_net add 10000010 000000000001
```

Tento příkaz vytvoří interní IPX síť s adresou *10000010* a s adresou uzlu *000000000001*. Adresa interní sítě musí být stejně jako všechny ostatní adresy IPX sítí jedinečná. Adresa uzlu může být libovolná, protože na interní síti existuje pouze jediný hostitel. Každý hostitel může mít pouze jedinou interní síť a pokud ji má, bude tato síť vždy fungovat jako primární.

Interní síť můžete zrušit příkazem:

```
# ipx_internal_net del
```

Interní síť nemá naprosto žádný význam jindy než v případech, kdy hostitel poskytuje nějaké služby inzerované protokolem SAP a zároveň má více síťových rozhraní.

Připojení vzdáleného svazku NetWare

Protokol IPX se často používá k připojení svazků NetWare do souborového systému Linuxu. Tím je umožněno souborové sdílení dat mezi dalšími operačními systémy a Linuxem. Volker Lendecke vyvinul pro Linux NCP klienta a další nástroje, které sdílení dat umožňují.

V prostředí NFS připojujeme vzdálené souborové systémy příkazem **mount**. Souborový systém NCP má bohužel zcela specifické požadavky, které z praktických důvodů neumožňují zabudovat jeho podporu přímo do příkazu **mount**. Proto má Linux samostatný příkaz **ncpmount**. Příkaz **ncpmount** je jedním z nástrojů ve Volkerově balíku `ncpfs`, který je přímo k dispozici ve většině moderních distribucí Linuxu a ve zdrojové podobě je k dispozici na *ftp.gwdg.de* v adresáři `/pub/linux/misc/ncpfs`. V současné době je aktuální verze 2.2.0.

Před připojením svazků NetWare musíte mít správně nastavena IPX rozhraní (postupem popsaným dříve). Dále musíte znát údaje potřebné pro přihlášení na server NetWare, tedy uživatelské jméno a heslo. A konečně potřebujete vědět, který svazek chcete připojit a do kterého lokálního adresáře jej chcete připojit.

Jednoduchý příklad příkazu ncpmount

Jednoduchý příklad použití příkazu **ncpmount** vypadá takto:

```
# ncpmount -S ALES_F1 -U rick -P d00-b-gud /mnt/brewery
```

Tento příkaz připojí všechny svazky souborového serveru `ALES_F1` do adresáře `/mnt/brewery`, pro připojení k serveru NetWare použije účet `rick` s heslem `d00-b-gud`.

Příkaz **ncpmount** mívá normálně nastaven setuid bit a může jej tedy použít kterýkoliv uživatel. Dané spojení pak bude vlastnit tento uživatel a svazek může odpojit buď on nebo superuživatel.

Systém NetWare pracuje s termínem *svazek*, který je analogií souborového systému v Linuxu. Svazek NetWare je logickou reprezentací souborového systému NetWare serveru a může jít o jednu nebo více logických a fyzických diskových oblastí. Standardně chápe podpora NCPFS v Linuxu svazky jako podadresáře většího logického souborového systému, který je představován samotným serverem NetWare. Příkaz **ncpmount** způsobí, že jednotlivé svazky serveru NetWare se budou jevit jako samostatné adresáře pod připojovacím bodem. Je to pohodlné pro přístup k celému serveru, vzhledem ke komplikovaným technickým důvodům však nebudete moci tyto adresáře reexportovat systémem NFS. Složitější řešení, které tento problém obchází, si popíšeme za chvíli.

Podrobnosti o příkazu ncpmount

Příkaz **ncpmount** má celou řadu řádkových parametrů, které umožňují velmi pružně měnit vlastnosti připojení. Nejdůležitější parametry jsou popsány v tabulce 15.2.

Tabulka 15.2 – Parametry příkazu ncpmount

Parametr	Popis
–S *server*	Název serveru k němuž se připojujeme.
–U *uživatel*	Přihlašovací jméno uživatele na serveru NetWare.
–P *heslo*	Heslo uživatele na serveru NetWare.
–n	Tento parametr se uvádí u účtů, které nemají přiděleno heslo.
–C	Tento parametr vypíná automatickou konverzi hesel na velká písmena.
–c *klient*	Tento parametr umožňuje specifikovat kdo vlastní spojení se souborovým serverem. Je užitečný pro tisk, o kterém budeme hovoři za chvíli.
–u *uid*	Uživatelské ID Linuxu, které bude použito jako vlastník připojeného adresáře. Není-li tento parametr uveden, použije se ID uživatele, který spustil příkaz **ncpmount**.
–g *gid*	Identifikátor skupiny Linuxu, který bude použit jako vlastník připojeného adresáře. Pokud není uveden, použije se skupina uživatele, který spustil příkaz **ncpmount**.
–f *file_mode*	Tento parametr umožňuje specifikovat souborový režim (práva) souborů v připojeném adresáři. Hodnota se udává osmičkově, například 0664. Skutečná práva budou dána právy nastavenými tímto příkazem a maskovanými právy, které má uživatel zadaný parametrem `-U` na serveru NetWare. Abyste mohli k souborům přistupovat, musí mít tento uživatel dostatečná práva k serveru a jednotlivým souborům. Implicitně se používá hodnota odvozená od aktuální umasky.
–d *dir_mode*	Tento parametr umožňuje specifikovat adresářová práva v připojovaném adresáři. Chová se stejně jako parametr `-f` s tím rozdílem, že se implicitní práva neodvozují od aktuální umasky. Právo spouštění se přidělí pokud je přiděleno právo čtení.
–V *svazek*	Tento parametr umožňuje zadat název jednoho svazku NetWare, který se má k připojovacímu bodu připojit místo připojení všech svazků na cílovém serveru. Použití tohoto parametru je nezbytné pokud chcete připojený svazek dále reexportovat systémem NFS.
–t *timeout*	Tento parametr umožňuje definovat čas, po který bude klient NCPFS čekat na odpověď od serveru. Implicitní hodnota je 60 ms a nastavuje se v setinách sekundy. Pokud budete mít problémy se stabilitou připojení svazků, měli byste tento interval prodloužit.
–r *opakování*	Kód NCP klienta se pokouší odesílat datagramy na server opakovaně a teprve pak usoudí, že server je nedostupný. Tento parametr umožňuje změnit počet opakování ze standardních pěti.

Skrytí přihlašovacího hesla NetWare

Uvádět heslo přímo na přihlašovacím řádku, tak jak jsme to v příkazu **ncpmount** dělali, představuje jisté bezpečnostní riziko. Ostatní uživatelé totiž budou moci pomocí příkazů jako **top** nebo **ps** toto heslo spatřit. Aby se snížilo riziko toho, že vám jiní uživatelé zcizí přihlašovací heslo NetWare, příkaz **ncpmount** je schopen některá nastavení načíst z konfiguračního souboru v domovském adresáři uživatele. V tomto souboru můžete mít uložena uživatelská jména a hesla ke všem serverům, k nimž se chcete připojovat. Soubor se jmenuje `~/.nwclient` a aby jej ostatní uživatelé nemohli číst, musí mít nastavena práva `0600`. Pokud práva nejsou nastavena správně, příkaz **ncpmount** jej odmítne použít.

Soubor má velmi jednoduchou syntaxi. Všechny řádky začínající znakem # se chápou jako komentáře a ignorují se. Zbylé řádky používají následující formát:

```
název_serveru/uživatelské_jméno heslo
```

Název serveru definuje server, z nějž budete svazky připojovat. Uživatelské jméno je jméno, pod kterým se k tomuto serveru přihlašujete. Heslo je nepovinné. Pokud je neuvedete, příkaz **ncpmount** se vás při připojování svazku na heslo zeptá. Pokud je místo hesla uveden znak `-`, nepoužívá se žádné heslo – je to ekvivalent řádkového parametru `-n`.

Soubor může obsahovat více údajů, názvy serveru na jednotlivých řádcích však musí být jedinečné. K rozhodnutí o tom, který z údajů v souboru `~/.nwclient` použít, využívá příkaz **ncpmount** parametr `-s`. Pokud tento parametr nezadáte, použije se první server v souboru, který je tak považován za váš preferovaný server. Na první místo souboru byste proto měli uvést server, k němuž se připojujete nejčastěji.

Složitější příklad použití příkazu ncpmount

Podívejme se nyní na složitější příklad použití příkazu **ncpmount**, ve kterém budeme využívat některé z funkcí, které jsme si právě popsali. Nejprve si vytvoříme jednoduchý soubor `~/.nwclient`:

```
# Přihlašování k serverům v pivovaru a vinařství
#
# Pivovar
ALES_F1/MATT staoic1
#
# Vinařství
REDS01/MATT staoic1
#
```

Nezapomeňte správně nastavit přístupová práva:

```
$ chmod 600 ~/.nwclient
```

Nyní připojíme jeden svazek ze serveru vinařství do sdíleného adresáře, přičemž nastavíme taková souborová a adresářová oprávnění, aby ke svazku mohli přistupovat i jiní uživatelé:

```
$ ncpmount -S REDS01 -V RESEARCH -f 0664 -d 0775 /usr/share/winery/data/
```

Tento příkaz společně s výše uvedeným souborem `~/.nwclient` připojí svazek `RESEARCH` serveru `REDS01` do adresáře `/usr/share/winery/data`, přičemž použije NetWare účet `MATT` a heslo zjištěné ze souboru `~/.nwclient`. Práva připojených souborů budou `0664` a adresářů `0775`.

Práce s dalšími nástroji IPX

Balík `ncpfs` obsahuje celou řadu užitečných nástrojů, o kterých jsme zatím nehovořili. Řada z těchto nástrojů emuluje různé původní nástroje systému NetWare. V této části se budeme věnovat těm nejužitečnějším z nich.

Seznam serverů

Příkaz **slist** vypíše všechny servery, které jsou z našeho hostitele dostupné. Informace se ve skutečnosti zjišťují od nejbližšího IPX směrovače. Původně byl tento příkaz zřejmě určen k tomu, aby uživatelé mohli zjistit, ke kterým serverům se mohou připojit. Funguje však zároveň jako velmi užitečný diagnostický nástroj, protože umožňuje administrátorům zjistit, které informace jsou protokolem SAP šířeny:

```
$ slist
NPPWR-31-CD01                          23A91330  000000000001
V242X-14-F02                           A3062DB0  000000000001
QITG_284ELI05_F4                       78A20430  000000000001
QRWMA-04-F16                           B2030D6A  000000000001
VWPDE-02-F08                           35540430  000000000001
NMCS_33PARK08_F2                       248B0530  000000000001
NCCRD-00-CD01                          21790430  000000000001
NWGNG-F07                              53171D02  000000000001
QCON_7TOMLI04_F7                       72760630  000000000001
W639W-F04                              D1014D0E  000000000001
QCON_481GYM0G_F1                       77690130  000000000001
VITG_SOE-MAIL_F4R                      33200C30  000000000001
```

Příkaz **slist** nemá žádné parametry. Výstup obsahuje název serveru, jeho IPX adresu a adresu uzlu.

Poslání zprávy uživatelům systému NetWare

Systém NetWare podporuje mechanismus pro zasílání zpráv uživatelům. Jedná se o podobnou funkci, jakou má v Linuxu příkaz **nsend**. Abyste mohli zprávu odeslat, musíte být k serveru přihlášeni, takže na příkazovém řádku musíte kromě přihlašovacího jména adresáta a textu zprávy uvést také název serveru a vaše přihlašovací údaje:

```
# nsend -S vbrew_f1 -U gary -P j0yj0y supervisor\
      "Než se nám to vytiskne, pojďme na pivo!"
```

V tomto příkladu posílá uživatel `gary` lákavé pozvání uživateli `supervisor` na serveru `ALES_F1`. Pokud neuvedeme přihlašovací údaje, použije se implicitní server, účet a heslo.

Prohlížení a manipulace s daty v bindery

Každý server NetWare udržuje databázi informací o uživatelích a dalších nastaveních. Tato databáze se jmenuje *bindery*. Linux obsahuje nástroje, které umožňují údaje z této databáze číst a pokud máte práva správce serveru, můžete údaje i měnit a mazat. Přehled jednotlivých nástrojů je uveden v tabulce 15.3.

Tabulka 15.3 – Nástroje pro práci s bindery

Název příkazu	Popis
nwfstime	Zobrazí nebo nastaví datum a čas serveru NetWare.
nwuserlist	Vypíše uživatele přihlášené k serveru.
nwvolinfo	Zobrazí informace o svazcích NetWare.
nwbocreate	Vytvoří objekt bindery.
nwbols	Vypíše seznam objektů bindery.
nwboprops	Vypíše vlastnosti objektu bindery.
nwborm	Odstraní objekt bindery.
nwbpcreate	Vytvoří vlastnost objektu bindery.
nwbpvalues	Vypíše obsah vlastnosti objektu bindery.
nwbpadd	Nastaví obsah vlastnosti objektu bindery.
nwbprm	Odstraní vlastnost objektu bindery.

Tisk do front NetWare

Balík `ncpfs` obsahuje krátký nástroj nazvaný **nprint**, který umožňuje prostřednictvím NCP spojení posílat tiskové úlohy do tiskových front NetWare. Tento příkaz v případě potřeby vytvoří spojení se serverem (přičemž k připojení používá údaje ze souboru `~/.nwclient`, o kterém jsme už hovořili). Kromě toho je možné parametry připojení specifikovat řádkovými parametry stejnými jako u příkazu **ncpmount**, takže se o nich nebudeme zmiňovat. V dalších příkladech si popíšeme nejdůležitější parametry tohoto příkazu, podrobnosti najdete na manuálové stránce `nprint(1)`.

Jediným povinným parametrem příkazu **nprint** je název souboru, který se má vytisknout. Pokud je zadán název –, nebo pokud není zadán žádný název, **nprint** přijímá tiskovou úlohu z `stdin`. Nejdůležitější parametry programu **nprint** definují tiskový server a tiskovou frontu, kam chcete úlohu poslat. Přehled důležitých parametrů uvádí tabulka 15.4.

Tabulka 15.4

Parametr	Popis
-S *server_name*	Název serveru NetWare, který obsluhuje vámi používanou frontu. Bývá výhodné mít pro server záznam v `~/.nwclient`. Tento parametr je povinný.
-q *queue_name*	Tisková fronta do níž chcete tisknout. Tento parametr je povinný.
-d *job_description*	Text který se objeví na tiskové konzoli pro danou tiskovou úlohu.
-l *lines*	Počet řádků na stránku, standard je 66.
-r *columns*	Počet sloupců na stránku, standard je 80.
-c *copies*	Počet tisknutých kopií, standard je 1.

Jednoduché použití příkazu **nprint** by vypadalo takto:

```
$ nprint -S REDS01 -q PSLASER -c 2 /home/matt/ethylene.ps
```

Tento příkaz vytiskne dvě kopie souboru `/home/matt/ethylene.ps` na tiskárně `PSLASER` serveru `REDS01` a použije uživatelské jméno a heslo uvedené v souboru `~/.nwclient`.

Použití příkazu nprint s tiskovým démonem

Možná si vzpomenete, že jsme se zmiňovali, že volba `-c` programu **ncpmount** je užitečná pro tisk. Konečně si vysvětlíme proč a jak.

Linux obvykle používá řádkové tiskové programy typu BSD. Tiskový démon (**lpd**) je démon kontrolující lokální tiskový adresář a hledá v něm soubory, které se mají vytisknout. **lpd** si přečte název tiskárny a případné další parametry ze speciálně naformátovaného souboru v tiskovém adresáři a zapíše data na tiskárnu, případě je přitom zpracuje nějakým filtrem, který je transformuje nebo jinak upraví.

Démon **lpd** používá jednoduchou databázi pojmenovanou `/etc/printcap` v níž má uloženy informace o konfiguraci tiskáren včetně používaných filtrů. Obvykle tento démon běží se speciálními právy uživatele *lp*.

Program **nprinter** můžete nakonfigurovat jako filtr pro démona **lpd**, takže uživatelé na Linuxu budou moci přímo tisknout na vzdálené tiskárny serveru NetWare. Aby to bylo možné, uživatel *lp* musí mít právo tisknout na serveru NetWare.

Jednoduchá možnost jak to zajistit bez nutnosti aby uživatel *lp* vytvářel vlastní spojení se serverem NetWare a přihlašoval se k němu je prohlásit *lp* za vlastníka spojení vytvořeného jiným uživatelem. Úplný příkad toho jak nastavit tiskový systém Linuxu pro tisk na serveru NetWare se dá shrnout do tří kroků:

1. Napište obalovací skript:

   Soubor `/etc/printcap` nedefinuje práva předávaná filtrům. Proto je nutné napsat krátký skript spouštějící příkaz společně s potřebnými volbami. Tento skript může být velmi jednoduchý:

   ```
   #!/bin/sh
   # p2pslaser - skript pro přesměrování stdin na
   # frontu PSLASER na serveru REDS01
   #
   /usr/bin/nprint -S REDS01 -U stuart -q PSLASER
   #
   ```

   Skript uložte jako `/usr/local/bin/p2pslaser`.

2. Vytvořte záznam v `/etc/printcap`.

   V souboru `/etc/printcap` musíme právě vytvořený skript `p2pslaser` nastavit jako výstupní filtr. Bude to vypadat asi takto:

   ```
   pslaser|Postscript Laser Printer na serveru NetWare:\
   :lp=/dev/null:\
   :sd=/var/spool/lpd/pslaser:\
   :if=/usr/local/bin/p2pslaser:\
   :af=/var/log/lp-acct:\
   :lf=/var/log/lp-errs:\
   :pl#66:\
   :pw#80:\
   :pc#150:\
   :mx#0:\
   :sh:
   ```

3. V příkazu **ncpmount** zadejte parametr `-c`:

```
ncpmount -S REDS01 .... -c lp ....
```

Lokální uživatel *stuart* musí při připojení k serveru NetWare definovat uživatele *lp* jako vlastníka spojení.

Kterýkoliv uživatel nyní může použít tiskárnu s názvem `pslaser` a použít tiskového démona. Příslušná úloha se pošle na server NetWare.

Správa tiskové fronty

Příkazem **pqlist** vypíšete seznam všech tiskových front, které jsou vám na zadaném serveru k dispozici. Pokud na příkazovém řádku pomocí parametru `-s` nezadáte server nebo jméno a heslo, převezmou se ze souboru `~/.nwclient`:

```
# pqlist -S vbrew_f1 -U guest -n
Server: ALES_F1
Print queue name                                        Queue ID
--------------------------------------------------------------
TEST                                                    AA02009E
Q2                                                      EF0200D9
NPI223761_P1                                            DA03007C
Q1                                                      F1060004
I-DATA                                                  0D0A003B
NPI223761_P3                                            D80A0031
```

Náš příklad ukazuje seznam front dostupných uživateli `guest` na serveru `ALES_F1`[110].

Seznam úloh ve frontě získáte příkazem **pqstat**. Jako parametr zadáte jméno fronty a příkaz vypíše všechny úlohy v této frontě. Nepovinně můžete zadat i další parametry a specifikovat například, kolik úloh chcete vypsat. Následující příklad výstupu jsme poněkud upravili, aby se vešel do knihy:

```
$ pqstat -S ALES_F1 NPI223761_P1

Server: ALES_F1     Queue: NPI223761_P1          Queue ID: 6A0E000C
   Seq  Name      Description                    Status   Form  Job ID
-------------------------------------------------------------------------
     1  TOTRAN    LyX document - proposal.lyx    Active      0  02660001
```

Ve frontě vidíme jedinou úlohu vlastněnou uživatelem `TOTRAN`. Další údaje obsahují popis úlohy, její stav a identifikátor.

Příkazem **pqrm** je možné odstranit úlohu z fronty. K odstranění úlohy, kterou jsme si právě vypsali, použijeme příkaz

```
$ pqrm -S ALES_F1 NPI223761_P1 02660001
```

Příkaz je velmi jednoduchý, pokud jej ale chcete zadat rychle, používá se špatně. Je rozumné napsat si jednoduchý skript, který rušení úloh z fronty usnadní.

110 Vypadá to, jako kdyby správce serveru NetWare před pojmenováváním front absolvoval exkurzi v našem virtuálním pivovaru. Doufejme, že fronty na vašem serveru se jmenují rozumněji.

Emulace serveru NetWare

Pro Linux existují dva volně dostupné emulátory serveru NetWare. **lwared** vyvinul Aleš Dryák, **mars_nwe** vyvinul Martin Stover. Oba tyto balíky zajišťují pod Linuxem základní emulaci serveru NetWare a umožňují klientům NetWare připojit si linuxové svazky stejně jako by je nabízel server NetWare. Server **lwared** se snadno konfiguruje, naopak **mars_nwe** je podstatně schopnější. Instalace a konfigurace těchto produktů je mimo náplň tohoto textu, oba jsou však popsány v dokumentu IPX-HOWTO.

Správa programu Taylor UUCP

Historie

Protokol UUCP byl navržen koncem sedmdesátých let Mikem Leskem ze společnosti AT&T Bell Laboratories. Tento protokol měl poskytovat jednoduché připojení k síti pomocí veřejných telefonních linek. Navzdory oblibě protokolů PPP a SLIP stále řada lidí, kteří chtějí mít na svém domácím počítači elektronickou poštu a usenetové news, používá protokol UUCP, protože je často levnější (zejména v zemích, kde se místní hovory tarifikují časově, nebo v případě, kdy lokální poskytovatel neexistuje a pro připojení se volá meziměsto). I přes vysoký počet implementací tohoto protokolu, které lze provozovat na širokém spektru hardwarových platforem a operačních systémů, jsou tyto implementace navzájem vysoce kompatibilní.

Nicméně jako u většiny softwaru, u něhož se až v průběhu let vyvinul určitý „standard“, neexistuje ani u protokolu UUCP taková implementace, která by se nazývala jednoduše *UUCP*. Protokol UUCP procházel od své první verze, jež byla uvedena v roce 1976, neustálým evolučním vývojem. V současné době existují dva hlavní druhy tohoto protokolu, které se navzájem značně odlišují jak v podporovaném hardwaru, tak i ve své vlastní konfiguraci. Z těchto dvou druhů protokolu UUCP se vyvinula spousta různých implementací, které se od svých předků liší jen minimálně.

Jeden druh se nazývá „protokol UUCP verze 2“ a jeho vznik je datován do roku 1977, kdy Mike Lesk, David A. Novitz a Greg Chesson vytvořili novou implementaci protokolu UUCP. I když je tato implementace poměrně stará, stále se hojně používá. Novější implementace verze 2 poskytuje spoustu vymožeností obsažených v novějších druzích protokolu UUCP.

Druhý typ protokolu byl vytvořen v roce 1983 a běžně se označuje názvy BNU (Basic Networking Utilities – základní síťové utility) nebo HoneyDanBer UUCP, zkráceně HDB. Tento název je odvozen ze jmen autorů, P. Honeymana, D. A. Novitze a B. E. Redmana. Protokol HDB byl vymyšlen proto, aby odstranil některé nedostatky protokolu UUCP verze 2. Byly například přidány nové transportní protokoly a odkládací adresář byl rozdělen tak, že v současnosti existuje jeden adresář pro každý systém, se kterým komunikujete na bázi protokolu UUCP.

V současnosti se společně s operačním systémem Linux dodává implementace protokolu UUCP pod názvem Taylor UUCP verze 1.06[111]. Protokol Taylor UUCP verze 1.06 byl uvolněn v srpnu roku 1995. Kromě tradičních konfiguračních souborů ovládá i konfigurační soubory nové generace – takzvané Taylorovy konfigurační soubory.

111 Napsal jej Ian Taylor v roce 1995.

Ve většině distribucí Linuxu je obvykle protokol UUCP přeložen tak, aby byl buď kompatibilní s utilitami BNU nebo kompatibilní s Taylorovým konfiguračním schématem, případně s oběma verzemi. Protože je Taylorovo konfigurační schéma mnohem flexibilnější a pravděpodobně i snáze pochopitelné, než často nesrozumitelné konfigurační soubory utilit BNU, budeme v této kapitole popisovat Taylorovo konfigurační schéma.

Účelem této kapitoly není poskytnout vyčerpávající popis všech parametrů pro všechny existující příkazy protokolu UUCP ani popis funkce všech těchto příkazů, nýbrž úvod do problematiky nastavení funkčního uzlu UUCP. V první části popisujeme implementaci vzdáleného spouštění a přenosu souborů pomocí protokolu UUCP. Nejste-li úplným nováčkem v oblasti práce s protokolem UUCP, můžete tuto kapitolu přeskočit a přejít na část zabývající se konfiguračními soubory protokolu UUCP, která vysvětluje použití různých souborů pro nastavení protokolu UUCP.

Budeme předpokládat, že znáte programy z balíku UUCP, konkrétně programy **uucp** a **uux**. Popis těchto příkazů naleznete na manuálových stránkách.

Kromě veřejně dostupných programů **uux** a **uucp** obsahuje balík UUCP také množství příkazů, které se používají pouze k administrativním účelům. Používají se k monitorování UUCP provozu ve vašem uzlu, k odstraňování starých souborů se záznamy nebo k sestavování statistik. Ani jeden z těchto příkazů zde nebudeme rozebírat, protože nejsou pro hlavní činnost protokolu UUCP důležité. Kromě toho je jejich dokumentace na velmi slušné úrovni a člověk jim snadno porozumí. Existuje však ještě třetí kategorie, která představuje skutečnou výkonnou část protokolu UUCP. Tyto programy se nazývají **uucico** (kde zkratka *cico* znamená copy-in copy-out) a **uuxqt**, který spouští úkoly přijaté ze vzdálených systémů. V této kapitole se budeme věnovat právě těmto důležitým souborům.

Ti, co nenajdou vše potřebné v této kapitole, by si měli přečíst dokumentaci, která je součástí balíku UUCP. Tato dokumentace se skládá ze souborů ve formátu Texinfo, popisujících nastavení při použití Taylorova konfiguračního schématu. Soubory ve formátu Texinfo je možné převést na formát **dvi** pomocí nástroje **texi2dvi** (který je součástí balíku Texinfo) a soubory **dvi** je možné prohlížet příkazem **xdvi**.

Dalším dobrým zdrojem informací o UUCP v prostředí Linuxu je dokument UUCP-HOWTO Guylhema Aznara. Je k dispozici na stránkách linuxového dokumentačního projektu a pravidelně se objevuje ve skupině comp.os.linux.answers.

Dále existuje diskusní skupina zabývající se protokolem UUCP, která se nazývá comp.mail.uucp. Máte-li specifické dotazy ohledně Taylorova konfiguračního schématu, bude lepší, když se na ně zeptáte přímo v této diskusní skupině, než ve skupinách comp.os.linux.*.

Přenos souborů a vzdálené spouštění

Pro pochopení protokolu UUCP je důležité porozumět konceptu úloh. Každý přenos iniciovaný uživatelem pomocí příkazů **uucp** nebo **uux** se nazývá úloha. Ten se skládá z příkazu, který má být spuštěn na vzdáleném systému, ze souborů, které mají být mezi propojenými systémy přeneseny, popřípadě z obojího.

Jako příklad budeme předpokládat, že jste na svém hostiteli spustili následující příkaz, který sdělí protokolu UUCP, aby překopíroval soubor `netguide.ps` na hostitele *pablo* a aby na něm spustil příkaz **lpr**, který tento soubor vytiskne:

```
$ uux -r pablo!lpr !netguide.ps
```

Protokol UUCP se obecně nespojuje se vzdáleným systémem hned (okamžitě to lze provést pomocí programu **kermit**). Místo toho dočasně uloží popis dané úlohy. Tento proces se nazývá *dočasné odkládání*. Strom s adresáři, ve kterém jsou uloženy jednotlivé úlohy, se nazývá *dočasný adresář* a je zpravidla umístěn v adresáři `/var/spool/uucp`. V našem příkladu budou informace o úloze obsahovat vzdálený příkaz (**lpr**), který se má spustit, jméno uživatele, který o toto spuštění žádá, a několik dalších položek. Kromě popisů jednotlivých úloh musí protokol UUCP uložit také kopírovaný soubor, tedy soubor `netguide.ps`.

Přesné umístění a názvy dočasných odkládacích souborů se mohou lišit v závislosti na nastavení některých voleb při sestavování. Protokol UUCP verze HDB zpravidla dočasně ukládá soubory v adresáři s názvem `/var/spool/uucp/`*`site`*, kde *site* je název vzdáleného systému. Když je protokol UUCP přeložen s podporou Taylorova konfiguračního schématu, vytvoří protokol UUCP pro různé typy dočasně odkládaných souborů podadresáře v dočasném adresáři konkrétního systému.

Protokol UUCP se pak v pravidelných intervalech spojuje se vzdáleným systémem. Když se uskuteční spojení se vzdáleným systémem, přenese protokol UUCP soubory popisující danou úlohu a všechny vstupní soubory. Příchozí úlohy nebudou spuštěny okamžitě, ale až po skončení spojení. To provede příkaz **uuxqt**, který se rovněž stará o doručení úloh, které jsou určeny pro jiný systém.

Aby protokol UUCP rozlišil důležité a méně důležité úkoly, přiřazuje každé úloze *prioritu*. Priorita je definována jediným znakem v rozmezí od 0 do 9, od A do Z a od a do z. Hodnota 0 odpovídá nejvyšší prioritě, hodnota z odpovídá prioritě nejnižší. Pošta je obvykle dočasně ukládána s prioritou B nebo C, zatímco news jsou ukládány s prioritou N. Úlohy s vyšší prioritou jsou přenášeny dříve. Prioritu můžete přiřadit příkazům **uucp** a **uux** pomocí parametru `-g`.

V určitých časových intervalech můžete také zabránit přenášení úloh, které mají nižší prioritu, než zadáte. Tato vlastnost se také označuje jako takzvaná *maximální odkládací priorita* povolená při komunikaci a implicitně je nastavena na hodnotu z, což znamená, že se přenese vše. Všimněte si této terminologické dvojsmyslnosti: soubor je přenesen pouze v případě, že má prioritu *vyšší* nebo *shodnou* s maximální odkládací prioritou.

Vnitřní funkce programu uucico

Abychom pochopili, proč potřebuje program **uucico** znát určitá data, bude vhodné uvést rychlý popis toho, jakým způsobem se tento program ve skutečnosti spojuje se vzdáleným systémem.

Když na příkazové řádce spustíte příkaz **uucico –s *system***, musí se příkaz **uucico** nejprve fyzicky připojit. Provedené akce budou záviset na typu navazovaného spojení – například při použití telefonní linky musí příkaz **uucico** nejprve najít modem a vytočit číslo. U spojení pomocí protokolu TCP musí nejprve zavolat funkci `gethostbyname`, aby převedl název hostitele na síťovou adresu, poté musí zjistit port, který se má otevřít, a přiřadit adresu odpovídajícímu socketu.

Po navázání spojení je třeba provést ověření totožnosti. To se zpravidla skládá z výzvy vzdáleného systému, aby bylo zadáno přihlašovací jméno a volitelně i heslo. Tento proces se obecně nazývá *login chat*. Procedura ověření totožnosti se buď provede pomocí běžného balíku **getty/login**, nebo při použití protokolu TCP vlastním programem **uucico**. Pakliže je ověření totožnosti úspěšné, spustí se na vzdáleném konci program **uucico**. Místní kopie programu uucico, která byla původcem spojení, se označuje jako *řídicí* a vzdálená kopie se označuje jako *řízená*.

Dále následuje takzvaná *handshake fáze*: řídicí program pošle název svého hostitele společně s několika příznaky. Řízený program ověří povolení k přihlášení, k posílání a příjmu souborů a podobně. Příznaky popisují (kromě jiných vlastností) maximální prioritu dočasně ukládaných souborů, které se budou přenášet. Pokud je povolena, uskuteční se dále kontrola počtu hovorů, takzva-

ný *sequence number check*. Tato volba způsobí, že si budou oba systémy hlídat počet úspěšných spojení a ty pak porovnají. Pokud si počty úspěšných spojení neodpovídají, pak inicializační fáze neuspěje. Tato vlastnost je užitečná při ochraně vašeho systému před potenciálními podvodníky.

Nakonec se oba programy uucico pokusí dohodnout na společném *přenosovém protokolu*. Tento protokol řídí způsob, jakým jsou přenášena data, jak se ověřuje konzistence dat a jak se v případě chyby posílají data znovu. Více přenosových protokolů je zapotřebí z důvodu odlišných podporovaných typů spojení. Například telefonní linky vyžadují „bezpečný" protokol, který je z hlediska výskytu chyb značně pesimistický, zatímco přenos pomocí protokolu TCP je již ze své podstaty spolehlivý, a proto může používat efektivnější protokol, který vynechává většinu dodatečných detekcí chyb.

Po skončení handshake fáze začne samotný přenos. Oba konce zapnou ovladač vybraného protokolu. Ovladače mohou eventuálně provést inicializační sekvenci, která závisí na typu vybraného protokolu.

Nejprve pošle řídicí program vzdálenému systému všechny soubory, které čekají ve frontě a jejichž priorita dočasného uložení je dostatečně veliká. Když tento přenos dokončí, bude řízený program informován o ukončení přenosu a o tom, že eventuálně může zavěsit. Řízený program nyní může buď souhlasit se zavěšením, nebo může převzít řízení komunikace. Takže de facto dojde k výměně rolí: nyní se program na vzdáleném systému stane řídicím a program na místním hostiteli se stane řízeným. Nový řídicí program nyní pošle své soubory. Po skončení tohoto přenosu souborů si oba programy **uucico** vymění zavěšující řetězce a spojení se přeruší.

Zajímají-li vás další podrobnosti, nahlédněte buď do zdrojového kódu programu, anebo do nějaké kvalitní knihy, která se zabývá problematikou protokolu UUCP. Kromě toho se někde po Internetu pohybuje poměrně starý článek od Davida A. Novitze, ve kterém najdete detailní popis protokolu UUCP[112]. Informační bulletin FAQ k protokolu Taylor UUCP také obsahuje některé podrobnosti ohledně způsobu implementace protokolu UUCP. Pravidelně je posílán na adresu comp.mail.uucp.

Řádkové parametry programu uucico

V této části popisujeme nejdůležitější parametry příkazu **uucico**.

`- - system`, `-s` *system*	Zavolá uvedený systém, pokud to není v době, kdy je to zakázáno
`-S` *system*	Bez jakýchkoliv podmínek zavolá uvedený systém system.
`- -master`, `-r1`	Spustí program **uucico** v řídícím režimu. Tento režim je implicitní, použijete-li parametry `-s` nebo `-S`. Samotná volba `-r1` způsobí, že se program **uucico** pokusí zavolat všechny systémy uvedené v souboru *sys* (popsaném dále), pokud toto volání není zakázáno časovým rozmezím pro volání nebo dobou pro opětovný pokus.
`- -slave`, `-r0`	Spustí program **uucico** v řízeném režimu. Tento režim je implicitní, pokud nejsou zadány parametry `-s` nebo `-S`. V řízeném režimu se předpokládá, že standardní vstup/výstup bude připojen buď na sériový port, nebo se použije port pro protokol TCP zadaný pomocí volby `-p`.
`- -ifwork`, `-c`	Tato volba je doplňující k `-s` nebo `-S` a říká, že se má provést spojení se vzdáleným systémem pouze v případě, že jsou pro něj nějaké úlohy.

112 Najdete jej i v manuálu správce systému 4.4BSD.

`- -debug` *type*, `-x` *type*, `-X` *type*

Zapne ladění zadaného typu. Pomocí seznamu odděleného čárkami může být zadáno několik typů ladění. Platné jsou následující typy: *abnormal*, *chat*, *handshake*, *uucp-proto*, *proto*, *port*, *config*, *spooldir*, *execute*, *incoming*, *outgoing*. Použijete-li klíčové slovo *all*, zapnou se všechny volby. Z důvodu kompatibility s ostatními implementacemi protokolu UUCP může být zadáno místo názvu číslo, které zapíná ladění prvních n položek z výše uvedeného seznamu.

Ladící informace budou zapsány do souboru `Debug`, který se nachází v adresáři `/var/spool/uucp`.

Konfigurační soubory protokolu UUCP

Na rozdíl od jednoduchých programů pro přenos souborů byl protokol UUCP navržen tak, aby se uměl automaticky postarat o všechny přenosy. Když se vám povede tento protokol správně nastavit, nebude nutný každodenní zásah ze strany správce systému. Informace potřebné k automatickému přenosu jsou umístěny v několika konfiguračních souborech, které se nacházejí v adresáři `/usr/lib/uucp`. Většina z těchto souborů je používána pouze při volání směrem z vašeho systému.

Úvod do Taylorova UUCP

Řekneme-li, že konfigurace UUCP je náročná, tak to ještě pořád není dostačující. Jde skutečně o nepříjemnou záležitost a stručný formát konfiguračních souborů vám to rozhodně neulehčí (i když Taylorův formát je v porovnání se staršími formáty protokolů HDB nebo verze 2 poměrně čitelný).

Abychom si ukázali jak spolu všechny konfigurační soubory souvisí, představíme si alespoň nejdůležitější z nich a podíváme se na vzorové položky těchto souborů. Teď se nebudeme zabývat detaily; přesnější vysvětlení najdete v následujících samostatných statích. Chcete-li, aby váš počítač podporoval protokol UUCP, bude nejlepší začít s několika vzorovými soubory, které budete postupně upravovat. Jako vzorové soubory můžete použít níže uvedené příklady nebo soubory, které jsou součástí vaší oblíbené distribuce operačního systému Linux.

Všechny soubory popisované v této stati jsou uloženy v adresáři `/etc/uucp` nebo v některém z jeho podadresářů. Některé distribuce Linuxu obsahují binární soubory protokolu UUCP, které mají přeloženou podporu jak pro konfiguraci podle standardu HDB, tak i pro Taylorovu konfiguraci, a pro každý druh konfiguračních souborů používají jiné podadresáře. V adresáři `/usr/lib/uucp` se obvykle nachází soubor `README`.

Aby protokol UUCP správně fungoval, musí být vlastníkem jeho souborů uživatel **uucp**. Některé z těchto souborů obsahují hesla a telefonní čísla, a proto by měly mít nastavena přístupová práva 600. Prakticky většina příkazů protokolu UUCP musí mít nastaveno setuid bit, nicméně program **uuchk** takto nastaven být *nesmí*. V opačném případě by si mohli uživatelé prohlížet všechna hesla i přesto, že konfigurační soubory budou mít režim 600.

Ústředním souborem protokolu UUCP je soubor `/etc/uucp/config`, který se používá pro nastavení obecných parametrů. Nejdůležitějším z těchto parametrů (a momentálně i jediným důležitým parametrem) je název vašeho hostitele protokolu UUCP. Ve společnosti Virtual Brewery používají jako bránu protokolu UUCP hostitele **vstout**:

```
# /etc/uucp/config - hlavní konfigurační soubor UUCP
nodename         vstout
```

Dalším důležitým konfiguračním souborem je soubor `sys`. V něm jsou obsaženy veškeré informace týkající se systémů, s nimiž jste propojeni. Tyto informace obsahují název systému a informace o vlastním spojení, například telefonní číslo používané při modemovém spojení. Typický záznam pro systém s názvem **pablo** připojený pomocí modemu vypadá asi takto:

```
# /usr/lib/uucp/sys - sousední počítače UUCP
#
system   pablo
time     Any
phone    555-22112
port       serial1
speed      38400
chat       ogin: vstout ssword: lorca
```

Pole *time* určuje časy, ve kterých se lze s daným systémem spojit. Pole *chat* popisuje přihlašovací komunikační skripty – souslednost řetězců, které si musí systémy navzájem vyměnit dříve, než se může program **uucico** přihlásit k hostiteli **pablo**. Ke komunikačním skriptům se vrátíme později. Položka *port* pouze definuje záznam v souboru `port`. (Viz obrázek 16.1.) Můžete použít libovolný název, ale musí odkazovat na korektní záznam v souboru `port`.

Soubor `port` obsahuje informace týkající se vlastního spojení. U modemových spojení tento soubor popisuje speciální soubor používaného zařízení, rozsah podporovaných rychlostí a typ volacího zařízení, které je připojeno k danému portu. Níže uvedený záznam popisuje zařízení `/dev/ttyS1` (což odpovídá sériovému portu COM 2), ke kterému je připojen modem NakWell, jenž je schopný pracovat s rychlostmi až do 38400 bps. Název záznamu byl zvolen tak, aby odpovídal názvu portu, který je uveden v souboru `sys`.

```
# /etc/uucp/port - porty UUCP
# /dev/ttyS1 (COM2)
port           serial1
type           modem
device  /dev/ttyS1
speed          38400
dialer  nakwell
```

Informace vztahující se k vytáčecímu zařízení jsou uchovávány v dalším speciálním souboru, jehož název jistě uhodnete – `dial`. Pro každý typ vytáčeného zařízení obsahuje tento soubor sekvenci příkazů, které musí být spuštěny, aby se pomocí zadaného čísla vytočil vzdálený systém. Této sekvenci příkazů se také říká komunikační skript. Například záznam pro výše uvedený modem NakWell by vypadal nějak takto:

```
# /etc/uucp/dial - informace o vytáčecích zařízeních
# modemy NakWell
dialer          nakwell
chat            "" AT&F OK ATDT\T CONNECT
```

Řádek začínající polem *chat* definuje komunikační řetězec modemu skládající se z posloupnosti příkazů poslaných na modem a přijatých z modemu. Tyto příkazy modem inicializují a vytáčejí požadované číslo. Sekvenci znaků „\ T“ nahradí program **uucico** zadaným telefonním číslem.

Abychom vám mohli poskytnout hrubé schéma způsobu práce programu **uucico** s konfiguračními soubory, budeme předpokládat, že jste spustili na příkazovém řádku následující příkaz:

```
$ uucico -s pablo
```

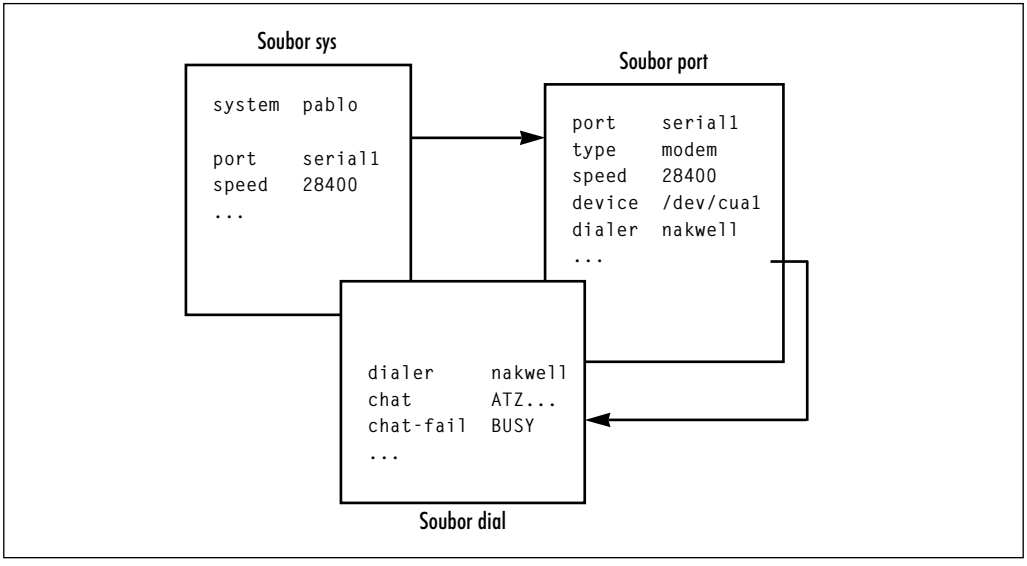
Program **uucico** nejprve vyhledá hostitele **pablo** v souboru `sys`. Ze záznamu hostitele **pablo**, který je umístěn v souboru `sys`, program **uucico** zjistí, že ke spojení by měl použít port serial1. Soubor `port` sdělí programu **uucico**, že požadovaný port je modemový port a že je k tomuto portu připojen modem NakWell.

Nyní vyhledá program **uucico** v souboru `dial` záznam, který popisuje modem NakWell, a když takový záznam nalezne, otevře sériový port `/dev/ttyS1` a spustí vytáčecí komunikační skript. To znamená, že pošle modemu příkaz **AT&F** a bude čekat na odpověď **OK** a tak dále. Jakmile najde řetězec \ T, nahradí ho skutečným telefonním číslem (555-22112), které získá ze souboru `sys`.

Poté, co modem vrátí odpověď **CONNECT**, bylo navázáno vlastní spojení a komunikační skript modemu skončí. Nyní se program **uucico** vrátí k souboru **sys** a spustí přihlašovací komunikační skript. V našem příkladu bude čekat na výzvu **login:**, potom pošle jméno uživatele (*vstout*), vyčká na výzvu **password:** a následně pošle heslo (*lorca*).

Po skončení ověřování totožnosti se předpokládá, že vzdálený konec spustí svůj vlastní program **uucico**. Potom oba tyto programy vstoupí do inicializační fáze, která byla popsána v předcházejícím textu.

Způsob, jakým na sobě závisí jednotlivé konfigurační soubory, je ukázán na obrázku 16.1.



Obrázek 16.1 – Interakce konfiguračních souborů Taylorova schématu

Co potřebuje protokol UUCP vědět

Dříve, než začnete sestavovat konfigurační soubory protokolu UUCP, musíte získat některé informace, které potřebuje protokol UUCP vědět.

Nejprve musíte zjistit, ke kterému sériovému zařízení je připojen váš modem. Porty COM 1 až COM 4 (v DOSu) se obvykle mapují na speciální soubory zařízení `/dev/ttyS0` až `/dev/ttyS3`. Některé distribuce, například Slackware, vytvoří odkaz `/dev/modem` na patřičný soubor zařízení `ttyS*`. Tyto distribuce pak nakonfigurují programy **kermit**, **seyon** a další tak, aby používaly obecný soubor `modem`. V tomto případě byste měli ve své konfiguraci protokolu UUCP také používat soubor `/dev/modem`.

Soubor `modem` byste měli používat z toho důvodu, že všechny programy obsluhující modem používají k signalizaci používání daného sériového portu takzvané zamykací soubory. Názvy těchto zamykacích souborů se tvoří kombinací řetězce `LCK..` s názvem souboru příslušného zařízení, například `LCK..ttyS1`. Pokud by programy používaly pro stejné zařízení odlišné názvy, nebudou schopny zjistit vzájemné zamykací soubory. V důsledku toho dojde k rušení relací, které jsou spuštěny současně. Mohlo by k tomu dojít zejména tehdy, pokud připojování protokolem UUCP plánujete záznamem v `crontab`. O konfiguraci sériových portů jsme hovořili v kapitole 4.

Dále musíme vědět, jakou rychlostí komunikuje modem s operačním systémem. Tuto rychlost musíte nastavit na maximální očekávanou efektivní přenosovou rychlost. Efektivní přenosová rychlost může být mnohem vyšší, než skutečná fyzická rychlost podporovaná vaším modemem. Například mnoho modemů posílá a přijímá data rychlostí 56 kb/s. Při použití kompresních protokolů, jako je V.42bis, se může skutečná přenosová rychlost vyšplhat i přes 100 kb/s.

Má-li být protokol UUCP k něčemu užitečný, budete samozřejmě potřebovat telefonní číslo volaného systému. Taktéž budete potřebovat korektní přihlašovací jméno a případně i heslo[113].

Dále musíte *přesně* vědět, jak se lze do daného systému přihlásit. Musíte třeba předtím, než se objeví výzva k přihlášení, stisknout klávesu Enter? Zobrazuje výzva řetězec `login:` nebo `user:`? Tyto údaje jsou důležité pro sestavení *komunikačního skriptu*. Pokud tyto údaje neznáte nebo pokud selže obvyklý komunikační skript, pokuste se dovolat na daný systém pomocí terminálového programu jako je **kermit** nebo **minicom** a zapište si přesně ty úkony, které musíte provést.

Pojmenování systému

Stejně jako u sítí na bázi protokolu TCP/IP musí mít váš hostitel i v sítích na bázi protokolu UUCP nějaký název. Pokud hodláte používat protokol UUCP pouze na přenos souborů do nebo ze systémů, se kterými se přímo spojujete, nebo hodláte-li používat tento protokol pouze pro přenos souborů v místní síti, nemusí tento název splňovat žádné standardy[114].

Používáte-li však protokol UUCP pro příjem pošty nebo news, měli byste zvážit, zda by nebylo lepší nechat si zaregistrovat váš název pomocí UUCP Mapping project. Tento projekt je popsán v kapitole 17. Dokonce i když jste členem domény, měli byste zvážit použití oficiálního názvu pro protokol UUCP.

Lidé si často zvolí svůj název pro protokol UUCP tak, aby odpovídal první části plně kvalifikovaného názvu domény. Předpokládejme, že adresa domény vašeho systému je **swim.twobirds.com**, potom by název vašeho hostitele pro protokol UUCP měl být **swim**. Můžete si to představit tak, že UUCP systémy se znají pouze podle křestního jména. Samozřejmě že pro protokol UUCP můžete použít i název, který nemá s doménovým názvem nic společného.

Ujistěte se však, že v poštovních adresách nepoužíváte nekvalifikovaný název systému, pokud ho nemáte zaregistrován jako oficiální název pro protokol UUCP. Pošta poslaná na neregistrovaného hostitele protokolu UUCP by v lepším případě zmizela v nějaké černé díře. Použijete-li název, kte-

113 Chcete-li si UUCP vyzkoušet, sežeňte si číslo nějakého archivního systému ve svém okolí. Zapište si jméno a heslo – bývají zveřejněny aby byl možný anonymní přístup. Ve většině případů vypadají tak nějak jako uucp/uucp nebo nuucp/uucp.

114 Jediným omezením je, že by neměl být delší než sedm znaků, aby to nespletlo implementace UUCP běžící na systémech, které mají omezeny délky názvů souborů. Jména delší než sedm znaků UUCP často zkracuje. Některé verze omezují délku jména dokonce jen na šest znaků.

rý již používá nějaký jiný systém, bude pošta směrována na tento systém a způsobí poštmistrům tohoto systému značné problémy.

Balík UUCP implicitně používá název systému nastavený pomocí příkazu hostname. Tento název je obecně nastaven při startu systému prostřednictvím `rc` skriptů a nachází se v souboru `/etc/HOSTNAME`. Pokud se váš název pro protokol UUCP liší od názvu, který jste přiřadili vašemu hostiteli, musíte použít v souboru `config` volbu **hostname**, která sdělí programu **uucico** název pro protokol UUCP. Tento postup bude popsán dále.

Taylorovy konfigurační soubory

Nyní se vrátíme zpět ke konfiguračním souborům. Protokol Taylor UUCP získává potřebné informace z následujících souborů:

config	Toto je hlavní konfigurační soubor. Zde můžete zadat název vašeho systému pro protokol UUCP.
sys	Tento soubor popisuje všechny systémy, které znáte. Pro každý systém je v tomto souboru uveden název systému, časy, ve kterých se lze s tímto systémem spojit, volané číslo (pokud nějaké existuje), typ používaného zařízení a způsob, jakým se lze k tomuto systému přihlásit.
port	Obsahuje položky popisující každý dostupný port, u kterého jsou uvedeny podporovaná rychlost linky a typ používaného vytáčecího zařízení.
dial	Popisuje vytáčecí zařízení, které se používá pro spojení po telefonní lince.
dialcode	Obsahuje expanze pro jednotlivé symbolické vytáčecí kódy.
call	Obsahuje přihlašovací jméno a heslo, které se použije při volání daného systému. Používá se jen zřídka.
passwd	Obsahuje přihlašovací jména a hesla, která mohou systémy používat při přihlašování. Tento soubor se používá pouze v případě, že si program **uucico** sám provádí ověření hesla.

Taylorovy konfigurační soubory se zpravidla skládají z řádků obsahujících dvojice klíčové slovo a hodnota. Symbol # značí komentář, který sahá až do konce příslušného řádku. Chcete-li použít samotný symbol #, zadejte ho společně se zpětným lomítkem: `\#`.

Pomocí těchto konfiguračních souborů můžete ovládat poměrně velké množství voleb. Nebudeme zde probírat všechny parametry, ukážeme si pouze nejdůležitější z nich. Tyto volby by vám měly pomoci nastavit modemové spojení pomocí protokolu UUCP. V dalším textu budeme popisovat úpravy, které je třeba provést, budete-li chtít používat protokol UUCP v síti na bázi protokolu TCP/IP nebo společně s přímou sériovou linkou. Kompletní popis je uveden v Texinfo dokumentech, které jsou součástí zdrojového kódu protokolu Taylor UUCP.

Pokud si myslíte, že máte svůj systém s protokolem UUCP kompletně nakonfigurován, můžete si svoji konfiguraci zkontrolovat pomocí nástroje **uuchk** (ten najdete v adresáři `/usr/lib/uucp`). Tento nástroj přečte vaše konfigurační soubory a vytiskne podrobnou zprávu obsahující konfigurační hodnoty, které jste použili u každého systému.

Všeobecné konfigurační volby – soubor config

Do tohoto souboru obvykle nebudete zapisovat příliš mnoho vlastností. Většinou ho použijete jen k definici názvu hostitele pro protokol UUCP. Ten bude implicitně používat název, který nastaví-

te pomocí příkazu **hostname**, ale je užitečné explicitně nastavit vlastní název pro protokol UUCP. Následuje vzorový soubor `config`:

```
# /usr/lib/uucp/config - hlavní konfigurační soubor UUCP
hostname        vstout
```

Samozřejmě existuje mnoho rozmanitých parametrů, které lze nastavit, příkladem může být název dočasného adresáře nebo přístupová práva k anonymní službě UUCP. Definice přístupových práv k anonymní službě UUCP bude popsána v části *Anonymní UUCP*.

Jak sdělit protokolu UUCP informace o jiných systémech – soubor sys

Soubor *sys* popisuje systémy, které zná váš počítač. Záznam je uveden klíčovým slovem *system*; další řádky až po následující direktivu *system* definují parametry, které se vztahují k danému systému. Typicky definuje záznam systému takové parametry, jako je například telefonní číslo a přihlašovací komunikační skript.

Parametry uvedené před prvním výskytem klíčového slova *system* definují implicitní hodnoty, které se budou používat pro všechny systémy. V sekci implicitních hodnot budou obvykle uvedeny parametry protokolu a podobné vlastnosti.

Následuje poměrně podrobný popis nejvýznamnějších polí.

Název systému

Příkaz *system* definuje název vzdáleného systému. Musíte zadat korektní název vzdáleného systému, ne pouze přezdívku, kterou jste si vymysleli, protože program **uucico** porovná tento název s názvem, který pošle vzdálený systém v okamžiku vašeho přihlášení[115].

Každý název systému se může objevit pouze jedenkrát. Pokud chcete pro daný systém používat několik skupin konfigurací (například různá telefonní čísla, která by měl program **uucico** postupně zkoušet vytáčet), můžete zadat takzvané *zástupce*, o kterých budeme hovořit později.

Telefonní číslo

Má-li být vzdálený systém dosažitelný pomocí telefonní linky, určuje pole *phone* číslo, které by měl modem vytočit. Toto pole může obsahovat několik symbolů, které posléze zpracuje vytáčecí procedura programu **uucico**. Symbol rovnítka (=) říká programu **uucico**, aby vyčkal na sekundární vytáčecí tón a symbol pomlčky (-) generuje pauzu dlouhou jednu sekundu. Některé telefonní rozvody se totiž „ucpou“, když neuděláte pauzu mezi číslem, které musíte vytočit, abyste se dostali z vnitřní sítě, a mezi vlastním telefonním číslem[116].

Často bývá výhodné namísto předčíslí různých míst uvádět přímo jejich názvy. Soubor `dialcode` umožňuje přiřadit názvu telefonní číslo a tento název pak můžete využít při zadávání čísel vzdálených systémů. Řekněme, že budete mít takovýto soubor `dialcode`:

```
# /usr/lib/uucp/dialcode - převod vytáčecích kódů
Bogoham         024881
Coxton          035119
```

115 Starší implementace UUCP verze 2 nevysílají při přihlašování své jméno, nicméně novější implementace to často dělají, stejně jako Taylorova implementace.

116 V řadě společností s vlastní ústřednou se například dostanete na státní linku až po vytočení nuly nebo devítky.

Pomocí těchto definic můžete v souboru `sys` používat například telefonní číslo *Bogoham7732*, které snad vypadá přehledněji a navíc usnadní aktualizaci záznamů, pokud by se předčíslí z nějakých důvodů změnilo.

Port a rychlost

Volby *port* a *speed* se používají k výběru zařízení, jež bude používáno k vytáčení vzdáleného systému a k nastavení maximální rychlosti, na kterou může být dané zařízení nastaveno[117]. Záznam *system* může používat buď jednu z těchto voleb, nebo kombinaci obou voleb. Při vyhledávání vhodného zařízení v souboru `port` vybere systém pouze ty porty, které odpovídají názvu portu a/nebo zvolenému rozsahu rychlostí.

Obvykle by měla stačit pouze volba *speed*. Máte-li v souboru `port` definováno pouze jedno sériové zařízení, zvolí program **uucico** vždy právě toto zařízení, takže pro ně musíte pouze nastavit požadovanou rychlost. Máte-li k systému připojeno několik modemů, ani tehdy zřejmě nebudete definovat konkrétní port, protože když program **uucico** zjistí, že danému požadavku vyhovuje více zařízení, bude postupně zkoušet každé z těchto zařízení, dokud nenajde nějaké nepoužívané.

Přihlašovací komunikační skript

V předchozím výkladu jsme se již setkali s přihlašovacím komunikačním skriptem, který sděluje programu **uucico**, jakým způsobem se má přihlásit ke vzdálenému systému. Přihlašovací komunikační skript se skládá ze seznamu symbolů, které definují očekávané řetězce a řetězce posílané místním procesem **uucico**. Cílem je, aby program **uucico** počkal, dokud vzdálený počítač nepošle výzvu k přihlášení. Místní proces pak vrátí přihlašovací jméno, počká až vzdálený systém pošle výzvu k zadání hesla a nakonec pošle vlastní heslo. Očekávané a posílané řetězce se vzájemně střídají. Program **uucico** automaticky přidá ke všem posílaným řetězcům znak konce řádku (\r). Jednoduchý komunikační skript by mohl vypadat asi takto:

```
ogin: vstout ssword:  catch22
```

Všimněte si, že pole očekávaných řetězců neobsahují celé výzvy. To proto, aby přihlášení uspělo i v případě, kdy vzdálený systém vyšle místo řetězce **login:** řetězec **Login:**. Pokud odesílaný nebo očekávaný řetězec obsahuje mezery, musíte jej uzavřít do uvozovek.

Program **uucico** umožňuje i určitý druh podmíněného spouštění, například když je potřeba program **getty** na vzdáleném počítači znovu inicializovat. V tomto případě můžete k očekávanému řetězci připojit komunikační podřetězce, které jsou vzájemně odděleny pomocí pomlček. Komunikační podřetězce se spustí pouze v případě, že selže hlavní řetězec, tedy při překročení časového limitu. Jedním z případů využití této vlastnosti je odeslání kódu BREAK v situaci, kdy vzdálený systém nepošle výzvu k přihlášení. Následující příklad uvádí všeobecný komunikační skript, který bude fungovat i v případě, že před zobrazením výzvy k přihlášení musíte stisknout Enter. První prázdný řetězec "" říká UUCP, aby na nic nečekal a okamžitě pokračoval v posílání řetězce.

```
"" \n\r\d\r\n\c ogin:-BREAK-ogin: vstout ssword: catch22
```

V komunikačním skriptu lze použít řadu speciálních řetězců a únikových znaků. Následuje neúplný seznam korektních symbolů, které mohou být uvedeny v očekávaných řetězcích:

117 Rychlost *tty* zařízení pak musí být alespoň rovna zadané hodnotě.

`""` Prázdný řetězec. Tento řetězec sděluje programu **uucico**, aby na nic nečekal a okamžitě pokračoval v posílání následujícího řetězce.

`\t` Tabulátor.

`\r` Návrat vozíku (CR).

`\s` Mezera. Tento znak potřebujete k vložení mezer do komunikačního řetězce.

`\n` Nový řádek (LF).

`\\` Zpětné lomítko.

V posílaných řetězcích se mohou kromě výše uvedených symbolů vyskytnout následující únikové znaky a řetězce:

EOT Konec přenosu (^D).

BREAK Přerušení.

`\c` Potlačí na konci řetězce poslání znaku návrat vozíku.

`\d` Přeruší posílání na jednu sekundu.

`\E` Povolí kontrolu opakováním. Program uucico očekává, že vše, co zapíše na komunikační zařízení, si z něj také přečte a až pak pokračuje v komunikaci. Tato volba je užitečná zejména při použití modemových komunikačních skriptů (se kterými se setkáme dále). Implicitně je kontrola opakováním vypnutá.

`\e` Vypne kontrolu opakováním.

`\K` Stejné jako BREAK.

`\p` Zastaví chod na zlomek sekundy.

Zástupci

Někdy je vhodné mít k dispozici několik záznamů daného systému, například tehdy, je-li systém dosažitelný pomocí několika modemových linek. U protokolu Taylor UUCP je možné pro jeden systém definovat několik záznamů pomocí takzvaných *zástupců*.

Záznam zástupce přebírá veškerá nastavení z hlavního záznamu systému a jsou v něm zadány pouze ty hodnoty, které by měly potlačit implicitní hodnoty záznamu systému nebo které by měly přidat hodnoty k tomuto záznamu. Zástupce je oddělen od hlavního záznamu systému řádkem, který obsahuje klíčové slovo *alternate*.

Chcete-li u hostitele **pablo** používat dvě telefonní čísla, upravíte soubor `sys` takto:

```
system       pablo
phone        123-456
... další záznamy...
alternate
phone        123-455
```

Když nyní budete volat hostitele **pablo**, pokusí se nejprve program **uucico** vytočit číslo 123-456 a pokud toto volání neuspěje, pokusí se vytočit číslo zástupce. Záznam zástupce přebírá veškerá nastavení z hlavního záznamu systému a potlačuje pouze nastavení telefonního čísla.

Omezení času volání

Protokol Taylor UUCP nabízí množství způsobů pro omezení časů, ve kterých je možné uskutečnit hovory se vzdáleným systémem. Tato omezení můžete zavést buď kvůli omezením, která klade vzdálený hostitel na své služby během pracovní doby, nebo proto, abyste se vyhnuli dobám

s vysokými tarifními poplatky. Pomocí voleb `-s` nebo `-f` je možné v programu **uucico** vždy potlačit nastavená omezení času volání.

Protokol Taylor UUCP implicitně zakazuje spojení v jakémkoliv čase, takže *musíte* v souboru `sys` použít volbu, která povolí nějaký časový rozsah volání. Pokud vás nezajímá omezení času volání, můžete zadat do souboru `sys` volbu *time* s hodnotou *Any*.

Nejjednodušší způsob omezení času nabízí volba *time*, za níž následuje řetězec tvořený poli den a čas. Pole den může nabývat libovolné kombinace hodnot *Mo*, *Tu*, *We*, *Th*, *Fr*, *Sa*, *Su* (pondělí až neděle) nebo hodnot *Any* (kdykoliv), *Never* (nikdy) nebo *Wk* (pracovní dny). Pole čas se skládá ze dvou 24hodinových časových hodnot, které jsou odděleny pomlčkou. Tyto hodnoty udávají časový rozsah, ve kterém se mohou hovory uskutečňovat. Kombinace těchto polí musí být zapsána bez mezery. Pomocí čárek může být seskupen libovolný počet denních a časových údajů, například:

```
time            MoWe0300-0730,Fr1805-2000
```

Tento příkaz povoluje volání v pondělí a ve středu v době od 03:00 do 07:30 a v pátek v době od 18:05 do 20:00. Když časové pole překračuje půlnoc, například *Mo1830-0600*, znamená to, že hovory budou povoleny v pondělí v době od půlnoci do 06:00 a v době od 18:30 do půlnoci.

Speciální řetězce *Any* a *Never* fungují přesně tak, jak naznačují: hovory se mohou uskutečnit kdykoliv, respektive nikdy.

Taylorovo UUCP navíc umožňuje zadat další speciální hodnoty, například *NonPeak* nebo *Night*. Tyto dvě konkrétně znamenají *Any2300-0800,SaSu0800-1700* a *Any1800-0700,SaSu*.

Příkaz time může obsahovat volitelný druhý parametr, který definuje čas v minutách, po jehož uplynutí se má uskutečnit nový pokus. Když selže pokus o spojení, nepovolí program **uucico** po určitou předem zadanou dobu další pokus o spojení se vzdáleným hostitelem. Zadáte-li například čas opakování pokusu 5 minut, odmítne program uucico zavolat vzdálený systém dříve, než 5 minut po posledním neúspěšném volání. Program **uucico** implicitně používá exponenciální zpětné schéma, při kterém se s každým opakovaným selháním zvyšuje čas do nového pokusu.

Příkaz *timegrade* umožňuje přidělit plánu maximální priority. Předpokládejme například, že v záznamu *system* máte následující příkazy *timegrade*:

```
timegrade           N Wk1900-0700,SaSu
timegrade           C Any
```

Tyto příkazy umožní přenos úkolů s prioritou C nebo vyšší (pošta se obvykle uchovává s prioritou B nebo C) při každém uskutečněném spojení, zatímco news (které se obvykle uchovávají s prioritou N) budou přenášeny pouze v noci a o víkendech.

Stejně jako u příkazu *time* lze zadat i u příkazu *timegrade* volitelný třetí parametr, který definuje čas (v minutách) do nového pokusu.

Ohledně priorit odesílaných souborů však existuje následující potíž: volba *timegrade* se vztahuje pouze na soubory, které posílá *váš* systém; vzdálený systém může i přesto přenášet cokoliv ho napadne. Chcete-li explicitně definovat požadavek na vzdálený systém, aby posílal pouze úkoly, které mají vyšší než zadanou prioritu, použijte k tomu volbu *call-timegrade*; nemáte však žádnou záruku, že se bude vzdálený systém tímto požadavkem řídit[118].

118 Pokud na vzdáleném systému běží Taylorovo UUCP, tak se tímto požadavkem řídit bude.

Podobně nebude ani pole *timegrade* ověřeno v případě, kdy spojení naváže vzdálený systém. V takovém případě mu budou poslány všechny připravené úkoly. Vzdálený systém však může explicitně požádat váš program **uucico**, aby se omezil pouze na určitou prioritu.

Jaká zařízení existují – soubor port

Soubor `port` informuje program **uucico** o dostupných portech. Těmito porty mohou být jak modemové porty, tak i ostatní podporované typy portů, jako jsou přímé sériové linky nebo sockety protokolu TCP.

Stejně jako soubor `sys`, je i soubor `port` složen ze samostatných záznamů, které začínají klíčovým slovem *port*, za kterým následuje název daného portu. Tento název můžete použít v souboru `sys` v příkazu *port*. Název nemusí být jedinečný; existuje-li více portů se shodným názvem, bude program **uucico** zkoušet postupně všechny porty, dokud nenajde port, který se v daném okamžiku nepoužívá.

Za příkazem *port* by měl následovat příkaz *type*, který definuje typ popisovaného portu. Korektní typy jsou *modem*, typ *direct* pro přímá spojení a typ *tcp* pro sockety protokolu TCP. Pokud příkaz *port* chybí, bude implicitní typ portu nastaven na *modem*.

V této stati budeme probírat pouze modemové porty; porty pro protokol TCP a přímé linky budou probrány v dalších statích.

U modemových a přímých portů musíte pomocí příkazu *device* zadat volací zařízení. Obvykle je to název speciálního souboru zařízení, který se nachází v adresáři `/dev`, například `/dev/ttyS1`.

V případě modemového zařízení určuje záznam portu také typ modemu, který je k danému portu připojen. Různé typy modemů musí být nakonfigurovány odlišným způsobem. Dokonce i modemy, které tvrdí, že jsou kompatibilní se standardem Hayes, nemusí být ve skutečnosti navzájem kompatibilní. Z tohoto důvodu musíte programu **uucico** sdělit, jak má inicializovat daný modem a jak má vytočit požadované telefonní číslo. Protokol Taylor UUCP uchovává popisy všech vytáčecích zařízení v souboru `dial`. Chcete-li použít některé z těchto zařízení, musíte pomocí příkazu *dialer* zadat název požadovaného vytáčecího zařízení.

Někdy budete chtít používat modem odlišným způsobem podle typu volaného systému. Například některé starší modemy nerozumí tomu, když se nějaký vysokorychlostní modem pokouší spojit rychlostí 56 kb/s; tyto modemy linku prostě zavěsí, místo aby se snažily spojit například rychlostí 9600 b/s. Víte-li, že nějaký systém používá právě takovýto typ hloupého modemu, musíte nastavit svůj modem odlišným způsobem. K tomu potřebujete další záznam v souboru `port`, který bude definovat odlišný typ vytáčecího zařízení. Nyní můžete přiřadit novému portu odlišný název, například *serial1-slow,* a v souboru `sys` použít v záznamu pro příslušný systém tento typ zařízení.

Výhodnější je ale rozdělit porty podle podporovaných rychlostí. Dva záznamy portů mohou pro výše popsanou situaci vypadat následovně:

```
# NakWell modem; spojení na vysoké rychlosti
port    serial1       # jméno portu
type    modem         # modem
device  /dev/ttyS1    # COM2
speed   115200        # podporovaná rychlost
dialer  nakwell       # vytáčecí zařízení

# NakWell modem; spojení na nízké rychlosti
port    serial1       # jméno portu
type    modem         # modem
```

```
device  /dev/ttyS1    # COM2
speed   9600          # podporovaná rychlost
dialer  nakwell-slow  # nezkoušej vysokou rychlos připojení
```

Záznam pro systém se starým modemem přiřadí portu název serial1, bude ale požadovat, aby se používala pouze rychlost 9600 b/s. Potom program **uucico** automaticky vybere druhý záznam portu. Všechny ostatní systémy, které mají v záznamu systému uvedenu rychlost 115200 b/s, budou volány s použitím prvního záznamu portu. Platí, že se vždy vybere první položka, u níž vyhovuje rychlost

Jak vytočit číslo – soubor dial

Soubor `dial` popisuje způsob, jakým se používají různá vytáčecí zařízení. Protokol UUCP z historických důvodů používá termín „vytáčecí zařízení“ a nikoliv „modem“, protože dříve bývalo obvyklé jedno (drahé) automatické vytáčecí zařízení, které obsluhovalo sadu modemů. Dnes má většina modemů zabudovanou podporu pro vytáčení čísel, takže toto rozlišování poněkud ztratilo smysl.

Přesto však mohou různá vytáčecí zařízení nebo modemy vyžadovat odlišné konfigurace. Každou z těchto konfigurací popíšete v souboru `dial`. Záznamy v tomto souboru začínají příkazem *dialer*, který přiřazuje název vytáčecímu zařízení.

Kromě této položky je nejdůležitější komunikační skript, který se zadává pomocí příkazu *chat*. Stejně jako tomu bylo u přihlašovacího komunikačního skriptu, skládá se i tento skript ze sekvence řetězců, které program **uucico** posílá na vytáčecí zařízení, a z odpovědí, které očekává z vytáčecího zařízení. Typicky se tento skript používá k inicializaci modemu do nějakého známého stavu a k vytočení požadovaného čísla. Následující příklad záznamu vytáčecího zařízení ukazuje typický komunikační skript pro modemy kompatibilní s modemovým standardem Hayes:

```
# NakWell modem; spojení na vysoké rychlosti
dialer  nakwell  # jméno vytáčecího zařízení
chat    "" AT&F OK\r ATH1E0Q0 OK\r ATDT\T CONNECT
chat-fail       BUSY
chat-fail       ERROR
chat-fail       NO\sCARRIER
dtr-toggle      true
```

Komunikační skript modemu začíná řetězcem "", což je prázdný očekávaný řetězec. Program **uucico** proto okamžitě pošle první příkaz (**AT&F**). Příkaz **AT&F** inicializuje modemy kompatibilní se standardem *Hayes* do výchozího stavu. Pak bude **uucico** čekat, dokud modem nepošle odpověď `OK` a pak pošle další příkaz, který vypne místní opakování znaků. Jakmile modem opět pošle odpověď `OK`, pošle program **uucico** příkaz pro vytáčení (**ATDT**). Sekvence `\T` v řetězci bude nahrazena skutečným telefonním číslem, které systém získá z příslušného záznamu v souboru `sys`. Potom bude program **uucico** čekat, dokud modem nepošle odpověď `CONNECT`, která bude signalizovat, že bylo spojení se vzdáleným systémem úspěšné.

Občas se stává, že se modemu nepodaří spojení se vzdáleným systémem, například když už vzdálený systém komunikuje s nějakým jiným systémem a linka je obsazená. V takovém případě vrátí modem nějakou chybovou zprávu, která označuje příčinu neúspěchu. Komunikační skripty modemu nejsou schopny detekovat takovéto zprávy, takže program **uucico** bude čekat na očekávaný řetězec tak dlouho, dokud nedojde k překročení časového limitu. Proto se v logovacím souboru protokolu UUCP objeví místo pravého důvodu pouze neurčitá zpráva „timed out in chat script“.

Avšak protokol Taylor UUCP umožní sdělit příčinu chyby programu **uucico**. Lze to provést pomocí příkazu *chat-fail*, který je uveden ve výše uvedeném výpisu. Když program **uucico** detekuje při provádění komunikačního skriptu modemu řetězec, který odpovídá některému ze záznamů *chat-fail*, zruší aktuální volání a zapíše do logovacího souboru protokolu UUCP chybovou zpráva.

Poslední příkaz v příkladu sděluje protokolu UUCP, aby přepnul linku DTR dříve, než se spustí komunikační skript modemu. Typicky totiž ovladač sériového portu ohlašuje modemu signálem DTR to, že jej nějaký proces otevřel a bude jej používat. Přepínač *dtr-toggle* funguje tak, že způsobí krátké vynulování signálu DTR a jeho nové nastavení. Většina modemů může být nakonfigurována tak, aby v reakci na signál DTR zvedla linku a přešla do příkazového režimu[119].

Použití protokolu UUCP v sítích na bázi protokolu TCP

Na první pohled se může zdát použití protokolu UUCP pro přenos dat po sítích na bázi protokolu TCP absurdní, ale v zásadě to není špatný nápad, zvláště při přenosu velkého množství dat typu usenetových news. Na TCP sítích se zpravidla vyměňují news pomocí protokolu NNTP, s jehož pomocí se o články žádá a jsou odesílány individuálně, bez komprese a bez jakékoliv další optimalizace. Ačkoliv tato technika vyhovuje rozsáhlým systémům s několika souběžnými news, není příliš vhodná pro malé systémy, které získávají news přes pomalá spojení, jako je ISDN. Takové systémy budou obvykle chtít kombinovat výhody protokolu TCP s výhodami, které přináší posílání news ve velkých skupinách, které lze zkomprimovat a pak je přenést s menší režií. Standardní způsob přenášení takových skupin spočívá v použití protokolu UUCP přes TCP.

V souboru `sys` musíte následujícím způsobem definovat systém, který má být volán pomocí protokolu TCP:

```
system          gmu
address         news.groucho.edu
time            Any
port            tcp-conn
chat            ogin: vstout word: clouseau
```

Příkaz *address* udává IP adresu hostitele nebo jeho plně kvalifikované doménové jméno. Odpovídající záznam v souboru port by měl vypadat asi takto:

```
port            tcp-conn
type            tcp
service         540
```

Záznam uvádí, že spojení pomocí protokolu TCP by mělo být použito tehdy, pokud záznam v souboru `sys` obsahuje hodnotu *tcp-conn,* a že program **uucico** by se měl pokusit spojit se vzdáleným hostitelem pomocí sítě TCP na portu 540. Toto je implicitní číslo portu pro službu UUCP. Místo čísla portu můžete v příkazu *service* uvést symbolický název portu. Číslo portu, které odpovídá tomuto názvu, bude vyhledáno v souboru `/etc/services`. Běžný název pro službu UUCP je uucpd.

Použití přímého spojení

Předpokládejme, že pro spojení vašeho systému **vstout** s hostitelem **tiny** používáte přímou linku. Stejně jako v případě použití modemu budete muset pro daný systém zapsat do souboru `sys` záznam. Příkaz *port* označuje sériový port, ke kterému je hostitel **tiny** připojen.

119 Některé modemy však tuto signalizaci nemají rády a občas přitom zavěsí.

```
system          tiny
time            Any
port            direct1
speed           38400
chat            ogin: cathcart word: catch22
```

V souboru `port` musíte popsat sériový port, který se používá pro přímé spojení. Příkaz *dialer* není nutný, protože u přímé linky není třeba vytáčet žádná čísla.

```
port            direct1
type            direct
speed           38400
dev             /dev/ttyS1
```

Řízení přístupu k funkcím UUCP

UUCP je velmi pružný systém. S touto pružností zároveň přichází nutnost pečlivě řídit přístup k jeho funkcím, aby se zabránilo úmyslnému či neúmyslnému zneužití. Hlavními funkcemi, na něž administrátor dává pozor, je vzdálené spouštění příkazů, přenos souborů a předávání. Taylorova implementace UUCP umožňuje omezit možnosti jak mohou vzdálení hostitelé jednotlivé funkce použít. S pečlivým nastavením oprávnění je možné zajistit, aby byla zachována bezpečnost systému.

Spouštění příkazů

Úkolem protokolu UUCP je kopírovat soubory z jednoho systému do druhého a žádat spouštění určitých příkazů na vzdálených hostitelích. Samozřejmě, že jako správce systému budete chtít řídit práva, která budete přidělovat ostatním systémům – rozhodně není dobré povolit na svém systému spouštění libovolných příkazů.

Jedinými příkazy, které na vašem počítači povolí protokol Taylor UUCP spouštět ostatním systémům, jsou implicitně příkazy **rmail** a **rnews**, které se obecně používají pro spojení pomocí protokolu UUCP k výměně elektronické pošty a usenetových news. Chcete-li pro konkrétní systém změnit množinu příkazů, použijte k tomu klíčové slovo *commands*, které přidáte do souboru `sys`. Dále můžete chtít omezit prohledávací cestu pouze na ty adresáře, v nichž se povolené příkazy nacházejí. Prohledávací cestu povolenou pro vzdáleného hostitele můžete změnit příkazem *command-path*. Chcete-li například, aby mohl systém **pablo** spouštět společně s příkazy **rmail** a **rnews** i příkaz **bsmtp**, upravte soubor `sys` takto[120]:

```
system          pablo
...
commands        rmail rnews bsmtp
```

Přenosy souborů

Protokol Taylor UUCP také dovoluje velice podrobné nastavení přenosu souborů. Na jedné straně můžete zcela zakázat přenosy souborů z konkrétního systému nebo do konkrétního systému. Stačí jen nastavit volbu *request* na hodnotu *no* a vzdálený systém nebude moci ani přijímat soubory z vašeho systému, ani žádné soubory vašemu systému posílat. Podobně můžete pomocí vol-

120 **bstmp** slouží k dávkovému přenosu SMTP pošty.

by *transfer*, kterou nastavíte na hodnotu *no*, zakázat svým uživatelům přenos souborů do vašeho systému nebo z vašeho systému. Implicitně je jak místním uživatelům, tak i uživatelům ze vzdáleného systému povoleno posílat a přijímat soubory.

Kromě toho můžete nastavit adresáře, do kterých nebo ze kterých mohou být soubory kopírovány. Obvykle budete chtít omezit přístup ze vzdálených systémů pouze jen na jednu hierarchii adresářů a zároveň budete chtít umožnit vašim uživatelům posílání souborů ze svého domovského adresáře. Uživatelé mohou typicky přijímat ze vzdáleného systému pouze soubory z veřejného adresáře protokolu UUCP, což je adresář `/var/spool/uucppublic`. Je to tradiční umístění, kde je možné soubory zpřístupnit veřejnosti; je to velmi podobné serverům FTP na Internetu[121].

Protokol Taylor UUCP poskytuje čtyři příkazy, které se používají ke konfiguraci adresářů pro přijímání a posílání souborů. Jsou to konkrétně příkazy *local-send*, který specifikuje seznam adresářů, z nichž může uživatel protokolem UUCP odesílat soubory, *local-recieve*, který uvádí seznam adresářů, do nichž může uživatel přijímat soubory, a příkazy *remote-send* a *remote-receive*, které se chovají analogicky při požadavku z cizího systému. Uvažte následující příklad:

```
system         pablo
...
local-send     /home
local-receive  /home ~/receive
remote-send    ~ !~/incoming !~/receive
remote-receive ~/incoming
```

Příkaz *local-send* povoluje uživatelům posílat z vašeho hostitele na hostitele **pablo** libovolné soubory z adresářové struktury pod adresářem `/home` a z veřejného adresáře protokolu UUCP. Příkaz *local-receive* povoluje těmto uživatelům přijímat soubory buď do volně dostupného adresáře `receive`, který se nachází pod adresářem `uucppublic`, nebo do libovolného adresáře (s právem zápisu), který se nachází v adresářové struktuře pod adresářem `/home`. Příkaz *remote-send* povoluje uživatelům z hostitele **pablo** žádat o soubory z adresáře `/var/spool/uucppublic`, vyjma souborů z adresářových struktur pod adresáři `incoming` nebo `receive`. Tyto vyjmuté adresáře jsou signalizovány programu **uucico** tak, že před jejich názvy je uveden vykřičník. A konečně poslední řádek povoluje uživatelům z hostitele **pablo** přenášet libovolné soubory do adresáře `incoming`.

Jedním z největších problémů souvisejících s přenosem souborů prostřednictvím protokolu UUCP je skutečnost, že lze přijímat soubory pouze do adresářů, do kterých lze veřejně zapisovat. To může svádět některé uživatele k tomu, aby chystali pastičky na ostatní uživatele. Tomuto ale nelze nijak zabránit, ledaže byste zablokovali veškeré přenosy souborů pomocí protokolu UUCP.

Předávání

Protokol UUCP nabízí mechanismus, pomocí něhož můžete na ostatních systémech spustit přenosy souborů pod svým jménem. Řekněme například, že máte UUCP přístup k hostiteli **seci** a nemáte přístup k hostiteli **uchile**. Pak můžete dosáhnout toho, že **seci** vyzvedne soubory z **uchile** a pošle je vám. Provede to následující příkaz:

```
$ uucp -r seci!uchile!~/find-ls.gz ~/uchile.filez.gz
```

Tato technika, pomocí níž lze předávat úkol přes několik systémů, se nazývá *předávání*. Pokud provozujete systém s protokolem UUCP, budete chtít omezit doručovací službu pouze na

121 K odkazu na veřejný adresář UUCP můžete použít tildu (~), funguje to však pouze v konfiguračních souborech UUCP. Kdekoliv jinde tento symbol označuje domovský adresář uživatele.

několik důvěryhodných hostitelů, kteří vám nevyženou telefonní účet do neúnosných mezí například tím, že by chtěli po vašem hostiteli stáhnout poslední verzi zdrojového kódu balíku X11R6.

Implicitně má protokol Taylor UUCP zablokované veškeré předávání. Chcete-li povolit předávání nějakému konkrétnímu systému, použijte k tomu příkaz *forward*. Tento příkaz udává seznam systémů, které vás mohou požádat o předávání. Například správce systému **seci** bude muset přidat do souboru `sys` následující řádky, které povolí hostiteli **pablo** požádat o předání úlohy pro hostitele **uchile**:

```
######################
# pablo
system  pablo
...
forward uchile
######################
# uchile
system  uchile
...
forward-to     pablo
```

Položka *forward-to* je pro hostitele **uchile** nutná, aby jakékoliv soubory poslané z tohoto hostitele byly předány hostiteli **pablo**. V opačném případě by je protokol UUCP zahodil. Tato položka je obměnou příkazu *forward* a hostiteli **uchile** povoluje posílat soubory pouze na hostitele **pablo** přes hostitele **seci**; jakákoliv jiná cesta je nepřípustná.

Chcete-li povolit předávání do libovolného systému, použijte k tomu speciální klíčové slovo *ANY* (vyžadují se velká písmena).

Nastavení systému pro příchozí volání

Chcete-li nastavit svůj systém pro příchozí volání, musíte u svých sériových portů povolit přihlášení a přizpůsobit některé ze systémových souborů tak, aby poskytovaly účty pro protokol UUCP. To vše bude náplní dalšího textu.

Poskytování účtů protokolu UUCP

Nejprve budete muset nastavit uživatelské účty, které umožní vzdáleným systémům přihlášení do vašeho systému a uskutečnění spojení pomocí protokolu UUCP. Zpravidla poskytnete každému systému, s nímž jste v kontaktu, samostatné přihlašovací jméno. Když budete nastavovat účet pro hostitele **pablo**, přiřadíte mu asi jméno uživatele **Upablo**. Volba přihlašovacích jmen se nemusí řídit žádnými pravidly, bývá ale rozumné volit je tak, aby z názvu účtu bylo jasné, k jakému systému se vztahují.

Účty systémů, které k vám volají po sériovém portu, musíte obvykle přidat do systémového souboru hesel `/etc/passwd`. Osvědčilo se vložit všechny účty protokolu UUCP do speciální skupiny, například s názvem **uuguest**. Domovský adresář daného účtu by měl být nastaven do `/var/spool/uucppublic`; jako přihlašovací interpret musí být program **uucico**.

Abyste mohli obsluhovat systémy UUCP, které se připojují k vašemu systému pomocí sítě na bázi protokolu TCP, musíte nastavit superserver **inetd**, aby obsluhoval příchozí spojení na portu služby UUCP. Do souboru `/etc/inetd.conf` přidáte následující řádek:

```
uucp stream tcp nowait root /usr/sbin/tcpd /usr/lib/uucp/uucico -l
```

Parametr `-l` sděluje programu **uucico**, aby prováděl při přihlášení vlastní proceduru ověření totožnosti. Program **uucico** vyzve, stejně jako standardní program **login**, k zadání přihlašovacího jména a hesla, avšak bude se spoléhat na svoji vlastní soukromou databázi, a nikoliv na soubor `/etc/passwd`. Tento soukromý soubor s hesly se nazývá `/etc/uucp/passwd` a obsahuje páry tvořící přihlašovací jména a hesla[122]:

```
Upablo  IslaNegra
Ulorca  co'rdoba
```

Samozřejmě tento soubor musí vlastnit uživatel **uucp** a musí mu být přidělena přístupová práva 600.

Pokud se vám použití takovéto samostatné databáze líbí a chtěli byste ji použít i pro přístup přes sériové porty, je to za určitých okolností možné. Potřebujete takový program **getty**, kterému můžete říct, aby pro uživatele UUCP spouštěl **uucico** namísto **/bin/login**[123]. Spuštění programu **uucico** bude vypadat takto:

```
/usr/lib/uucp/uucico -l -u uživatel
```

Parametr `-u` umožňuje zadat jméno uživatele a program **uucico** se na ně nebude ptát[124].

Abyste chránili své uživatele protokolu UUCP od volajících, kteří udávají falešný název systému a kteří sledují veškerou poštu vašich uživatelů, měli byste ke každému záznamu systému v souboru `sys` přidat příkaz *called-login*. Ten je popsán v následujícím textu.

Jak se chráníme před podvodníky

Jeden z největších problémů s protokolem UUCP spočívá v tom, že volající systém může lhát o svém názvu; on sice po přihlášení ohlásí svůj název volanému systému, ale server nemá žádnou možnost, jak by ho mohl ověřit. Takhle se může útočník přihlásit ke svému účtu protokolu UUCP a předstírat, že je někdo jiný a vyzvednout si tak poštu tohoto systému. To je problematické zejména v situaci, kdy nabízíte přihlášení pomocí anonymní služby UUCP, jejíž heslo je veřejně dostupné.

Před tímto typem podvodníků se *musíte* chránit. Lék na tuto nemoc spočívá v tom, že budete po každém systému vyžadovat, aby používal konkrétní přihlašovací jméno, což lze provést uvedením volby *called-login* v souboru `sys`. Vzorový záznam pro daný systém by mohl vypadat asi takto:

```
system          pablo
... obvyklé volby ...
called-login    Upablo
```

Výsledkem je, že kdykoliv se systém přihlásí a bude předstírat, že je hostitel **pablo**, zkontroluje program **uucico**, zda se tento systém přihlásil jako uživatel **Upablo**. Jestliže se volající systém nepřihlásil jako tento uživatel, bude odmítnut a spojení bude zavěšeno. Měli byste si zvyknout přidat volbu *called-login* ke každému záznamu systému, který přidáte do souboru `sys`. Je důležité, abyste toto opatření provedli u *všech* systémů, bez ohledu na to, zda se k vám budou či nebudou připojovat. U systémů, které vám nevolají, byste měli nastavit volbu *called-login* na nějaké úplně vymyšlené jméno uživatele, například na jméno **neverlogsin**.

---

122 **tcpd** má obvykle nastaven režim 700, takže jej musíte spouštět jako **root** a ne jako uživatel **uucp**. O démonu **tcpd** jsme hovořili v kapitole 12.

123 Můžete například použít **mgetty** Gerta Doeringa. Běží na řadě platforem včetně SCO Unixu, AIX, SunOS, HP-UX a Linux.

124 Ve verzi 1.04 není tato volba možná.

Buďte paranoidní – sekvenční kontrola hovorů

Další způsob, který může detekovat a odrazit podvodníky, představuje použití sekvenčních kontrol hovorů. Sekvenční kontrola hovorů vám může pomoci s ochranou před vetřelci, kteří si nějakým způsobem opatřili přihlašovací heslo vašeho systému s protokolem UUCP.

Při použití sekvenční kontroly hovorů si oba počítače uchovávají záznamy o počtu v minulosti uskutečněných hovorů. S každým spojením se tento počet zvyšuje. Po procesu přihlášení vyšle volající svůj sekvenční počet hovorů a volaný si tento počet zkontroluje se svým vlastním počtem uskutečněných hovorů. Jestliže si počty neodpovídají, bude pokus o uskutečnění spojení odmítnut. Je-li počáteční hodnota volena náhodně, bude uhodnutí správné hodnoty velmi obtížné.

Avšak sekvenční kontrola hovorů toho dokáže ještě více: dokonce i když se nějaké velmi chytré osobě povede detekovat váš sekvenční počet hovorů společně s vaším heslem, stejně na to přijdete. Když nějaký útočník zavolá na systém UUCP vašeho zásobitele a ukradne vám poštu, zvýší se u vašeho zásobitele sekvenční číslo hovoru o jedničku. Když se v budoucnu budete chtít spojit s vaším zásobitelem a pokusíte-li se přihlásit, vzdálený program **uucico** vás odmítne, protože čísla se nebudou shodovat!

Jestliže máte povolenou sekvenční kontrolu hovorů, měli byste pravidelně sledovat logovací soubory, jestli se v nich nevyskytují chybová hlášení, která by mohla naznačovat možné útoky. Jestliže váš systém odmítne sekvenční počet hovorů, který poskytne volající systém, vloží program **uucico** do logovacího souboru zprávu, která říká něco ve smyslu „Out of sequence call rejected“. Jestliže je váš systém odmítnut svým zásobitelem, protože sekvenční čísla nejsou synchronní, vloží do logovacího souboru hlášení „Handshake failed (RBADSEQ)“.

Chcete-li povolit sekvenční kontrolu, musíte přidat do záznamu systému následující řádku:

```
# povolení sekvenční kontroly hovorů
sequence        true
```

Kromě toho musíte vytvořit soubor obsahující vlastní sekvenční čísla. Protokol Taylor UUCP uchovává sekvenční čísla v souboru s názvem `.Sequence`, jenž se nachází v dočasném adresáři vzdáleného hostitele. Vlastníkem tohoto souboru *musí* být uživatel **uucp** a musí mu být přiřazen režim 600 (to znamená, že do něj může zapisovat a z něho může číst pouze uživatel **uucp**). Nejlepší je tento soubor založit s libovolnou, předem dohodnutou počáteční hodnotou. Jednoduše lze tento soubor vytvořit takto:

```
# cd /var/spool/uucp/pablo
# echo 94316 > .Sequence
# chmod 600 .Sequence
# chown uucp.uucp .Sequence
```

Samozřejmě, že vzdálený systém musí také povolit sekvenční kontrolu hovorů a ta musí mít nastavenu počáteční hodnotu počitadla na přesně stejnou hodnotu, na jakou ho máte nastaven vy.

Anonymní přístup pomocí protokolu UUCP

Chcete-li poskytovat anonymní přístup pomocí protokolu UUCP, musíte nejprve dříve popsaným způsobem nastavit speciální účet pro tuto službu. Běžně se tomuto účtu přiřazuje přihlašovací jméno a heslo **uucp**.

Kromě toho musíte ještě pro neznámé systémy nastavit několik bezpečnostních voleb. Můžete jim například zakázat spouštění jakýchkoliv příkazů na vašem systému. Avšak tyto parametry nemůžete nastavit v záznamu v souboru `sys`, protože příkaz *system* vyžaduje název systému, který vy

zatím neznáte. Protokol Taylor UUCP řeší tento problém pomocí dalšího příkazu *unknown*. Příkaz *unknown* lze použít v souboru `config` ke specifikaci libovolného příkazu, který by za normálních okolností byl uveden v záznamu systému:

```
unknown          remote-receive ~/incoming
unknown          remote-send ~/pub
unknown          max-remote-debug none
unknown          command-path /usr/lib/uucp/anon-bin
unknown          commands rmail
```

První dva řádky povolí neznámým systémům stahování souborů pouze z adresářové struktury, která se nachází pod adresářem `pub`, a posílání souborů povolí pouze do adresáře `incoming`, které se nacházejí ve `/var/spool/uucppublic`. Další řádek sdělí programu **uucico**, aby ignoroval veškeré požadavky ze vzdáleného systému, které by požadovaly zapnutí místního ladění. Poslední dva řádky povolují neznámým systémům spouštět příkaz **rmail**; avšak cesta uvedená u tohoto příkazu sděluje programu **uucico**, aby hledal příkaz **rmail** pouze v soukromém adresáři s názvem `anon-bin`. To vám umožní poskytovat speciální příkaz **rmail**, který bude například umět doručit superuživateli veškerou poštu k prozkoumání. Umožníte tak anonymním uživatelům například zastihnout správce systému, ale zabráníte jim vložit jakoukoliv poštu, která je určena pro jiné systémy.

Abyste povolili anonymní službu UUCP, musíte zadat do souboru `config` minimálně jeden příkaz *unknown*. V opačném případě by program **uucico** odmítl všechny anonymní systémy.

Nízkoúrovňové protokoly protokolu UUCP

Ke sjednání řízení relace a přenosu souborů se vzdáleným hostitelem používá program **uucico** skupinu standardizovaných zpráv. Tato skupina zpráv se často označuje jako takzvaný **vysokoúrovňový protokol**. Během inicializační fáze a během fáze zavěšování se tyto zprávy posílají jako řetězce. Avšak během přenosové fáze se používá doplňkový nízkoúrovňový protokol, který je většinou pro vyšší úrovně transparentní. To proto, aby se například při použití nespolehlivých linek mohla provádět detekce chyb.

Celkový pohled na protokol

Protože se protokol UUCP používá pro rozdílné typy spojení, jako jsou sériové linky nebo sítě na bázi protokolu TCP nebo dokonce X.25, jsou zapotřebí nízkoúrovňové protokoly. Kromě toho různé implementace protokolu UUCP obsahují odlišné protokoly, které v zásadě provádějí stejné služby.

Protokoly se mohou rozčlenit do dvou kategorií: na **proudové** a **paketové** protokoly. Proudové protokoly přenášejí soubor jako celek, eventuálně mohou u tohoto souboru vypočítat kontrolní součet. Tento typ nemá žádnou režii, avšak vyžaduje spolehlivé spojení, protože jakákoliv chyba způsobí, že soubor musí být celý poslán znovu. Tyto protokoly se zpravidla používají u spojení na bázi protokolu TCP, avšak jejich použití není vhodné u telefonních linek. I když moderní modemy odvádějí poměrně dobrou práci, co se týká korekce chyb, přesto nejsou perfektní a navíc ani neexistuje žádná detekce chyb vzniklých mezi vaším počítačem a modemem.

Na druhou stranu paketové protokoly rozdělují soubor na několik stejně velikých kousků. Každý paket je posílán a přijímán odděleně, je vypočítáván kontrolní součet a zasílateli je podáváno potvrzení o přijetí. Aby se doprava ještě více zefektivnila, byly vyvinuty protokoly s proměnlivou délkou okna, které připouštějí v libovolném okamžiku omezený počet nevyřízených potvrzení. To

značně sníží množství času, které stráví program **uucico** během přenosu vyčkáváním. Vzhledem k vyšší režii oproti proudovým protokolům jsou paketové protokoly nevýhodné pro sítě jako TCP, jsou však ideální pro telefonní linky.

Rozdíl je i v šířce přenosové cesty. Někdy je po sériovém spojení nemožné poslat osmibitové znaky, například když spojení prochází přes hloupý terminálový server, který nejvyšší bit ořezává. Při posílání 8bitových znaků po 7bitové lince se musí osmibitové znaky nahrazovat speciální sekvencí. V nejhorším případě se může objem přenášených dat zdvojnásobit, i když toto zvětšení může být nakonec určitým způsobem kompenzováno interní hardwarovou kompresí. Linky, jež jsou schopny přenášet libovolné osmibitové znaky, se označují jako *8-bit clean*. To je případ všech spojení na bázi protokolu TCP, stejně tak je tomu i u většiny modemových spojení.

V protokolu Taylor UUCP verze 1.06 je dostupná řada UUCP protokolů. Nejběžnějšími jsou:

g Toto je nejběžnější protokol a měly by mu rozumět snad všechny programy **uucico**. Protokol *g* umí důkladnou detekci chyb, a proto je vhodný zejména pro nekvalitní telefonní linky. Protokol *g* vyžaduje osmibitovou datovou cestu. Je to paketově orientovaný protokol používající techniku proměnlivých oken.

i Obousměrný paketový protokol, který může současně posílat a přijímat soubory. Tento protokol vyžaduje spojení s obousměrným přenosem dat a osmibitovou datovou cestu. V současné době je podporován pouze v Taylor UUCP.

t Tento protokol je navržen pro použití u spojení na bázi protokolu TCP nebo u jiných skutečně bezchybných sítí. Používá pakety o velikosti 1024 bajtů a vyžaduje osmibitovou datovou cestu.

e Tento protokol dělá v zásadě totéž, co protokol *t*. Hlavní odlišnost spočívá v tom, že protokol *e* je proudový protokol, a je tedy určen pro spolehlivá síťová spojení.

f Tento protokol byl navržen pro použití u spolehlivých spojení X.25. Je to proudový protokol a používá sedmibitovou datovou cestu. Osmibitové znaky jsou kódovány, což může být značně neefektivní.

G Toto je verze protokolu *g* z balíku System V Release 4. Tento protokol také zvládají některé další verze protokolu UUCP.

a Tento protokol je podobný protokolu *ZMODEM*. Vyžaduje osmibitové spojení, avšak kóduje určité kontrolní znaky, jako je například XON a XOFF.

Nastavení přenosového protokolu

Všechny protokoly umožňují určitou změnu velikosti paketů, změnu hodnot překročení časového limitu a podobně. Ve standardních podmínkách obvykle fungují implicitní hodnoty dobře, avšak v určitých situacích nemusí být optimální. Například protokol *g* může používat velikosti oken v rozmezí od 1 do 7 a velikosti paketů, odpovídajících mocninám čísla 2, v rozsahu od 64 do 4096. Jestliže je vaše telefonní linka většinou rušená tak, že se po ní ztrácí více než 5 procent všech paketů, měli byste asi snížit velikost paketu a zmenšit velikost okna. Na druhou stranu, u velmi kvalitních telefonních linek vám může připadat režie protokolu příliš nehospodárná, protože protokol bude posílat potvrzení pro každých 128 bajtů. V tomto případě můžete chtít zvýšit velikost paketu na hodnotu 512 bajtů nebo dokonce na 1024 bajtů.

Protokol Taylor UUCP umožňuje nastavit parametry protokolů příkazem *protocol-parameter* v souboru `sys`. Například chcete-li pro komunikaci s hostitelem **pablo** nastavit u protokolu *g* velikost paketu na 512 bajtů, přidejte do souboru `sys` následující řádku:

```
system          pablo
...
protocol-parameter g  packet-size 512
```

Nastavitelné parametry a jejich názvy se liší protokol od protokolu. Jejich popis najdete v dokumentaci, která je součástí zdrojového kódu protokolu Taylor UUCP.

Výběr konkrétních protokolů

Ne každá implementace programu **uucico** komunikuje a rozumí každému protokolu, takže během inicializační fáze se musí oba procesy dohodnout na použití společného protokolu. Řídicí program **uucico** nabídne řízenému programu uucico seznam podporovaných protokolů (pošle řetězec `P`*seznam*), ze kterého si může řízený program vybrat nějaký protokol.

Na základě typu používaného portu (modem, protokol TCP nebo přímá linka) sestaví program **uucico** implicitní seznam protokolů. U modemových a přímých spojení bude tento seznam zpravidla obsahovat protokoly *i*, *a*, *g*, *G* a *j*. U spojení na bázi protokolu TCP bude seznam obsahovat protokoly *t*, *e*, *i*, *a*, *g*, *G*, *j* a *f*. Tento seznam můžete přepsat pomocí příkazu *protocols*, jenž může být uveden buď v záznamu systému, nebo v záznamu portu. Například v souboru `port` můžete upravit záznam vašeho modemového portu následujícím způsobem:

```
port            serial1
...
protocols       igG
```

Tento příkaz bude vyžadovat, aby veškerá přicházející a odcházející spojení na tomto portu používala protokoly *i*, *g* nebo *G*. Jestliže vzdálený systém nepodporuje ani jeden z těchto protokolů, pak bude rozhovor neúspěšný.

Hledání a odstraňování problémů

Tato stať popisuje to, co by mohlo zlobit u vašeho spojení pomocí protokolu UUCP a dává vám doporučení, kde byste měli hledat chyby. I když dále popsané problémy představují ty nejčastější potíže, může dojít i k úplně jiným komplikacím.

Máte-li jakékoliv problémy, povolte ladění pomocí volby `-xall` a podívejte se na výpis `Debug`, který se nachází v dočasném adresáři. Tento výpis by vám měl rychle pomoci rozpoznat, v čem problém vězí. Pokud se modem nemůže připojit, bývá užitečné zapnout mu hlasový výstup. U modemů kompatibilních se standardem Hayes to lze provést přidáním řetězce „`ATL1M1 OK`“ do komunikačního skriptu modemu, jenž se nachází v souboru `dial`.

Nejprve byste měli vždy zkontrolovat, zda jsou správně nastavena všechna práva u souborů. Program **uucico** by měl mít nastaveno uid na **uucp** a všechny soubory v adresářích `/usr/lib/uucp`, `/var/spool/uucp` a `/var/spool/uucppublic` by měly být vlastněny uživatelem **uucp**. V dočasném adresáři existuje také několik skrytých souborů[125], které musí rovněž vlastnit uživatel **uucp**.

Pokud jsou všechna práva správně nastavena a stále něco není v pořádku, můžete se zkusit orientovat podle chybových hlášení. Nyní si představíme některé nejčastější chyby a problémy.

125 Tedy souborů jejichž název začíná tečkou. Příkaz **ls** takové soubory standardně nevypisuje.

Program uucico vypíše „Wrong time to call“

Toto hlášení obvykle znamená, že jste buď v záznamu systému v souboru `sys` nezadali příkaz *time*, který určuje, v jakých časech se můžete spojit se vzdáleným systémem, nebo jste v něm zadali takovou hodnotu, jež v současné době zakazuje volání. Jestliže není poskytnut žádný časový plán volání, bude program **uucico** předpokládat, že se nedá se systémem nikdy spojit.

Program uucico uvádí, že daný systém je již zablokován

To znamená, že program **uucico** detekoval pro daný vzdálený systém zamykací soubor v adresáři `/var/spool/uucp`. Zamykací soubor může zůstat z dřívějšího spojení se systémem, které zhavarovalo nebo které bylo zrušeno. Další možnost je, že existuje ještě jeden proces **uucico**, který se snažil dovolat na vzdálený systém a zůstal viset někde v komunikačním skriptu nebo se zablokoval z jiné příčiny.

Jestliže se procesu **uucico** nepodaří úspěšně spojit se vzdáleným systémem, ukončete jej příkazem `kill` a smažte všechny zamykací soubory, které po tomto procesu zůstaly.

Lze se spojit se vzdáleným systémem, ale komunikační skript nefunguje správně

Podívejte se na text, jenž obdržíte ze vzdáleného systému. Jestliže je zkomolený, může to naznačovat, že jde o problém s rychlostí. V opačném případě si ověřte, že obdržený text skutečně souhlasí s textem, který očekává váš komunikační skript. Komunikační skript vždy začíná očekávaným řetězcem. Jestliže obdržíte výzvu k přihlášení a pošlete svoje uživatelské jméno, ale už neobdržíte výzvu k zadání hesla, vložte mezi tyto řetězce (nebo dokonce mezi jednotlivé znaky) nějakou prodlevu.

Modem nevytáčí

Jestliže váš modem neindikuje vzestup linky DTR, když se váš program **uucico** snaží volat směrem ze systému, pravděpodobně jste programu **uucico** nesdělili správné zařízení. Jestliže modem rozeznává linku DTR, ověřte si pomocí terminálového programu, že na toto zařízení můžete posílat znaky. Jestliže posílání znaků funguje, zapněte na začátku komunikačního skriptu modemu opakování znaků pomocí direktivy *\E*. Jestliže se ani po zapnutí opakování znaků nebudou během komunikačního skriptu modemu zobrazovat vaše příkazy, zkontrolujte, zda není na váš modem rychlost linky příliš vysoká nebo nízká. Jestliže vidíte opakování znaků, zkontrolujte, zda jste vypnuli odpovědi modemu nebo zda jste je nenastavili na číselné kódy. Zkontrolujte, zda je vlastní komunikační skript správný. Pamatujte si, že pokud chcete poslat na modem zpětné lomítko, musíte za sebou napsat dvě zpětná lomítka.

Modem volá, ale nedovolá se

Do telefonního čísla vložte prodlevu, zejména pokud voláte z vnitřní telefonní sítě společnosti. Zkontrolujte, zda používáte správný typ volby, protože některé společnosti nepodporují pulsní i tónovou volbu. A konečně zkontrolujte, zda voláte správné číslo.

Přihlášení uspělo, další komunikace však ne

To může být způsobeno řadou příčin. Hodně by vám měly pomoci záznamy uvedené v logovacím souboru. Podívejte se na seznam protokolů, které nabízí vzdálený systém (vzdálený systém pošle během inicializační fáze řetězec `P` *seznam*). Aby mohla komunikace pokračovat, oba systémy musejí podporovat alespoň jeden společný protokol.

Jestliže vzdálený systém pošle řetězec `RLCK`, v tom případě na vzdáleném systému existuje pro váš systém starý zamykací soubor. Jestliže to není z důvodu, že jste již spojeni se vzdáleným systémem na nějaké jiné lince, požádejte o jeho odstranění.

Jestliže vzdálený systém pošle řetězec `RBADSEQ`, v tom případě je na vzdáleném systému zapnuta sekvenční kontrola hovorů, avšak sekvenční počty hovorů si neodpovídají. Jestliže vzdálený systém pošle řetězec `RLOGIN`, nebylo vám povoleno přihlášení pod daným účtem.

Logovací soubory a ladění

Pokud byl váš balík UUCP sestaven s logováním podle implementace Taylor UUCP, budete mít pouze tři globální logovací soubory, které budou umístěny v adresáři fronty. Hlavní logovací soubor se jmenuje `Log` a obsahuje všechny informace o uskutečněných spojení a o přenesených souborech. Typický výpis z tohoto souboru vypadá asi následovně (po malém přeformátování, aby se vešel na šířku stránky):

```
uucico pablo - (1994-05-28 17:15:01.66 539) Calling system pablo (port cua3)
uucico pablo - (1994-05-28 17:15:39.25 539) Login successful
uucico pablo - (1994-05-28 17:15:39.90 539) Handshake successful
               (protocol 'g' packet size 1024 window 7)
uucico pablo postmaster (1994-05-28 17:15:43.65 539) Receiving D.pabloB04aj
uucico pablo postmaster (1994-05-28 17:15:46.51 539) Receiving X.pabloX04ai
uucico pablo postmaster (1994-05-28 17:15:48.91 539) Receiving D.pabloB04at
uucico pablo postmaster (1994-05-28 17:15:51.52 539) Receiving X.pabloX04as
uucico pablo postmaster (1994-05-28 17:15:54.01 539) Receiving D.pabloB04c2
uucico pablo postmaster (1994-05-28 17:15:57.17 539) Receiving X.pabloX04c1
uucico pablo - (1994-05-28 17:15:59.05 539) Protocol 'g' packets: sent 15,
                resent 0, received 32
uucico pablo - (1994-05-28 17:16:02.50 539) Call complete (26 seconds)
uuxqt pablo postmaster (1994-05-28 17:16:11.41 546) Executing X.pabloX04ai
               (rmail okir)
uuxqt pablo postmaster (1994-05-28 17:16:13.30 546) Executing X.pabloX04as
               (rmail okir)
uuxqt pablo postmaster (1994-05-28 17:16:13.51 546) Executing X.pabloX04c1
               (rmail okir)
```

Další důležitý logovací soubor se jmenuje `Stats` a jsou v něm vypsány statistiky přenosů. Část ze souboru `Stats`, která odpovídá výše uvedenému přenosu, vypadá asi takto:

```
postmaster pablo (1994-05-28 17:15:44.78)
                  received 1714 bytes in 1.802 seconds (951 bytes/sec)
postmaster pablo (1994-05-28 17:15:46.66)
                  received 57 bytes in 0.634 seconds (89 bytes/sec)
postmaster pablo (1994-05-28 17:15:49.91)
                  received 1898 bytes in 1.599 seconds (1186 bytes/sec)
postmaster pablo (1994-05-28 17:15:51.67)
                  received 65 bytes in 0.555 seconds (117 bytes/sec)
postmaster pablo (1994-05-28 17:15:55.71)
                  received 3217 bytes in 2.254 seconds (1427 bytes/sec)
postmaster pablo (1994-05-28 17:15:57.31)
                  received 65 bytes in 0.590 seconds (110 bytes/sec)
```

Třetím souborem je soubor `Debug`. Do tohoto souboru se zapisují ladicí informace. Pokud používáte ladění, měli byste se ujistit, že má tento soubor nastavena přístupová práva 600. V závislosti na vybraném ladicím režimu může tento soubor obsahovat přihlašovací jména a hesla, která používáte při spojení se vzdáleným systémem.

Pokud používáte nějaké nástroje, které předpokládají logovací soubory v tradičním formátu používaném HDB kompatibilními implementacemi, můžete Taylorovo UUCP přeložit tak, aby vytvářelo logovací záznamy v tomto formátu. Jedná se o pouhé nastavení příslušné volby při překladu v souboru `config.h`.

KAPITOLA 17

Elektronická pošta

Elektronická pošta představuje jedno z nejvýznamnějších využití síťových služeb od doby, kdy byla síť vynalezena. Původně to byla jednoduchá služba, která kopírovala soubor z jednoho počítače do druhého, kde ho připojila k souboru *poštovní schránky* příjemce. V zásadě tento mechanismus zůstal stejný, i když stále rostoucí síť vedla se svými složitými směrovacími požadavky a se svým stále se zvyšujícím počtem zpráv k nutnosti vytvoření propracovanějšího schématu.

Byly vyvinuty různé standardy výměny elektronické pošty. Systémy v síti Internet se drží standardu RFC 822 doplněného některými dalšími RFC standardy, které umožňují prostřednictvím elektronické pošty strojově nezávislý přenos prakticky *čehokoliv* včetně grafiky, zvuku a nestandardních znaků[126]. Společnost CCITT definovala další standard, nazvaný X.400. Stále se používá v některých velkých společnostech a ve státních sítích, postupně se však vytrácí.

Na platformě Unix byl implementován velký počet programů pro přenos pošty. Jedním z nejznámějších programů je **sendmail**, vyvinutý Ericem Allmanem na University of California v Berkeley. Eric Allman dnes nabízí **sendmail** na komerční bázi, samotný program však zůstává k dispozici zdarma. **sendmail** se v některých distribucích Linuxu používá jako standardní poštovní program. O jeho konfiguraci budeme hovořit v kapitole 18.

Dále se na Linuxu používá **Exim** napsaný Philipem Hazelem na University of Cambridge. Konfiguraci programu **Exim** popisujeme v kapitole 19.

V porovnání se **sendmailem** je **Exim** docela mladý. Pro většinu aplikací budou vyhovovat oba tyto programy.

Jak **Exim** tak **sendmail** vycházejí z řady konfiguračních souborů, které si musíte upravit podle potřeb vašeho systému. Kromě nastavení, která jsou pro činnost elektronické pošty nezbytná (například název systému), existuje celá řada dalších parametrů, které je možné doladovat. Hlavní konfigurační soubor programu **sendmail** je na první pohled velmi těžko čitelný. Vypadá skoro jako něco, co napsala kočka pobíhající po klávesnici se zapnutým Shiftem. Konfigurační soubory programu **Exim** jsou strukturovanější a snáze čitelné. **Exim** však nenabízí přímou podporu UUCP a zpracovává pouze doménové adresy. Dnes už to není tak zásadní omezení jako v minulosti a v řadě případů možnosti programu **Exim** plně dostačují. Ať už použijete kterýkoliv z těchto programů, nároky na jejich nastavení jsou přibližně stejné.

V této kapitole si vysvětlíme co to elektronická pošta je a s jakými problémy se musí její administrátor vyrovnat. V kapitolách 18 a 19 uvádíme návod pro počáteční nastavení programů **sendmail** a **Exim**. Uvedené informace by měly postačit ke zprovoznění pošty na typickém systému, kromě toho je však k dispozici i celá řada dalších voleb a doladováním těch nejjemnějších nuancí můžete strávit spoustu šťastných hodin.

126 Pokud tomuto tvrzení nevěříte, přečtěte si dokument RFC 1437.

Ke konci této kapitoly se stručně zmíníme o programu **elm**, což je na unixových systémech velmi často používaný poštovní klient.

Další informace týkající se elektronické pošty na Linuxu najdete v dokumentu Electronic Mail HOWTO Guylhema Aznara[127], který se pravidelně objevuje na comp.os.linux. Zdrojové distribuce programů **elm**, **Exim** a **sendmail** rovněž obsahují rozsáhlou dokumentaci, která by vám měla zodpovědět většinu otázek týkajících se jejich instalace. V příslušných místech textu se na tyto dokumentace odkazujeme. Pokud potřebujete nějaké obecné informace o elektronické poště, existuje na toto téma řada RFC dokumentů. Ty jsou uvedeny na konci knihy.

Co je to poštovní zpráva?

Poštovní zpráva se zpravidla skládá z těla zprávy, což je samotný text zprávy, a ze speciálních administrativních dat, která označují příjemce, přenosové médium a podobně – trochu připomínají informace na dopisní obálce.

Tato administrativní data spadají do dvou kategorií; v první kategorii jsou zastoupena všechna data vztahující se k transportnímu médiu, jako je adresa odesílatele a adresa příjemce. Z tohoto důvodu se tato kategorie nazývá *obálka*. V závislosti na tom, kudy prochází daná zpráva, mohou být data z této kategorie transportním softwarem transformována.

Druhou kategorii představují všechna data, která jsou nutná pro manipulaci s poštovní zprávou a která se nevztahují k žádnému transportnímu mechanismu. Příkladem může být řádka s předmětem zprávy, seznam všech příjemců nebo datum zaslání dané zprávy. V mnoha sítích se stalo standardem, že tato kategorie dat je uvedena před vlastní poštovní zprávou a tvoří takzvanou *poštovní hlavičku*. Od *textu zprávy* je hlavička oddělena prázdným řádkem[128].

Ve světě Unixu používá většina transportních programů formát hlavičky, který byl definován v dokumentu RFC 822. Jeho původním záměrem bylo vytvořit standard pro použití v sítích ARPANET, ale protože byl navržen jako nezávislý na prostředí, byl jednoduše upraven pro použití v dalších sítích, včetně mnoha sítí založených na protokolu UUCP.

Nicméně dokument RFC 822 je pouze největším obecným standardem. Vznikly i novější standardy, a to z důvodu zvyšujících se potřeb, jako je šifrování dat, podpora mezinárodních znakových sad a podpora multimediálního rozšíření pošty (MIME, viz RFC 1341).

Ve všech těchto standardech se hlavička zprávy skládá z několika řádků, za nimiž následuje prázdný řádek. Řádek obsahuje název pole, které začíná ve sloupci jedna, a vlastní pole, které je odděleno dvojtečkou a mezerou. Formát a sémantika každého pole jsou odlišné a závisí na názvu daného pole. Pole hlavičky může pokračovat i na novém řádku v případě, že následující řádek začíná mezerou nebo tabulátorem. Pole mohou být uspořádána v libovolném pořadí.

Typická hlavička poštovní zprávy vypadá asi takto:

```
Return-Path: <ph10@cus.cam.ac.uk>
Received: ursa.cus.cam.ac.uk (cusexim@ursa.cus.cam.ac.uk [131.111.8.6])
    by al.animats.net (8.9.3/8.9.3/Debian 8.9.3-6) with ESMTP id WAA04654
    for <terry@animats.net>; Sun, 30 Jan 2000 22:30:01 +1100
Received: from ph10 (helo=localhost) by ursa.cus.cam.ac.uk with local-smtp
    (Exim 3.13 #1) id 12EsYC-0001eF-00; Sun, 30 Jan 2000 11:29:52 +0000
Date: Sun, 30 Jan 2000 11:29:52 +0000 (GMT)
From: Philip Hazel <ph10@cus.cam.ac.uk>
Reply-To: Philip Hazel <ph10@cus.cam.ac.uk>
```

127 Guylhema můžete kontaktovat na adrese *guylhem@danmark.linux.eu.org*.

128 Kromě toho bývá zvykem připojovat ke zprávě *signaturu* (`.sig`), která obsahuje informace o odesílateli a případně nějaký vtip nebo citát. Ta bývá od těla zprávy oddělena řádkem obsahujícím znaky „--„ a mezeru.

```
To: Terry Dawson <terry@animats.net>, Andy Oram <andyo@oreilly.com>
Subject: Electronic mail chapter
In-Reply-To: <38921283.A58948F2@animats.net>
Message-ID: <Pine.SOL.3.96.1000130111515.5800A-200000@ursa.cus.cam.ac.uk>
```

Všechna potřebná pole hlavičky jsou obvykle vygenerována používaným poštovním klientem, což může být například **elm**, **pine**, **mush** nebo **mailx**. Některá pole jsou volitelná a může je přidat sám uživatel. Například program **elm** umožní upravit část hlavičky zprávy. Další pole jsou přidána poštovním transportním programem. Pokud se podíváte do své poštovní schránky, zjistíte, že každá zpráva začíná řádkem „From“ (bez mezery!) Nejedná se o hlavičku dle standardu RFC 822, tento řádek doplnil poštovní klient kvůli pohodlí programů, kterými poštu prohlížíte. Aby se předešlo případným problémům s řádky textu, které začínají textem „From“, stalo se standardem začínat takovéto řádky znakem „>“.

Následuje seznam běžně používaných polí hlavičky a jejich význam:

`From:` Toto pole obsahuje adresu odesílatele a může eventuálně obsahovat i jeho „skutečné jméno“. Pro toto pole se používá obrovské množství formátů.

`To:` Toto pole obsahuje adresu příjemce.

`Cc:` Seznam adresátů, kteří obdrží kopii zprávy. Jednotlivé adresy se od sebe oddělují čárkami.

`Bcc:` Seznam adresátů, kteří obdrží kopii zprávy. Rozdíl mezi poli „Cc:“ a „Bcc:“ spočívá v tom, že adresy uvedené v poli „Bcc:“ se neobjeví ve zprávách doručených jednotlivým adresátům. Tímto způsobem můžete poslat kopii zprávy více lidem, aniž by se jednotliví adresáti o sobě navzájem dozvěděli. Jednotlivé adresy se od sebe oddělují čárkami.

`Subject:` Toto pole obsahuje popis obsahu zprávy vyjádřený několika slovy.

`Date:` Toto pole obsahuje datum, kdy byla zpráva odeslána.

`Reply-To:` Toto pole určuje adresu, na kterou chce odesílatel směrovat odpověď od příjemce. Pole může být užitečné v případě, že máte několik účtů, ale přitom chcete dostávat poštu pouze na adresu, kterou používáte nejčastěji. Toto pole je volitelné.

`Organization:` Toto pole obsahuje společnost, která vlastní počítač a z něhož pochází daná pošta. Máte-li svůj počítač v soukromém vlastnictví, nechte toto pole buď volné, nebo sem zadejte řetězec „private“, případně nějaký nesmysl. Toto pole není definováno žádným RFC dokumentem a je úplně nepovinné. Některé poštovní programy je podporují, jiné ne.

`Message-ID:` Toto pole obsahuje řetězec, který vygeneroval transportní systém původního odesílatele. Představuje jednoznačný identifikátor zprávy.

`Received:` Toto pole vloží do hlavičky každý systém, který zpracoval vaši poštu (včetně počítačů odesílatele a příjemce). V něm je uveden název daného systému, identifikátor zprávy, čas a datum, kdy daný systém obdržel tuto zprávu, dále od kterého systému tato zpráva pochází a který transportní program byl použit k jejímu doručení. Tyto informace se uvádějí z toho důvodu, abyste mohli sledovat směrování pošty a případně si stěžovat příslušné zodpovědné osobě.

`X-cokoliv`: Žádný z poštovních programů by neměl mít potíže s jakoukoliv hlavičkou začínající řetězcem X-. Tento řetězec se používá kvůli implementaci doplňkových vlastností, jež zatím nebyly vloženy do standardu RFC, nebo které do něj ani vloženy nebudou. Tento řetězec například používala poštovní konference Linux Activists, kde se pomocí hlavičky `X-Mn-Key:` vybíral požadovaný kanál konference.

Jak se pošta doručuje?

Typicky vytvoříte poštu pomocí nějakého poštovního rozhraní, například pomocí programu **mail** nebo **mailx**; nebo pomocí dokonalejších programů, jako je **elm**, **mush** nebo **pine**. Takový program se nazývá *poštovní uživatelský agent* (*mail user agent*), zkráceně MUA. Když se posílá nějaká poštovní zpráva, ve většině případů předá uživatelský agent poštu k doručení dalšímu programu. Tento program se nazývá *poštovní přenosový agent* (*mail transport agent*), zkráceně MTA. Ve většině systémů se pro lokální i vzdálené doručování používá stejný přenosový agent, kterým typicky bývá `/usr/sbin/sendmail` (nebo na systémech nekompatibilních s FSSTN `/usr/lib/sendmail`). U systémů založených na UUCP se obvykle používají dva přenosoví agenti, **rmail** pro vzdálené doručení a **lmail** pro lokální doručení.

Místní doručování znamená samozřejmě více, než jen pouhé přidání příchozí zprávy do poštovní schránky příjemce. Místní MTA bude obvykle rozumět aliasům (což je nastavení, kdy místní adresy příjemce odkazují na další adresy) a předávání (což je přesměrování pošty uživatele na nějakou jinou destinaci). Také zprávy, které nemohou být doručeny, musí být *odmítnuty*, což znamená, že je systém musí vrátit odesílateli společně s nějakou chybovou zprávou.

U vzdáleného doručování závisí použitý transportní program na povaze daného spojení. Při doručování pošty po sítích TCP/IP se nejčastěji používá protokol SMTP (*Simple Mail Transfer Protocol*), popsaný v dokumentu RFC 821. Protokol SMTP byl navržen pro přímé doručení pošty na počítač adresáta, přičemž přenos pošty se sjedná s démonem SMTP, který je spuštěn na vzdálené straně. Dnes bývá obvyklé, že větší společnosti používají pro doručování pošty všem příjemcům jediný počítač, který je pak správně předá tam, kam patří.

V sítích na bázi protokolu UUCP nebývá obvykle pošta doručována přímo, ale je doručena požadovanému hostiteli přes několik zprostředkujících systémů. Posílá-li se zpráva pomocí spojení na bázi protokolu UUCP, spustí MTA odesílatele obvykle pomocí příkazu **uux** na doručujících systémech program **rmail** a pošle mu na standardní vstup danou zprávu.

Protože se tento proces provádí samostatně pro každou zprávu, může způsobit jak značné pracovní zatížení na hlavních poštovních uzlech, tak i přeplnění dočasné fronty protokolu UUCP stovkami malých souborů, které zabírají neúměrné množství místa na disku[129]. Proto někteří MTA umožní seskupit několik zpráv pro vzdálený systém do jednoho dávkového souboru. Dávkový soubor obsahuje příkazy protokolu SMTP, které by za normálních okolností spustil místní hostitel, kdyby se použilo přímé spojení. Tento postup se označuje jako protokol BSMTP neboli *dávkový* protokol SMTP. Dávka je potom předána programům **rsmtp** nebo **bsmtp** na vzdáleném systému, které zpracují vstup stejným způsobem, jako kdyby došlo k normálnímu SMTP spojení.

129 Diskový prostor se totiž typicky alokuje v blocích o velikosti 1024 bajtů, takže i pár desítek bajtů dlouhá zpráva zabere celý kilobajt.

Adresy elektronické pošty

U elektronické pošty se adresa skládá přinejmenším ze dvou částí. Jednou částí je název *poštovní domény*, který se finálně přeloží buď na adresu počítače příjemce nebo na adresu jiného počítače, který poštu pro příjemce přijme. Druhou částí je nějaký způsob jednoznačné identifikace uživatele, což může být přihlašovací jméno, skutečné jméno ve tvaru „Křestní.Příjmení“ nebo jakákoliv přezdívka, která se přeloží na uživatele nebo skupinu uživatelů. Jiná poštovní adresní schémata, jako je X.400, používají obecnější množinu „atributů“, které slouží k vyhledání hostitele příjemce na adresářovém serveru X.500.

Způsob interpretace poštovní adresy značně závisí na používaném typu sítě. Zaměříme se na to, jak poštovní adresy interpretují sítě TCP/IP a UUCP.

RFC 822

Systémy v síti Internet dodržují standard popsaný v RFC 822, který vyžaduje notaci *user@host.domain*, kde *host.domain* je plně kvalifikované doménové jméno daného hostitele. Spojovacím členem je znak „zavináč“ (@), který se v tomto kontextu často vyslovuje jako „at“[130].

Tato notace nespecifikuje cestu k cílovému počítači. Směrování zprávy je ponecháno na jiných mechanismech, o kterých se ještě krátce zmíníme.

Pokud používáte systém připojený k Internetu, potkáte se se standardem RFC 822 poměrně často. Jeho použití se nevztahuje jen na poštu, ale i na jiné služby, jako jsou například news. Použití standardu RFC 822 pro službu news popisujeme v kapitole 20.

Starší formáty adres

V původním prostředí protokolu UUCP se používaly adresy *path!host!user*, kde cesta *path* definuje sekvenci hostitelů, jimiž musí zpráva projít, než dorazí k cílovému hostiteli *host*. Tato forma zápisu se nazývá *vykřičníková notace*[131], podle znaku vykřičníku, který se používá v jejím zápisu. Dnes již mnoho sítí založených na protokolu UUCP adoptovalo standard z dokumentu RFC 822, a tudíž bude rozumět i doménové adrese.

Na jiných sítích se stále používají i jiné formáty adres. Například sítě založené na DECnet používají jako oddělovač dvě dvojtečky a pracují s adresami ve tvaru *host::user*[132]. Standard X.400 používá úplně odlišné adresační schéma, které příjemce popisuje dvojicemi příznak-hodnota, jako například stát nebo organizace.

Konečně FidoNet identifikuje každého uživatele zápisem jako 2:320/204.9, který obsahuje čtyři číselné údaje definující zónu (2 je Evropa), síť (320 je Paříž a okolí), uzel (lokální rozbočovač) a bod (počítač koncového uživatele). Adresy sítě FidoNet je možné mapovat na adresy formátu RFC 822, předchozí zápis by se zadal takto: Thomas.Quinot@p9.f204.n320.z2.fidonet.org. Je celkem zřejmé, že doménové adresy z toho všeho vycházejí nejlépe...

Kombinace různých formátů

Je celkem jasné, že když se dá dohromady spousta různých systémů a spousta chytrých lidí, budou hledat způsoby jak odlišné systémy propojit tak, aby spolu mohly komunikovat. Existuje pro-

130 Pozn. překladatele: Vyslov *et* – tedy „uživatel XY *na* počítači ABC“.

131 Pozn. překladatele: V angličtině *bang path*, v telegrafní angličtině se totiž vykřičník čte „bang“ (podobně jako u nás tečka „stop“.

132 Pokud byste chtěli doručit poštu uživateli sítě DECnet z prostředí RFC 822, můžete použít adresu *"host::user"@relay*, kde *relay* je název nějakého předávacího stroje mezi Internetem a DECnetem.

to řada poštovních bran, které jsou schopny propojovat různé poštovní systémy a zajistit tak předávání pošty z jednoho systému na jiný. Těmito branami se nebudeme nijak podrobně zabývat, zaměříme se spíš na některé komplikace, k nimž může při použití takovýchto systémů dojít.

Představme si situaci kdy kombinujeme vykřičníkové adresy a RFC 822. Tyto dva typy adres se příliš dobře neslučují. Předpokládejme adresu ve tvaru *domainA!user@domainB*. Není zcela jasné, zda znak „@“ bude mít přednost před cestou nebo tomu bude naopak. Pošleme zprávu na doménu B, která ji předá na *domainA!user*, nebo ji pošleme na doménu A, která ji předá na *user@domainB*?

Adresy, v nichž jsou zastoupeny různé typy operátorů adres, se nazývají *hybridní adresy*. Nejznámější z nich vidíte ve výše uvedeném příkladu. Tato adresa je obvykle vyřešena tak, že operátor „@“ dostane přednost před cestou. Ve výše uvedeném příkladu to znamená, že zpráva bude odeslána nejprve na doménu B.

Existuje však způsob, jak ve formátu odpovídajícím RFC 822 zadat přesné směrování zprávy. Adresa *<@domainA,@domainB:user@domainC>* definuje adresu uživatele domény C, ke které se dostaneme přes domény A a B (v tomto pořadí). Tento typ adresy se často označuje jako *směrovaná adresa*. Není nicméně rozumné s tímto chováním počítat, protože revize dokumentu RFC popisující směrování pošty doporučují ignorovat směrované adresy a pokusit se o přímé doručení na vzdálený systém.

Dále existuje ještě adresový operátor „%“: U adresy *user%domainB@domainA* bude zpráva nejprve poslána na doménu A, kde se nejpravější (a v našem případě jediný) znak „%“ nahradí znakem „@“.Takže nyní budeme mít adresu *user@domainB* a zpráva bude spokojeně doručena danému uživateli na doméně B. Tento typ adresy se někdy označuje jako „Ye Olde ARPAnet Kludge“ a jeho používání se nedoporučuje.

Použití odlišných typů adres má určité důsledky, které si popíšeme dále. V prostředí, kde se používá standard RFC 822, byste neměli používat žádné jiné adresy než absolutní doménové adresy tvaru *user@host.domain*.

Jak pracuje směrování pošty?

Proces doručování zprávy hostiteli příjemce se nazývá *směrování*. Kromě nalezení cesty ze systému odesílatele do cílového systému v sobě tento proces zahrnuje i kontrolu chyb a optimalizaci rychlosti a nákladů.

Existuje obrovský rozdíl ve způsobu, jakým se starají o směrování pošty UUCP systémy a internetové systémy. V Internetu provede hlavní práci související se směrováním dat na hostitele příjemce (který je známý prostřednictvím své IP adresy) síťová úroveň IP, zatímco v zóně protokolu UUCP musí být směrování provedeno uživatelem nebo je musí vygenerovat poštovní přenosový agent.

Směrování pošty v Internetu

V Internetu záleží pouze na konfiguraci cílového hostitele, zda se vůbec uskuteční nějaké konkrétní směrování pošty. Implicitně je nastaveno přímé doručení zprávy cílovému hostiteli, tak, že se nejprve zjistí, kterému hostiteli má být zpráva doručena, a pak se mu přímo doručí. Většina systémů obvykle preferuje doručení veškeré příchozí pošty na jeden spolehlivý poštovní server, který je schopen postarat se o veškerou poštu a který ji potom předá místním hostitelům. Tato služba je oznamována v takzvaném MX záznamu v DNS databázi domény. Zkratka MX znamená *Mail Exchanger* a v zásadě říká, že uvedený server slouží jako předávací systém pro veškerou poštu v doméně. Po-

mocí MX záznamů lze také obsluhovat poštu pro hostitele, kteří nejsou přímo připojeni k Internetu, například sítě UUCP nebo FidoNet musejí poštu přijímat přes nějakou bránu.

MX záznamy mají vždy přiřazenu takzvanou *preferenci*, což je celočíselná hodnota. Existuje-li pro jednu doménu více poštovních serverů, bude se pošta doručovat tomu s nejnižším preferenčním číslem a pouze pokud by se to nepovedlo, bude použit další s vyšším preferenčním číslem. Je-li nějaký hostitel zároveň poštovním serverem pro určitou cílovou adresu, smí poštu na tuto adresu doručovat pouze prostřednictvím systémů, které mají preferenční číslo nižší než on sám; což je bezpečný způsob, jak zabránit vytváření poštovních smyček. Pokud pro nějakou doménu neexistuje MX záznam, transportní agent zjistí, zda samotné doméně není přiřazena IP adresa, a pokud je, doručuje poštu přímo na tuto adresu.

Předpokládejme, že například organizace Foobar Inc. chce veškerou poštu spravovat na svém počítači s názvem **mailhub**. Potom bude mít v databázi DNS přibližně následující záznam MX:

```
green.foobar.com          IN   MX     5   mailhub.foobar.com.
```

Tento záznam říká, že na adrese **mailhub.foobar.com** je poštovní server pro doménu **green.foobar.com** s preferencí 5. Hostitel, který chce doručit zprávu na adresu **joe@green.foobar.com**, nahlédne do DNS a nalezne MX záznam, který směřuje na počítač **mailhub**. Pokud neexistuje žádný jiný MX záznam s prioritou menší než 5, bude zpráva doručena poštovnímu serveru **mailhub**, který ji posléze odešle na hostitele **green**.

Výše uvedený příklad je jen zjednodušeným znázorněním toho, jak pracují záznamy MX. Podrobnější informace o směrování pošty na Internetu naleznete v dokumentech RFC 821, RFC 974 a RFC 1123.

Směrování pošty v sítích UUCP

Směrování pošty v sítích na bázi protokolu UUCP je mnohem komplikovanější než v síti Internet, protože vlastní transportní software neprovádí žádné směrování. Dříve musela být veškerá pošta adresována pomocí vykřičníkové notace, která uváděla seznam hostitelů, přes které se doručovala pošta. Jednotliví hostitelé byli odděleni vykřičníkem, za kterým následovalo jméno uživatele. Pokud jste chtěli adresovat dopis uživateli Janet na počítač s názvem **moria**, museli jste zadat cestu jako **eek!swim!moria!janet**. Tato formule poslala poštu z vašeho hostitele na hostitele **eek**, z něj pak na hostitele **swim** a nakonec dorazila pošta z hostitele **swim** na hostitele **moria**.

Zcela zřejmá nevýhoda této techniky spočívá v tom, že musíte znát podrobnosti o síťové topologii, rychlosti linek a podobně. Ještě horší věcí je, že změna v uspořádání síťové topologie – například odstranění některých spojení nebo odstranění nějakého hostitele – může způsobit nedoručení zprávy, protože jste nebyli obeznámeni s provedenými změnami. A konečně v případě, že změníte místo svého působení, budete pravděpodobně muset aktualizovat veškerá směrování.

Použití tohoto mechanismu směrování bylo dáno jedním důvodem – totiž neexistencí centrálně přidělovaných názvů. Předpokládejme, že existují dva systémy s názvem **moria**, jeden ve Spojených státech a druhý ve Francii. Na který systém ale adresa **moria!janet** odkazuje? To bude jasné až tehdy, uvedete-li cestu, pomocí které se lze spojit s hostitelem **moria**.

Prvním krokem, který vedl k odstraňování nejednoznačných názvů hostitelů, byl vznik The UUCP Mapping Project. Tento projekt je umístěn na Rutgers University a registruje všechny oficiální názvy UUCP počítačů společně s informacemi o jejich sousedech a geografickém umístění. Projekt zajišťuje, aby nebyl žádný název hostitele použit dvakrát. Informace získané projektem mapování názvů jsou zveřejňovány jako takzvané *Usenetové mapy*, které jsou pravidelně distribuovány po-

mocí Usenetu. Typická položka systému v mapě vypadá (po odstranění komentářů) následovně[133]:

```
moria
        bert(DAILY/2),
        swim(WEEKLY)
```

Tato položka uvádí, že hostitel **moria** má spojení s hostitelem **bert**, se kterým se spojuje dvakrát denně. Dále má spojení s hostitelem **swim**, s nímž se spojuje každý týden. K formátu souboru s mapami se ještě vrátíme.

Pomocí informací o jednotlivých spojeních, které jsou uvedeny v souborech s mapami, je možné automaticky vygenerovat celou cestu z vašeho hostitele k libovolnému cílovému systému. Tyto informace se obvykle ukládají v souboru `paths`, někdy se tento soubor také nazývá *databáze cest*. Předpokládejme, že v souboru map stojí, že s hostitelem **bert** se můžete spojit přes hostitele **ernie**. V tom případě by položka pro hostitele **moria** v souboru `paths`, který byl vygenerován z výše uvedené části mapy, mohla vypadat asi takto:

```
moria           ernie!bert!moria!%s
```

Pokud nyní zadáte cílovou adresu janet@moria.uucp, vybere MTA agent výše uvedené směrování a pošle zprávu na hostitele **ernie** a jako adresu uvede **bert!moria!janet**.

Není ovšem vhodné vytvářet soubor `paths` ze všech usenetových map. Informace v těchto mapách jsou často poměrně zkreslené a mnohdy i zastaralé. Z tohoto důvodu se vytváří soubor `paths` ze všech světových map protokolu UUCP pouze na několika hlavních hostitelích. Většina systémů spravuje pouze informace o systémech, které jsou v jejich okolí, a poštu pro hostitele, které nenajdou ve své databázi, pošlou chytřejším hostitelům, jež mají kompletnější směrovací informace. Toto schéma se nazývá *smart-host routing*. Hostitelé, kteří udržují pouze jediné poštovní spojení pomocí protokolu UUCP (takzvané *listové systémy*), sami neprovádí žádné směrování; plně se spoléhají na svého chytřejšího hostitele.

Kombinace UUCP a RFC 822

Zatím je nejlepším lékem na problémy se směrováním pošty v sítích na bázi protokolu UUCP převzetí systému DNS do sítí UUCP. Samozřejmě, že pomocí protokolu UUCP se nemůžete dotazovat jmenného serveru. Některé systémy UUCP však vytvořily malé domény, které vnitřně koordinují směrování pošty. V mapách tyto domény zveřejní pouze jednoho nebo dva hostitele, kteří budou označení jako poštovní brány. Tím pádem nemusí v mapách existovat položka pro každého hostitele, který se nachází v příslušné doméně. Brány se starají o veškerou poštu, která přichází do domény nebo která z dané domény odchází. Směrovací schéma uvnitř domény je pro okolní svět neviditelné.

Tento princip funguje velmi dobře se směrováním na chytřejší hostitele. O globální směrovací informace se starají pouze brány; vedlejší hostitelé příslušné domény vystačí pouze s malým ručně vytvořeným souborem `paths`, který uvádí směrování uvnitř jejich domény a směrování na bránu. Dokonce ani poštovní brány nemusí mít informace o směrování na každého UUCP hostitele na světě. Kromě kompletních informací o směrování uvnitř jimi obsluhované domény potřebují mít ve svých databázích pouze směrování na všechny ostatní domény. Například níže uvedená položka cesty bude směrovat veškerou poštu určenou pro systémy v doméně **sub.org** na hostitele **smurf**:

133 Mapy systémů registrovaných pod UUCP Mapping Project jsou distribuovány prostřednictvím skupiny comp.mail.maps, různé organizace mohou dále zveřejňovat vlastní mapy svých sítí.

```
.sub.org          swim!smurf!%s
```

Pošta směřující na adresu **claire@jones.sub.org** bude poslána na hostitele **swim** s adresou **smurf!jones!claire**.

Hierarchické uspořádání prostoru s názvy domén umožňuje poštovním serverům kombinovat přesnější směrování s méně přesným směrováním. Například systém ve Francii může mít přesné směrování pro subdomény v rámci domény **fr**, ale veškerou poštu určenou pro hostitele v doméně **us** bude směrovat na nějaký systém ve Spojených státech. Takovéto doménové směrování značně redukuje velikost směrovacích databází a stejně tak i potřebnou administrativní režii.

Hlavním přínosem použití názvu domén v prostředí sítí UUCP je to, že shoda se standardem RFC 822 umožňuje mezi sítěmi na bázi protokolu UUCP a sítí Internet jednoduchou průchodnost dat. V dnešní době má spousta domén UUCP spojení s internetovou bránou, která se chová jako jejich chytřejší hostitel. Posílání zpráv po Internetu je rychlejší a směrování informací je mnohem spolehlivější, protože hostitelé v Internetu mohou používat místo usenetových map systém DNS.

Aby mohl být systém na bázi protokolu UUCP dosažitelný z Internetu, uvádí obvykle internetové brány záznam MX pro domény na bázi protokolu UUCP (záznamy MX byly popsány výše v části *Směrování pošty v Internetu*). Předpokládejme třeba, že hostitel **moria** patří do domény **orcnet.org**. Hostitel **gcc2.groucho.edu** se chová vůči této doméně jako internetová brána. Z toho důvodu použije hostitel **moria** jako svého chytřejšího hostitele **gcc2**, takže veškeré zprávy pro cizí domény budou doručovány pomocí Internetu. Na druhou stranu hostitel **gcc2** uvede záznam MX pro doménu ***.orcnet.org** a tím pádem může doručit veškerou příchozí poštu určenou pro systémy v doméně **orcnet** na hostitele **moria**. Znak * v ***.ornet.org** znamená všechny hostitele v dané doméně, takže pro ně nemusíte mít samostatné záznamy. Takovéto nastavení je typické pro čistě UUCP domény.

Nyní zůstal už jen poslední problém, totiž že transportní programy protokolu UUCP neumí obsloužit plně kvalifikovaná doménová jména. Většina balíků UUCP byla navržena tak, aby zvládla názvy systémů do délky osmi znaků, některé dokonce i méně, a použití nealfanumerických znaků, jako jsou například tečky, je u většiny z nich absolutně nepřípustné.

Proto je nutná existence určitého převodu mezi názvy odpovídajícími standardu RFC 822 a názvy hostitelů protokolu UUCP. Způsob, jakým je tento převod prováděn, zcela závisí na typu implementace. Obecný způsob, jak namapovat názvy FQDN na názvy protokolu UUCP, spočívá v použití mapování přímo v souboru cest:

```
moria.orcnet.org   ernie!bert!moria!%s
```

Tato formule vytvoří z adresy, která udává plně kvalifikovaný název domény, čistokrevnou vykřičníkovou notaci pro použití v protokolu UUCP. Některé mailery k tomuto účelu poskytují speciální soubory; například program **sendmail** používá soubor `uucpxtable`.

Někdy se při posílání pošty ze sítě na bázi protokolu UUCP do Internetu vyžaduje zpětná transformace (hovorově zvaná *doménizace*). Pokud odesílatel pošty použije jako cílovou adresu plně kvalifikované doménové jméno, lze se tomuto problému vyhnout tak, že se při doručování zprávy chytřejšímu hostiteli neodstraní název domény z adresy na obálce. Nicméně stále ještě existují UUCP systémy, které nejsou součástí žádné domény. Jejich doménizace se typicky provede jejich přiřazením do fiktivní domény **uucp**.

Databáze cest poskytuje v sítích na bázi protokolu UUCP hlavní informace o směrování. Typická položka vypadá asi takto (název systému je od cesty oddělen tabulátory):

```
moria.orcnet.org  ernie!bert!moria!%s
moria             ernie!bert!moria!%s
```

Tento záznam uvádí, že každá zpráva určená pro hostitele **moria** bude doručena přes hostitele **ernie** a **bert**. Je zde zadán jak plně kvalifikovaný název, tak i název pro protokol UUCP. Oba názvy jsou zde uvedeny pro případ, že by mailer neměl samostatný způsob pro mapování mezi těmito dvěma jmennými prostory.

Pokud chcete směrovat všechny zprávy určené pro hostitele v rámci nějaké domény na jejich poštovní brány, můžete v databázi cest zadat také cestu, která bude mít jako cíl název domény, před nímž bude uvedena tečka. Pokud mohou být například hostitelé v doméně **sub.org** dosažitelní přes poštovní bránu **swim!smurf**, mohla by položka v databázi cest vypadat následovně:

```
.sub.org       swim!smurf!%s
```

Vytvoření souboru cest je přijatelné pouze v případě, že provozujete systém, který neobsahuje příliš mnoho informací o směrování. Provádíte-li směrování pro velké množství hostitelů, bude lepší k tomuto účelu použít příkaz **pathalias**, který umí vytvořit soubor cest ze souborů map. Mapy lze spravovat daleko snáze, protože konkrétní systém lze přidat nebo odstranit tak, že v mapě upravíte jeho položku a necháte znovu vytvořit soubor map. I když se mapy zveřejňované usenetovým projektem mapování názvů už ke směrování moc nepoužívají, mohou menší sítě na bázi protokolu UUCP poskytovat informace o směrování pomocí svých vlastních skupin map.

Soubor map se skládá především ze seznamu systémů, ve kterém má každý systém uveden seznam systémů, s nimiž se může spojit nebo které se mohou spojit s ním. Název systému začíná ve sloupci jedna a za ním následuje seznam jednotlivých spojení, která jsou vzájemně oddělena čárkami. Seznam může pokračovat i na dalších řádcích v případě, že následující řádek začíná znakem tabulátor. Každý záznam o spojení je tvořen názvem systému, za kterým je v hranatých závorkách uvedena jeho režie. Režie představuje aritmetický výraz složený z čísel a symbolických výrazů jako DAILY nebo WEEKLY. Řádky začínající znakem # jsou ignorovány.

Jako příklad uvažujme hostitele **moria**, který se spojuje dvakrát denně s hostitelem **swim.twobirds.com** a jedenkrát týdně s hostitelem **bert.sesame.com**. Pro spojení s hostitelem **bert** používá pouze pomalý modem s rychlostí 2400 b/s. Hostitel **moria** by měl v souboru map zveřejnit následující položku:

```
moria.orcnet.org
        bert.sesame.com(DAILY/2),
        swim.twobirds.com(WEEKLY+LOW)
moria.orcnet.org = moria
```

Poslední řádek říká, že hostitel **moria** může být rozpoznán i na základě svého názvu pro protokol UUCP. Všimněte si, že musí být uvedena režie *DAILY/2*, protože volání dvakrát denně ve skutečnosti snižuje režii tohoto spojení na polovinu.

Při použití takovýchto souborů map je schopen příkaz **pathalias** vypočítat optimální směrování na libovolný cílový systém, který je uveden v souboru cest. Dále je schopen z těchto souborů vytvořit databázi cest, která může být poté používána pro směrování do těchto systémů.

Příkaz **pathalias** poskytuje množství dalších vlastností, například skrytí systému (tedy to, že systémy mohou být přístupné pouze pomocí brány). Podrobnosti i úplný seznam příkazů pro ohodnocování spojů najdete na manuálových stránkách příkazu **pathalias**.

Komentáře v souboru map obvykle obsahují doplňkové informace o systémech, které jsou v tomto souboru popsány. Tyto komentáře musí odpovídat pevně stanovenému formátu, takže je lze získat z map. Například program **uuwho** používá k zobrazení těchto informací, které jsou mimochodem naformátovány moc hezky, databázi vytvořenou ze souborů map. Pokud svůj systém zaregistrujete u organizace, která distribuuje svým členům soubory map, budete obvykle muset v mapě vyplnit přibližně takovýto nějaký záznam. Následuje vzorová položka mapy (je to záznam popisující Olafův systém):

```
#N      monad, monad.swb.de, monad.swb.sub.org
#S      AT 486DX50; Linux 0.99
#O      private
#C      Olaf Kirch
#E      okir@monad.swb.de
#P      Kattreinstr. 38, D-64295 Darmstadt, FRG
#L      49 52 03 N / 08 38 40 E
#U      brewhq
#W      okir@monad.swb.de (Olaf Kirch); Sun Jul 25 16:59:32 MET DST
#
monad   brewhq(DAILY/2)
# Domains
monad = monad.swb.de
monad = monad.swb.sub.org
```

Za prvními dvěma znaky je vždy zapsán tabulátor. Význam většiny polí je zřejmý; detailní popis obdržíte od jakékoliv domény, u které se zaregistrujete. Největší zábavou je rozluštit význam pole *L*: to uvádí vaše geografické umístění ve formátu zeměpisná šířka/zeměpisná délka a používá se při vykreslování postscriptových map, které ukazují všechny systémy v rámci dané země nebo všechny systémy na světě[134].

Konfigurace programu elm

Název **elm** znamená „electronic mail“ a tento program je tak jedním z nejvýstižněji pojmenovaných unixových nástrojů. Program **elm** poskytuje celoobrazovkové rozhraní s propracovanou nápovědou. Nebudeme se zde zabývat způsobem jeho použití, ale zdržíme se pouze u jeho konfiguračních voleb.

Teoreticky můžete provozovat program **elm** bez jakékoliv konfigurace a přitom bude vše pracovat správně – pokud budete mít štěstí. Existuje však několik voleb, které je potřeba nastavit, i když budou potřeba jen při určitých událostech.

Při startu si program **elm** přečte několik konfiguračních proměnných ze souboru `elm.rc`, který se nachází v adresáři `/etc/elm`. Potom se pokusí přečíst z vašeho domovského adresáře soubor `.elm/elmrc`. Tento soubor si obvykle nebudete vytvářet sami. Program ho vytvoří za vás, když z nabídky options vyberete volbu „Save new options“.

Sada voleb použitá v privátním souboru `elmrc` je také dostupná v globálním souboru `elm.rc`. Většina nastavení uvedená v soukromém souboru `elmrc` potlačí nastavení uvedená v globálním souboru.

134 Pravidleně jsou zveřejňovány na news.lists.ps–maps. Pozor – jsou OBROVSKÉ.

Globální nastavení programu elm

V globálním souboru `elm.rc` musíte nastavit volby, které se týkají názvu vašeho hostitele. Například ve společnosti Virtual Brewery by mohl soubor pro hostitele **vlager** obsahovat následující informace:

```
#
# Jméno hostitele
hostname = vlager
#
# Jméno domény
hostdomain = .vbrew.com
#
# Plně kvalifikované doménové jméno
hostfullname = vlager.vbrew.com
```

Tyto volby poskytují programu **elm** informace o názvu místního hostitele. I když se tyto informace používají jen zřídka, měli byste je mít nastaveny. Tyto informace mají význam pouze v případě, že je uvedete v globálním konfiguračním souboru; nacházejí-li se ve vašem soukromém souboru `elmrc`, budou ignorovány.

Mezinárodní znakové sady

V minulosti byly navrženy doplňky standardu definovaného v dokumentu RFC 822, které by umožnily podporu různých typů zpráv, například prostý text, binární soubory, postscriptové soubory a podobně. Tato skupina standardů se běžně označuje jako MIME (Multipurpose Internet Mail Extensions). Kromě jiného tyto standardy umožňují příjemci zjistit, zda byla při psaní zprávy použita jiná znaková sada než standardní znaková sada ASCII, například francouzské akcenty nebo německé přehlásky. V určitých mezích tyto standardy podporuje i program **elm**.

Znaková sada, kterou vnitřně používá operační systém Linux, se obvykle označuje jako ISO-8859-1, což je název standardu, který tato znaková sada splňuje. Tento standard je také známý pod označením Latin-1. Všechny zprávy používající znaky z této znakové sady by měly mít ve své hlavičce následující řádek:

```
Content-Type:  text/plain; charset=iso-8859-1
```

Přijímací systém by měl toto pole rozeznat a při zobrazování zprávy by měl provést příslušná opatření. Pro zprávy typu *text/plain* (prostý text) je implicitně nastaveno pole znakové sady *charset* na hodnotu *us-ascii*.

Aby bylo možné zobrazit zprávy napsané v jiné znakové sadě, než je znaková sada ASCII, musí program **elm** vědět, jak má tyto znaky zobrazit. Pokud program **elm** obdrží zprávu, ve které má pole *charset* jinou hodnotu než *us-ascii* (nebo jiného typu než je *text/plain*), pokusí se implicitně zobrazit zprávu pomocí příkazu **metamail**. Zprávy vyžadující pro zobrazení program **metamail** mají na obrazovce v přehledu zpráv zobrazeno v prvním sloupci písmeno **M.**

Protože je implicitní znakovou sadou operačního systému Linux znaková sada ISO-8859-1, není nutné pro zobrazení zpráv vytvořených pomocí této znakové sady používat program **metamail**. Je-li programu **elm** sděleno, že zobrazovací systém odpovídá standardu ISO-8859-1, pak se program **metamail** nepoužije, ale zpráva bude zobrazena přímo. To zajistíte nastavením následující volby v globálním souboru `elm.rc`:

```
displaycharset = iso-8859-1
```

Pamatujte, že tuto volbu byste měli nastavit i v případě, že neplánujete posílání nebo přijímání zpráv, které obsahují jiné znaky než definuje standard ASCII. Tato volba se uvádí proto, že lidé posílající takovýto typ zpráv obvykle nakonfigurují svůj mailer tak, aby vložil do hlavičky pošty patřičné pole Content-Type:, a to bez ohledu na skutečnost, zda je či není daná zpráva napsána ve znakové sadě ASCII.

Ale pouze nastavení této volby v souboru `elm.rc` nestačí. Problém spočívá v tom, že při zobrazování zprávy pomocí vestavěného prohlížeče zavolá program **elm** pro každý zobrazovaný znak příslušnou funkci knihovny, aby zjistil, zda daný znak je či není zobrazitelný. Implicitně bude tato funkce považovat za zobrazitelné pouze znaky splňující standard ASCII a všechny ostatní znaky zobrazí jako `^?`. Tento problém lze vyřešit tak, že nastavíme proměnnou prostředí LC_CTYPE na hodnotu ISO-8859-1. Tato proměnná sdělí knihovně, aby považovala za zobrazitelné všechny znaky ze znakové sady Latin-1. Podpora této i dalších vlastností je dostupná od verze knihovny 4.5.8.

Při posílání zpráv, které obsahují speciální znaky ze znakové sady ISO-8859-1, byste se měli ujistit, že jste v souboru `elm.rc` nastavili ještě další dvě proměnné:

```
charset = iso-8859-1
textencoding = 8bit
```

Tyto volby způsobí, že program **elm** uvede v hlavičce pošty, že příslušná zpráva byla vytvořena pomocí znakové sady ISO-8859-1 a že bude poslána ve formě 8bitových hodnot (implicitně se veškeré znaky překládají na 7bitové hodnoty).

Samozřejmě, že jakoukoliv z těchto voleb lze místo v globálním konfiguračním souboru uvést v soukromém souboru `elmrc` takže uživatelé mohou používat odlišné volby v případě, že jim globální nastavení nevyhovuje.

Program sendmail

Říká se, že *skutečným* správcem Unixu nebudete, dokud neupravíte soubor `sendmail.cf`. Ale také se říká, že musíte být blázen, když to chcete zkoušet i podruhé.

Program **sendmail** je neuvěřitelně výkonný nástroj. Většina lidí ho chápe jen velmi obtížně a ještě hůře se ho učí. Jakýkoliv program, jehož kompletní manuál (manuál k programu **sendmail** Bryana Costalese a Erica Allmana vydalo nakladatelství O'Reilly and Associates) má 1050 stran, zcela pochopitelně vyděsí většinu lidí. Na konci této knihy uvádíme odkazy na dokumentaci k programu **sendmail**.

Nové verze programu **sendmail** jsou naštěstí odlišné. Odstraňují nutnost úprav záhadného souboru `sendmail.cf` a obsahují nástroje, které jej vytvoří automaticky z daleko jednodušších makrosouborů. Nemusíte proto znát celou syntaxi souboru `sendmail.cf`, protože v makrosouborech ji nepotřebujete. Namísto toho pouze uvedete názvy položek, například názvy funkcí, které má vaše konfigurace podporovat, a uvedete nějaké parametry, jak se má funkce chovat. Tradiční unixový nástroj **m4** pak vezme vámi vytvořené makrosoubory, zkombinuje je se šablonami definujícími syntaxi souboru `sendmail.cf` a vytvoří tento soubor automaticky.

V této kapitole se s programem **sendmail** seznámíme, ukážeme si jak jej nainstalovat, nakonfigurovat a otestovat, přičemž jako příklad použijeme síť virtuálního pivovaru. Věříme, že vám zde uvedené příklady pomohou pochopit konfiguraci programu **sendmail** a že na jejich základě budete schopni vytvářet i podstatně složitější konfigurace.

Instalace programu sendmail

Transportní agent **sendmail** bývá obvykle součástí většiny linuxových distribucí. V takovém případě je instalace velmi jednoduchá. I přesto ale bývá rozumné instalovat **sendmail** přímo ze zdrojových kódů, zejména pokud vám záleží na bezpečnosti. Program **sendmail** je velmi složitý a v průběhu let získal pověst programu s řadou bezpečnostních slabin. Jedním z nejznámějších příkladů je červ RTM, který využíval chyby přetečení bufferu ve starších verzích programu **sendmail**. Stručně jsme o něm hovořili v kapitole 9. Většina takovýchto ohrožení staví na tom, že kopie programu **sendmail** na různých systémech jsou identické a ukládají data vždy na určitá místa. Přesně k tomu dochází, pokud **sendmail** nainstalujete přímo z distribuce. Pokud si jej sami přeložíte, riziko se zmenšuje. Novější verze programu **sendmail** jsou podstatně odolnější, protože s nárůstem významu bezpečnosti prodělaly velmi přísné testování.

Zdrojový kód programu **sendmail** je k dispozici na anonymním FTP na adrese *ftp.sendmail.org*.

Překlad je velice jednoduchý, protože zdrojový balík přímo podporuje Linux. **sendmail** přeložíte následujícím postupem:

```
# cd /usr/local/src
# tar xvfz sendmail.8.9.3.tar.gz
# cd src
# ./Build
```

K instalaci vzniklých binárních souborů (kterou provedete následujícími příkazy) potřebujete práva superuživatele:

```
# cd obj.Linux.2.0.36.i586
# make install
```

Tím nainstalujete program **sendmail** do adresáře `/usr/sbin`. Kromě toho se v adresáři `/usr/bin` vytvoří několik symbolických odkazů. Zmíníme se o nich, až se budeme zabývat některými příklady spouštění programu **sendmail**.

Konfigurační soubory – přehled

Tradiční program **sendmail** se nastavuje pomocí systémového konfiguračního souboru (obvykle je to soubor `/etc/mail/sendmail.cf` nebo na starších distribucích `/etc/sendmail.cf` nebo dokonce `/usr/lib/sendmail.cf`), který se nepodobá žádnému z jazyků, s nimiž jste se mohli až doposud setkat. Editace souboru `sendmail.cf` tak, aby se program choval požadovaným způsobem, může být náročnou zkušeností.

V současné době jsou všechna konfigurační nastavení řízena pomocí maker, která používají snadno pochopitelnou syntaxi. Konfigurace generovaná pomocí maker pokrývá většinu potřeb, stále však ještě můžete vzniklý soubor `sendmail.cf` upravit ručně tak, aby pracoval i ve velmi složitých prostředích.

Soubory sendmail.cf a sendmail.mc

Makroprocesor **m4** vytvoří soubor `sendmail.cf` z makrosouboru, který vytváří správce systému. Ve zbytku textu budeme tento konfigurační soubor označovat jako `sendmail.mc`.

Proces konfigurace v zásadě spočívá ve vytvoření správného souboru `sendmail.mc`, který obsahuje makra popisující požadovanou konfiguraci. Makra představují výrazy, kterým rozumí procesor **m4**, který je pak převede do složité syntaxe souboru `sendmail.cf`. Makro se skládá z názvu makra (text psaný velkými písmeny na začátku), který může být propojen s nějakou funkcí nějakého programovacího jazyka, a z nějakých parametrů (textu v závorkách), které se použijí při rozkladu makra. Tyto parametry se mohou buď přímo promítnout do souboru `sendmail.cf`, nebo mohou ovlivnit způsob, jakým bude makro zpracováno.

Soubor `sendmail.mc` pro minimální konfiguraci (UUCP nebo SMTP s předáváním veškeré pošty jedinému přímo připojenému hostiteli) bude dlouhý 10 až 15 řádků, nepočítaje v to komentáře.

Dva příklady souboru sendmail.mc

Pokud spravujete více poštovních systémů, nemusíte konfigurační soubor pojmenovat `sendmail.mc`. Namísto toho se běžně pojmenovává názvem hostitele, pro nějž je určen – v našem případě to bude `vstout.m4`. Na názvu příliš nezáleží, hlavně že se výstup bude jmenovat `sendmail.cf`. Pojmenovávání jednotlivých souborů názvy hostitelů umožňuje ukládat je všechny v jednom adresáři, což je otázka pohodlí administrátora. Podívejme se na dva příklady konfiguračních souborů, čistě abychom viděli, o co jde.

Většina konfigurací programu **sendmail** dnes používá čistě SMTP. Konfigurace **sendmailu** pro podporu SMTP je velice jednoduchá. Konfigurace v příkladu 18.1 předpokládá existenci DNS serveru, který zajistí překlad názvů na adresy, a veškerou poštu pro všechny systémy bude doručovat čistě protokolem SMTP.

Příklad 18.1 – Konfigurační soubor vstout.smtp.m4

```
divert(-1)
#
# Příklad konfigurace pro vstout - jen smtp
#
divert(0)
VERSIONID('@(#)sendmail.mc    8.7 (Linux) 3/5/96')
OSTYPE('linux')
#
# Podpora lokálního a smtp doručování.
MAILER('local')
MAILER('smtp')
#
FEATURE(rbl)
FEATURE(access_db)
# end
```

Soubor `sendmail.mc` pro počítač vstout ve virtuálním pivovaru vidíme na příkladu 18.2. Pro komunikaci se všemi hostiteli na síti pivovaru používáme protokol SMTP a můžete si všimnout podobnosti s právě uvedeným příkladem. Kromě toho se veškerá pošta pro ostatní počítače na Internetu odesílá protokolem UUCP přes počítač moria.

Příklad 18.2 – Konfigurační soubor vstout.uucpsmtp.m4

```
divert(-1)
#
# Příklad konfigurace pro vstout
#
divert(0)
VERSIONID('@(#)sendmail.mc 8.7 (Linux) 3/5/96')
OSTYPE('linux')
dnl
# moria je předávací hostitel, používáme přenos "uucp-new".
define('SMART_HOST', 'uucp-new:moria')
dnl
# Podpora lokálního, smtp a uucp doručování.
MAILER('local')
MAILER('smtp')
MAILER('uucp')
LOCAL_NET_CONFIG
# Toto pravidlo zajišťuje, že všechna lokální pošta se posílá přes
# smtp, ostatní pošta jde přes předávacího hostitele.
R$* < @ $* .$m. > $*   $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
dnl
#
FEATURE(rbl)
FEATURE(access_db)
# end
```

Pokud si obě konfigurace porovnáte, pravděpodobně se vám podaří odhadnout, co jednotlivé konfigurační parametry dělají. Postupně si je všechny podrobně popíšeme.

Typicky používané parametry souboru sendmail.mc

Několik položek souboru `sendmail.mc` je zapotřebí vždycky, jiné je možné vynechat v případě, že vám budou vyhovovat implicitní nastavení. Obecné konfigurační příkazy souboru `sendmail.mc` jsou:

1. VERSIONID
2. OSTYPE
3. DOMAIN
4. FEATURE
5. Definice lokálních maker
6. MAILER
7. Pravidla LOCAL_*

V dalším textu budeme o jednotlivých parametrech hovořit a jejich použití si vysvětlíme na příkladech 18.1 a 18.2.

Komentáře

Řádky souboru `sendmail.mc`, které začínají znakem #, program **m4** nezpracovává a standardně je přímo přepisuje do výsledného souboru `sendmail.cf`. Je to užitečné, abyste okomentovali jednotlivá nastavení jak ve vstupním, tak i ve výstupním souboru.

Pokud chcete v souboru `sendmail.mc` použít komentáře, které se *nemají* přepsat do výstupního souboru, můžete použít tokeny *divert* a *dnl* programu **m4**. Nastavení *divert(-1)* potlačí veškerý výstup, *divert(0)* obnoví původní chování výstupu. Jakýkoliv výstup generovaný mezi těmito dvěma řádky bude potlačen. V našich příkladech tato nastavení používáme k vytvoření komentářů, které zůstanou pouze v souboru `sendmail.mc`. Pokud bychom chtěli dosáhnout stejného efektu na jediném řádku, můžeme použít token *dnl*, který doslova znamená „smaž všechny znaky až po začátek dalšího řádku a přidej na výstup znak nového řádku“. I tento token v našich příkladech používáme.

Jedná se o standardní příkazy programu **m4** a můžete se o nich dozvědět více na manuálových stránkách.

VERSIONID a OSTYPE

```
VERSIONID('@(#)sendmail.mc 8.9 (Linux) 01/10/98')
```

Makro `VERSIONID` je nepovinné a často se používá k zaznamenání verze konfigurace do souboru `sendmail.cf`. Proto jej zpravidla v příkladech najdete a i my vám doporučujeme je používat. V každém případě však nezapomeňte na makro

```
OSTYPE('linux')
```

Jedná se o pravděpodobně nejdůležitější definici. Makro `OSTYPE` způsobí přidání konfiguračních nastavení, která představují správné standardní hodnoty pro daný cílový systém. Většina definicí v makru `OSTYPE` zahrnuje nastavení cest k různým konfiguračním souborům, cesty k poštovním programům a jejich parametry, a umístění adresářů, v nichž **sendmail** ukládá zprávy. Standardní distribuce programu **sendmail** obsahuje nastavení potřebných hodnot i pro Linux a tato nastave-

ní zahrne právě uvedené makro. Některé distribuce Linuxu, zejména Debian, obsahují vlastní definiční soubor, který je plně kompatibilní s Linux-FHS. Pokud vaše distribuce takovýto soubor obsahuje, měli byste jej asi použít namísto standardního nastavení.

Definice OSTYPE by měla být uvedena hned z počátku konfiguračního souboru, protože na ní závisí řada dalších definic.

DOMAIN

Makro DOMAIN je užitečné v případě, že chcete na stejné síti nakonfigurovat větší počet počítačů stejným způsobem. Pokud nastavujete jen několik počítačů, pravděpodobně jeho použití nestojí za to. Typicky se tímto makrem konfigurují nastavení platná pro všechny počítače v síti, například název předávacího hostitele a podobně.

Standardní instalace obsahuje adresář šablon programu **m4**, které slouží k řízení konfiguračního procesu. Tento adresář se obvykle jmenuje `/usr/share/sendmail.cf` nebo nějak podobně. Naleznete zde podadresář `domain`, který obsahuje konfigurační šablony platné pro celou doménu. Abyste mohli použít makro DOMAIN, musíte si vytvořit vlastní makrosoubor obsahující standardní nastavení platná pro vaše systémy a uložit jej do podadresáře `domain`. Typicky se zde uvádějí definice specifické pro vaši doménu, například definice předávacího hostitele nebo poštovního uzlu, můžete však nastavit i další hodnoty.

Distribuce programu **sendmail** obsahuje celou řadu doménových konfiguračních souborů, pomocí nichž si můžete vytvořit svůj vlastní.

Pokud si vlastní doménový soubor uložíte jako `/usr/share/sendmail.cf/domain/vbrew.m4`, můžete jej pak v souboru `sendmail.mc` použít takto:

```
DOMAIN('vbrew')
```

FEATURE

Makro FEATURE umožňuje zahrnout do konfigurace předdefinované funkce programu **sendmail**. Tyto funkce umožňují velmi jednoduchou konfiguraci. Je jich velké množství a v této kapitole se zmíníme jen o některých nejdůležitějších. Podrobný popis všech předdefinovaných funkcí naleznete v `CF` souboru dodávaném s programem **sendmail**.

K zahrnutí kterékoliv funkce stačí v souboru `sendmail.mc` uvést takovýto řádek:

```
FEATURE(název)
```

kde jako *`název`* zapíšete název požadované funkce. Některé funkce mají nepovinné parametry. Pokud byste chtěli použít jiné než standardní nastavení, zadáte položku takto:

```
FEATURE(název, parametr)
```

kde *`parametr`* je hodnota požadovaného parametru.

Definice lokálních maker

Standardní konfigurační makrosoubory programu **sendmail** obsahují řadu propojení a proměnných, které lze použít k úpravě konfigurace. Obecně se označují jako *definice lokálních maker*. Řada z nich je uvedena v souboru `CF` v distribuci **sendmailu**. Definice lokálních maker se obvykle používají uvedením názvu makra s parametrem udávajícím hodnotu předávanou proměnné, jíž makro udržuje. V dalších příkladech v této kapitole si popíšeme některá základní lokální makra.

Definice transportních protokolů pošty

Pokud chcete, aby **sendmail** doručoval poštu i jinak než jenom lokálně, musíte mu říct, jaký způsob přenosu má použít. Velmi to zjednodušuje makro `MAILER`. Současné verze programu **sendmail** podporují řadu transportních protokolů, některé jsou experimentální, jiné se používají jen zřídka.

V naší síti potřebujeme přenos protokolem SMTP pro příjem a odesílání pošty mezi počítači v lokální síti a přenos protokolem UUCP pro příjem a odesílání pošty na předávacího hostitele. Dosáhneme toho jednoduše tak, že nadefinujeme přenosové protokoly `smtp` a `uucp`. Přenos typu `local` je implicitně zapnut, pro větší názornost jej však můžete zapnout i explicitně. Pokud používáte protokoly `smtp` i `uucp`, je nutné protokol `smtp` uvést jako první. Následující seznam představuje některé běžně používané přenosové protokoly:

`local` Tento transportní protokol zahrnuje jednak lokálního agenta pro doručování zpráv do schránek uživatelů místního systému a jednak doručovač prog používaný pro odesílání zpráv lokálním programům. Tento protokol se aktivuje standardně.

`smtp` Tento protokol implementuje přenos protokolem SMTP (Simple Mail Transport Protocol), což je nejběžnější poštovní přenosový protokol v Internetu. Při jeho aktivaci se zapínají čtyři doručovací systémy: `smtp` (základní SMTP), `esmtp` (rozšířené SMTP), `smtp8` (osmibitové binární SMTP) a `relay` (pro předávání zpráv mezi poštovními branami).

`uucp` Přenos protokolem UUCP zahrnuje dva programy: `uucp-old` pro tradiční UUCP přenosy a `uucp-new`, který umožňuje přenos zpráv více adresátům v jedné zásilce.

`usenet` Tento systém umožňuje přímé odesílání zpráv do usenetových news. Lokální zprávy určené na adresu *news.group.usenet* budou předány do sítě news a skupiny *news.group*.

`fax` Pokud máte nainstalován program HylaFAX, můžete tímto transportním systémem směrovat poštu na něj, takže získáte vestavěnou faxovou bránu. Tato funkce je v experimentálním stádiu a další informace o ní můžete zjistit na adrese *http://www.vix.com/hylafax*.

Existují i další transportní systémy, například pop, procmail, mail11, phquery a cyrus, které jsou sice užitečné, ale méně používané. Pokud vás zajímají, můžete se o nich dozvědět více buď v již zmíněné knize věnované sendmailu, nebo přímo v dokumentaci k tomuto programu.

Konfigurace směrování pošty pro lokální hostitele

Konfigurace pošty ve virtuálním pivovaru je pravděpodobně složitější, než tomu bude ve většině reálných aplikací. Většina dnešních sítí používá pouze protokol SMTP a vůbec se nezabývá UUCP přenosy. V našem příkladu jsme nastavili „chytrého hostitele“, kterého jsme použili pro obsluhu veškeré odchozí pošty. Protože ale na lokální síti používáme SMTP, musíme programu **sendmail** říct, aby poštu pro lokální systémy neposílal přes poštovní bránu. Makro `LOCAL_NET_CONFIG` umožňuje vložit přímo do souboru `sendmail.cf` pravidla, která definují způsob obsluhy lokální pošty. O přepisovacích pravidlech budeme podrobněji hovořit později, pro tuto chvíli nám stačí vědět, že jsme nadefinovali pravidlo, které nařizuje veškerou poštu pro doménu *vbrew.com* doručovat přímo na cílové hostitele protokolem SMTP.

Vygenerování souboru sendmail.cf

Po dokončení úprav konfiguračních souborů pro program **m4** je nyní musíte zpracovat a vytvořit z nich soubor `/etc/mail/sendmail.cf`, kterému rozumí **sendmail**. Tento krok je velmi jednoduchý, jak vidíte z následujícího příkladu:

```
# cd /etc/mail
# m4 /usr/share/sendmail.cf/m4/cf.m4 vstout.uucpsmtp.mc >sendmail.cf
```

Tento příkaz spustí makroprocesor **m4** a předá mu názvy dvou definičních makrosouborů, aby je v zadaném pořadí zpracoval. Prvním z nich je standardní makrošablona dodávaná přímo se **sendmailem**, druhým je samozřejmě námi vytvořený soubor. Výstup příkazu směrujeme do souboru `/etc/mail/sendmail.cf`, což je požadovaný cílový soubor.

Nyní můžete spustit **sendmail** s novou konfigurací.

Interpretace a vytváření přepisovacích pravidel

Bezesporu nejmocnější funkcí programu **sendmail** jsou přepisovací pravidla. Přepisovací pravidla používá **sendmail** ke zjištění, jak zpracovat přijaté zprávy. **sendmail** předává adresy z hlaviček zprávy přes soubory přepisovacích pravidel, takzvané *sady pravidel*. Přepisovací pravidlo transformuje poštovní adresu z jednoho tvaru do jiného a můžete si je představit jako příkaz editoru, který nahrazuje text vyhovující nějaké podmínce jiným textem.

Každé pravidlo má levou a pravou stranu, které jsou odděleny alespoň jedním tabulátorem. Když **sendmail** zpracovává poštu, prohlíží přepisovací pravidla a hledá shodu s levou stranou. Pokud adrese vyhovuje levá straně pravidla, nahradí se pravou stranou a zpracovává se znovu.

Příkazy R a S

V souboru `sendmail.cf` se sady pravidel definují příkazem `S`*n*, kde *n* nastavuje číslo aktuální sady pravidel.

Samotná pravidla se definují příkazem `R`. Každý příkaz `R` se po svém přečtení přidá do aktuální sady pravidel.

Pokud pracujete čistě se souborem `sendmail.mc`, nemusíte se o příkazy `S` starat, makra je vytvoří za vás. Ručně musíte vytvořit pouze `R` příkazy.

Sada pravidel programu **sendmail** tedy vypadá takto:

```
Sn
Rlhs rhs
Rlhs2 rhs2
```

Některá užitečná makra

sendmail interně používá některá standardní makra. V rámci přepisovacích pravidel jsou nejužitečnější tato:

$j Plně kvalifikované doménové jméno tohoto hostitele.

$w Hostitelská část plně kvalifikovaného doménového jména hostitele.

$m Doménová část plně kvalifikovaného doménového jména hostitele.

Tato makra můžeme použít v přepisovacích pravidlech. Konkrétně v pravidlech pro virtuální pivovar jsme použili makro `$m`.

Levá strana

Na levé straně přepisovacího pravidla definujete vzor, kterému má vyhovovat adresa, jíž chcete transformovat. Většina znaků se musí shodovat přesně, existují však i znaky se speciálním významem, které popisuje následující přehled. Na levé straně přepisovacích pravidel je možné použít následující symboly:

`$@` Vyhovuje právě žádný token.

`$*` Vyhovuje žádný nebo více tokenů.

`$+` Vyhovuje jeden nebo více tokenů.

`$-` Vyhovuje právě jeden token

`$=`*x* Vyhovuje jakákoliv fráze třídy *x*.

`$~`*x* Vyhovuje jakékoliv slovo mimo třídu *x*.

Token je řetězec znaků oddělených mezerou. Neexistuje žádná možnost, aby mezera byla součástí tokenu a není to ani nutné, protože výrazové šablony jsou natolik pružné, že umožňují tuto potřebu obejít. Když adresa pravidlu vyhovuje, text vyhovující jednotlivým částem šablony bude přiřazen do speciálních proměnných, které je možné použít na pravé straně pravidla. Jedinou výjimkou je výraz `$@`, kterému nevyhovuje žádný token, a proto nikdy negeneruje text, který by bylo možné použít na pravé straně pravidla.

Pravá strana

Když adresa vyhovuje levé straně pravidla, původní text se vymaže a nahradí se pravou stranou pravidla. Všechny tokeny na pravé straně se zkopírují přesně, výjimkou jsou ty začínající symbolem dolaru. Stejně jako na levé straně je možné i na pravé straně použít celou řadu metasymbolů. Popisuje je následující seznam. Pravá strana pravidel může obsahovat symboly:

`$`*n* Tento metasymbol se nahradí *n*tým výrazem z levé strany.

`$[`*název*`$]` Tento metasymbol převede název hostitele na kanonické jméno. Nahrazuje se kanonickým názvem zadaného názvu hostitele.

`$[`*mapa klíč* `$@`*parametry* `$:`*implicitní* `$]`

Obecnější tvar vyhledání. Výstupem je výsledek hledání *klíče* v *mapě* s použitím *parametrů*. *Mapa* může být jakákoliv mapa podporovaná **sendmailem**, například `virtusertable`, o které budeme hovořit za chvíli. Pokud vyhledání není úspěšné, výstup bude *implicitní*. Pokud vyhledání není úspěšné a není nastavena implicitní hodnota, vstup se nezmění a výstupem bude *klíč*.

`$>`*n* Tento symbol způsobí zpracování zbytku řádku pravidly *n*té sady. Výstup volané sady pravidel bude použit jako výstup tohoto pravidla. Tento mechanismus umožňuje pravidlům volat další sady pravidel.

`$#`*mailer* Tento metasymbol způsobí ukončení vyhodnocování sady pravidel a definuje mailer, který má být použit k doručení zprávy v dalším kroku. Tento metasymbol by měl být použit pouze v sadě *0* nebo v nějaké její subrutině. Jedná se o závěrečnou fázi zpracování adresy a měly by za ním být uvedeny následující dva metasymboly.

`$@`*`hostitel`* Tento metasymbol specifikuje hostitele, na nějž bude zpráva předána. Pokud je cílem místní hostitel, nemusí být tento symbol uveden. Hodnota *hostitel* může být dvojtečkami oddělený seznam cílových hostitelů, které budou v zapsaném pořadí vyzkoušeny k odeslání zprávy.

`&:`*`uživatel`* Tento metasymbol specifikuje cílového adresáta zprávy.

Pokud adresa pravidlu vyhovuje, za normálních okolností se vzniklý výstup znovu testuje proti stejnému pravidlu a v případě shody se transformace opakuje tak dlouho, dokud výstup pravidlu nevyhovuje. Teprve pak se provádí zpracování následujícím pravidlem v sadě. Toto chování je možné změnit tak, že na začátku pravé strany uvedeme jeden ze dvou speciálních metasymbolů pravé strany, které jsou popsány v následující tabulce. Metasymboly pro řízení smyček jsou:

`$@` Tento symbol způsobí, že zbytek pravé strany bude předán jako výstup celé sady pravidel. Nezpracovává se už žádné další pravidlo v sadě.

`$:` Tento metasymbol způsobí okamžité ukončení pravidla, vyhodotí se však následující pravidla v sadě.

Příklad jednoduchého pravidla

Abychom lépe pochopili jak fungují substituce v adresách, podívejme se nejprve na příklad levé strany pravidla:

```
$* < $+ >
```

Tomuto pravidlu vyhovuje „žádný nebo více tokenů následovaný symbolem <, následovaný jedním nebo více tokeny a ukončený znakem >“.

Pokud bychom pravidlo použili na adresy `brewer@vbrew.com` nebo `Head Brewer < >`, nebudou pravidlu vyhovovat. První adresa nevyhovuje, protože neobsahuje znak <, druhá nevyhovuje, protože symbol `$+` představuje *jeden nebo více* tokenů a mezi znaky < a > v adrese není žádný token. Když adresa pravidlu nevyhovuje, pravá strana pravidla se neuplatní.

Pokud bychom pravidlo použili na adresu `Head Brewer < brewer@vbrew.com >`, adresa vyhovuje a na pravé straně bude symbol `$1` obsahovat text `Head Brewer` a symbol `$2` bude obsahovat `brewer@vbrew.com`.

Pokud bychom pravidlo použili na adresu `< brewer@vbrew.com >`, bude vyhovovat rovněž, protože symbolu `$*` vyhovuje *žádný nebo více tokenů* a symbol `$1` na pravé straně bude prázdný řetězec.

Sémantika sad pravidel

Každá ze sad pravidel se volá k provedení jiné fáze zpracování pošty. Při vytváření pravidel je důležité chápat, co má která sada pravidel dělat. Podíváme se na jednotlivé sady pravidel, které nám konfigurační skripty programu **m4** umožňují měnit.

`LOCAL_RULE_3` Sada 3 zodpovídá za konverzi adresy z libovolného formátu do formátu, který **sendmail** dále zpracovává. Výstupní formát sady má známý tvar *lokální_část@hostitel_a_doména*. Sada 3 by měla umístit hostitelskou část konvertované adresy mezi symboly < a >, čímž se usnadní zpracování dalšími sadami. Sada 3 se používá jako první v sekvenci zpracování adres, takže pokud **sendmail** používáte k předávání pošty pro nějaký systém, který

používá atypický formát adres, měli byste pomocí makra `LOCAL_RULE_3` doplnit pravidlo, které tento formát zkonvertuje do normálního formátu.

`LOCAL_RULE_0` a `LOCAL_NET_CONFIG`

Sada 0 se uplatní na adresu příjemce po zpracování adresy sadou 3. Makro `LOCAL_NET_CONFIG` způsobí vložení pravidel do *spodní poloviny* sady 0. Sada 0 zodpovídá za doručení zprávy příjemce, takže musí vygenerovat trojici mailer, hostitel a uživatel. Pravidla se uplatní před případnou definicí „chytrého" hostitele, takže pokud přidáte pravidla zajišťující správný převod adres, adresy vyhovující těmto pravidlům nebudou obslouženy tímto hostitelem. Tímto mechanismem jsme zajistili přímé doručení protokolem `smtp` pro hostitele v naší lokální síti.

`LOCAL_RULE_1` a `LOCAL_RULE_2`

Sada 1 se uplatní na všechny adresy odesílatele, sada 2 na všechny adresy příjemce. Typicky jsou obě prázdné.

Interpretace pravidla v našem příkladu

V příkladu 18.3 jsme použili makro `LOCAL_NET_CONFIG` k vytvoření pravidla, které zajistí, že jakákoliv pošta pro naši doménu bude doručena přímo prostřednictvím maileru **smtp**. Nyní víme, jak se pravidla vytvářejí, takže si můžeme vysvětlit, jak pravidlo funguje. Podívejme se na ně znovu.

Příklad 18.3 – Přepisovací pravidlo z vstout.uucpsmtp.m4

```
LOCAL_NET_CONFIG
# Toto pravidlo zajišťuje, že všechna lokální pošta se posílá přes
# smtp, ostatní pošta jde přes předávacího hostitele.
R$* < @ $* .$m. > $*  $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
```

Víme už, že makro `LOCAL_NET_CONFIG` způsobí přidání pravidla někam na konec sady 0, nicméně před definici předávacího hostitele. Víme také, že sada 0 je poslední prováděnou sadou a že jejím výsledkem by měla být trojice údajů definující mailer, uživatele a hostitele.

Můžeme samozřejmě ignorovat oba komentářové řádky, ty nemají na nic vliv. Samotné pravidlo je řádek začínající znakem `R`. Víme už, že `R` je příkaz **sendmail**u a přidává pravidlo do aktuální sady pravidel, což je v našem případě sada 0. Podívejme se nejprve na levou stranu a pak na pravou stranu pravidla.

Levá strana vypadá takto: `$* < @ $* .$m. > $*`

Sada 0 předpokládá existenci znaků < a >, které doplní sada 3. Sada 3 konvertuje adresy do běžného formátu a aby usnadnila další zpracování, umístí hostitelskou část adresy mezi znaky < a >.

Tomuto pravidlu bude vyhovovat jakákoliv adresa v formátu typu `CílovýUživatel < @ nějakýhostitel.našedoména. > nějaký text`. Vyhovují mu tedy zprávy pro jakéhokoliv uživatele v naší doméně.

Připomeňme si, že text vyhovující metasymbolům na levé straně pravidla je přiřazen metasymbolům pro použití na pravé straně. V našem případě bude prvnímu symbolu `$*` vyhovovat všechno po znak <. Veškerý tento text bude přiřazen symbolu `$1` na pravé straně. Další část textu (mezi < a >) vyhovuje druhému symbolu `$*` a na pravé straně bude uložena v `$2`. Podobně poslední část adresy bude uložena v `$3`.

Levá strana pravidla by měla být jasná. Tomuto pravidlu vyhovuje pošta pro jakéhokoliv uživatele na jakémkoliv počítači v naší doméně. Uživatel bude uložen v `$1`, hostitel v `$2` a případný další text v `$3`. Pak se volá pravá strana pravidla, která adresu zpracuje.

Nejprve se podívejme na to, co bychom měli na výstupu dostat. Pravá strana pravidla vypadá takto: `$#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3`.

Při zpracování pravé strany se interpretují jednotlivé metasymboly a provádějí se příslušné substituce.

Symbol `$#` definuje mailer, v našem případě *smtp*.

Symbol `$@` definuje cílového hostitele. V našem případě je tento hostitel definován jako `$2.$m.`, což je plně kvalifikované doménové jméno hostitele v naší doméně. Toto FQDN vytvoříme z hostitelské části jména přiřazené symbolu `$2`, ke které připojíme doménové jméno (`$m`).

Symbol `$:` definuje uživatele, kterého rovněž známe z levé strany a máme jej zapsaného v symbolu `$1`. Nakonec zachováme text části <> a případný ukončovací text, přičemž použijeme hodnoty zjištěné v levé části pravidla.

Protože výsledkem pravidla je mailer, předá se mu zpráva k doručení. V našem případě bude zpráva předána cílovému hostiteli k doručení protokolem SMTP.

Konfigurace nastavení programu sendmail

Program **sendmail** má řadu voleb, které umožní upravit provádění různých operací. Je jich velké množství, proto se v následujícím seznamu omezíme pouze na některé nejčastěji používané.

Jednotlivé volby můžete nastavit buď v souboru **m4** (což je vhodnější postup), nebo je můžete uvést přímo v souboru `sendmail.cf`. Pokud budeme například chtít, aby **sendmail** pro každou zprávu spustil novou úlohu, můžeme do konfiguračního souboru `sendmail.mc` přidat následující řádek:

```
define('confSEPARATE_PROC','true')
```

V souboru `sendmail.cf` se vytvoří odpovídající řádek

```
O ForkEachJob=true
```

Následující seznam představuje běžné volby v souboru `sendmail.mc` (a jejich ekvivalenty v souboru `sendmail.cf`).

`confMIN_FREE_BLOCKS`

`(MinFreeBlocks)` Mohou nastat situace, kdy nějaké problémy znemožní okamžité doručení zprávy a ta pak musí být uložena v poštovní frontě. Pokud váš hostitel zpracovává velké objemy pošty, může se fronta rozrůst natolik, že zaplní celý souborový systém, na němž je uložena. Aby se tomu zabránilo, umožňuje **sendmail** nastavit tímto parametrem minimální počet volných diskových bloků, které musí být k dispozici před přijetím zprávy. Tím zajistíte, že **sendmail** nikdy nezaplní celý diskový prostor. (Implicitní hodnota je 100.)

`confME_TOO (MeToo)` Když se expanduje cílová adresa, například při překladu aliasu, může se stát, že odesílatel se objeví také jako adresát zprávy. Tato volba řídí, zda bude autorovi zprávy poslána její kopie v případě, že se objeví na se-

znamu příjemců. Platné hodnoty jsou "true" a "false". (Implicitní hodnota je false.)

`confMAX_DAEMON_CHILDREN (MaxDaemonChildren)`

Kdykoliv **sendmail** přijme SMTP spojení ze vzdáleného hostitele, vytvoří svou kopii, která příchozí zprávu obslouží. Díky tomu je **sendmail** schopen současně obsluhovat více příchozích spojení. Je to sice užitečné chování, nicméně každá nová kopie zabírá paměť hostitelského systému. Pokud by vzniklo velké množství současných spojení (například zotavení po výpadku nebo kvůli útoku na systém), mohly by démony **sendmail** zabrat veškerou dostupnou paměť. Tato volba umožní nastavit, kolik démonů může být současně spuštěno. Po dosažení limitu budou další spojení odmítána, dokud neskončí nějaké předchozí spojení. (Implicitní hodnota není definována.)

`confSEPARATE_PROC (ForkEachJob)`

Při zpracovávání fronty a odesílání zpráv zpracovává **sendmail** jednu zprávu po druhé. Při zapnutí této volby bude **sendmail** vytvářet svou kopii pro každou doručovanou zprávu. Toto chování je užitečné zejména v případě, že ve frontě se nachází větší počet zpráv například kvůli problémům cílového systému. (Implicitní hodnota je false.)

`confSMTP_LOGIN_MSG (SmtpGreetingMessage)`

Kdykoliv **sendmail** přijme připojení, vypíše uvítací hlášení. Implicitně toto hlášení obsahuje název hostitele, název přenosového agenta, verzi sendmailu, verzi lokální konfigurace a datum. Standard RFC 821 říká, že první část uvítacího hlášení musí být plně kvalifikované doménové jméno hostitele, zbytek může být libovolný. Můžete použít makra programu **sendmail**, která budou při použití expandována. Jediný kdo toto hlášení uvidí bude zvědavý správce systému, který se snaží diagnostikovat problémy poštovního systému, nebo příliš zvědaví uživatelé, které zajímá, jak váš poštovní systém funguje. Jejich námahu můžete odměnit nějakou vtipnou poznámkou, nicméně buďte příjemní. **sendmail** automaticky za první část hlášení doplní slovo ESMTP, které vzdálenému hostiteli signalizuje, že nás systém podporuje protokol ESMTP. (Implicitní hodnota je `$j Sendmail $v/$Z; $b`).

Některá užitečná nastavení sendmailu

Možných konfigurací **sendmailu** je obrovské množství. V dalším textu si popíšeme jenom několik důležitých druhů konfigurace, které mohou být užitečné pro řadu různých instalací.

Uživatelé mohou nastavovat údaj From:

Občas je užitečné přepsat pole `From:` odchozí zprávy. Řekněme, že máte webový systém pro odesílání pošty. Normálně by odchozí pošta pocházela od uživatele, který vlastní proces webového serveru. Můžeme chtít adresu odesílatele přepsat, aby se pošta tvářila, že pochází od jiného uživatele či adresy našeho systému. **sendmail** umožňuje definovat, kteří uživatelé systému mají právo tuto hodnotu měnit.

Volba `use_ct_file` zapíná tuto možnost a aktivuje soubor se seznamem jmen, která mají právo měnit adresu odesílatele. Implicitně má toto právo jen malý počet uživatelů (například `root`). Implicitně se pro tyto uživatele používá soubor `/etc/mail/trusted-users` na systémech, které používají adresář `/etc/mail` nebo soubor `/etc/sendmail.ct` na systémech, které tento adresář nepoužívají. Název a umístění souboru je možné změnit parametrem `confCT_FILE`.

Funkci povolíte tím, že v souboru `sendmail.mc` uvedete `FEATURE(use_ct_file)`.

Poštovní aliasy

Poštovní aliasy jsou mocná funkce, která umožňuje přesměrování pošty do schránek s alternativními názvy uživatelů nebo procesů na cílovém hostiteli. Bývá například obvyklé, že odezva a komentáře týkající se webových stránek se odesílají uživateli *webmaster*. Velmi často na cílovém počítači neexistuje žádný uživatel *webmaster* a jedná se o pouhý alias jiného uživatele systému. Aliasy běžně využívají servery obsluhující poštovní konference, kdy alias směruje poštu serveru konference.

Aliasy se definují v souboru `/etc/aliases`. Při přijetí příchozí zprávy se **sendmail** v tomto souboru dívá, jak se zprávou naložit. Pokud v souboru nalezne záznam odpovídající adresátovi zprávy, přesměruje zprávu tak, jak mu to záznam nařizuje.

Pomocí aliasu je možné provést tři operace:

- Definují odkazy nebo známá jména adresátů, u nichž se pošta směruje na konkrétního uživatele nebo více uživatel.
- Dokáží spustit program a předat mu zprávu jako vstup.
- Mohou poslat zprávu do souboru.

Aby systém odpovídal RFC standardu, musí mít definovány aliasy Postmaster a MAILER-DAEMON.

Při vytváření aliasů, které spouštějí programy nebo zapisují do souborů, postupujte vždy velice opatrně, protože **sendmail** běží s právy superuživatele.

Podrobnosti týkající se poštovních aliasů najdete na manuálových stránkách `alias(5)`. Příklad souboru `aliases` vidíte na následujícím výpisu:

Příklad 18.4 – Soubor aliases

```
#
# Následující aliasy musí existovat podle RFC.
# Je nutné je směrovat někomu, kdo čte poštu pravidelně
#
postmaster:    root                              # povinný údaj
MAILER-DAEMON: postmaster                        # povinný údaj
#
#
# příklady běžných aliasů
#
usenet:        janet                             # alias osoby
admin:         joe,janet                         # alias pro více lidí
newspak-users: :include:/usr/lib/lists/newspak # čte adresáty ze souboru
changefeed:    |/usr/local/lib/gup              # alias spouštějící program
complaints:    /var/log/complaints               # alias zapisující do souboru
#
```

Kdykoliv upravíte soubor `aliases`, nezapomeňte spustit příkaz

```
# /usr/bin/newaliases
```

Tento příkaz vygeneruje nové tabulky aliasů, které **sendmail** interně používá. Příkaz `/usr/bin/newaliases` je symbolický odkaz přímo na **sendmail** a takto spuštěný volá **sendmail** následujícím způsobem:

```
# /usr/lib/sendmail -bi
```

Příkaz **newaliases** je jednodušší a pohodlnější způsob, jak toto volání provést.

Použití předávacího hostitele

Občas narazíte na zprávu, kterou nedokážete přímo poslat cílovému systému. Bývá pohodlné mít na síti jeden systém, který se stará o doručování pošty na obtížně dosažitelné cíle namísto řešení, kdy budou tyto operace provádět všechny systémy.

Existuje několik dobrých důvodů proč ke správě pošty používat jediného hostitele. Usnadníte si administraci tím, že budete mít pouze jediného hostitele s podrobnou konfigurací, který bude umět přenášet poštu na všechny typy systémů včetně UUCP, Usenet a podobně. Všechny ostatní počítače potřebují pouhý jeden poštovní protokol k poslání pošty na tento centrální systém. Hostitelé plnící úlohy takovýchto centrálních směrovacích a předávacích uzlů se označují jako *předávací hostitelé*. Pokud máte předávacího hostitele, který od vás přijímá poštu, můžete mu poslat cokoliv a nestaráte se o to, jak konkrétní zprávy doručit.

Další výhodné použití předávacího hostitele spočívá v řízení přenosu pošty přes firewall. Organizace může mít instalovánu privátní IP síť s neregistrovanými IP adresami. Tato síť může být k Internetu připojena přes firewall. Odeslání a příjem pošty protokolem SMTP u hostitelů v privátní síti není možné, protože tito hostitelé nemohou vytvářet a přijímat přímá spojení s Internetem. Můžeme však mít firewall, který bude zároveň plnit funkci předávacího hostitele[135]. Předávací hostitel na firewallu může navazovat přímá spojení s ostatními hostiteli v privátní síti a s hostiteli na Internetu. Tento hostitel tak může přijmout poštu od kohokoliv a zajistit její odeslání přímo na správného adresáta.

Předávací hostitel se obvykle použije pokud selžou všechny ostatní metody transportu. V případě organizace s privátní sítí je rozumné řešení nastavit počítače tak, aby se nejprve pokusily doručit poštu přímo, a pokud se jim to nepovede, aby použily předávací systém. Tím se předávacímu systému dost odlehčí, protože veškerou interní poštu si budou počítače na privátní síti posílat přímo.

sendmail umožňuje jednoduché nastavení předávacího hostitele volbou `SMART_HOST`, v konfiguraci pro virtuální pivovar jsme ji použili. Odpovídající část konfiguračního souboru vypadala takto:

```
define('SMART_HOST', 'uucp-new:moria')
LOCAL_NET_CONFIG
# Toto pravidlo zajišťuje, že všechna lokální pošta se posílá přes
# smtp, ostatní pošta jde přes předávacího hostitele.
R$* < @ $* .$m. > $*   $#smtp $@ $2.$m. $: $1 < @ $2.$m. > $3
```

135 Pozn. překladatele: Toto řešení je sice *možné*, ale je neskutečně *nevýhodné*. Firewall (má-li za něco stát) by měl být systém, na kterém neběží vůbec *nic*, který vůbec *nic* nedělá a jenom si třikrát rozmyslí, jestli přijatý paket taky někam pošle. Spustit na stroji který ma zajišťovat bezpečnost celé sítě něco tak nebezpečného jako je sendmail, to už je zbytečné firewall vůbec vytvářet.

Makro `SMART_HOST` umožňuje definovat hostitele pro předávání veškeré odchozí pošty, kterou nelze doručit přímo, a zároveň umožňuje definovat protokol, který se má pro komunikaci s tímto hostitelem použít.

V našem případě jsme použili přenos `uucp-new` přes UUCP hostitele *moria*. Pokud bychom chtěli **sendmail** nastavit pro použití SMTP předávacího hostitele, mohli bychom to udělat například takto:

```
define('SMART_HOST', 'mail.isp.net')
```

Nemusíme specifikovat typ přenosu, protože SMTP je implicitní.

Tušíte už, co může dělat makro `LOCAL_NET_CONFIG` a přepisovací pravidla?

Makro `LOCAL_NET_CONFIG` umožňuje doplnit přepisovací pravidla programu sendmail, která budou definovat poštu doručovanou přímo lokálním systémem. V našem příkladu jsme použili pravidlo, jemuž vyhovovaly všechny adresy naší domény (`.$m.`) a přepsali jsme adresu pro přímé doručení SMTP systémem. Tím zajistíme, že veškerá pošta pro hostitele v naší doméně bude předána SMTP maileru a odeslána cílovému hostiteli aniž bychom použili předávacího hostitele.

Manipulace s nevyžádanou poštou (spam)

Pokud se přihlásíte na nějakou poštovní konferenci, zveřejníte svou adresu na webových stránkách nebo pošlete článek na UseNet, pravděpodobně vám začne docházet nevyžádaná reklamní pošta. Je celkem běžné, že různá individua hledají na Internetu poštovní adresy a zařazují je do seznamů, které pak prodávají společnostem, které hledají různé metody inzerce svého zboží. Tento typ hromadného rozesílání pošty se běžně označuje jako spam.

Free On-line Dictionary of Computing[136] definuje spam (týkající se elektronické pošty) jako:

Nerozlišeně rozesílat velké objemy nevyžádané pošty k inzerci výrobku nebo služby. Spam v tomto smyslu představuje elektronický ekvivalent reklamních poštovních zásilek rozesílaných obyvatelům.

V 90. letech s nárůstem komerčního potenciálu sítě vznikají společnosti, které nabízejí spamming jako „službu“ firmám, které chtějí inzerovat na síti. Provádějí to odesíláním pošty na seznamy adres, usenetové news a poštovní konference. Tyto praktiky vedou spoustu uživatelů k rozhořčeným a agresivním akcím proti společnostem, které je provozují.

Naštěstí **sendmail** obsahuje určité mechanismy, které mohou usnadnit manipulaci s nevyžádanou poštou.

Real-time Blackhole List

Služba Real-time Blackhole List je veřejná služba, jejímž cílem je omezit objemy nevyžádané reklamní pošty. V databázi, kterou je možné z Internetu prohledávat, jsou umístěny adresy a systémy známé rozesíláním spamu. Databázi doplňují uživatelé, kteří nevyžádanou poštu z různých adres přijímají. Někdy se na tomto seznamu ocitnou i dobře známé domény, které zanedbají kontrolu spamových aktivit. I když občas někdo protestuje proti konkrétním záznamům v tomto seznamu, jedná se o velmi populární službu a případné stížnosti bývají řešeny velmi rychle. Podrobnosti o tom, jak služba funguje, je možné najít na domovských stránkách Mail Abuse Protection System na adrese *http://maps.vix.com/rbl/*.

136 Free On-Line Dictionary of Computing bývá součástí řady distribucí Linuxu a je k dispozici na adrese *http://wombat.doc.ic.ac.uk/foldoc/*.

Pokud v **sendmailu** zapnete příslušnou funkci, bude veškerou příchozí poštu testovat proti tomuto seznamu a rozhodne, zda zprávu přijmout. Pokud provozujete velký systém s řadou uživatelů, můžete tím ušetřit značný diskový prostor. Parametrem služby je název serveru, který má být dotazován. Implicitně se používá hlavní server na adrese *rbl.maps.vix.com*.

Službu Real-time Blackhole List nastavíte následujícím makrem v souboru `sendmail.mc`:

```
FEATURE(rbl)
```

Pokud budete chtít použít jiný RBL server, můžete zadat makro takto:

```
FEATURE(rbl,'rbl.host.net')
```

Přístupová databáze

Alternativní řešení, které nabízí větší míru kontroly za cenu větších konfiguračních nároků je funkce access_db programu **sendmail**. Přístupová databáze umožňuje nastavit, od kterých uživatelů nebo hostitelů budete přijímat poštu a pro které budete poštu předávat.

Správné nastavení hostitelů, pro něž předáváte poštu, je velmi důležité, protože se jedná o další techniku běžně používanou spammery k obejití systémů jako jsou Real-time Blackhole List. Namísto přímého odeslání pošty na váš systém ji spammer pošle na nějakého nic netušícího hostitele, který vám ji předá. Příchozí spojení pak nepochází od spammera ale od předávajícího hostitele. Aby nemohl být váš vlastní poštovní systém tímto způsobem zneužit, měli byste poštu předávat pouze pro známé systémy. Verze **sendmail** 8.9.0 a novější mají předávání pošty standardně vypnuto, takže pokud chcete předávání povolit, musíte to nastavit v přístupové databázi.

Základní myšlenka je jednoduchá. Když **sendmail** přijme příchozí spojení, zjistí si informace z hlavičky a pak se podívá do přístupové databáze, zda se má zprávou dále zabývat.

Přístupová databáze je soustava pravidel popisujících, co se má provést pro zprávy odeslané z uvedených hostitelů. Implicitně je přístupová databáze uložena v souboru `/etc/mail/access`. Tato tabulka má jednoduchý formát. Každý řádek obsahuje jedno pravidlo. Může to být úplná poštovní adresa, název hostitele nebo IP adresa. Za tím je uvedeno, co se má provést. Existuje pět typů možných akcí:

`OK`	Přijme zprávu.
`RELAY`	Přijme zprávu od tohoto hostitele nebo uživatele dokonce i v případě, že není určena pro místní systém, umožní tedy předání zprávy na další systém.
`REJECT`	Odmítne zprávu s obecným hlášením.
`DISCARD`	Zruší zprávu mailerem `$#discard`.
### text	Vrátí chybovou zprávu kde *###* je kód chyby (dle RFC 821) a *text* je chybové hlášení.

Příklad souboru `/etc/mail/access` může vypadat takto:

```
friends@cybermail.com   REJECT
aol.com                 REJECT
207.46.131.30           REJECT
postmaster@aol.com      OK
linux.org.au            RELAY
```

Tento příklad odmítá zprávu z adresy *friends@cybermail.com*, od jakéhokoliv systému v doméně *aol.com* a od systému *207.46.131.30*. Další pravidlo povoluje příjem zpráv z adresy *postmas-*

ter@aol.com i když zbytek domény je zakázán. Poslední pravidlo povoluje předávání pošty všem hostitelům v doméně *linux.org.au*.

Řízení přístupu databází zapnete následujícím příkazem v souboru `sendmail.mc`:

```
FEATURE(access_db)
```

Při implicitním nastavení se databáze vytváří příkazem `hash -o /etc/mail/access`, který z prostého textového souboru udělá jednoduchou hashovanou databázi. Ve většině případů to naprosto postačuje. Existují i další možnosti, které možná budete chtít použít v případě, že máte velikou přístupovou databázi. Podrobnosti naleznete v knize o **sendmailu** nebo přímo v jeho dokumentaci.

Zabránění uživatelům v příjmu pošty

Pokud máte uživatele nebo automatické procesy, které poštu odesílají ale nepotřebují ji přijímat, bývá užitečné pro takovéto adresáty zakázat příjem pošty. Tím se šetří diskový prostor od ukládání zpráv, které nikdo nebude číst. Funkce blacklist_recipients společně s funkcí access_db umožňuje zablokovat příjem pošty pro zvolené lokální uživatele.

Pokud chcete tuto funkci zapnout, musíte v souboru `sendmail.mc` uvést:

```
FEATURE(access_db)
FEATURE(blacklist_recipients)
```

Pokud chcete pro nějakého uživatele zakázat příjem pošty, uveďte jej v přístupové databázi. Obvykle použijete akci typu `###` s nějakým rozumným chybovým hlášením, aby odesílatel věděl, že pošta nebyla doručena. Tato funkce funguje i pro uživatele ve virtuálních poštovních doménách, v takovém případě uvedete kromě jména uživatele i název virtuální domény. Příklad souboru `/etc/mail/access` může vypadat takto:

```
daemon          550 Daemon does not accept or read mail.
flacco          550 Mail for this user has been administratively disabled.
grump@dairy.org 550 Mail disabled for this recipient.
```


Konfigurace virtuálních poštovních hostitelů

Virtuální poštovní hostitelé umožňují aktivovat funkci, kdy počítač přijímá a doručuje poštu pro řadu různých domén tak, jako kdyby se jednalo o řadu samostatných systémů. Typicky virtuální hostitele nabízejí poskytovatelé internetového připojení v kombinaci s virtuálními webovými servery. Tato funkce se nastavuje poměrně jednoduše a protože nikdy nevíte, kdy budete chtít svého hostitele použít jako virtuálního hostitele pro konferenci týkající se nějakého pěkného linuxového projektu, popíšeme si, jak to udělat.

Příjem pošty pro jiné domény

Když **sendmail** přijme zprávu, porovnává cílového hostitele v hlavičce zprávy s názvem lokálního systému. Pokud si odpovídají, **sendmail** zprávu přijme k lokálnímu doručení. Pokud si neodpovídají, **sendmail** se rozhodne, zda zprávu přijmout a předat ji uvedenému cílovému hostiteli (viz předchozí text *Přístupová databáze*).

Pokud chceme poskytovat virtuální poštovní hostitele, musíme **sendmailu** v první řadě říct, aby přijímal poštu i pro domény, jejichž jsme hostitelem. To se naštěstí provede velmi jednoduše.

Funkce use_cw_file umožňuje definovat název souboru v němž jsou uvedeny domény, pro něž má **sendmail** přijímat poštu. Funkci aktivujete následujícím příkazem v souboru `sendmail.mc`:

```
FEATURE(use_cw_file)
```

Soubor s hostěnými doménami se standardně jmenuje `/etc/mail/local-host-names` na systémech používajících adresář `/etc/mail` nebo `/etc/sendmail.cw` na ostatních. Název a umístění souboru můžete změnit makrem `confCW_FILE` takto:

```
define('confCW_FILE', '/etc/virtualnames')
```

Přidržíme-li se standardního souboru a budeme-li chtít obsluhovat poštu domén *bovine.net*, *dairy.org* a *artist.org*, vyvoříme soubor `/etc/mail/local-host-names`, který bude vypadat takto:

```
bovine.net
dairy.org
artist.org
```

Když to uděláme a za předpokladu, že pro dané domény existují DNS záznamy které směrují poštu na náš systém, bude **sendmail** přijímat poštu těchto domén stejně jako by byla určena přímo pro naši doménu.

Předávání pošty virtuálních domén na jejich adresáty

Funkce virtusertable konfiguruje podporu tabulek virtuálních uživatelů, které používáme k obsluze pošty virtuálních domén. Tabulka virtuálních uživatelů mapuje příchozí poštu určenou pro *user@host* na poštu pro *otheruser@otherbost*. Můžete to chápat jako trochu chytřejší alias, který se netýká jen uživatelského jména, ale i doménového jména.

Tabulky virtuálních uživatelů zapnete následujícím příkazem v souboru `sendmail.mc`:

```
FEATURE(virtusertable)
```

Implicitně se soubor s překladovou tabulkou nachází v `/etc/mail/virtusertable`. Můžete nastavit jiné umístění pomocí parametru makra, podrobnosti o dostupných možnostech najdete v dokumentaci k **sendmailu**.

Formát tabulky virtuálních uživatelů je velmi jednoduchý. Levá strana každého řádku obsahuje pravidlo definující původní adresu, pravá strana pak adresu na níž má být virtuální adresa mapována.

Následující příklad ukazuje tři možné typy položek:

```
samiam@bovine.net     colin
sunny@bovine.net      darkhorse@mystery.net
@dairy.org            mail@jhm.org
@artist.org           $1@red.firefly.com
```

V tomto případě sloužíme jako virtuální hostitel pro domény *bovine.net*, *dairy.org* a *artist.org*.

První záznam směruje poštu uživatele v doméně *bovine.net* na lokálního uživatele našeho systému. Druhý záznam směruje poštu jiného uživatele této domény na úplně jiného uživatele v úplně jiné doméně. Třetí příklad směruje poštu adresovanou komukoliv v doméně *dairy.org* na jednu konkrétní adresu. Konečně poslední příklad směruje poštu jakéhokoliv uživatele v doméně *ar-*

tist.org na stejného uživatele v jiné doméně, tedy pošta pro *julie@artist.org* bude poslána uživateli *julie@red.firefly.com*.

Testování konfigurace

Program **m4** zpracuje definiční makrosoubor podle svých vlastních syntaktických pravidel, aniž by cokoliv věděl o syntaxi používané programem **sendmail**. Pokud tedy v makrosouboru něco zadáte špatně, neobjeví se žádné chybové hlášení. Proto je nesmírně důležité, abyste celou konfiguraci pečlivě otestovali. Naštěstí **sendmail** nabízí poměrně jednoduchý způsob, jak to udělat.

sendmail podporuje režim „testování adres“, který umožňuje otestovat konfiguraci a odhalit chyby. V tomto režimu spouštíme **sendmail** z příkazové řádky a on se ptá na sadu pravidel a na cílovou adresu. Pak zpracuje zadanou adresu podle zvolených pravidel a postupně vypíše výstup jednotlivých použitých pravidel. V tomto režimu **sendmail** spustíte s parametrem `-bt`:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
>
```

Implicitní konfigurační soubor se nazývá `sendmail.cf`, alternativní soubor můžete otestovat pomocí volby `-C`. Abychom otestovali naši konfiguraci, potřebujeme zadat řadu různých adres, na kterých uvidíme, že jsou zpracovány tak, jak potřebujeme. Budeme si to demonstrovat na složitější konfiguraci používající UUCP podle příkladu 18.2.

Nejprve otestujme, zda je **sendmail** schopen doručit poštu lokálnímu uživateli. V následujících testech předpokládáme, že všechny adresy budou přepsány na mailer *local* na lokálním systému:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac
rewrite: ruleset   3   input: isaac
rewrite: ruleset  96   input: isaac
rewrite: ruleset  96 returns: isaac
rewrite: ruleset   3 returns: isaac
rewrite: ruleset   0   input: isaac
rewrite: ruleset 199   input: isaac
rewrite: ruleset 199 returns: isaac
rewrite: ruleset  98   input: isaac
rewrite: ruleset  98 returns: isaac
rewrite: ruleset 198   input: isaac
rewrite: ruleset 198 returns: $# local $: isaac
rewrite: ruleset   0 returns: $# local $: isaac
```

Výstup ukazuje, jak program sendmail vnitřně zpracovává adresu *isaac*. Každý řádek ukazuje, co se předává sadě pravidel a výsledek zpracování hodnoty sadou. Říkáme programu, že chceme adresu zpracovat sadami 3 a 0. Sada 0 se testuje standardně, sadu 3 jsme si vynutili, protože ta se standardně netestuje. Na posledním řádku vidíme, že výsledkem sady 0 je doručení zprávy na adresu *isaac* mailerem *local*.

Dál otestujeme poštu adresovanou na naši SMTP adresu *isacc@vstout.vbrew.com*. Měli bychom dostat stejný výsledek jako v předchozím případě:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vstout.vbrew.com
rewrite: ruleset   3   input: isaac @ vstout . vbrew . com
rewrite: ruleset  96   input: isaac < @ vstout . vbrew . com >
rewrite: ruleset  96 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   3 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   0   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198 returns: $# local $: isaac
rewrite: ruleset   0 returns: $# local $: isaac
```

Test dopadl úspěšně. Teď vyzkoušíme naši UUCP adresu *vstout!isacc*.

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 vstout!isaac
rewrite: ruleset   3   input: vstout ! isaac
rewrite: ruleset  96   input: isaac < @ vstout . UUCP >
rewrite: ruleset  96 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   3 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset   0   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 199 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset  98 returns: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198   input: isaac < @ vstout . vbrew . com . >
rewrite: ruleset 198 returns: $# local $: isaac
rewrite: ruleset   0 returns: $# local $: isaac
```

Test rovněž dopadl správně. Těmito testy jsme ověřili, že pošta určená uživateli lokálního systému bude doručena bez ohledu na to, jak mu byla adresována. Pokud jste pro svůj systém definovali jakékoliv aliasy, virtuální hostitele a podobně, měli byste testy vyzkoušet pro všechna možná jména, pod nimiž je hostitel znám, abyste ověřili, že vše funguje správně.

Dále si ověříme, že pošta adresovaná jiným systémům v doméně *vbrew.com* bude doručena přímo danému systému protokolem SMTP:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vale.vbrew.com
rewrite: ruleset   3   input: isaac @ vale . vbrew . com
rewrite: ruleset  96   input: isaac < @ vale . vbrew . com >
rewrite: ruleset  96 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset   3 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset   0   input: isaac < @ vale . vbrew . com . >
rewrite: ruleset 199   input: isaac < @ vale . vbrew . com . >
```

```
rewrite: ruleset 199 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset  98   input: isaac < @ vale . vbrew . com . >
rewrite: ruleset  98 returns: isaac < @ vale . vbrew . com . >
rewrite: ruleset 198   input: isaac < @ vale . vbrew . com . >
rewrite: ruleset 198 returns: $# smtp $@ vale . vbrew . com . /
    $: isaac < @ vale . vbrew . com . >
rewrite: ruleset   0 returns: $# smtp $@ vale . vbrew . com . /
    $: isaac < @ vale . vbrew . com . >
```

Vidíme, že test nařizuje předání pošty mailerem SMTP na hostitele *vale.vbrew.com* jeho uživateli *isacc*. Potvrdili jsme si, že definice v makru `LOCAL_NET_CONFIG` funguje správně. Aby test dopadl úspěšně, musí být možné provést překlad cílového jména, tedy buď musíme mít příslušný záznam v `/etc/hosts` nebo v DNS. Co se stane, pokud jméno nepůjde převést, si vyzkoušíme tak, že záměrně zadáme neplatné jméno hostitele:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> 3,0 isaac@vXXXX.vbrew.com
rewrite: ruleset   3   input: isaac @ vXXXX . vbrew . com
rewrite: ruleset  96   input: isaac < @ vXXXX . vbrew . com >
vXXXX.vbrew.com: Name server timeout
rewrite: ruleset  96 returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset   3 returns: isaac < @ vXXXX . vbrew . com >
== Ruleset 3,0 (3) status 75
rewrite: ruleset   0   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 199   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 199 returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  98   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  98 returns: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset 198   input: isaac < @ vXXXX . vbrew . com >
rewrite: ruleset  95   input: < uucp-new : moria > isaac </
    @ vXXXX . vbrew . com >
rewrite: ruleset  95 returns: $# uucp-new $@ moria $: isaac </
    @ vXXXX . vbrew . com >
rewrite: ruleset 198 returns: $# uucp-new $@ moria $: isaac </
    @ vXXXX . vbrew . com >
rewrite: ruleset   0 returns: $# uucp-new $@ moria $: isaac </
    @ vXXXX . vbrew . com >
```

Výsledek je dost odlišný. Nejprve sada 3 vrací chybu udávající, že název nelze převést. S touto situací se vyrovnáme díky dalšímu klíčovému prvku naší konfigurace, kterým je předávací hostitel. Smyslem předávacího hostitele je doručit cokoliv, co neumíme doručit jinak. Jméno hostitele, které jsme v příkladu zadali, nebylo možno převést a pravidla rozhodla, že pošta bude předána našemu předávacímu hostiteli *moria* mailerem *uucp-new*. Ten může být připojen lépe a třeba bude vědět, co s poštou udělat.

Poslední test ověří, že pošta adresovaná mimo naši doménu bude předána předávacímu hostiteli. Výsledek by měl být podobný předchozímu testu:

```
# /usr/sbin/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
```

```
> 3,0 isaac@linux.org.au
rewrite: ruleset   3   input: isaac @ linux . org . au
rewrite: ruleset  96   input: isaac < @ linux . org . au >
rewrite: ruleset  96 returns: isaac < @ linux . org . au . >
rewrite: ruleset   3 returns: isaac < @ linux . org . au . >
rewrite: ruleset   0   input: isaac < @ linux . org . au . >
rewrite: ruleset 199   input: isaac < @ linux . org . au . >
rewrite: ruleset 199 returns: isaac < @ linux . org . au . >
rewrite: ruleset  98   input: isaac < @ linux . org . au . >
rewrite: ruleset  98 returns: isaac < @ linux . org . au . >
rewrite: ruleset 198   input: isaac < @ linux . org . au . >
rewrite: ruleset  95   input: < uucp-new : moria > isaac </
    @ linux . org . au . >
rewrite: ruleset  95 returns: $# uucp-new $@ moria $: isaac </
    @ linux . org . au . >
rewrite: ruleset 198 returns: $# uucp-new $@ moria $: isaac </
    @ linux . org . au . >
rewrite: ruleset   0 returns: $# uucp-new $@ moria $: isaac </
    @ linux . org . au . >
```

Z výsledku testu vidíme, že název hostitele byl převeden a že zpráva bude předána předávacímu hostiteli. Definice `LOCAL_NET_CONFIG` tedy funguje správně a v obou případech se chová jak má. Celý test proběhl dobře, takže věříme, že konfigurace je správná a můžeme ji použít.

Spuštění programu sendmail

Démon **sendmail** může být spuštěn dvěma způsoby. Jedna možnost je spouštět jej prostřednictvím démona **inetd**, druhá a obvyklejší je spouštět jej samostatně. Je také obvyklé, že poštovní programy spouštějí **sendmail** jako uživatelský příkaz k doručení nově napsané pošty.

Při samostatném spouštění programu **sendmail** jej uveďte v souboru `rc` aby se spouštěl automaticky při startu systému. Běžně se používá následující syntaxe:

```
/usr/sbin/sendmail -bd -q10m
```

Parametr `-bd` říká programu, aby se spustil jako démon. Vytvoří svou kopii a poběží v pozadí. Parametr `-q10m` říká, aby testoval frontu každých 10 minut. Můžete nastavit jiný testovací interval.

Pokud chcete **sendmail** spouštět démonem **inetd**, použijete takovýto záznam:

```
smtp  stream  tcp nowait  nobody  /usr/sbin/sendmail -bs
```

`Parametr -bs` říká programu, aby na standardním vstupu a výstupu používal protokol SMTP, což je podmínka pro použití s démonem **inetd.**

Příkaz **runq** je obvykle symbolický odkaz na soubor **sendmail** a představuje jednodušší variantu příkazu

```
# sendmail -q
```

Po spuštění tímto způsobem zpracuje **sendmail** poštu ve frontě. Pokud **sendmail** spouštíte démonem **inetd**, musíte vytvořit také úlohu pro démona **cron**, která bude pravidelně spouštět příkaz **runq**, aby bylo zajištěno pravidelné odesílání pošty z fronty.

Záznam v tabulce démona **cron** může vypadat takto:

```
# Výběr poštovní fronty každých 15 minut
0,15,30,45     *     *     *     *     /usr/bin/runq
```

Na většině instalací se **sendmail** každých 15 minut spouští a odešle všechnu poštu z fronty přesně tak, jako by to dělal při výše uvedeném nastavení.

Tipy a triky

Existuje celá řada postupů jak zefektivnit práci s programem **sendmail**. Balík tohoto programu obsahuje řadu administrativních nástrojů, některé z nich si krátce představíme.

Správa poštovní fronty

Pošta k odeslání se ukládá v adresáři `/var/spool/mqueue`. Tento adresář představuje poštovní frontu. Program **sendmail** umožňuje zobrazit seznam úloh ve frontě a jejich status.

Příkaz **/usr/bin/mailq** je symbolický odkaz na **sendmail** a chová se stejně jako

```
# sendmail -bp
```

Výstup zobrazí identifikátor zprávy, její velikost, čas kdy byla zařazena do fronty, kdo ji odesílá a status zprávy. Příklad ukazuje zprávu, která uvízla ve frontě kvůli problémům:

```
$ mailq
                Mail Queue (1 request)
--Q-ID-- --Size-- -----Q-Time----- ------------Sender/Recipient------------
RAA00275      124 Wed Dec  9 17:47 root
                 (host map: lookup (tao.linux.org.au): deferred)
                                   terry@tao.linux.org.au
```

Zpráva je stále ve frontě protože nelze zjistit IP adresu cílového hostitele.

Okamžité zpracování fronty můžeme nařídit příkazem **/usr/bin/runq**.

Příkaz **runq** nemá žádný výstup, **sendmail** začne na pozadí zpracovávat poštovní frontu.

Okamžité zpracování fronty vzdáleným hostitelem

Pokud používáte občasné připojení k Internetu s *pevnou* IP adresou a používáte nějakého hostitele k ukládání vaší pošty po dobu, kdy nejste připojeni, určitě uvítáte možnost přinutit tohoto hostitele, aby vám vaši poštu předal hned poté, co se k Internetu připojíte.

Distribuce programu **sendmail** obsahuje krátký skript v Perlu, který tuto operaci usnadňuje u hostitelů, jež ji podporují. Skript **etrn** má na vzdáleném hostiteli podobný efekt jako příkaz **runq** na lokálním hostiteli. Pokud zadáme následující příkaz:

```
# etrn vstout.vbrew.com
```

donutíme tak hostitele *vstout.vbrew.com* aby zpracoval zprávy ve frontě určené pro náš systém.

Typicky tento příkaz přidáte do PPP skriptu `ip-up` takže se provede hned poté, co se naváže síťové spojení.

Analýza poštovních statistik

sendmail shromažďuje informace o přenesené poště a o hostitelích, jimž poštu doručoval. Tyto informace se zobrazí pomocí dvou příkazů: **mailstats** a **hoststat**.

mailstats

Příkaz **mailstats** zobrazí statistiky týkající se objemu přenesené pošty. Nejprve se vytiskne čas v němž byla statistika vytvořena a pak následuje tabulka s jedním řádkem pro každý používaný mailer a s řádkem celkové statistiky. Každý řádek obsahuje osm údajů:

Údaj	Význam
`M`	Číslo maileru (transportního protokolu).
`msgsfr`	Počet zpráv přijatých mailerem.
`bytes_from`	Počet KB přijatých mailerem.
`msgto`	Počet zpráv odeslaných mailerem.
`bytes_to`	Počet KB odeslaných mailerem.
`msgsrej`	Počet odmítnutých zpráv.
`msgsdis`	Počet zrušených zpráv.
`Mailer`	Název maileru.

Příklad výstupu příkazu **mailstats** ukazuje výpis 18.5.

Příklad 18.5 – Výstup programu mailstats

```
# /usr/sbin/mailstats
Statistics from Sun Dec 20 22:47:02 1998
 M   msgsfr  bytes_from   msgsto    bytes_to  msgsrej msgsdis  Mailer
 0        0          0K       19        515K        0       0  prog
 3       33        545K        0          0K        0       0  local
 5       88        972K      139       1018K        0       0  esmtp
=============================================================
 T      121       1517K      158       1533K        0       0
```

Tato data se shromažďují, pokud je v souboru `sendmail.cf` povolena volba *StatusFile* a pokud stavový soubor existuje. Typicky uvedete v souboru `sendmail.cf`

```
# stavový soubor
O StatusFile=/var/log/sendmail.st
```

K vynulování statistik musíte stavovému souboru nastavit nulovou délku:

```
> /var/log/sendmail.st
```

a restartovat **sendmail**.

hoststat

Příkaz **hoststat** vypisuje informace o hostitelích, jimž se **sendmail** pokusil doručit poštu. Příkaz **hoststat** je ekvivalentní následujícímu příkazu:

```
sendmail -bh
```

Výstup obsahuje na každém řádku jednoho hostitele a u každého pak čas kdy se spojení uskutečnilo a status zprávy.

Následující příklad ukazuje, jak může výstup programu **hoststat** vypadat. Většina položek obsahuje úspěšně doručené zprávy. Výsledek u hostitele *earthlink.net* naopak upozorňuje na neúspěch. Stavový text občas napomůže ke zjištění, proč k chybě došlo. V tomto případě došlo k timeoutu spojení, pravděpodobně protože hostitel byl nedostupný.

Příklad 18.6 – Výstup programu hoststat

```
# hoststat
 ---------- Hostname ---------- How long ago ---------Results---------
 mail.telstra.com.au                  04:05:41 250 Message accepted for
 scooter.eye-net.com.au              81+08:32:42 250 OK id=0zTGai-0008S9-0
 yarrina.connect.com.a               53+10:46:03 250 LAA09163 Message acce
 happy.optus.com.au                  55+03:34:40 250 Mail accepted
 mail.zip.com.au                      04:05:33 250 RAA23904 Message acce
 kwanon.research.canon.com.au        44+04:39:10 250 ok 911542267 qp 21186
 linux.org.au                        83+10:04:11 250 IAA31139 Message acce
 albert.aapra.org.au                  00:00:12 250 VAA21968 Message acce
 field.medicine.adelaide.edu.au      53+10:46:03 250 ok 910742814 qp 721
 copper.fuller.net                   65+12:38:00 250 OAA14470 Message acce
 amsat.org                            5+06:49:21 250 UAA07526 Message acce
 mail.acm.org                        53+10:46:17 250 TAA25012 Message acce
 extmail.bigpond.com                 11+04:06:20 250 ok
 earthlink.net                       45+05:41:09 Deferred: Connection time
```

Příkazem **purgestat** se vymažou schromážděná data a jeho ekvivalentem je příkaz

```
# sendmail -bH
```

Statistiky se rozrůstají dokud je nesmažete. Můžete je mazat periodickým spouštěním příkazu **purgestat**, abyste mohli snáze najít novější záznamy, zejména pokud je váš systém hodně zatížený. Tento příkaz můžete uvést v tabulce démona **cron**, nebo jej můžete čas od času spouštět ručně.

Nastavení a spuštění programu Exim

V této kapitole uvádíme stručný přehled jak nastavit a provozovat program Exim. I když po stránce chování je Exim dobře kompatibilní se **sendmailem**, jejich konfigurační soubory se značně liší.

Konfigurační soubor se na většině distribucí Linuxu typicky jmenuje `/etc/exim.conf` nebo `/etc/exim/config`, popřípadě `/usr/lib/exim/config` na starších distribucích. Kde je konfigurační soubor uložen, zjistíte příkazem

```
$ exim -bP configure_file
```

Konfigurační soubor bude potřeba upravit aby obsahoval hodnoty specifické pro váš systém. U většiny typických konfigurací nejsou změny obtížné a jednou nastavená funkční konfigurace se obvykle nemění.

Implicitně Exim veškerou došlou poštu zpracovává a doručuje okamžitě. Pokud máte velký provoz, můžete Exim nastavit tak, aby všechny zprávy řadil do fronty a zpracovával je pouze v pravidelných intervalech.

Při obsluze pošty na síti TCP/IP běží Exim často jako démon, při spuštění systému se spouští příkazem `/etc/init.d/exim`[137]. Po spuštění se přepne na pozadí, kde čeká na příchozí spojení na portu SMTP (obvykle port 25). Je to výhodné zejména pokud očekáváte větší provoz, protože Exim nemusíte spouštět pro každé příchozí spojení. Alternativně je možné svěřit správu portu SMTP programu **inetd**, který bude spouštět Exim vždy při přijetí spojení na tomto portu. Tato konfigurace je výhodná, pokud máte málo paměti a malý poštovní provoz.

Exim má komplikovanou soustavu řádkových parametrů, z nichž se řada shoduje se sendmailem. Program ovšem nemusíte spouštět se soustavou parametrů tak, aby vyhovoval tomu, co potřebujete udělat, nejběžnější operace můžete implementovat pomocí tradičních příkazů jako jsou **rmail** a **rsmtp**. Jedná se o symbolické odkazy na Exim (a pokud ne, můžete je tak nastavit). Když spustíte některý z těchto příkazů, Exim si zjistí pod jakým názvem byl spuštěn a sám si nastaví správné parametry.

Dva odkazy na Exim by měly existovat ve všech případech: `/usr/bin/rmail` a `/usr/sbin/sendmail`[138]. Když napíšete a odešlete zprávu pomocí uživatelského agenta jako je například **elm**, pře-

137 Další možná umístění jsou `/etc/rc.d/init.d` a `rc.inet2`. Poslední metoda je obvyklá na systémech používajících strukturu souborů v adresáři `/etc` ve stylu BSD systémů.

138 Jedná se o nové umístění programu **sendmail** podle standardu Linux File System Standard. Dalším obvyklým umístěním je `/usr/lib/sendmail`, které obvykle používají poštovní programy jež nebyly nastaveny speciálně pro Linux. Oba soubory můžete definovat jako symbolické odkazy na Exim, takže programy a skripty spouštějící sendmail spustí se stejným efektem Exim.

dává se zpráva k doručení rourou programům **sendmail** nebo **rmail**, proto by oba měly ukazovat na Exim. Seznam adresátů zprávy se Eximu předává na příkazovém řádku[139]. To samé platí pro poštu přicházející protokolem UUCP. Potřebné příkazy můžete nastavit tak aby ukazovaly na Exim následujícím postupem:

```
$ ln -s /usr/sbin/exim /usr/bin/rmail
$ ln -s /usr/sbin/exim /usr/sbin/sendmail
```

Pokud se budete chtít zabývat konfigurací programu Exim podrobněji, měli byste si prostudovat jeho popis. Pokud není součástí vaší distribuce Linuxu, najdete jej ve zdrojových kódech programu Exim, nebo si jej můžete přečíst na webových stránkách *http://www.exim.org*.

Spuštění programu Exim

Před spuštěním programu Exim se musíte rozhodnout, zda má obsluhovat příchozí SMTP spojení samostatným démonem, nebo zda má být port SMTP spravován démonem **inetd**, který spustí Exim pouze tehdy, když si nějaký klient SMTP spojení vyžádá. Obvykle se dává přednost spuštění samostatného démona, protože se tím poštovní server zatěžuje daleko méně, než opakovaným vytvářením nových procesů pro každé příchozí spojení. Protože poštovní server navíc doručuje většinu příchozí pošty přímo uživatelům, na ostatních počítačích byste měli zvolit spouštění démonem **inetd**.

Ať už zvolíte kterýkoliv z režimů operace, měli byste zkontrolovat, že soubor `/etc/services` obsahuje následující řádek:

```
smtp            25/tcp          # Simple Mail Transfer Protocol
```

Tím je definováno, který TCP port se používá pro SMTP konverzaci. Port s číslem 25 je standard definovaný RFC dokumentem „Assigned Numbers“ (RFC 1700).

Když Exim běží v režimu démona, přepne se na pozadí a čeká na spojení na portu SMTP. Když nastane spojení, rozdvojí se a vzniklý synovský proces ošetří komunikaci s partnerským procesem na klientském počítači. Démon Exim se obvykle spouští z `rc` skriptu při startu systému takto:

```
/usr/sbin/exim -bd -q15m
```

Parametr `-bd` jej spouští v režimu démona, parametrem `-q15m` se programu nařizuje zpracovat eventuální poštu ve frontě vždy každých 15 minut.

Pokud chcete raději použít démona **inetd**, musí soubor `/etc/inetd.conf` obsahovat řádek jako

```
smtp    stream  tcp nowait  root  /usr/sbin/exim  in.exim -bs
```

Nezapomeňte, že po provedení změn musíte démona **inetd** donutit znovu načíst konfiguraci tím, že mu pošlete signál HUP[140].

Režim démona a spouštění pomocí **inetd** se navzájem vylučují. Pokud spouštíte Exim jako démona, měli byste zkontrolovat, že v souboru `inetd.conf` není definován žádný řádek pro službu SMTP. Naopak pokud spoušíte Exim prostřednictvím démona **inetd**, zkontrolujte, že se nespouští přímo v žádném `rc` skriptu.

---

139 Někteří uživatelští agenti nicméně předávají zprávu transportnímu agentu protokolem SMTP, který pak volají s parametry `-bs`.

140 Použijte příkaz `kill HUP pid`, kde *pid* je číslo procesu **inetd**, které zjistíte příkazem `ps`.

Že je Exim správně nastaven k příjmu příchozích SMTP spojení si můžete ověřit telnetem na port SMTP. Takto nějak bude vypadat úspěšné připojení k Eximu:

```
$ telnet localhost smtp
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
220 richard.vbrew.com ESMTP Exim 3.13 #1 Sun, 30 Jan 2000 16:23:55 +0600
quit
221 richard.brew.com closing connection
Connection closed by foreign host.
```

Pokud se v tomto textu neobjeví vítací hlášení protokolu SMTP (řádek začínající číslem 220), zkontrolujte, zda existuje proces démona Exim, nebo zda je **inetd** správně nakonfigurován. Pokud bude všechno v pořádku, podívejte se do logovacích souborů programu Exim (budeme o nich hovořit za chvíli), abyste zjistili, zda chyba není v konfiguraci Eximu.

Když pošta nefunguje

K řešení instalačních potíží existuje celá řada možností. První z nich je kontrola logovacích souborů programu. Na linuxových systémech se typicky nacházejí v adresáři `/var/log/exim/log` a jmenují se `exim_mainlog`, `exim_rejectlog` a `exim_paniclog`. Na jiných operačních systémech bývají často v adresáři `/var/spool/exim/log`. Kde se logovací soubory nacházejí můžete zjistit příkazem

```
exim -bP log_file_path
```

Hlavní log obsahuje všechny transakce, log zamítných zpráv obsahuje podrobnosti o zprávách, které byly odmítnuty z důvodů nastavených politik. Záznamy paniky se vztahují k chybám v konfiguraci a k podobným událostem.

Typická část hlavního logu je uvedena níže. Každý záznam je tvořen jedním řádkem, který obsahuje datum a čas. Rozdělili jsme jej na několik řádků, aby se vešel na šířku stránky:

```
2000-01-30 15:46:37 12EwYe-0004WO-00 <= jack@vstout.vbrew.com
  H=vstout.vbrew.com [192.168.131.111] U=exim P=esmtp S=32100
  id=38690D72.286F@vstout.vbrew.com
2000-01-30 15:46:37 12EwYe-0004WO-00 => jill <jill@vbrew.com>
  D=localuser T=local_delivery
2000-01-30 15:46:37 12EwYe-0004WO-00 Completed
```

Tento záznam ukazuje, že pošta od *jack@vstout.vbrew.com* pro *jill@vbrew.com* byla úspěšně doručena do schránky na lokálním počítači. Přijetí zprávy ukazuje symbol <=, odchod zprávy symbol =>.

Existují dva typy chyb doručení: trvalé a dočasné. Trvalá chyba je zaznamenána v následujícím záznamu a označuje se symboly **:

```
2000-01-30 14:48:28 12EvcH-0003rC-00 ** bill@lager.vbrew.com
  R=lookuphost T=smtp: SMTP error from remote mailer after RCPT TO:
  <bill@lager.vbrew.com>: host lager.vbrew.com [192.168.157.2]:
  550 <bill@lager.vbrew.com>... User unknown
```

Po podobné chybě odešle Exim odesílateli zprávu o chybě.

Dočasné chyby jsou označeny symboly ==.

```
2000-01-30 12:50:50 12E9Un-0004Wq-00 == jim@bitter.vbrew.com
  T=smtp defer (145): Connection timed out
```

Takovéto chyby jsou typické v situaci, kdy Exim správně zjistí, že zpráva má být doručena na vzdáleného hostitele, nepodaří se mu ale spojit s jeho SMTP službou. Hostitel může být vypnutý, nebo mohou být problémy se síťovým spojením. Kdykoliv je zpráva tímto způsobem *odložena*, zůstává ve frontě Eximu a pokusy o doručení se pravidelně opakují. Pokud se ji ovšem v nějaké poměrně dlouhé době nepodaří doručit (typicky několik dnů), dojde k trvalé chybě a zpráva bude zahozena.

Pokud se vám příčinu chyby nepodaří zjistit z hlášení, která Exim generuje, můžete zapnout ladicí režim. Provedete to pomocí parametru `-d`, za kterým může následovat úroveň podrobností (hodnota 9 je nejobšírnější režim). Pak Exim zobrazuje hlášení o činnosti na obrazovce a můžete tak získat podrobnější údaje o tom, co není v pořádku.

Překlad programu Exim

Exim stále prochází aktivním vývojem. Verze, které jsou součástí distribucí, pravděpodobně nejsou posledními dostupnými verzemi. Pokud potřebujete nějakou funkci nebo opravu, která je dostupná až v novější verzi, musíte získat zdrojový kód a přeložit si program sami. Poslední verzi získáte prostřednictvím odkazů na adrese *http://www.exim.org*.

Linux je jeden z mnoha operačních systémů, které zdrojový kód Eximu podporují. Když chcete Exim přeložit pro Linux, musíte upravit soubor `src/EDITME` a výsledek umístit do souboru `Local/Makefile`. V souboru `src/EDITME` jsou komentáře, které říkají, k čemu se různá nastavení používají. Pak spustíte program **make**. Podrobnosti o sestavení programu najdete v dodávané dokumentaci.

Režimy doručování pošty

Jak už bylo řečeno, Exim je schopen buď doručovat poštu okamžitě, nebo ji řadit do fronty a zpracovat později. Všechna příchozí pošta je uložena v adresáři `input` pod `/var/spool/exim`. Pokud se nepoužívá řazení do fronty, provádí se doručení každé zprávy ihned poté, co dorazí. V opačném případě zůstává ve frontě tak dlouho, dokud ji nevyzvedne proces zvaný *queue-runner*. Zpracování fronty může být nastaveno napevno parametrem `queue_only` v konfiguračním souboru, nebo může být zapínáno podmíněně na základně zatížení systému v poslední minutě takto:

```
queue_only_load = 4
```

Tento příkaz způsobí zpracování fronty, pokud zatížení systému přesáhne 4[141].

Pokud nemáte počítač trvale připojen k Internetu, můžete zapnout řazení do fronty pouze u zpráv na vzdálené adresy, doručování lokálních zpráv však může probíhat okamžitě. Dosáhnete toho nastavením

```
queue_remote_domains = *
```

141 Zatížení systému je standardní unixová metrika založená na tom, kolik procesů je uloženo ve frontě a čeká na spuštění. Příkaz **uptime** zobrazí průměrné zatížení v poslední minutě, 5 a 15 minutách.

Pokud zapnete jakýkoliv typ řazení zpráv do fronty, musíte zajitit, aby fronta byla pravidelně vybírána, typicky každých 10 až 15 minut. I pokud nemáte zapnuto řazení zpráv do fronty, stále je nutné frontu vybírat, protože se do ní řadí i zprávy odložené kvůli dočasné chybě. Spouštíte-li Exim v režimu démona, parametrem `-q15m` na příkazovém řádku mu nařídíte výběr fronty každých 15 minut. Kromě toho můžete Exim pravidelně spouštět prostřednictvím démona **cron**.

Zprávy zařazené ve frontě můžete vypsat pomocí parametru `-bp`. Můžete také vytvořit **mailq** jako odkaz na Exim a spouštět **mailq**:

```
$ mailq
 2h    52K 12EwGE-0005jD-00 <sam@vbrew.com>
         D bob@vbrew.com
           harry@example.net
```

Tento výstup říká, že ve frontě čeká jedna zpráva od *sam@vbrew.com*, určená dvěma adresátům. Byla úspěšně doručena na adresu *bob@vbrew.com*, nicméně se ji zatím nepodařilo doručit na adresu *harry@example.net*, i když je ve frontě už dvě hodiny. Velikost zprávy je 52K a Exim pro ni používá identifikátor `12EwGE-0005jD-00`. Příčinu proč nebyla zpráva doručena můžete zjistit z logovacího souboru této zprávy, který se nachází v adresáři `msglog`. Jednoduše to provedete parametrem `-Mvl`:

```
$ exim -Mvl 12EwGE-0005jD-00
2000-01-30 17:28:13 example.net [192.168.8.2]: Connection timed out
2000-01-30 17:28:13 harry@example.net: remote_smtp transport deferred:
  Connection timed out
```

Logovací soubory jednotlivých zpráv obsahují záznamy o konkrétních zprávách, takže je můžete snadno ověřit. Některé informace je možné zjistit z hlavního logovacího souboru pomocí nástroje **exigrep**:

```
$ exigrep 12EwGE-0005jD-00 /var/log/exim/exim_mainlog
```

Tento příkaz může trvat déle, zejména na zatíženém systému, kde může být logovací soubor dlouhý. Nástroj **exigrep** je užitečný zejména pokud hledáme informace o více zprávách. Prvním parametrem je regulární výraz a vypíše všechny logovací záznamy o všech zprávách, u nichž alespoň jeden ze záznamů obsahuje řetězec odpovídající danému regulárnímu výrazu. Díky tomu je možné snadno zjistit všechny zprávy určené na nějakou adresu nebo všechny přicházející nebo odcházející na nějakého hostitele.

Celkový přehled o činnosti programu Exim můžete získat tak, že na jeho hlavní logovací soubor spustíte **tail**. Další možnost je spustit nástroj **eximon**, dodávaný přímo s Eximem. Jedná se o X11 aplikaci, která obsahuje výpis hlavního logovacího souboru, a kromě toho zobrazuje seznam zpráv čekajících na doručení a nějaké statistiky o aktivitě programu.

Různé konfigurační volby

Uveďme si několik nejužitečnějších voleb, které můžete nastavit v konfiguračním souboru.

message_size_limit Tento parametr omezuje velikost zpráv, které bude Exim přijímat.

return_size_limit Tento parametr omezuje velikost původní zprávy, kterou Exim vrací jako součást chybové zprávy.

deliver_load_max Pokud zatížení systému dosáhne nastavené hodnoty, doručování pošty se pozastaví, nicméně zprávy budou i nadále přijímány.

smtp_accept_max Maximální počet současných SMTP spojení, které Exim přijímá.

log_level Množství podrobností zapisovaných v logovacích souborech. Kromě toho existují i další parametry začínající *log_*, které nařizují záznam konkrétních údajů.

Směrování a doručování zpráv

Exim rozděluje doručení zpráv na tři samostatné úlohy: směrování, doručení a přenos. Pro každou operaci existuje řada různých modulů, každý z nich se konfiguruje samostatně. V konfiguračním souboru tak bývá typicky nastaveno více směrovacích, doručovacích a přenosových modulů.

Směrovače analyzují vzdálené adresy, určují kterému hostiteli má být zpráva poslána a jaký doručovací modul má být použit. U hostitelů připojených k Internetu bývá často nastaven jediný směrovač, který provádí analýzu jmen pomocí DNS. Alternativně může být samostatný směrovač pro adresy hostitelů v lokální síti a druhý, který bude všechny ostatní zprávy směrovat na *předávacího hostitele*, například server poskytovatele internetového připojení.

Lokální adresy se předávají doručovacím modulům, kterých je typicky několik, a které obsluhují aliasy a předávání a také identifikaci lokálních schránek. Doručovací seznamy mohou být obsluhovány aliasovým nebo předávacím doručovačem. Pokud je pro nějakou adresu použit alias nebo předání, nová adresa bude nezávisle zpracována směrovačem nebo doručovačem. Nejběžnějším doručením je doručení do schránky, zprávy však mohou být předávány příkazům nebo připojovány k jiným souborům než jsou poštovní schránky.

Transportní moduly implementují způsob přenosu, například přenesení zprávy SMTP spojením nebo uložení do lokální schránky. Jaký transportní modul pro danou adresu použít rozhodují směrovače a doručovače. Pokud se transport nezdaří, Exim buď zprávu zahodí a vygeneruje chybovou zprávu, nebo zprávu odloží pro pozdější doručení.

V konfiguraci jednotlivých služeb máte velkou volnost. U každé z nich je k dispozici množství ovladačů, ze kterých si volíte ty, které potřebujete. Pak je Eximu popíšete v samostatných částech konfiguračního souboru. Nejprve se definují transportní moduly, pak doručovače a nakonec směrovače. Neexistují žádné implicitně používané moduly, i když se Exim dodává se standardním konfiguračním souborem, který vyhovuje ve většině jednoduchých nasazení. Pokud chcete změnit směrovací politiky nebo metody transportu, je jednodušší vyjít ze standardní konfigurace a tu upravit, než se pokoušet o vytvoření celé nové konfigurace úplně od začátku.

Směrování zpráv

Když má Exim adresu k doručení, podívá se nejprve, zda nejde o doménu, kterou obsluhuje lokální hostitel – to zjistí porovnáním se seznamem domén `local_domains` v konfiguračním souboru. Pokud není tato volba nastavena, za lokální doménu se považuje pouze doména odpovídající názvu lokálního hostitele. Pokud je doména lokální, předá se adresa doručovacímu modulu. Pokud není, obsluhuje ji směrovač a zjistí, kterému stroji zásilku předat[142].

142 Tento popis je zjednodušený. Je možné, že direktor předá transportnímu modulu adresu k doručení na vzdáleného hostitele a naopak směrovač může předat transportnímu modulu adresu k lokálnímu doručení, která vede k zapsání zprávy do souboru nebo roury. Směrovače mohou také za určitých okolností předávat adresy doručovacím modulům.

Doručování zpráv na lokální adresy

Ve většině případů je lokální adresa čistě přihlašovací jméno uživatele, v takovém případě se zásilka doručí do poštovní schránky uživatele, tedy do `/var/spool/mail/`*`uživatel`*. Dalšími případy mohou být aliasy, názvy poštovních konferencí a pošta předávaná lokálním uživatelem. V těchto případech se lokální adresa přeloží na nový seznam adres, které mohou být místní nebo vzdálené.

Kromě těchto „normálních“ adres umí Exim obsluhovat další typy lokálních cílů, jako jsou soubory a roury. Při doručování do souboru připojí Exim zprávu na konec zadaného souboru, který v případě potřeby vytvoří. Soubory a roury nejsou adresovatelné přímo, nemůžete tedy poslat poštu na adresu řekněme /etc/passwd@vbrew.com a předpokládat, že dojde k přepsání souboru s hesly – doručování do souboru je možné pouze tehdy, je-li soubor specifikován aliasem nebo forwardingem. Adresa /etc/passwd@vbrew.com je nicméně ze syntaktického hlediska platnou adresou a pokud pro ni Exim přijme zprávu, bude (typicky) hledat uživatele s přihlašovacím jménem /etc/passwd, nenajde jej a zprávu zahodí.

V seznamu aliasů a v předávacím souboru je za název souboru považováno cokoliv, co začíná lomítkem a nepředstavuje to plně kvalifikovanou poštovní adresu. Takže `/tmp/junk` v předávacím nebo aliasovém souboru znamená soubor, /tmp/junk@vbrew.com bude chápáno jako poštovní adresa, i když zřejmě nepříliš užitečná. Nicméně platné adresy tohoto typu můžete potkat při odesílání pošty přes X.400 brány, protože adresy podle standardu X.400 začínají lomítkem.

Podobně je rourovým příkazem cokoliv začínající symbolem roury (`|`), pokud to nepředstavuje platnou adresu. Pokud nezměníte konfiguraci programu, Exim nespouští příkazy v příkazovém interpretu, ale rozdělí je na sám název příkazu a na parametry a spustí jej přímo. Zpráva se pak příkazu předává na standardní vstup.

Například pokud chcete poštovní konferenci směrovat na nějakou lokální skupinu, můžete použít skript pojmenovaný **gateit** a vytvořit lokální alias, který všechny pro danou konfrenci doručí do skriptu příkazem `|gateit`. Pokud příkazový řádek obsahuje čárku, musí být celý i s úvodní rourou uzavřen v uvozovkách.

Lokální uživatelé

Lokální adresa nejčastěji znamená schránku uživatele. Ty se normálně nacházejí v adresáři `/var/spool/mail` a jde o soubor pojmenovaný jménem uživatele, který jej také vlastní. Pokud tento soubor neexistuje, Exim jej automaticky vytvoří.

V některých konfiguracích je skupina souboru nastavena na skupinu daného uživatele a práva jsou 0600. V těchto případech běží doručovací proces jako uživatel a ten může celou schránku smazat. V jiných konfiguracích je skupina nastavena na mail a soubor má režim 0660, doručovací proces běží pod uid systému a skupinou mail, a uživatel nemůže smazat soubor schránky, může jej nicméně vyprázdnit.

Momentálně standardním místem pro ukládání schránek je adresář `/var/spool/mail`, nicméně některé poštovní programy mohou být přeloženy tak, aby používaly jiné cesty, například `/usr/spool/mail`. Pokud systematicky dochází k chybám doručování pošty lokálním uživatelům, mohlo by pomoci pokud `/usr/spool/mail` vytvoříte jako symbolický odkaz na `/var/spool/mail`.

V souboru aliasů by měly být uvedeny adresy MAILER-DAEMON a postmaster, které by měly reprezentovat adresu správce systému. Adresu MAILER-DAEMON Exim používá při odesílání chybových zpráv. Doporučuje se také, aby jako alias pro administrátora systému byla nastavena adresa root, zejména pokud se doručování provádí s právy uživatele, aby se tak předešlo situaci, kdy bude jakákoliv pošta doručována v superuživatelském režimu.

Předávání

Uživatel může přesměrovat svou poštu na alternativní adresu vytvořením souboru `.forward` v domovském adresáři. Tento soubor obsahuje seznam adres oddělených čárkami a/nebo oddělovači řádků. Čtou se a interpretují všechny řádky souboru. Typickým příkladem souboru `.forward` pro dobu dovolené může být

```
janet, "vacation"
```

V popisech starších souborů `.forward` můžete vidět, že jména uživatelů jsou uvedena obráceným lomítkem. Tento mechanismus byl nutný u některých starších transportních agentů, a zakazoval jim hledat v souboru `.forward` v domovském adresáři nového uživatele, protože by to mohlo vést ke vzniku smyček. U Eximu to není nutné, protože vzniku podobných smyček zabraňuje automaticky[143]. Nicméně použití obráceného lomítka je povoleno a dává smysl v případě, kdy je najednou obsluhováno více domén. Bez uvedení obráceného lomítka se nekvalifikované uživatelské jméno doplní implicitní doménou, při použití obráceného lomítka zůstane doména z původní adresy.

První adresa v souboru přeposílá příchozí zprávu to poštovní schránky uživatele *janet*, příkaz **vacation** vrátí odesílateli krátké upozornění[144].

Kromě klasických předávacích souborů umí Exim také jejich složitější variantu, takzvané *filtry*. Namísto prostého předání může filtr nejprve otestovat obsah a předat ji například pouze tehdy, pokud předmět obsahuje slovo „urgent“. Tuto možnost musí uživatelům povolit správce systému.

Aliasy

Exim umí pracovat se soubory aliasů kompatibilními s formátem programu **sendmail**. Položky v souboru mají následující tvar:

```
alias: adresáti
```

Adresáti jsou tvořeni čárkami odděleným seznamem adres, jimiž bude alias nahrazen. Seznam adresátů může pokračovat i na následujícím řádku za předpokladu, že je tento řádek odsazen od okraje.

Další speciální funkce umožňuje, aby Exim obsluhoval poštovní seznamy uložené odděleně od souboru aliasů. Pokud místo adresátů zadáte údaj `:include:`*soubor*, Exim přečte zadaný soubor a jeho obsah bude chápat jako seznam adresátů. Další možnost správy poštovních konferencí je popsána dále v části *Poštovní konference*.

Hlavní soubor aliasů se jmenuje `/etc/aliases`. Pokud bude nastaven jako volně zapisovatelný nebo zapisovatelný skupinou, Exim jej odmítne použít a doručování lokálních zpráv pozastaví. Test používaný pro kontrolu režimu souboru můžete nastavit změnou hodnoty `modemask` v direktoru `system_aliases`.

Soubor aliasů může vypadat takto:

---

143 Pokud má být zpracována adresa, kterou direktor vygeneroval zpracováním původní adresy, tak už se direktor znovu nevolá.

144 Pokud používáte nějaký takový program, zajistěte, aby neodpovídal na zprávy posílané z poštovních seznamů. Je docela nepříjemné, pokud někdo odjede na dovolenou a ke každé zprávě došlé do konference se přidá zpráva oznamující, že on je na dovolené. Pro správce konferencí: právě proto není rozumné nastavovat hodnotu `Reply-To:` ve zprávách posílaných konferencí na adresu, kterou se do konference přispívá.

```
# vbrew.com, soubor /etc/aliases
hostmaster: janet
postmaster: janet
usenet: phil
# Poštovní konference vývojářů
development: joe, sue, mark, biff,
        /var/mail/log/development
owner-development: joe
# Obecná oznámení se zasílají všem uživatelům
announce: :include: /etc/Exim/staff,
        /var/mail/log/announce
owner-announce: root
# poštovní konference ppp se směruje na lokální skupinu
ppp-list: "|/usr/local/bin/gateit local.lists.ppp"
```

Pokud jsou v souboru aliasů tak jako zde uvedeny soubory a roury, je nutné Eximu říct, pod jakým uživatelským ID má doručování probíhat. V konfiguračním souboru musí být nastavena volba `user` (a případně `group`), a to v části direktoru aliasů nebo v transportním modulu, který doručování provádí.

Pokud dojde k chybě při doručování pošty na adresu uvedenou v souboru aliasů, Exim jako obvykle pošle chybové hlášení odesílateli zprávy, což nemusí stačit. Pomocí volby `errors_to` je možné nastavit doručování chybových zpráv na jinou adresu, například na správce poštovního systému.

Poštovní konference

Kromě aliasů je možné poštovní konference obsluhovat také direktorem `forwardfile`. Konference jsou ukládány v jednom adresáři, například `/etc/exim/lists` a konference pojmenovaná například *nag-bugs* pak bude uložena v souboru `lists/nag-bugs`. Tento soubor obsahuje adresy příjemců oddělené buď čárkami nebo oddělovači řádků. Řádky začínající symbolem # představují komentáře. Takto nastavená data je možné používat následujícím direktorem:

```
lists:
  driver = forwardfile
  file = /etc/exim/lists/${local_part}
  no_check_local_user
  errors_to = ${local_part}-request
```

Při spuštění direktoru dojde k expandování hodnot v údajích `file` a `errors_to`. Expanze znamená, že řetězce začínající symbolem $ budou při každém použití řetězce nahrazeny. Nejjednodušším způsobem expanze je použití jedné z proměnných programu Exim tak, jak to děláme tady. Řetězec `${local_part}` bude nahrazen hodnotou `$local_part`, což je lokální část právě zpracovávané adresy.

Pro každou poštovní konferenci by měl existovat uživatel (nebo alias), který bude pojmenován *`názevkonference`*`-request`. Chyby při doručování nebo expanzi adres budou oznamovány na tuto adresu.

Ochrana před spammingem

Spam, nebo také nevyžádaná elektronická pošta, je nepříjemný problém řady uživatelů. Byl založen projekt, který má napomoci řešení tohoto problému. Jmenuje se Mail Abuse Protection Sys-

tem (MAPS) a v jeho rámci byl vytvořen mechanismus, který redukuje problém spammingu. Tento mechanismus se jmenuje Real Time Blackhole List (RBL). O tom, jak MAPS RBL funguje, se můžete informovat na adrese *http://maps.vix.com/rbl/*. Myšlenka je jednoduchá. Systémy odesílající spam jsou zařazeny do databáze a programy jako Exim si mohou v databázi ověřit, že právě příchozí pošta nepochází z nějaké takové adresy.

Se vznikem projektu RBL vznikla i řada dalších seznamů. Jedním z nich je Dial-Up List (DUL), který obsahuje IP adresy telefonicky připojovaných hostitelů. Ti by měli veškerou odchozí poštu směrovat zásadně přes poštovní server svého poskytovatele. Řada systémů blokuje příjem pošty od telefonicky připojených hostitelů, protože když takový hostitel obchází poštovní server svého poskytovatele, neznamená to obvykle nic dobrého.

Exim obsahuje podporu tohoto typu seznamů. Konfiguruje se velmi snadno. Do souboru `/etc/exim.conf` přidáte následující řádek:

```
# Vixie / MAPS RBL (http://maps.vix.com/rbl)
rbl_domains = rbl.maps.vix.com : dul.maps.vix.com
```

Tento příklad testuje seznamy RBL a DUL a odmítá veškerou poštu od hostitelů, kteří jsou v některém ze seznamů uvedeni. Volba `rbl_hosts` umožňuje nastavit, pro jaké skupiny hostitelů se má nebo nemá kontrola proti RBL seznamu provádět. Implicitní nastavení je

```
rbl_hosts = *
```

Znamená to, že kontrola se bude provádět pro všechny hostitele. Pokud chcete přepsat nastavení seznamu a od určitých hostitelů přijímat poštu vždy, můžete zadat například

```
bl_hosts = ! nocheck.example.com : *
```

Vykřičník před prvním údajem znamená negaci údaje: pokud se k vám připojí hostitel *nocheck.example.com*, vyhovuje tomuto údaji. Protože se jedná o negovaný údaj, nebude kontrola proti RBL provedena. U všech ostatních hostitelů se kontrola provádět bude.

Nastavení pro UUCP

Exim neobsahuje podporu přenosu pošty přes UUCP a nepodporuje vykřičníkové adresy formátu UUCP. Pokud se ale používá doménové adresování, je možné poměrně jednoduše Exim pro práci s UUCP nastavit. Následující příklad představuje konfiguraci, v níž se určité domény posílají na UUCP:

```
# Transport
uucp:
  driver = pipe
  user = nobody
  command = "/usr/local/bin/uux -r - \
    ${substr_-5:$host}!rmail ${local_part}"
  return_fail_output = true

# Router
uucphost:
  transport = uucp
  driver = domainlist
  route_file = /usr/exim/uucphosts
  search_type = lsearch
```

V celém konfiguračním souboru by se sekce `Transport` nacházela mezi ostatními doručovači a sekce `Router` mezi ostatními směrovači. Soubor `/usr/exim/uucphosts` obsahuje údaje jako:

```
darksite.example.com:          darksite.UUCP
```

který bude interpretován jako „Pošli poštu adresovanou doméně *darksite.example.com* na UUCP hostitele *darksite*“. Konfiguraci je možné zjednodušit tak, že směrovač nebude k názvu *darksite* přidávat příponu *UUCP*, kterou musí transportní modul zase odstranit, nicméně použité nastavení má výhodu v tom, že je jasný rozdíl mezi doménovým jménem *darksite.example.com* a UUCP jménem *darksite*.

Kdykoliv směrovač narazí na adresu uvedenou v souboru tras, pošle ji transportnímu modulu UUCP, který ji následně předá příkazu **uux** (popsanému v kapitole 16). Pokud dojde k chybě, **uux** vygeneruje nějaký výstup a skončí s nenulovým výstupním kódem. Nastavení `return_fail_output` zajistí, že chybové hlášení bude předáno zpět odesílateli.

Pokud jsou příchozí UUCP zprávy sdružovány v souborech ve formátu dávkového SMTP, je možné je Eximu přímo předat příkazem

```
exim -bS </var/uucp/incoming/001
```

Je zde nicméně jeden problém. Když Exim obdrží zprávu lokálně, předpokládá, že odesílatelem je lokální uživatel, který Exim volal, nicméně u UUCP dávek chceme, aby byl odesílatel převzat z příchozí zprávy. Exim to provede, pokud byl zavolán procesem označeným jako *prověřený uživatel*. Pokud budete příchozí UUCP poštu obsluhovat uživatelem uucp, musíte v konfiguračním souboru zadat nastavení

```
trusted_users = uucp
```

Tím se zajistí, že adresy budou správně převzaty ze zpráv.

Síťové news

Síťové news, nebo Usenet news, zůstávají i dnes jednou z nejdůležitějších a nejoceňovanějších služeb. I přes nevyžádanou reklamní poštu a záplavu pornografie síťové news stále udržují několik vysoce kvalitních diskusních skupin, které před nástupem webu představovaly základní informační zdroj. I v současné době, kdy existují miliardy webových stránek, síťové news stále představují online nápovědu a komunikaci na řadu různých témat.

Historie Usenetu

Myšlenka síťových news se zrodila v roce 1979, kdy dva absolventi univerzity, Tom Truscott a Jim Ellis, začali uvažovat o využití protokolu UUCP ke spojení počítačů za účelem výměny informací mezi uživateli Unixu. V Severní Karolině vytvořili malou síť tvořenou třemi počítači.

Původně byl provoz sítě obsluhován několika skripty příkazového interpretu (které byly později přepsány do jazyka C), ale ty se nikdy nedostaly na veřejnost. Rychle je nahradily takzvané „A news“, které byly prvním veřejným vydáním softwaru news.

Systém „A news“ nebyl navržen k obsluze více než několika článků na skupinu a den. Když začal jejich objem narůstat, rozhodli se ho Mark Horton a Matt Glickman přepsat, a výsledek svého snažení nazvali vydání „B“ (takzvané B News). První veřejné vydání B News mělo číslo verze 2.1 a stalo se tak v roce 1982. Postupně byl systém rozšiřován o některé nové vlastnosti. Současná verze B News má číslo 2.11. Pomalu však zastarává, nehledě k tomu, že oficiální správce přešel na INN.

O další přepis se postarali Geoff Colyer a Hanry Spencer v roce 1987; šlo o verzi „C“, neboli „C News“. Následovala spousta oprav, z nichž nejvýznamnější bylo vydání C News Performance Release. V systémech, které spravují velké množství diskusních skupin, se režie způsobená častým spouštěním programu **relaynews**, který je zodpovědný za odesílání příchozích článků jiným hostitelům, stává významnou. Verze Performance Release přidává novou volbu programu **relaynews**, která umožňuje spouštět tento program v režimu démona, který běží na pozadí.

Verze Performance Release je verzí C News, která je v současné době součástí většiny linuxových distribucí. Program „C News“ popisujeme v kapitole 21.

Všechna vydání až po verzi „C“ jsou určena především pro sítě UUCP, i když je lze stejně dobře používat i v jiných prostředích. Efektivní přenos v sítích typu TCP/IP, DECNet nebo jim podobných vyžaduje nové schéma. Z toho důvodu byl v roce 1986 uveden protokol NNTP (Network News Transfer Protocol). Je založen na síťových spojeních a specifikuje množství příkazů pro interaktivní předávání a získávání článků.

Na síti existuje spousta aplikací založených na protokolu NNTP. Jednou z nich je i balík **nntpd**, jehož autory jsou Brian Barber a Phil Lapsley. Tento balík umožňuje přístup ke službě news více počítačům na lokální síti. Byl navržen jako doplněk balíků B news nebo C News, které měl rozšířit

o vlastnosti protokolu NNTP. Pokud chcete C News použít ve spolupráci s NNTP serverem, v kapitole 22 popisujeme konfiguraci NNTP démona.

Jiným balíkem vycházejícím z protokolu NNTP je INN, neboli Internet News. Nejedná se o koncové rozhraní, ale spíše o plnohodnotný systém news. Obsahuje důmyslného démona pro přenos news, který je schopen efektivně spravovat několik souběžných spojení NNTP a z toho důvodu je vhodným serverem pro mnoho systémů připojených k Internetu. Hovoříme o něm v kapitole 23.

Co je to vlastně Usenet?

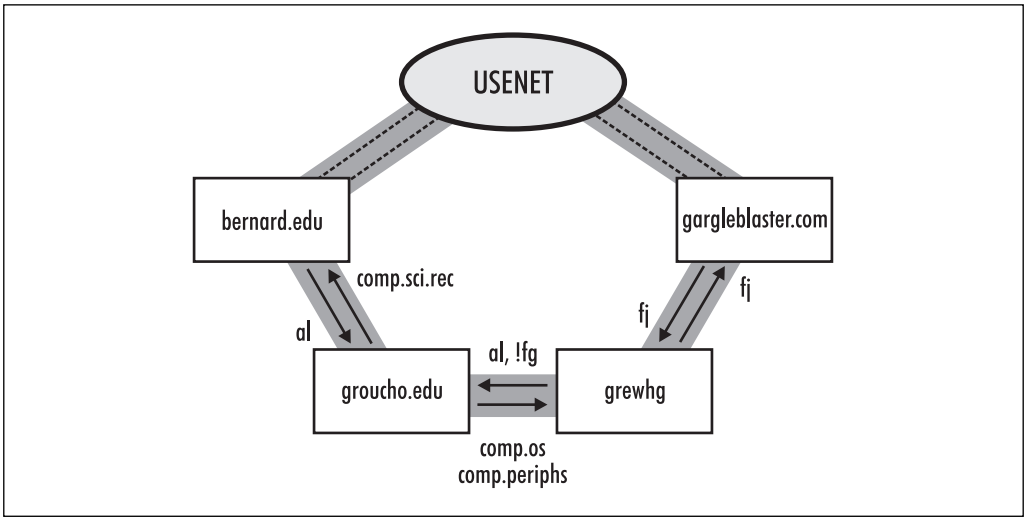
Jedním z ohromujících faktů ohledně Usenetu je skutečnost, že není součástí žádné organizace ani jej neřídí žádná centralizovaná autorita pro správu sítě. Součástí usenetové tradice je fakt, že kromě technické specifikace nejste schopni definovat, *co* to vlastně Usenet je. S vědomím, že to možná bude znít hloupě, bychom mohli Usenet definovat jako spolupráci vzájemně oddělených míst, která si mezi sebou vyměňují usenetové news. Aby se váš systém stal součástí Usenetu, potřebujete nalézt jiný takový systém a sjednat si dohodu s jeho vlastníky nebo správci o výměně news. Poskytování news jinému systému se někdy také říká „krmení“, z čehož vychází další obecný axiom usenetové filosofie: „Dej mu nažrat a jsi v něm.“

Nejmenší jednotkou usenetových news je *článek*. Jedná se o zprávu, kterou uživatel napíše a „zveřejní“ na síti. Aby s ním mohl systém news pracovat, je opatřen administrativními informacemi, takzvanou hlavičkou článku. Ta je podobná formátu poštovní hlavičky specifikované standardem internetové pošty RFC 822 v tom, že je tvořena několika řádky textu, z nichž každý začíná názvem pole ukončeným dvojtečkou, za nímž následuje hodnota tohoto pole[145].

Články jsou postupovány jedné nebo více diskusním skupinám. Diskusní skupina se zabývá články týkajícími se jednoho tématu. Všechny diskusní skupiny jsou hierarchicky organizovány, přičemž název každé skupiny zároveň udává její pozici v této hierarchii. Díky tomu se dá snadno poznat, čeho se skupina týká. Například z názvu diskusní skupiny **comp.os.linux.announce** může každý poznat, že slouží pro oznámení týkající se počítačového operačního systému jménem Linux.

Tyto články si pak mezi sebou vymění všechna usenetová místa, která mají zájem o news z příslušné diskusní skupiny. Pokud dvě místa souhlasí s výměnou news, mohou si mezi sebou předávat libovolné diskusní skupiny a mohou do nich přidávat své místní hierarchicky uspořádané skupiny. Například server **groucho.edu** může mít odkaz na server **bernard.edu**, který je hlavním zdrojem news, a několik odkazů na menší servery, které naopak zásobuje svými news. Takto může Bernyard College obdržet všechny usenetové skupiny, zatímco GMU má zájem jen o některé hlavní skupiny typu **sci**, **comp** nebo **rec**. Některé z pomalých serverů, řekněme například UUCP server **brewhq**, bude mít zájem o ještě menší počet skupin, z důvodu nedostatku síťových nebo hardwarových zdrojů. Na druhou stranu může mít server **brewhq** zájem o diskusní skupiny z větve **fj**, které GMU nepřijímá. Pro takové případy udržuje jiný odkaz na **gargleblaster.com**, kde je umístěna celá hierarchie **fj** a ten předá serveru **brewhq**. Tok news ukazuje obrázek 20.1.

145 Formát hlaviček usenetových news je definován v dokumentu RFC 1036, *Standard for interchange of USENET messages.*



Obrázek 20.1 – Tok usenetových news na Groucho Marx University

Popisky šipek vedoucích ze serveru **brewhq** vyžadují bližší vysvětlení. Implicitně chce totiž tento server všechny místní news posílat na adresu **groucho.edu**. Avšak na tomto serveru není uchovávána skupina **fj** a není tedy důvod tomuto systému zprávy z této skupiny posílat. Proto je předávání zpráv z **brewhq** do GMU označeno jako `all,!fj`, což znamená, že jsou univerzitě posílány všechny diskusní skupiny s výjimkou **fj**.

Jak Usenet obsluhuje news?

Dnes se Usenet rozrostl do obrovských rozměrů. Systémy, které uchovávají celé síťové news obvykle přenášejí něco kolem šedesáti megabajtů dat denně[146]. Samozřejmě, že to vyžaduje mnohem více než jen postrkování souborů. Podívejme se tedy na způsob, jakým většina unixových systémů obsluhuje usenetové news.

Všechno začíná, když uživatel vytvoří a zveřejní článek. Uživatel zadá svůj text ve speciální aplikaci zvané newsreader, která jej správně naformátuje pro přenos na lokální news server. V prostředí Unixu používá newsreader k přenosu zpráv na newsserver protokolem TCP/IP typicky program **inews**. Je ale možné napsat článek přímo jako soubor ve speciálním adresáři, takzvané frontě news. Jakmile je článek přenesen na lokální news server, ten zodpovídá za jeho distribuci dalším uživatelům news.

News jsou po síti šířeny různými druhy přenosů. Historickým médiem je UUCP, ale hlavní dopravu dnes obstarává Internet. Použitému směrovacímu algoritmu se říká *flooding*. Každý systém udržuje několik spojení (*news feeds*) s jinými systémy. Každý článek vytvořený nebo obdržený lokálním systémem news bude předán okolním systémům s tou výjimkou, že pokud některý systém už daný článek má, nebude mu poslán znovu a bude smazán. Systém může na základě hlavičkového pole `Path:` zjistit, kterými systémy již daný článek prošel. Tato hlavička obsahuje seznam všech systémů, které článek předaly dále, v podobě vykřičníkové notace.

146 Momentík: 60 MB při rychlosti 9600 b/s, to máme 60 milionů krát 1024, děleno ... hmm, hmm, – to máme 34 hodin přenosu denně!

Aby bylo možné články rozlišovat a odhalovat duplicity, nese si s sebou každý usenetový článek identifikátor zprávy (uvedený v hlavičkovém poli `Message-Id:`), které kombinuje název odesílajícího systému a sériové číslo do tvaru `<sériové_číslo@odesílatel>`. Systém news uloží identifikátor každého článku, který zpracoval, do souboru *history*, s nímž jsou pak porovnávány všechny nově příchozí články.

Tok článků mezi dvěma místy může být omezen dvěma kritérii: zaprvé může být článku přiřazena takzvaná distribuce (v hlavičkovém poli `Distribution:`), která může omezit jeho doručení jen do určitých skupin míst. Kromě toho mohou být předávané skupiny news omezeny odesílatelem i příjemcem. Sada diskusních skupin a distribucí, které mají být přenášeny na okolní systémy, je obvykle uchovávána v souboru *sys*.

Velký počet článků zpravidla vyžaduje zdokonalení výše uvedeného schématu. V sítích UUCP je přirozené sesbírat články za určité období, zkombinovat je do jednoho souboru, ten potom zkomprimovat a poslat vzdálenému systému. Této proceduře říkáme *dávkování*.

Jinou alternativou je protokol *ihave/sendme*, který v prvé řadě zabraňuje přenosu duplicitních článků, čímž šetří přenosové kapacity. Místo aby umístil do dávkových souborů všechny články a poslal je dál, zkombinuje pouze identifikátory zpráv článků do jedné velké zprávy „ihave“ a pošle ji vzdálenému systému. Ten si ji přečte, porovná její obsah s obsahem souboru history a v souboru „sendme“ vrátí seznam článků, které požaduje. Pak mu jsou poslány pouze tyto články.

Samozřejmě, že protokol ihave/sendme má smysl jen u dvou velkých systémů, z nichž každý získává news z několika nezávislých zdrojů, které si vyměňují nové zprávy dostatečně často, aby celý mechanismus mohl být efektivní. Systémy, které jsou připojeny k Internetu, obecně spoléhají na TCP/IP software založený na NNTP (Network News Transfer Protocol). Protokol NNTP je definován v RFC 977 a zodpovídá za přenos news mezi newsservery a za přístup k Usenetu uživatelům na vzdálených hostitelích.

Protokol NNTP definuje tři různé způsoby přenosu news. Jedním z nich je verze real-time verze protokolu ihave/sendme, která se nazývá *pushing*. Druhým způsobem je metoda *pulling*, při které klient požaduje seznam článků v dané diskusní skupině nebo hierarchii, které dorazily na příslušný server po specifikovaném datu, a vybere si ty, jež nenajde ve svém souboru history. Třetí metoda slouží k interaktivnímu čtení news a dovoluje vám nebo vašemu prohlížeči news získávat články ze specifikovaných diskusních skupin, stejně jako posílat články s neúplnými hlavičkovými informacemi.

Na každém serveru jsou news uchovávány v adresářové hierarchii `/var/spool/news`, každý článek je uložen v samostatném souboru a každá diskusní skupina má svůj vlastní adresář. Název adresáře je odvozen z názvu diskusní skupiny, přičemž jednotlivé komponenty tohoto názvu tvoří cestu. Takto budou články ze skupiny **comp.os.linux.misc** uloženy v adresáři `/var/spool/news/comp/os/linux/misc`. Článkům jsou v jednotlivých skupinách přidělována čísla podle pořadí, v jakém přišly. Toto číslo slouží jako název souboru. Rozsah čísel článků, které jsou aktuálně k dispozici, je uchováván v souboru *active*, který současně slouží jako seznam diskusních skupin známých ve vašem systému.

Jelikož je diskový prostor omezený, musíte po určité době články mazat[147]. Tomuto mechanismu říkáme *expiring*. Doba platnosti článků z určitých skupin a hierarchií zpravidla vyprší po pevném počtu dnů od jejich doručení. Toto pravidlo může odesílatel potlačit za pomoci pole `Expires:` v hlavičce článku, ve kterém uvede datum vypršení platnosti.

147 Zlí jazykové tvrdí, že Usenet je spiknutím výrobců pevných disků a modemů.

Z předchozího přehledu jste si mohli vytvořit představu, co byste měli číst dále. Uživatelé protokolu UUCP by si měli přečíst o programu C News v kapitole 21. Pokud používáte protokol TCP/IP, přečtěte si o NNTP v kapitole 22. Pokud hodláte přenášet přiměřený počet zpráv mezi TCP/IP systémy, může vám vyhovovat server popsaný v této kapitole. Pokud chcete instalovat velmi zatížený server přenášející značné objemy zpráv, přečtěte si o InterNet News v kapitole 23.

C News

Jedním z nejoblíbenějších softwarových balíků pro síťové news je balík C News. Byl navržen pro systémy, které přenášejí news prostřednictvím protokolu UUCP. Tato kapitola se zabývá základními vlastnostmi C News a základy instalace a údržby tohoto balíku.

Program C News ukládá své konfigurační soubory do adresáře `/etc/news` a většina jeho binárních souborů je umístěna v adresáři `/usr/lib/news/`. Články jsou uchovávány pod adresářem `/var/spool/news`. Měli byste se ujistit, že všechny soubory v těchto adresářích jsou vlastněny uživatelem **news** nebo skupinou **news**. Většina problémů totiž souvisí s tím, že systém C News nemá přístup k příslušným souborům. Zvykněte si, že dříve než se čehokoliv v tomto adresáři dotknete, musíte se příkazem **su** přihlásit jako uživatel **news**. Jedinou výjimkou je program **setnewsids**, který slouží k nastavování skutečných uživatelských identifikátorů některých programů news. Vlastníkem těchto programů musí být uživatel **root** a musí mít nastaven setuid bit.

V této kapitole si podrobně popíšeme všechny konfigurační soubory systému C News a řekneme si, co je třeba udělat pro to, aby váš systém správně fungoval.

Doručování news

Články lze do systému C News dodávat několika způsoby. Když pošle článek místní uživatel, program pro čtení news (newsreader) ho obvykle předá příkazu **inews**, který doplní hlavičkové informace. Články ze vzdálených systémů, ať už se jedná o jediný článek nebo o celou dávku článků, jsou předány příkazu **rnews**, který je uloží do adresáře `/var/spool/news/in.coming`, odkud si je později vyzvedne program **newsrun**. Při použití obou výše zmíněných postupů však bude článek nakonec předán programu **relaynews**.

U každého článku příkaz **relaynews** nejprve vyhledá identifikátor zprávy v souboru `history`, čímž zjistí, zda již místní systém daný článek zná. Duplicitní články zruší. Potom se program **relaynews** podívá do hlavičky na řádek `Newsgroups:`, aby zjistil, zda daný systém přijímá články z některé z těchto skupin. Pokud ano a příslušná diskusní skupina je uvedena v souboru `active`, pokusí se program příslušný článek uložit do odpovídajícího adresáře v adresáři spool. Pokud takový adresář neexistuje, pak ho vytvoří. Identifikátor zprávy následně zaznamená do souboru `history`. V opačném případě program **relaynews** článek smaže.

Pokud se programu **relaynews** nepodaří uložit příchozí článek, protože skupina, do které byl zaslán, není uvedena v souboru `active`, umístí článek do skupiny **junk**[148]. Program **relaynews** také vyhledá staré články nebo články se špatným datem a zruší je. Příchozí dávky článků, jejichž zpracování z nějakého důvodu selže, umístí do adresáře `/var/spool/news/in.coming/bad` a do logovacího souboru zapíše chybu.

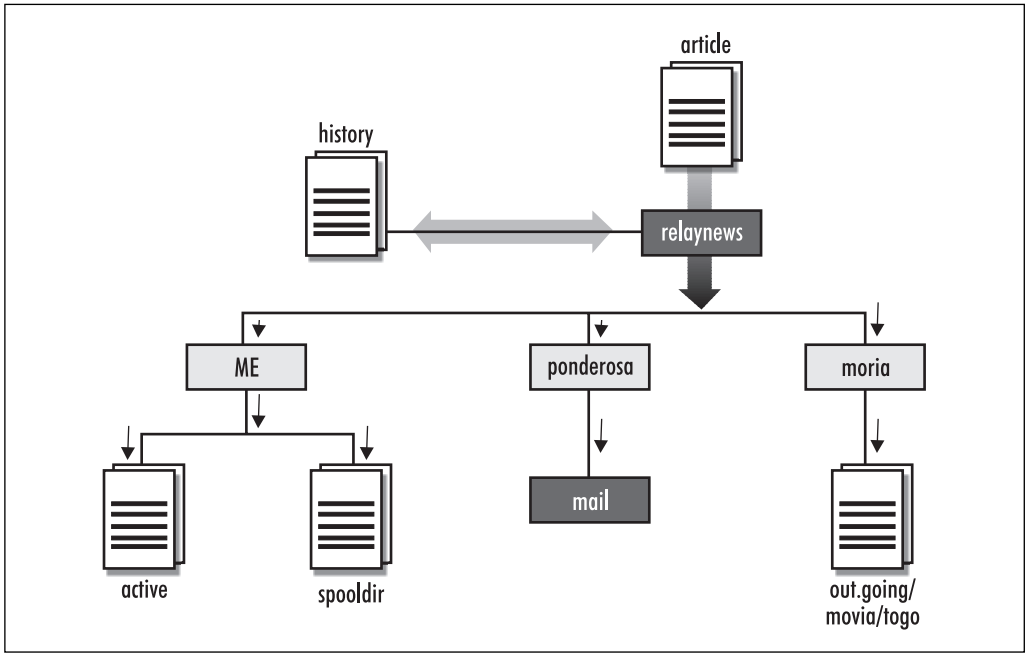
148 Rozdíl může být v tom, které skupiny na vašem systému existují a které skupiny chce systém přijímat. Například na seznamu odebíraných skupin může být uvedeno comp.all, takže budete dostávat všechny skupiny v hierarchii comp, nicméně ne všechny ze skupin pod comp budete mít uvedeny v souboru `active`. Články z těchto skupin se budou ukládat ve skupině **junk**.

Poté je článek přesměrován na všechny ostatní systémy, které požadovaly news z těchto skupin, přičemž se využije přenos specifikovaný pro konkrétní místo. Aby nebyl článek posílán serveru, který ho již viděl, je každé místo určení porovnáno s hlavičkovým polem `Path:`, které obsahuje seznam míst (zapsaný ve vykřičníkové notaci, viz kapitola 17), přes která článek zatím prošel. Příslušný článek mu bude poslán pouze v případě, že se název cílového místa nevyskytuje v tomto seznamu.

Systém C News je běžně používán k předávání news mezi systémy, které využívají protokol UUCP. Lze ho však používat i v prostředí NNTP. Pro doručování news (buď jednotlivých článků, nebo celých dávek) na vzdálený UUCP systém se za pomoci **uux** spustí na vzdáleném systému program **rnews** a tomu se na standardní vstup pošle článek nebo dávka. Podrobnosti o činnosti protokolu UUCP najdete v kapitole 16.

Termínem „dávkové zpracování“ rozumíme zasílání více zpráv najednou v jednom balíku. Je-li pro daný systém povoleno dávkové zpracování, nepošle systém C News příchozí článek okamžitě, ale připojí název jeho cesty do souboru, který se většinou nazývá `out.going/site/togo`. Periodicky je démonem **cron** spouštěn program pro vytváření dávek, který umístí články do jednoho nebo více souborů, případně je i zkomprimuje a pošle je programu **rnews** na vzdáleném systému[149].

Tok news zprostředkovaný programem **relaynews** ukazuje obrázek 21.1. Články mohou být přenášeny do lokálního systému (označeného jako *ME*) a prostřednictvím elektronické pošty do systému nazvaného **ponderosa** nebo místa nazvaného **moria**, pro které je zapnuto dávkové zpracování.



Obrázek 21.1 – Tok news zprostředkovaný programem relaynews

149 Záznam by měl být v **crontab** tabulce uživatele **news**, aby nedošlo k poškození vlastnictví souborů.

Instalace

U moderních distribucí by měl být program C News připraven k automatické instalaci, takže instalace bude velmi jednoduchá. Pokud tomu tak není, nebo pokud jej chcete nainstalovat ze zdrojové distribuce, samozřejmě to lze udělat také[150]. Ať už instalaci provedete jakkoliv, budete muset upravit konfigurační soubory programu. Jejich formát je popsán v následujícím seznamu.

`sys` V tomto souboru je uvedeno, které skupiny váš systém přijímá a předává. Budeme o něm podrobněji hovořit v samostatné části.

`active` Obvykle se neupravuje, obsahuje pravidla pro zpracování článků v jednotlivých přijímaných skupinách.

`organization` Název vaší organizace, například „Virtual Brewery, Inc“. Na domácím počítači zadejte něco na způsob „private site“. Pokud tento soubor nenastavíte správně, většina systémů bude váš počítač považovat za nesprávně nakonfigurovaný.

`newsgroups` Tento soubor obsahuje seznam všech skupin, u každé z nich je uveden její jednořádkový popis. Tyto popisy často využívají programy pro čtení zpráv k zobrazení seznamu skupin, k nimž jste přihlášeni.

`mailname` Poštovní adresa vašeho systému, tj. například **vbrew.com**.

`whoami` Název vašeho systému pro účely news. Často se používá UUCP-název místa, například **vbrew**.

`explist` Tento soubor byste asi měli upravit, protože definuje dobu platnosti některých speciálních diskusních skupin. Zde může hrát důležitou roli i disková kapacita.

Počáteční hierarchii skupin vytvoříte tak, že si ze systému, který vás zásobuje novými news, obstaráte soubory `active` a `newsgroups` a nainstalujete je do adresáře `/etc/news`. Také se ujistěte, že je jejich vlastníkem uživatel news a mají nastavena přístupová práva 644. Ze souboru `active` odstraňte všechny skupiny **to.*** a doplňte skupiny **to**.*mysite*, **to**.*feedsite*, **junk** a **control**. Skupiny **to.*** slouží k výměně zpráv protokolem ihave/sendme, ale měli byste jej vytvořit bez ohledu na to, zda s použitím tohoto protokolu počítáte, či nikoli. Dále nahraďte pomocí následujícího příkazu všechna čísla článků ve druhém a třetím poli v souboru `active`:

```
# cp active active.old
# sed 's/ [0-9]* [0-9]* / 0000000000 00001 /' active.old > active
# rm active.old
```

Druhý příkaz spouští editor **sed**. V tomto případě nahradí dva řetězce číslic řetězcem nul a řetězcem `000001`.

Nakonec vytvoříme adresář `spool` a jeho podadresáře používané pro ukládání příchozích a odchozích news:

```
# cd /var/spool
# mkdir news news/in.coming news/out.going news/out.master
# chown -R news.news news
# chmod -R 755 news
```

150 Zdrojovou distribuci balíku C News najdete na domovském serveru ftp.cs.toronto.edu v adresáři `/pub/c-news/c-news.tar.Z`.

Jestliže používáte pro čtení news jiný program, než jaký je dodáván se systémem C News, může se vám stát, že bude zprávy očekávat v `/usr/spool/news` a ne v `/var/spool/news`. Nenajde-li žádné články, vytvořte `/usr/spool/news` jako symbolický odkaz na `/var/spool/news`:

```
# ln -sf /usr/spool/news /var/spool/news
```

Nyní jste připraveni na příjem news. Všimněte si, že nemusíte vytvářet adresáře pro jednotlivé přijímané skupiny, protože když systém C News obdrží článek patřící do skupiny, pro kterou ještě neexistuje příslušný adresář, vytvoří jej automaticky.

Platí to pro *všechny* skupiny, pro něž se předávají zprávy. Po nějaké době proto zjistíte, že váš adresář `spool` bude přeplněn adresáři s názvy skupin, ke kterým jste se nikdy nepřihlásili, například **alt.lang.teco**. Tomu zabráníte tak, že buď odstraníte ze souboru `active` všechny nechtěné skupiny, nebo budete pravidelně spouštět skript, který odstraní všechny prázdné adresáře pod adresářem `/var/spool/news` (samozřejmě vyjma `out.going` a `in.coming`).

Systém C News potřebuje nějaký uživatelský účet, na který by mohl posílat chybové a stavové zprávy. Implicitně se tento účet nazývá **usenet**. Využijete-li toto implicitní nastavení, musíte nastavit alias, který by veškerou poštu došlou na tento účet přesměroval na odpovědné osoby. Toto chování lze také potlačit pomocí proměnné prostředí *NEWSMASTER*, které přiřadíte příslušný název účtu. Proměnnou je třeba nastavit v souboru `crontab` uživatele **news**, stejně jako při každém manuálním spuštění nějakého administrativního nástroje, takže zřízení aliasu je zřejmě jednodušší. Zřizování poštovních aliasů je popsáno v kapitolách 18 a 19.

Při úpravě souboru `/etc/passwd` se ujistěte, že každý uživatel má v poli *pw_gecos* (jde o čtvrté pole) zapsáno svoje skutečné jméno. Patří k etiketě sítě, že se v poli `From:` každého článku objeví skutečné jméno odesílatele. Stejně to musíte udělat, když budete používat elektronickou poštu.

Soubor sys

Soubor `sys`, umístěný v adresáři `/etc/news`, určuje, které skupiny přijímáte a předáváte dále. I když existují speciální nástroje pro práci s tímto souborem, programy **addfeed** a **delfeed**, myslíme si, že je lepší upravovat tento soubor ručně.

Soubor `sys` obsahuje záznam pro každý systém, kterému předáváte news, stejně jako popis skupin, které přijímáte. Prvním záznamem je údaj ME, popisující váš systém. Nejlepší je nastavit jej takto:

```
ME:all/all::
```

Kromě toho musíte přidat záznam pro každý systém, kterému budete news předávat. Jednotlivé řádky vypadají takto:

```
site[/exclusions]:grouplist[/distlist][:flags[:cmds]]
```

Pokud bude záznam pokračovat na více řádcích, ukončují se jednotlivé řádky obráceným lomítkem. Znakem # se uvozují komentáře.

`site` Název systému, kterého se záznam týká. Zpravidla se použije UUCP název příslušného místa. V souboru musí mít záznam i váš systém, jinak byste nedostávali žádné články. Váš systém se označuje speciálním názvem ME. Tento záznam definuje všechny skupiny, které chcete mít uloženy lokálně. Články, které nevyhovují řádku se záznamem ME, putují do skupiny **junk**. Aby se zabránilo vzniku smyček, C News zahazují všechny články, které systémem prošly. Zajišťují to tak, že hledají

název systému v hlavičce `Path:` článku. Některé systémy se mohou objevovat pod různými názvy, například uvádějí plně kvalifikované doménové jméno, nebo alias typu `news.`*`site.domain`*. Aby se předešlo vzniku smyček, musí být v poli `exclusions` uvedeny všechny používané aliasy, oddělené čárkami. V záznamu týkajícím se systému **moria** by mohlo příslušné pole obsahovat například řetězec **moria/moria.ocnet.org**. Pokud by systém **moria** používal rovněž alias **news.orcnet.org**, měli byste v souboru `sys` uvést **moria/moria.orcnet.org,news. orcnet.org**.

`grouplist` Toto je čárkami oddělený seznam skupin a hierarchií, které si zapsal konkrétní systém. Hierarchii je možné specifikovat prostřednictvím předpony (například **comp.os** pro všechny skupiny, jejichž název začíná touto předponou), za níž může volitelně následovat klíčové slovo **all** (tedy **comp.os.all**). Předání příslušné hierarchie nebo skupiny zabráníte tak, že před její název přidáte vykřičník. Při porovnávání názvu diskusní skupiny s tímto seznamem má nejvyšší prioritu nejdelší shoda. Když například pole *grouplist* obsahuje `!comp,comp.os.linux,comp.folklore.computers` nebudou příslušnému systému předány žádné skupiny z `comp` s výjimkou skupiny `comp.folklore.computers` a všech podskupin skupiny `comp.os.linux`. Pokud daný systém požaduje všechny news, které jste sami obdrželi, zadejte do pole *grouplist* údaj *all*.

`distlist` Tento údaj je oddělen od pole *grouplist* lomítkem a obsahuje seznam distribucí, které se mají předávat dále. I v tomto případě můžete některé distribuce vyřadit tím, že před ně umístíte vykřičník. Všechny distribuce označuje řetězec *all*. Pokud toto pole vynecháte, znamená to, že jste zvolili *all*. Můžete například použít seznam distribucí *all,!local*, čímž zabráníte šíření news, které jsou určeny jen pro lokální použití. Toto pole většinou obsahuje minimálně dvě distribuce: *world*, což je obvykle používaná implicitní distribuce, není-li uvedeno jinak, a *local*. Existují také další distribuce, které se týkají určité oblasti, státu, země a podobně. Nakonec zde máme dvě distribuce, které používá pouze systém C News. Jsou to *sendme* a *ihave* a využívá je protokol ihave/sendme. Využití distribucí je předmětem stálých sporů. Údaj `Distribution` v článku je uveden povinně, aby však měl smysl, musí mu server zpráv rozumět. Některé chybně se chovající programy vytvářejí nesmyslné distribuce díky tomu, že předpokládají, že nejvyšší úroveň v hierarchii článků zároveň představuje platnou distribuci. Předpokládá se například, že *comp* je platná distribuce při zadávání článku do skupiny *comp.os.linux.networking*. Sporné jsou i regionální distribuce, protože při šíření článku prostřednictvím Internetu může článek snadno opustit hranice regionu[151]. Smysl dávají naopak distribuce platné pro určitou organizaci – zabrání se tak, aby interní články firmy opustily její systém. K tomu se však dá lépe využít vlastní diskusní skupina nebo hierarchie.

`flags` Toto pole popisuje parametry přenosu. Může být prázdné nebo může obsahovat kombinaci následujících znaků: `F` – Tento příznak povoluje dávkové zpracování. `f` – Má téměř stejný význam jako `F`, umožňuje však přesnější výpočet velikosti dávek a měl by být používán přednostně. `I` – Tento příznak způsobí, že C News budou vytvářet seznam článků používaný protokolem ihave/sendme. Aby bylo možné tento protokol použít, jsou nutné další úpravy v souborech `sys` a `batchparms`. `n` – Vytváří dávkové soubory pro přenosové klienty využívající protokol NNTP, jako například **nntpxmit** (viz kapitola 22). Dávkové soubory obsahují název souboru článku a id

151 Není neobvyklé, aby článek vytvořený řekněme v Hamburku putoval do Frankfurtu přes systém reston.ans.net v Holandsku, nebo dokonce přes nějaký systém ve Spojených Státech.

zprávy. `L` – Tento příznak říká systému C News, aby přenášel pouze články vytvořené na vašem systému. Příznak může být následován desítkovým číslem *n*, které způsobí, že C News budou přenášet pouze články, které na váš systém dorazili přes maximálně *n* skoků. Počet skoků se určuje z pole `Path:`. `u` – Systém C News bude vytvářet dávky pouze z článků v nemoderovaných skupinách. `m` – Systém C News bude vytvářet dávky pouze z článků v moderovaných skupinách. Z parametrů `F`, `f`, `I` a `n` může být použit pouze jeden.

`cmds` Toto pole obsahuje příkaz, který se provede nad každým článkem, pokud není povoleno dávkové zpracování. Příslušný článek bude předán na standardní vstup příkazu. Tuto možnost byste měli využívat jen u malých dávek news, protože jinak by došlo k velkému zatížení obou systémů. Implicitní příkaz má formát `uux - -r -z` *`vzdálený_systém`*`!rnews Spustí tedy na vzdáleném systému program` **`rnews`** a předá mu článek na standardní vstup. Implicitní vyhledávací cesta pro příkazy předané v tomto poli je `/bin:/usr/bin:/usr/lib/news/bin/batch`. Poslední z těchto adresářů obsahuje několik skriptů příkazového interpretu, jejichž názvy začínají textem **via**. Podrobněji si je popíšeme dále. Je-li povoleno dávkové zpracování jedním z parametrů `F` nebo `f`, `I` nebo `n`, bude systém C News očekávat, že najde v tomto poli název souboru a ne příkaz. Pokud název souboru nezačíná lomítkem, předpokládá se, že jde o relativní cestu k adresáři `/var/spool/news/out.going`. Je-li toto pole prázdné, vezme se implicitní cesta *`vzdálený_systém`*`/togo`. Předpokládá se, že soubor bude mít stejný formát jako *`vzdálený_systém`*`/togo` a že obsahuje seznam článků k přenesení.

Při nastavování systému C News budete zřejmě muset vytvořit svůj vlastní soubor `sys`. Následující příklad představuje tento soubor ze systému **vbrew.com**, který si můžete upravit podle vašich požadavků.

```
# Bereme vše, co nám ostatní poskytnou.
ME:all/all::
# Vše co dostaneme posíláme na moria kromě lokálních článků.
# Používáme dávkový přenos.
moria/moria.orcnet.org:all,!to,to.moria/all,!local,!brewery:f:
# Skupinu comp.risks posíláme na adresu jack@ponderosa.uucp
ponderosa:comp.risks/all::rmail jack@ponderosa.uucp.
# Hostitel swim dostává několik skupin.
swim/swim.twobirds.com:comp.os.linux,rec.humor.oracle/all,!local:f:
# Zaznamenávej poštovní mapy pro pozdější zpracování.
usenet-maps:comp.mail.maps/all:F:/var/spool/uumaps/work/batch
```

Soubor active

Soubor `active` najdete v adresáři `/etc`. Obsahuje seznam všech skupin, které zná váš systém, a všech článků, které jsou zde v současné době uloženy. Jen výjimečně ho budete přímo upravovat, ale kvůli úplnosti si ho zde popíšeme. Záznamy v tomto souboru mají následující formát:

```
newsgroup high low perm
```

Pole *newsgroup* je samozřejmě název skupiny. Pole *low* a *high* udávají nejnižší a nejvyšší čísla článků, které jsou právě dostupné. Není-li dostupný žádný článek, je pole *low* rovno *high+1*.

Přinejmenším by takto mělo pole *low* fungovat. Nicméně systém C News kvůli efektivitě toto pole neaktualizuje. To by nebylo tak zlé, kdyby na něm nezávisely některé programy pro prohlížení news. Například program **trn** se na základě tohoto pole rozhoduje, zda může odstranit nějaké články ze své databáze vláken. K aktualizaci pole *low* je proto nutné pravidelně spouštět program **updatemin** (nebo v dřívějších verzích systému C News skript **upact**).

Parametr *perm* řídí přístup uživatelů, kteří mají povolen přístup k dané skupině. Může nabývat jedné z následujících hodnot:

`y` Uživatelé mohou do této skupiny přispívat.

`n` Uživatelé nemohou do této skupiny přispívat. Nicméně mohou z ní číst články.

`x` Tato skupina byla lokálně vypnuta. K tomu občas dochází, když se administrátoři news (nebo jejich nadřízení) kvůli nějakému poslanému článku urazí. Články určené pro tuto skupinu nejsou ukládány lokálně, nicméně jsou předávány dále systémům, které je požadují.

`m` Tato hodnota označuje moderovanou skupinu. Pokusí-li se uživatel zaslat do této skupiny nějaký článek, inteligentní program pro čtení news ho na to upozorní a místo do skupiny ho pošle moderátorovi. Adresu moderátora zjistí ze souboru `moderators`, který najde v adresáři `/var/lib/news`.

`=`*`skupina`* Takto označená skupina je lokálním aliasem jiné skupiny. Všechny články zaslané do této skupiny budou přesměrovány do skutečné skupiny.

V systému C News nebudete muset přímo pracovat s tímto souborem. Skupiny lze lokálně přidávat i odstraňovat pomocí nástrojů **addgroup** a **delgroup** (viz část *Nástroje pro údržbu*). Je-li do Usenetu přidána nebo z něj naopak odstraněna nějaká skupina, dozví se o tom všechny systémy na základě řídicí zprávy *newgroup* nebo *rmgroup*. Tento typ zprávy nikdy sami neposílejte! Máte-li zájem o podrobnosti týkající se vytváření diskusních skupin, pak si přečtěte měsíční příspěvky do konference **news.anounce.newusers**.

Se souborem `active` je úzce spjat soubor `active.times`. Kdykoliv je vytvořena nová skupina, zaznamená systém C News do tohoto souboru zprávu, která obsahuje název vytvořené skupiny, datum vytvoření, zda se tak stalo na základě zprávy *newgroup* nebo lokálně a kdo tak učinil. To kvůli programům pro čtení news, které pak mohou informovat uživatele o každé nově vytvořené skupině. Tuto informaci využívá také příkaz *NEWGROUPS* protokolu NNTP.

Dávkové zpracování článků

Dávky news dodržují stejný formát pro Bnews, C News a INN. Každému článku předchází následující řádek:

```
#! rnews count
```

kde *count* je počet bajtů článku. Použije-li se komprese dávek, je výsledný soubor zkomprimován jako celek a na začátek je umístěn další řádek, který udává příkaz, jenž se použije pro rozbalení. Standardním kompresním nástrojem je compress, který je označen zprávou

```
#! cunbatch
```

Někdy, když je potřeba poslat dávky za pomoci poštovního programu, který ze všech dat odstraňuje osmý bit, je možné zkomprimovanou dávku chránit prostřednictvím takzvaného c7-kódování. Takovéto dávky budou označeny **c7unbatch**.

Když je dávka na vzdáleném systému předána programu **rnews**, vyhledá tyto značky a dávku příslušným způsobem zpracuje. Některé systémy používají ještě jiné kompresní nástroje, například **gzip**, a před takto zkomprimované soubory předřazují zprávu **zunbatch**. Systém C News těmto nestandardním hlavičkám nerozumí. Má-li je podporovat, je třeba upravit zdrojový kód.

V systému C News je dávkování článků prováděno programem `/usr/lib/news/batch/sendbatches`, který vezme seznam článků ze souboru `system/togo` a umístí je do několika dávek. Měl by být spouštěn jednou za hodinu případně ještě častěji, v závislosti na objemu provozu. Jeho práci řídí soubor `batchparms` umístěný v adresáři `/var/lib/news`. Tento soubor popisuje maximální velikost dávky pro každý systém, dávkovací a volitelně i komprimační program, který se má použít, a přenos, který zajistí doručení příslušné dávky vzdálenému systému. Dávkovací parametry je možné zadat pro každý systém jednotlivě nebo jako sadu implicitních parametrů, použitou pro systémy, které nejsou uvedeny explicitně.

Při instalaci systému C News pravděpodobně zjistíte, že součástí distribuce je i soubor `batchparms`, který obsahuje rozumné implicitní záznamy, takže zřejmě tento soubor nebudete muset upravovat. Jeho formát si zde popíšeme jen pro úplnost. Každý řádek je složen ze šesti polí, které jsou navzájem odděleny mezerami nebo tabulátory:

`site size max batcher muncher transport`

`site` Udává název systému, kterého se daný záznam týká. Pro daný systém musí pod `/out.going/togo` existovat soubor `togo`. Název `/default/` označuje implicitní záznam.

`size` Udává maximální velikost vytvořených dávek článků (před kompresí). Je-li velikost jednoho článku větší než tento údaj, povolí systém C News výjimku a umístí ho do samostatné dávky.

`max` Udává maximální počet dávek vytvořených a naplánovaných pro přenos, než je dávkování pro daný systém zrušeno. To je užitečné v případě, že vzdálený systém dlouhou dobu nefunguje, protože se tak vyhnete zahlcení UUCP front miliony dávek news. Systém C News určuje počet čekajících dávek za pomoci skriptu **queuelen**, který najdete v adresáři `/usr/lib/news`. Pokud si nainstalujete standardní distribuci C News, nebudete muset skript nijak upravovat, pokud ale používáte jiné umístění nebo formát front (například při použití Taylor UUCP), budete jej muset upravit. Pokud vás počet položek ve frontě netrápí (jelikož počítač používáte sami a nepíšete megabajtové zprávy), můžete celé tělo skriptu nahradit jediným příkazem **exit 0**.

`batcher` Obsahuje příkaz, který slouží k vytváření dávky ze seznamu článků v souboru `togo`. Pro účely normálního přenosu je jím zpravidla program **batcher**. Pro jiné účely mohou být poskytovány jiné dávkovací programy. Například protokol ihave/sendme vyžaduje, aby byl seznam článků převeden do řídicích zpráv *ihave* a *sendme*, které jsou pak odeslány do skupiny **to**. To mají na starosti programy **batchih** a **batchsm**.

152 Standardně dodávaný **compcun** používá 12bitovou kompresi, která představuje nejmenší společný komprimační způsob na všech systémech. Můžete vytvořit kopii tohoto skriptu pojmenovanou například compcun16 a nastavit ji pro 16bitovou kompresi, nicméně efekt není nijak výrazný.

`muncher` Specifikuje příkaz, který se použije při kompresi. Zpravidla to bývá skript **compcun**, který vytvoří komprimovanou dávku[232]. Pro stejné účely je možné využít také program **gzip** a spouštět jej skriptem **gzipcun** (který si musíte sami napsat). Je třeba se ujistit, že program **uncompress** na vzdáleném systému je upraven tak, aby rozpoznal soubory komprimované programem **gzip**. Pakliže vzdálený systém nedisponuje příkazem **uncompress**, stačí použít volbu **nocomp**, která kompresi zakáže.

`transport` Popisuje přenos, který se má použít. K dispozici je několik příkazů, jejichž názvy začínají řetězcem **via** a slouží pro různé druhy přenosů. Program **sendbatches** jim předá na příkazové řádce název cílového systému. Pokud nejde o záznam */default/*, odvodí se název systému z pole *site*, přičemž se odstraní veškeré znaky za první tečkou nebo lomítkem včetně. V případě položky */default/* se použijí názvy podadresářů adresáře `out.going`.

Dávkové zpracování pro specifický systém spustíte následujícím příkazem

```
# su news -c "/usr/lib/news/bin/batch/sendbatches systém"
```

Spustíte-li program **sendbatches** bez parametrů, bude obsluhovat veškeré fronty dávek. Interpretace slova „veškeré“ závisí na přítomnosti implicitního záznamu v souboru `batchparms`. Pokud existuje, budou se kontrolovat všechny adresáře v adresáři `/var/spool/news/out.going`. V opačném případě bude program procházet všechny záznamy uvedené v souboru `batchparms`. Všimněte si, že program **sendbatches** bere při procházení adresáře `out.going` v úvahu pouze adresáře, které nemají v názvu tečku nebo @.

Program **uux** používá ke spouštění **rnews** na vzdáleném systému dva příkazy: **viauux** a **viauuxz**. Druhý z nich nastavuje příznak `-z` pro starší verze programu uux, který způsobí, že tento program nebude vracet zprávy o úspěšném doručení každého článku. Jiný příkaz, **viamail**, posílá dávky článků uživateli **rnews** na vzdáleném systému prostřednictvím elektronické pošty. Samozřejmě, že tento vzdálený systém musí veškerou poštu pro uživatele **rnews** přesměrovat do svého lokálního systému news. Úplný seznam všech přenosů naleznete v manuálových stránkách příkazu **newsbatch**.

Všechny příkazy z posledních tří polí musí být umístěny v adresáři `out.going/systém` nebo `/usr/lib/news/batch`. Většina z nich jsou skripty, takže si můžete nové nástroje snadno přizpůsobit svým potřebám. Jsou spouštěny pomocí roury. Seznam článků je předáván na standardní vstup dávkovacího programu, výstupem je příslušná dávka. Ta je předána programu na komprimaci dávek a tak dále.

Takto vypadá příklad souboru:

```
# soubor batchparms
# site         | size    |max     |batcher   |muncher     |transport
#--------------+---------+--------+----------+------------+------------
/default/       100000    22       batcher    compcun      viauux
swim            10000     10       batcher    nocomp       viauux
```

Vypršení platnosti news

V systému B News se vypršení platnosti zpráv zajišťuje programem **expire**, kterému jako parametry předáte seznam diskusních skupin a časový údaj, po jehož uplynutí vyprší platnost článků. Po-

kud chcete, aby různým hierarchiím vypršela platnost v jinou dobu, potřebujete skript, který by pro každou skupinu spouštěl program **expire** samostatně. Systém C News nabízí pohodlnější řešení: v souboru `explist` stačí uvést diskusní skupiny a příslušné časové intervaly. Typicky jednou denně je démonem **cron** spouštěn příkaz **doexpire**, který podle tohoto seznamu zpracuje všechny skupiny.

Někdy si možná budete chtít ponechat některé články z určitých skupin i po uplynutí doby jejich platnosti; například si budete chtít archivovat programy zaslané do skupiny **comp.sources.unix**. Tento postup se označuje jako *archivace*. Soubor `explist` umožňuje označit skupiny, které chcete archivovat.

Záznam v souboru `explist` vypadá následovně:

```
grouplist perm times archive
```

Pole *grouplist* je čárkami oddělený seznam diskusních skupin, kterých se daný záznam týká. Hierarchii skupin zadáte celým názvem skupiny, za nímž může nepovinně následovat řetězec *all*. Například záznam týkající se všech podskupin skupiny **comp.os** můžete v poli *grouplist* uvést jako **comp.os** nebo **comp.os.all**.

Při zjišťování vypršení platnosti news z nějaké skupiny je její název porovnán se všemi záznamy v souboru `explist`. Použije se první vyhovující záznam. Chcete-li například po čtyřech dnech zrušit většinu obsahu diskusní skupiny **comp** s výjimkou skupiny **comp.os.linux.announce**, jejíž obsah si chcete ponechat týden, vytvoříte záznam této skupiny, který bude udávat sedmidenní platnost článků v ní, a teprve za ním bude následovat záznam pro skupinu **comp**, který bude udávat platnost pouze čtyři dny.

Pole *perm* popisuje, zda se příslušný záznam týká moderovaných, nemoderovaných nebo všech skupin. Může nabývat hodnot *m*, *u* nebo *x*, které označují moderované, nemoderované a všechny skupiny.

Třetí pole, *times*, většinou obsahuje pouze jediné číslo. To je počet dnů, po jejichž uplynutí vyprší doba platnosti článků, které ji neměly explicitně nastavenu v poli `Expires:` v hlavičce článku. Jedná se o počet dnů od data, kdy článek dorazil do vašeho systému, nikoliv od data jeho vytvoření.

Pole *times* ovšem může být i složitější. Může obsahovat kombinaci až tří čísel, navzájem oddělených pomlčkou. První určuje počet dnů, které musí uplynout, aby se článek stal kandidátem na vyřazení, dokonce i v případě, že pole `Expires:` udává kratší dobu. Jen zřídka se používá jiná hodnota než nulová. Druhé pole je výše zmiňovaný implicitní počet dnů, po kterém vyprší platnost souboru. Třetí číslo specifikuje počet dnů, po kterých bude článek bezpodmínečně smazán, bez ohledu na to, zda má v hlavičce uvedeno pole `Expires:`. Uvedete-li pouze prostřední číslo, použijí se pro zbylé dvě pole implicitní hodnoty. Ty je možné specifikovat prostřednictvím speciálního záznamu */bounds/*, který bude popsán dále.

Čtvrté pole, *archive*, určuje, zda má být příslušná diskusní skupina archivována a kde. Pokud si archivaci nepřejete, pak zde uveďte pomlčku. V opačném případě zadejte buď úplnou cestu (odkazující na adresář), nebo znak @. Ten zastupuje implicitní archivní adresář, který pak musíte programu **doexpire** předat pomocí parametru `-a` v příkazové řádce. Archivní adresář by měl vlastnit uživatel **news**. Když program **doexpire** archivuje článek řekněme ze skupiny **comp.sources.unix**, uloží ho do adresáře `comp/sources/unix` pod archivním adresářem a pokud neexistuje, vytvoří nový. Vlastní archivní adresář ovšem nevytvoří.

Program **doexpire** spoléhá na dva speciální záznamy ze souboru `explist`. Místo názvů diskusních skupin používají klíčová slova */bounds/* a */expired/*. Záznam */bounds/* obsahuje implicitní hodnoty tří položek pole *times*, které jsme si před chvílí popsali.

Pole */expired/* určuje, jak dlouho bude systém C News uchovávat řádky v souboru `history`. Tato položka je nutná z toho důvodu, že systém C News neodstraní řádek s názvem příslušného článku ze souboru `history` ihned po jeho smazání, ale ponechá ho tam pro případ, že by po tomto datu přišel duplicitní článek. Pokud dostáváte články pouze z jediného systému, může být tato hodnota malá. V ostatních případech se pro sítě založené na protokolu UUCP doporučují použít hodnotu odpovídající dvěma týdnům, v závislosti na zkušenostech s prodlevami článků z těchto systémů.

Vzorový soubor explist s krátkými intervaly vypršení platnosti zde reprodukujeme:

```
# Záznamy v history uchováváme 14 dní, nic nepřežije 3 měsíce.
/expired/                       x     14     -
/bounds/                        x 0-1-90     -
# skupiny, které chceme uchovat déle než ostatní
comp.os.linux.announce          m     10     -
comp.os.linux                   x      5     -
alt.folklore.computers          u     10     -
rec.humor.oracle                m     10     -
soc.feminism                    m     10     -
# Archivovat skupiny *.sources
comp.sources,alt.sources        x      5     @
# Implicitní nastavení technických skupin
comp,sci                        x      7     -
# Stačí i při delším víkendu
misc,talk                       x      4     -
# Rychle zahazujeme junk
junk                            x      1     -
# řídící zprávy taky nejsou zajímavé
control                         x      1     -
# a toto platí pro všechno ostatní
all                             x      2     -
```

V souvislosti s vypršením vyvstává několik potenciálních problémů. Váš program pro čtení news může například spoléhat na třetí pole souboru `active`, které obsahuje číslo nejnižšího článku, který je k dispozici. Při vyřazování článků systém C News toto pole neaktualizuje. Pokud potřebujete (nebo chcete), aby toto pole odráželo aktuální stav, pak musíte po každém spuštění programu **doexpire** spustit i program **updatemin**. (Ve starších verzích balíku C News se místo něj používá skript **upact**.)

Systém C News dále neprovádí kontrolu platnosti souborů procházením adresáře příslušné diskusní skupiny, ale pouze na základě souboru `history`[153]. Budou-li v souboru `history` nesprávné údaje, mohou články zůstat na vašem systému navždy, protože C News na ně prostě zapomenou[154]. Tento problém lze opravit pomocí skriptu **addmissing**, který najdete v adresáři `/usr/lib/news/maint` a který přidá do souboru `history` chybějící články. Jiný skript, **mkhistory**, znovu vytvoří celý soubor `history`. Dříve než tento skript spustíte se nezapomeňte přihlásit jako uživatel **news**, jinak získáte soubor `history`, který bude pro systém C News nečitelný.

153 Doba, kdy článek dorazil, se ukládá jako prostřední údaj na řádcích v souboru `history` a uvádí se v sekundách od 1. ledna 1970.

154 Nevíme *proč*, ale občas se to stává.

Různé soubory

Chování systému C News řídí i další soubory, ty však nejsou nezbytné. Všechny jsou uloženy v adresáři `/etc/news`. Nyní si je krátce popíšeme.

`newsgroups` Tento soubor je doprovodným souborem k souboru `active` a obsahuje seznam názvů diskusních skupin společně s popisem jejich hlavního tématu. Jeho obsah je automaticky aktualizován, jakmile systém C News obdrží řídicí zprávu *checknews*.

`localgroups` Máte-li více lokálních skupin a nechcete, aby se systém C News ozýval pokaždé, když obdrží zprávu *checknews*, umístěte jejich názvy a popisy do tohoto souboru stejným způsobem, jak byste je zařadili do souboru newsgroups.

`mailpaths` Tento soubor obsahuje adresy moderátorů všech moderovaných skupin. Každý řádek obsahuje název skupiny následovaný poštovní adresou moderátora (odsazenou tabulátorem). Implicitní jsou dva speciální záznamy – *backbone* a *internet*. Oba označují (ve vykřičníkové notaci) cestu k nejbližšímu páteřnímu místu a systému, který rozumí adrese podle standardu RFC 822 (**uživatel@hostitel**). Implicitními záznamy jsou `internet backbone` Máte-li nainstalován jeden z programů `exim` nebo `sendmail`, pak nemusíte záznam *internet* měnit, protože tyto programy rozumí adresování podle standardu RFC 822. Záznam *backbone* se využije, když uživatel přispívá do moderované skupiny, jejíž moderátor není explicitně uveden. Je-li například název skupiny **alt.sewer** a záznam *backbone* obsahuje řetězec `path!%s`, pošle systém C News tento článek na adresu `path!alt-sewer` a doufá, že jej bude uvedený systém schopen správně obsloužit. Jakou máte použít cestu zjistíte od správce news na systému, který vás jimi zásobuje. Jako poslední útočiště můžete také použít adresu `uunet.uu.net!%s`.

`distributions` Tento soubor ve skutečnosti nepatří k systému C News, ale používají ho některé programy pro čtení news a **nntpd**. Obsahuje seznam distribucí, které rozpozná váš systém, a popis jejich (zamýšleného) efektu. Například společnost Virtual Brewery používá soubor, který obsahuje následující záznamy:

```
world
    everywhere in the world  local
    Only local to this site  nl
    Netherlands only  mugnet
MUGNET only  fr
    France only  de
    Germany only  brewery
Virtual Brewery only
```

`log` Tento soubor obsahuje záznam všech aktivit systému C News. Pravidelně ho vybírá program **newsdaily**. Kopie starších logovacích souborů jsou pojmenovávány `log.o`, `log.oo` a tak dále.

`errlog` Do tohoto souboru jsou zaznamenávány všechny chybové zprávy vyprodukované systémem C News. To se netýká článků, které byly odhozeny kvůli špatné skupině a podobně. Když program **newsdaily** zjistí, že je tento soubor neprázdný, automaticky ho pošle správci news (implicitně na účet **usenet**). Soubor `errlog` vyprazdňuje program **newsdaily**. Starší kopie tohoto souboru jsou uchovávány pod názvy typu `errlog.o`.

`batchlog` Sem jsou zaznamenávána veškerá spuštění programu **sandbatches**. Tento soubor má zpravidla jen malý význam. Udržuje jej také program **newsdaily**.

`watchtime` Jedná se o prázdný soubor, který se vytvoří vždy při spuštění programu **newswatch**.

Řídicí zprávy

Protokol usenetových news rozeznává speciální kategorii článků, které ze strany systému news vyvolávají jisté odpovědi nebo akce. Takovýmto článkům říkáme *řídicí zprávy*. Poznáte je podle přítomnosti pole `Control:` v hlavičce článku, které obsahuje název řídicí operace, která se má provést. Existuje jich několik typů a všechny jsou obsluhovány skripty umístěnými v adresáři `/usr/lib/news/ctl`.

Většina z nich provede příslušnou akci automaticky v době, kdy je článek zpracován systémem C News, aniž by o tom informovaly správce news. Implicitně jsou správci předány pouze zprávy *checkgroups*, ale toto chování lze úpravou příslušných skriptů změnit.

Zpráva cancel

Nejznámějším typem zprávy je zpráva *cancel*, s jejíž pomocí může uživatel zrušit článek, který dříve poslal. Pakliže tento článek dosud existuje, zajistí tato zpráva jeho odstranění z adresářů *spool*. Zpráva *cancel* je předána všem systémům, které dostávají news z příslušné skupiny, bez ohledu na to, zda příslušný článek již viděli či nikoli. To pro případ, že by se původní článek opozdil za rušící zprávou. Některé systémy news dovolují uživatelům rušit zprávy jiných osob, což by samozřejmě nemělo být v žádném případě možné.

Zprávy newgroup a rmgroup

Tyto dvě zprávy se týkají vytváření a rušení diskusních skupin. Nová skupina může být v „obvyklé“ hierarchii vytvořena až poté, co to odsouhlasí uživatelé Usenetu. Pravidla týkající se hierarchie skupiny **alt** dovolují vzniknout něčemu, co se hodně podobá anarchii. Více podrobností najdete ve skupinách **news.announce.newusers** a **news.announce.newgroups**. Nikdy neposílejte zprávu *newgroup* nebo *rmgroup*, pokud bezpodmínečně nevíte, že k tomu máte oprávnění.

Zpráva checkgroups

Zprávy *checkgroups* posílají správci news, aby všechny systémy v síti provedly vzájemnou synchronizaci svých souborů `active` s realitou Usenetu. Komerční poskytovatelé internetových služeb by mohli například poslat tuto zprávu svým zákazníkům. Jednou měsíčně pošle moderátor skupiny **comp.announce.newgroup** „oficiální“ zprávu *checkgroups* pro hlavní hierarchie. Tato zpráva je ovšem poslána jako běžný článek, nikoliv jako řídicí zpráva. Chcete-li provést operaci *checkgroup*, uložte tento článek do souboru, řekněme `/tmp/check`, odstraňte z něj vše kromě vlastní řídicí zprávy a předejte ho následujícím způsobem skriptu **checkgroups**:

```
# su news -c "/usr/lib/news/ctl/checkgroups"  < /tmp/check
```

Tento příkaz aktualizuje váš soubor `newsgroups`, do kterého přidá skupiny uvedené v souboru `localgroups`. Původní soubor `newsgroups` bude přejmenován na `newsgroups.bac`. Všimněte si, že pošlete-li tuto zprávu lokálně, bude jen zřídka fungovat, protože program **inews** odmítne tak velký článek přijmout.

Když systém C News zjistí rozpory mezi seznamem *checkgroups* a souborem `active`, vytvoří seznam příkazů, které zajistí aktualizaci vašeho systému, a pošle ho správci news. Výstup má zpravidla následující podobu:

```
From news Sun Jan 30 16:18:11 1994
Date: Sun, 30 Jan 94 16:18 MET
From: news (News Subsystem)
To: usenet
Subject: Problems with your active file
The following newsgroups are not valid and should be removed.
        alt.ascii-art
        bionet.molbio.gene-org
        comp.windows.x.intrisics
        de.answers
You can do this by executing the commands:
         /usr/lib/news/maint/delgroup alt.ascii-art
         /usr/lib/news/maint/delgroup bionet.molbio.gene-org
         /usr/lib/news/maint/delgroup comp.windows.x.intrisics
         /usr/lib/news/maint/delgroup de.answers
The following newsgroups were missing.
        comp.binaries.cbm
        comp.databases.rdb
        comp.os.geos
        comp.os.qnx
        comp.unix.user-friendly
        misc.legal.moderated
        news.newsites
        soc.culture.scientists
        talk.politics.crypto
        talk.politics.tibet
```

Pokud od systému news obdržíte zprávu tohoto typu, pak jí slepě nevěřte. V závislosti na tom, kdo poslal zprávu *checkgroups*, může postrádat několik skupin nebo i celé hierarchie. Skupiny byste tedy měli odstraňovat jen opatrně. Pokud zjistíte, že některé skupiny, jež chcete vést ve vašem systému, chybí, pak je musíte doplnit pomocí skriptu **addgroup**. Uložte seznam chybějících skupin do souboru a předejte ho následujícímu malému skriptu:

```
#!/bin/sh
#
WHOIAM=`whoami`
if [ "$WHOIAM" != "news" ]
then
        echo "You must run $0 as user 'news'" >&2
        exit 1
fi
#
cd /usr/lib/news
while read group; do
    if grep -si "^$group[[:space:]].*moderated" newsgroup; then
        mod=m
    else
        mod=y
    fi
    /usr/lib/news/maint/addgroup $group $mod
done
```

Zprávy sendsys, version a senduuname

Nakonec zde máme tři zprávy, které lze využít k získávání informací o síťové topologii. Jsou to zprávy *sendsys*, *version* a *senduuname*. Systém C News pošle po jejich obdržení odesílateli buď soubor `sys`, řetězec s verzí softwaru nebo výstup programu **uuname**. C News jsou při poskytování zprávy o verzi velmi skoupé; vrátí pouze písmeno `C`.

I zde platí, že tento typ zpráv byste neměli posílat, pokud si nejste absolutně jistí, že nemohou opustit vaši (regionální) síť. Odpovědi na zprávy *sendsys* mohou snadno shodit síť UUCP[155].

C News v prostředí NFS

Jednoduchý způsob, jak šířit news v rámci lokální počítačové sítě, je uchovávat všechny news na ústředním hostiteli a relevantní soubory exportovat prostřednictvím souborového systému NFS, kdy programy pro čtení news mohou přímo procházet články. Výhodou této metody oproti využití protokolu NNTP je mnohem nižší režie spojená se získáváním a řazením článků. Protokol NNTP na druhé straně vítězí v heterogenní síti, kde se vybavení jednotlivých hostitelů výrazně liší nebo kde uživatelé nemají potřebné účty na serveru.

Při použití souborového systému NFS musí být články zaslané lokálnímu hostiteli přesměrovány na centrální počítač, protože přístup k administrativním souborům by v opačném případě mohl vystavit systém nebezpečným podmínkám, které by způsobily nekonzistenci souborů. Nebo se můžete rozhodnout chránit vaši oblast spool tím, že ji exportujete jen pro čtení, což také vyžaduje přesměrování na centrální počítač.

Systém C News provádí tyto úkoly transparentně. Jakmile pošlete nějaký článek, váš program pro čtení news zpravidla spustí program **inews**, který vloží daný článek do systému news. Tento příkaz podrobí článek několika kontrolám, doplní hlavičku a projde soubor `server` v adresáři `/etc/news`. Pokud tento soubor existuje a obsahuje jiný název hostitele, než je název lokálního hostitele, spustí na tomto serveru prostřednictvím **rsh** program **inews**. Protože tento skript používá několik binárních příkazů a podporuje soubory systému C News, musíte mít buď lokálně nainstalován C News, nebo si připojit software news ze serveru.

Aby spojení rsh správně fungovalo, musí mít každý uživatel na serveru ekvivalentní účet, tedy takový, na který se může přihlásit bez hesla.

Ujistěte se, že název hostitele v souboru `server` přesně odpovídá výstupu příkazu **hostname** na serveru. Jinak by se totiž systém C News při doručování článku navždy zacyklil. O NFS hovoříme v kapitole 14.

Nástroje pro údržbu

Navzdory složitosti systému C News může být život jeho správce poměrně snadný, protože tento systém nabízí širokou paletu nástrojů pro údržbu. Některé z nich jsou určeny pro pravidelné spouštění démonem **cron**, například **newsdaily**. Používání těchto skriptů významně snižuje nároky na denní péči o vaši instalaci C News.

Nebude-li uvedeno jinak, jsou následující příkazy umístěny v adresáři `/usr/lib/news/maint`. (Nezapomeňte, že před spuštěním těchto příkazů se musíte přihlásit jako uživatel **news**. Pokud byste tyto příkazy spustili jako superuživatel, mohlo by dojít k tomu, že se důležité konfigurační soubory stanou pro systém C News nepřístupnými).

155 Neměli byste to zkoušet ani na Internetu.

`newsdaily` Význam tohoto souboru je zřejmý z jeho názvu. Jde o důležitý skript, který pomáhá udržet logovací soubory přijatelně velké a ponechává poslední tři kopie každého z nich. Také se pokouší napravit různé anomálie, jako jsou staré dávky v příchozích a odchozích adresářích, příspěvky do neznámých nebo moderovaných diskusních skupin a podobně. Výsledné chybové zprávy pak pošle správci news.

`newswatch` Tento skript byste měli spouštět pravidelně, zhruba jednou za hodinu, protože hledá anomálie v systému news. Má vystopovat problémy, které by měly okamžitý vliv na funkčnost systému news a poslat o tom správci hlášení. Mezi kontrolované oblasti patří zamykací soubory, které se nepodařilo odstranit, neobsloužené vstupní dávky a nedostatek místa na disku.

`addgroup` Přidá do vašeho lokálního systému novou skupinu. Správný formát tohoto příkazu je `addgroup` *`groupname`* `y|n|m|=`*`realgroup`* Druhý parametr má stejný význam, jako příznak v souboru `active` a může znamenat, že do dané skupiny může přispívat kdokoliv (y), nikdo (n), že je moderovaná (m) nebo že se jedná o alias pro jinou skupinu (*`=realgroup`*). Tento příkaz možná využijete také v případě, kdy první články nově vytvořené skupiny dorazí dříve, než řídicí příkaz *`newgroup`*, který má vyvolat její vytvoření.

`delgroup` Umožňuje lokální smazání nějaké skupiny. Spustíte ho jako `delgroup` *`groupname`* Pořád ještě ale musíte smazat články, které zůstaly v adresáři spool. Anebo můžete jejich odstranění svěřit přirozenému běhu věcí (tedy programu `expire`).

`addmissing` Přidá chybějící články do souboru `history`. Tento skript použijte, máte-li podezření, že se nějaké články v systému zasekly.

`newsboot` Tento skript by se měl spouštět při zavádění systému. Odstraní všechny zamykací soubory, které zbyly po zabití procesů news při ukončování systému a uzavře a provede všechny dávky, které zde zůstaly z NNTP spojení, jež byly přerušeny při ukončování systému.

`newsrunning` Tento skript najdete v adresáři `/usr/lib/news/input` a lze ho použít k zakázání rozbalování příchozích news, například v pracovní době. Rozbalování dávek vypnete následujícím způsobem `/usr/lib/news/input/newsrunning off` Rozbalování znovu zapnete, když místo řetězce *`off`* použijete *`on`*.

Protokol NNTP a démon nntpd

Protokol NNTP (Network News Transfer Protocol) představuje zcela odlišný mechanismus obsluhy news, než jaký používá C News a další programy. Namísto využití dávkových technologií jako je UUCP umožňuje předávání zpráv pomocí interaktivních síťových spojení. Nejedná se o nějaký program, ale o standard definovaný dokumentem RFC 977. Je založen na proudově orientovaném spojení, většinou prostřednictvím protokolu TCP, mezi klientem na libovolném místě v síti a serverem, který uchovává síťové news ve svém diskovém prostoru. Toto proudové spojení umožňuje klientovi a serveru interaktivně se dohodnout na přenosu článků s takřka nulovým zpožděním, čímž se daří udržovat nízký počet duplicitních článků. Společně s vysokými přenosovými rychlostmi současného Internetu se dosahuje přenosu news, který daleko překonává původní sítě UUCP. Zatímco před dvěma roky bylo zcela běžné, když článek dorazil do posledního uzlu Usenetu za dva týdny, podařilo se nyní tuto dobu zkrátit na méně než dva dny a v samotném Internetu je to často otázka několika minut.

K získávání, posílání a zveřejňování článků slouží klientům různé příkazy. Rozdíl mezi odesláním a zveřejněním je ten, že druhá metoda se může týkat článků s neúplnými hlavičkovými informacemi, obecně to znamená, že uživatel článek právě napsal[156]. Získání článku mohou využívat klienti pro přenos news stejně jako programy pro čtení news. Tím se stává protokol NNTP vynikajícím nástrojem, který umožňuje mnoha klientům v lokální síti přístup k news, přičemž se vyhnou komplikacím, k nimž dochází při použití NFS.

Protokol NNTP umožňuje aktivní a pasivní způsob přenosu news, kterému se hovorově říká „pushing“ a „pulling“. Pushing používá v podstatě stejný způsob, jako protokol C News ihave/sendme (viz kapitola 21). Klient nabízí serveru články prostřednictvím příkazu **IHAVE msgid** a server mu vrátí odpověď, z níž vyplývá, zda již daný článek má, nebo ho požaduje. Pokud o něj má zájem, klient mu článek pošle a ukončí ho tečkou na samostatném řádku.

Pushing má jednu nevýhodu v tom, že poměrně výrazně zatěžuje systém serveru, protože ten musí kvůli každému článku prohledávat databázi všech došlých článků.

Opačnou technikou je pulling, kdy klient požaduje seznam všech dostupných článků diskusní skupiny, které dorazily po specifikovaném datu. Takovýto dotaz předává pomocí příkazu **NEWNEWS**. Ze získaného seznamu identifikátorů zpráv si klient vybere ty články, které ještě nevlastní a postupně o ně požádá server příkazem **ARTICLE**.

156 Při zveřejňování článku protokolem NNTP přidá server minimálně jeden hlavičkový údaj – `NNTP-Posting-Host:`, který obsahuje název hostitele klienta.

Problémem pullingu je, že server potřebuje mít větší kontrolu nad skupinami a distribucemi, které povoluje klientovi požadovat. Musí se například ujistit, že nebude neautorizovaným klientům poslán tajný materiál z lokální diskusní skupiny.

Programy pro čtení news také disponují několika pohodlnými příkazy, které jim dovolují odděleně získat hlavičky a těla článků, případně i jednotlivé řádky článků. Takto je možné udržovat všechny news na centrálním hostiteli, k němuž mohou všichni uživatelé lokální sítě přistupovat prostřednictvím klientských programů pro čtení a posílání news, které pracující na bázi protokolu NNTP. Jedná se o alternativní řešení k exportu adresářů news prostřednictvím systému NFS, o kterém jsme hovořili v kapitole 21.

Celkový problém protokolu NNTP tkví v tom, že umožňuje informovanému člověku vložit do proudu news články s falešnou specifikací odesílatele. Tomu se říká *falšování news*[157]. Rozšíření protokolu NNTP umožňuje u určitých příkazů vyžadovat autentifikaci uživatele, takže pak existuje jistá ochrana před zneužitím systému.

V současné době existuje několik balíků protokolu NNTP. Jedním z nejpoužívanějších je démon NNTP (nntpd), nazývaný také *referenční implementace*. Jeho původními autory jsou Stan Barber a Phil Lapsley, kteří chtěli jeho prostřednictvím ilustrovat detaily specifikace RFC 977. Stejně jako u většiny dnes dostupných kvalitních programů bude tento program s největší pravděpodobností součástí vaší linuxové distribuce, případně si můžete pořídit jeho zdrojové kódy a přeložit jej sami. Pokud jej překládáte sami, musíte dobře znát strukturu vaší distribuce, abyste správně nakonfigurovali všechny používané cesty.

Balík nntpd tvoří server, dva klienti pro pushing a pulling news a najdete zde také náhradu programu **inews**. Vše je určeno pro prostředí B News, ale po několika úpravách to bude fungovat i v prostředí C News. Pokud však plánujete používat protokol NNTP i k jiným účelům, než jen pro zprostředkování přístupu k vašemu serveru news, není pro vás probíraná implementace tou pravou volbou. Proto budeme probírat pouze démona NNTP z balíku nntpd a vynecháme popis klientských programů.

Pokud chcete provozovat větší systém news, měli byste se poohlédnout po balíku *InterNet News*, zkráceně INN, jehož autorem je Rich Salz. Umožňuje přenos pomocí NNTP i UUCP. Pro přenos news je rozhodně lepší než program **nntpd**. Podrobněji o balíku INN hovoříme v kapitole 23.

Protokol NNTP

Hovořili jsme už o dvou klíčových příkazech protokolu NNTP, které se používají k pushingu a pullingu zpráv. Nyní si je popíšeme podrobněji v kontextu celé relace protokolu NNTP, abychom viděli, jak jednoduše protokol funguje. Postup si budeme demonstrovat s použitím jednoduchého **telnet** klienta, kterým se připojíme na INN news server virtuálního pivovaru, který se jmenuje *news.vbrew.com*. Kvůli krátkosti příkladů používá server nejjednodušší možnou konfiguraci. Jak server nakonfigurovat si ukážeme v kapitole 23. Při testování budeme vytvářet články pouze ve skupině *junk*, abychom jimi neobtěžovali nikoho jiného.

Připojení k serveru news

Připojení k serveru news se provede navázáním TCP spojení s jeho NNTP portem. Jakmile se připojíte, zobrazí se uvítací hlášení. Prvním z příkazů, které můžete vyzkoušet, je příkaz **help**. Odpověď serveru bude záviset na tom, zda vás považuje za vzdálený NNTP server nebo za klienta, pro-

157 Stejným problémem trpí i poštovní protokol SMTP, většina poštovních přenosových agentů však obsahuje mechanismy, které falšování pošty zabraňují.

tože používané příkazy se v obou případech liší. Režim chování serveru můžete změnit příkazem **mode**, o kterém budeme hovořit později.

```
$ telnet news.vbrew.com nntp
Trying 172.16.1.1...
Connected to localhost.
Escape character is '^]'.
200 news.vbrew.com InterNetNews server INN 1.7.2 08-Dec-1997 ready
help
100 Legal commands
        authinfo
                help
                ihave
                check
                takethis
                list
                mode
                xmode
                quit
                head
                stat
                xbatch
                xpath
                xreplic
For more information, contact "usenet" at this machine.
.
```

Odpověď serveru na libovolný příkaz vždy končí tečkou na samostatném řádku. Čísla uváděná na řádcích jsou *kódy odpovědí* a jejich prostřednictvím server oznamuje úspěšné nebo neúspěšné provedení příkazu. Všechny kódy jsou popsány v dokumentu RFC 977, o nejdůležitějších z nich se postupně zmíníme.

Pushing článku na server

Když jsme hovořili o pushingu článků na server, zmiňovali jsme se o příkazu **ihave**. Podívejme se nyní, jak tento příkaz doopravdy funguje:

```
ihave <123456@gw.vk2ktj.ampr.org>
335
From: terry@gw.vk2ktj.ampr.org
Subject: test message sent with ihave
Newsgroups: junk
Distribution: world
Path: gw.vk2ktj.ampr.org
Date: 26 April 1999
Message-ID: <123456@gw.vk2ktj.ampr.org>
Body:

This is a test message sent using the NNTP IHAVE command.

.
235
```

Příkazy protokolu NNTP nerozlišují velká a malá písmena, takže je můžete zadávat oběma způsoby. Příkaz **ihave** má jediný povinný parametr, kterým je identifikátor posílané zprávy. Každý článek news dostává při svém vytvoření jednoznačný identifikátor. Příkaz **ihave** umožňuje serveru NNTP říct, které články má k dispozici, když provádí pushing na jiný server. Odesílající server spustí příkaz **ihave** pro každý článek, který má k dispozici. Pokud vzdálený server vrátí kód *3xx*, odesílající server mu pošle úplný článek se všemi hlavičkami, přičemž článek ukončí tečkou na samostatném řádku. Pokud vzdálený server odpoví kódem *4xx*, nechce daný článek přijmout – například proto, poněvadž jej už má, nebo kvůli nějakým problémům, například nedostatku diskového místa.

Přechod do režimu čtenáře

Programy pro čtení zpráv používají při komunikaci se serverem vlastní skupinu příkazů. Tyto příkazy se aktivují přepnutím serveru do režimu *čtenáře*. Většina serverů používá tento režim implicitně, pokud připojení nevzniklo z IP adresy, kterou má nastavenu jako adresu partnerského serveru. V každém případě však protokol NNTP obsahuje příkaz, jímž je možné přepnutí provést explicitně.

```
mode reader
200 news.vbrew.com InterNetNews NNRP server INN 1.7.2 08-Dec-1997 ready/
    (posting ok).
help
100 Legal commands
  authinfo user Name|pass Password|generic <prog> <args>
  article [MessageID|Number]
  body [MessageID|Number]
  date
  group newsgroup
  head [MessageID|Number]
  help
  ihave
  last
  list [active|active.times|newsgroups|distributions|distrib.pats|/
      overview.fmt|subscriptions]
  listgroup newsgroup
  mode reader
  newgroups yymmdd hhmmss ["GMT"] [<distributions>]
  newnews newsgroups yymmddhhmmss ["GMT"] [<distributions>]
  next
  post
  slave
  stat [MessageID|Number]
  xgtitle [group_pattern]
  xhdr header [range|MessageID]
  xover [range]
  xpat header range|MessageID pat [morepat...]
  xpath MessageID
Report problems to <usenet@vlager.vbrew.com>
.
```

Server v režimu čtenáře nabízí řadu příkazů. Většina z nich usnadňuje život programům pro čtení zpráv. Už jsme se zmínili o příkazech pro samostatné poslání hlavičky a těla článku. Kromě to-

ho jsou k dispozici příkazy pro vypsání dostupných skupin a článků, a jiné příkazy, které umožňují zveřejnění článku na serveru.

Výpis dostupných skupin

Příkazem **list** dojde k vypsání různých informací, například skupin, které server podporuje:

```
list newsgroups
215 Descriptions in form "group description".
control                 News server internal group
junk                    News server internal group
local.general           General local stuff
local.test              Local test group
.
```

Výpis aktivních skupin

Příkaz **list active** vypíše jednotlivé podporované skupiny a uvede k nim další informace. Čísla na každém řádku představují nejnižší a nejvyšší číslo článku v dané skupině. Z těchto dvou čísel může čtenář poznat počet článků ve skupině. Budeme o nich hovořit za chvíli. Poslední údaj obsahuje příznaky uvádějící, zda je možné do skupiny přispívat, zda je skupina moderovaná a zda se články ukládají nebo jen posílají dále. Podrobněji jsou jednotlivé příznaky popsány v kapitole 23. Příklad výstupu vypadá takto:

```
list active
215 Newsgroups in form "group high low flags".
control 0000000000 0000000001 y
junk 0000000003 0000000001 y
alt.test 0000000000 0000000001 y
.
```

Publikování článku

Už jsme se zmínili o rozdílu mezi odesláním a publikováním článku. Když článek odesíláte, předpokládá se, že tento článek už existuje, že má identifikátor, který mu přiřadil server na němž byl původně publikován, a že obsahuje všechny potřebné hlavičky. Když článek publikujete, znamená to, že jej vytváříte a uvádíte pouze ty hlavičky, které se týkají vás, například předmět článku a skupinu v níž má být zveřejněn. Server na němž článek zveřejňujete pak musí doplnit ostatní potřebné hlavičky a přiřadit zprávě identifikátor, který se použije ke zveřejňování na jiných serverech. Znamená to, že publikovat článek je jednodušší než jej na server poslat. Příklad publikování článku může vypadat takto:

```
post
340 Ok
From: terry@richard.geek.org.au
Subject: test message number 1
Newsgroups: junk
Body:

This is a test message, please feel free to ignore it.

.
240 Article posted
```

Dále jsme vytvořili ještě dvě podobné nové zprávy, aby další příklady vypadaly realističtěji.

Vypsání nových článků

Když se program pro čtení zpráv poprvé připojí na server a uživatel si zvolí prohlíženou skupinu zpráv, musí program zjistit seznam nových článků, tedy článků, které byly na server poslány nebo publikovány od posledního přihlášení uživatele. Musí být uvedeny tři povinné parametry: název skupiny nebo skupin, které se mají prohlížet, počáteční datum a počáteční čas, od kterého se mají zprávy prohlížet. Datum i čas se zadávají jako šestice číslic, přičemž se postupuje od nejvýznamnějšího po nejméně významný údaj (tedy *rrmmdd* a *hhmmss*):

```
newnews junk 990101 000000
230 New news follows
<7g2o5r$aa$6@news.vbrew.com>
<7g5bhm$8f$2@news.vbrew.com>
<7g5bk5$8f$3@news.vbrew.com>
.
```

Výběr skupiny

Když si uživatel vybere skupinu k prohlížení, může program pro čtení zpráv označit tuto skupinu za aktivní. Tím se usnadňuje komunikace mezi programem a serverem, protože nebude nutné u každého příkazu uvádět, které skupiny se týká. Příkaz **group** požaduje jako parametr název skupiny. Většina následujících příkazů pak pracuje se zvolenou skupinou, pokud nebude explicitně uvedena jiná skupina.

```
group junk
211 3 1 3 junk
```

Potvrzení od serveru obsahuje počet zpráv ve skupině, číslo první a poslední zprávy a název skupiny. I když v našem případě je počet zpráv a číslo nejvyšší zprávy shodné, ve většině případů to nebude platit. Na aktivních serverech byly různé články smazány nebo zastaraly, takže počet zpráv se snižuje, nicméně číslo nejvyšší zprávy zůstává stejné.

Vypsání článků ve skupině

Aby mohl program pro čtení zpráv adresovat články ve skupině, musí vědět, která čísla odpovídají existujícím článkům. Příkaz **listgroup** vrací seznam čísel existujících článků v nastavené skupině nebo v explicitně uvedené skupině:

```
listgroup junk
211 Article list follows
1
2
3
.
```

Převzetí hlaviček článku

Uživatel potřebuje mít o článku nějaké základní údaje ještě předtím než se rozhodne, zda si chce článek přečíst. Už jsme se zmínili o tom, že existují příkazy, které zvlášť stáhnou hlavičky článků a jejich těla. Příkazem **head** se říká serveru, aby přenesl pouze hlavičky zadané zprávy. Pokud si

uživatel nebude chtít zprávu přečíst, ušetříme čas a zatížení sítě tím, že nebudeme zbytečně stahovat (klidně i dlouhé) tělo článku.

Na články je možné se odkazovat buď jejich čísly (zjištěnými příkazem **listgroup**) nebo jejich identifikátory.

```
head 2
221 2 <7g5bhm$8f$2@news.vbrew.com> head
Path: news.vbrew.com!not-for-mail
From: terry@richard.geek.org.au
Newsgroups: junk
Subject: test message number 2
Date: 27 Apr 1999 21:51:50 GMT
Organization: The Virtual brewery
Lines: 2
Message-ID: <7g5bhm$8f$2@news.vbrew.com>
NNTP-Posting-Host: localhost
X-Server-Date: 27 Apr 1999 21:51:50 GMT
Body:
Xref: news.vbrew.com junk:2
.
```

Převzetí těla článku

Pokud se uživatel rozhodne nějaký článek přečíst, musí mu program pro čtení zpráv předložit i tělo článku. K získání těla slouží příkaz **body**. Funguje stejně jako příkaz **head**, vrací však tělo článku:

```
body 2
222 2 <7g5bhm$8f$2@news.vbrew.com> body
This is another test message, please feel free to ignore it too.

.
```

Převzetí celého článku

I když je za normálních okolností výhodné přenášet zvlášť hlavičky a tělo článku, při některých příležitostech je rozumnější přenést celý článek najednou. Příkladem mohou být programy používané pro přenos všech článků ve skupině bez podrobnějšího výběru, například cachovací NNTP program **leafnode**[158].

Protokol NNTP samozřejmě takovou operaci umožňuje a nepřekvapí nás, že příkaz bude fungovat opět podobně jako příkaz **head**. Ke stažení celého článku naráz použijeme příkaz **article**, kterému zadáváme číslo nebo identifikátor článku.

```
article 1
220 1 <7g2o5r$aa$6@news.vbrew.com> article
Path: news.vbrew.com!not-for-mail
From: terry@richard.geek.org.au
Newsgroups: junk
Subject: test message number 1
Date: 26 Apr 1999 22:08:59 GMT
```

158 Program **leafnode** získáte pomocí anonymního FTP z počítače *upxx02.toxi.uni-wuerzbung.de* v adresáři `/pub`.

```
Organization: The Virtual brewery
Lines: 2
Message-ID: <7g2o5r$aa$6@news.vbrew.com>
NNTP-Posting-Host: localhost
X-Server-Date: 26 Apr 1999 22:08:59 GMT
Body:
Xref: news.vbrew.com junk:1

This is a test message, please feel free to ignore it.

.
```

Pokud se pokusíte přečíst neexistující článek, server vrátí odpověď s příslušným kódem a možná i s textovým vysvětlením:

```
article 4
423 Bad article number
```

Popsali jsme si nejdůležitější příkazy protokolu NNTP. Pokud byste chtěli vytvořit implementaci protokolu NNTP, měli byste si přečíst příslušnou RFC specifikaci. Tam naleznete všechny podrobnosti, o kterých jsme zde nehovořili.

Nyní se podíváme, jak je protokol NNTP implementován serverem *nntpd*.

Instalace serveru NNTP

Server *nntpd* lze přeložit dvěma způsoby v závislosti na očekávaném zatížení systému news. Kvůli implicitním hodnotám specifickým pro jednotlivé systémy, které jsou zakódovány přímo do spustitelného souboru, nejsou k dispozici žádné předkompilované verze. Veškerou konfiguraci provádí makro, které je definováno v souboru `common/conf.h`.

Server *nntpd* může být nakonfigurován buď jako samostatný server, který je spouštěn při zavádění systému z **rc** souboru, nebo jako démon řízený superserverem **inetd**. Ve druhém případě je třeba mít v souboru `/etc/inetd.conf` následující záznam:

```
nntp    stream   tcp nowait    news    /usr/etc/in.nntpd      nntpd
```

Syntaxe souboru `/etc/inetd.conf` je popsána v kapitole 12. Pokud konfigurujete *nntpd* jako samostatný server, ujistěte se, že je každý podobný řádek řádně zakomentován. V každém případě se musíte přesvědčit, že v souboru `/etc/services` nechybí následující řádek:

```
nntp    119/tcp   readnews untp       # Network News Transfer Protocol
```

Aby bylo možné ukládat příchozí články, potřebuje mít server *nntpd* v adresáři `spool` také adresář `.tmp`. Vytvoříte ho následovně:

```
# mkdir /var/spool/news/.tmp
# chown news.news /var/spool/news/.tmp
```

Omezení přístupu k NNTP serveru

Přístup ke zdrojům NNTP je řízen souborem `nntp_access`, který je uložen v adresáři `/etc/news`. Řádky v tomto souboru specifikují přístupová práva přidělená vzdáleným hostitelům. Každý řádek má následující formát:

```
site read|xfer|both|no post|no [!exceptgroup]
```

Jakmile se klient připojí na NNTP port, pokusí se server *nntpd* pomocí zpětného vyhledávání podle IP adresy získat plně kvalifikovaný doménový název vzdáleného systému. Název klienta a jeho IP adresa jsou porovnány s polem *site* každého záznamu v takovém pořadí, v jakém jsou uvedeny v souboru. Shoda může být buď částečná, nebo úplná. V případě úplné shody bude položka použita. V případě jen částečné shody bude položka použita jen pokud se nenajde žádná další lepší shoda. Pole *site* je možné specifikovat jedním z následujících způsobů:

hostname	Plně kvalifikovaný doménový název hostitele. Pokud přesně odpovídá kanonickému hostitelskému názvu klienta, použije se tento záznam a všechny další záznamy budou ignorovány.
IP adresa	IP adresa v tečkové notaci. Pokud jí IP adresa klienta odpovídá, použije se tento záznam a všechny následující záznamy budou ignorovány.
domainname	Název domény zadaný ve formě **.domain*. Pokud doména v názvu klienta odpovídá zadané hodnotě, pak záznam vyhovuje.
networkname	Název sítě specifikovaný v souboru `/etc/networks`. Pokud síťová část IP adresy klienta odpovídá číslu dané sítě v souboru `/etc/networks`, pak záznam vyhovuje.
`default`	Této volbě vyhovuje každý klient.

Záznamy s obecnější specifikací adres je dobré uvést na začátku souboru, protože případné shody budou pozdějšími přesnějšími shodami potlačeny.

Druhé a třetí pole popisují přístupová práva přidělená danému klientovi. Druhé pole uvádí přístupová práva pro získávání news metodou pullingu (*read*) a předávání news metodou pushingu (*xfer*). Hodnota *both* povoluje oba typy přístupu, zatímco hodnota *no* přístup úplně zakazuje. Třetí pole uděluje klientovi právo publikovat články, což znamená doručovat články s neúplnými hlavičkovými informacemi, které pak doplní software news. Pokud druhé pole obsahuje hodnotu *no*, je třetí pole ignorováno.

Čtvrté pole je volitelné a obsahuje čárkami oddělený seznam skupin, k nimž má klient zakázaný přístup.

Následuje ukázkový soubor `nntp_access`:

```
#
# standardně mohou všichni pushovat news, nemohou pushovat a publikovat
default              xfer     no
#
# hostitel public.vbrew.com poskytuje veřejný přístup po modemu
# povolíme pull a publikaci všude mimo lokální skupinu
public.vbrew.com     read     post     !local
#
# ostatní počítače pivovaru mají povolen pulling a publikaci
*.vbrew.com          read     post
```

Autorizace protokolem NNTP

Démon *nntpd* nabízí jednoduchý autorizační mechanismus. Pokud některé z přístupových tokenů v souboru `nntp_access` uvedete velkými písmeny, bude server před příslušnou operací poža-

dovat od klienta autorizaci. Pokud například uvedete oprávnění Xfer nebo XFER (oproti klasickému xfer), démon nepovolí klientovi přenos zpráv pokud nedojde k autorizaci.

Proces autorizace je implementován prostřednictvím nového příkazu protokolu NNTP, který se jmenuje **AUTHINFO**. Za pomoci tohoto příkazu předá klient serveru NNTP uživatelské jméno a heslo. Server *nntpd* je porovná s databází `/etc/passwd` a ověří, že příslušný uživatel patří do skupiny **nntp**.

Současná implementace NNTP autorizace je pouze experimentální a není tedy dostatečně přenositelná. Vzhledem k tomu funguje jen pro základní databáze hesel, nelze ji použít se stínovou databází. Pokud si démona přeložíte ze zdrojových kódů a máte instalován balík PAM, je možné mechanismus kontroly hesel snadno změnit.

Interakce serveru nntpd a systému C News

Po přijetí článku ho musí server *nntpd* doručit do subsystému news. V závislosti na tom, zda jej obdržel jako výsledek příkazu **IHAVE** nebo **POST**, je článek předán programu **rnews** nebo **inews**. Místo přímého spouštění programu **rnews** můžete server (v době překladu) nastavit tak, aby zpracovával příchozí články do dávky a výsledné dávky pak posílal do souboru `/var/spool/news/in.coming`, kde si je při příštím dotazu vyzvedne program **relaynews**.

Aby mohl správně využívat protokol ihave/sendme, musí mít server *nntpd* přístup k souboru `history`. V době překladu se proto musíte ujistit, že máte správně nastavenu příslušnou cestu. Také je třeba se přesvědčit, že se systém C News a *nntpd* dohodli na formátu souboru `history`. Systém C News používá při přístupu k němu hašovací funkce `dbm`, která má poměrně hodně různých a mírně nekompatibilních implementací. Pokud by byl systém C News slinkován s jinou knihovnou `dbm`, než kterou máte ve vaší knihovně `libc`, musíte s touto knihovnou slinkovat také server *nntpd*.

Nedorozumění mezi programem **nntpd** a systémem C News občas vedou k tomu, že se v logovacích záznamech objevují zprávy o tom, že **nntpd** nemůže otevřít zprávy, nebo můžete zjistit, že protokolem NNTP vám přicházejí zprávy duplicitně. Vhodným testem je vzít článek z adresáře `spool`, navázat telnetové spojení se serverem **nntpd** a nabídnout mu tento článek tak, jak to ukazuje následující příklad. Samozřejmě musíte nahradit řetězec *msg@id* správným identifikátorem článku.

```
$ telnet localhost nntp
Trying 127.0.0.1...
Connected to localhost
Escape characters is '^ ]'.
201 vstout NNTP[auth] server version 1.5.11t (16 November 1991) ready at
Sun Feb 6 16:02:32 1194 (no posting)
IHAVE msg@id
435 Got it.
QUIT
```

Tento rozhovor ukazuje správnou reakci serveru **nntpd**; zpráva „Got it“ říká, že již příslušný článek má. Pokud místo toho obdržíte zprávu **335 OK**, pak vyhledávání v souboru `history` z nějakého důvodu selhalo. Ukončete rozhovor stiskem kombinace kláves Ctrl+D. Co se stalo, zjistíte na základě systémového logu; server **nntpd** zapisuje všechny druhy zpráv do syslogu. Nekompatibilní knihovnu `dbm` odhalíte zpravidla podle zprávy se stížností na selhání programu `dbminit`.

Internet News

Démon Internet News (INN) je zřejmě v současnosti nejoblíbenějším serverem news. Je mimořádně pružný a vyhovuje prakticky pro jakékoliv nasazení s výjimkou velmi jednoduchých systémů[159]. Je velmi výhodný pro systémy s velkým zatížením.

Server INN obsahuje řadu komponent, přičemž každá používá vlastní konfigurační soubor, o nichž budeme za chvíli hovořit. Konfigurace serveru INN může být trochu namáhavá, nicméně si popíšeme celý její průběh a poskytneme vám dostatek informací k tomu, aby vám dávaly smysl manuálové stránky programu INN a dokumentace programu, takže budete moci vyrobit konfiguraci pro prakticky jakékoliv použití.

Podrobnosti o programu INN

Základem balíku INN je démon **innd**. Úkolem tohoto démona je přijímat příchozí články, lokálně je ukládat a podle potřeby je předávat okolním systémům. Spouští se při startu systému a běží trvale jako proces na pozadí. Jeho spuštění jako démona zvyšuje výkon systému, protože program čte svou konfiguraci pouze jednou. V závislosti na zatížení vašeho systému mohou mít některé soubory (například `history`, obsahující seznam všech přijatých článků) velikost od jednotek po desítky megabajtů.

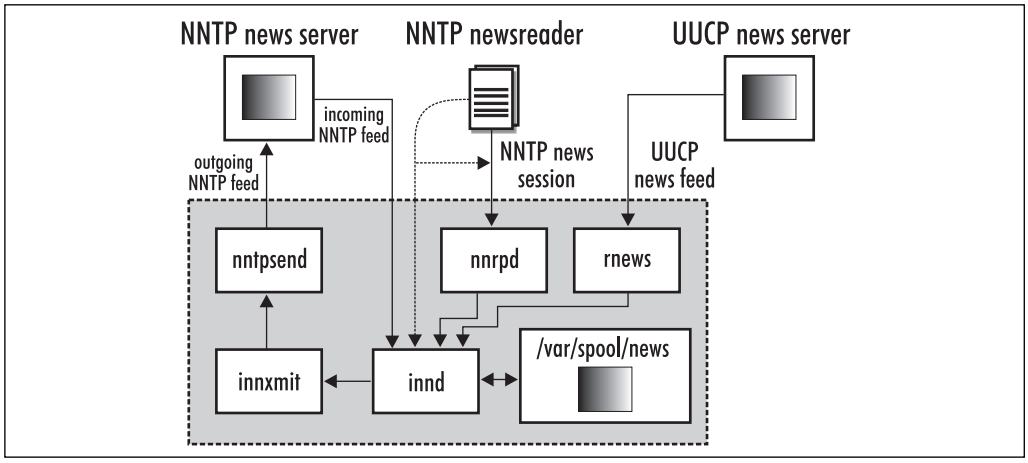
Další důležitou vlastností balíku INN je, že vždy běží jen jediná instance démona **innd**. I to je z hlediska výkonu velmi příznivé, protože démon může všechny články zpracovávat aniž by se zatěžoval synchronizací svých interních stavů s ostatními instancemi téhož démona, které se budou ometat kolem stejného adresáře se zprávami. Tím je však ovlivněn celkový návrh systému. Protože je nutné, aby byly příchozí zprávy obslouženy co nejrychleji, je neúnosné, aby byl server zablokován nepodstatnými úlohami jako je obsluha klienta čtoucího zprávy protokolem NNTP, nebo jako je rozbalování dávky článků přijaté protokolem UUCP. Proto jsou takové úlohy odděleny od hlavního serveru a implementují je samostatné programy. Na obrázku 23.1 jsou znázorněny vztahy mezi démonem **innd**, ostatními lokálními procesy, vzdáleným serverem news a klienty.

V současné době je nejběžnějším mechanismem přenosu zpráv protokol NNTP a démon **innd** proto přímo nepodporuje nic jiného. Znamená to, že démon **innd** poslouchá na TCP portu 119 a přijímá články protokolem „ihave“.

Články přijaté jiným mechanismem než je NNTP se zpracovávají tak, že je přijme samostatný proces a ten je protokolem NNTP předá démonu **innd**. Dávky článků přijímané protokolem UUCP typicky obsluhuje program **rnews**. Program **rnews**, který je součástí balíku INN, dávku rozbalí, a jednotlivé články postupně předá programu **innd**.

159 Velmi malé systémy mohou používat cachovací NNTP server **leafnode**, který lze získat na adrese *http:// upxx02.toxi.uni-wuerzbung.de/~krasel/leafnode.html*.

Klientské programy mohou články dodávat, pokud uživatel nějaký napíše. Protože obsluha programů pro čtení zpráv představuje samostatný úkon, vrátíme se k ní později.



Obrázek 23.1 – Architektura balíku INN (zjednodušeno)

Když démon **innd** přijme zprávu, nejprve hledá její identifikátor v souboru `history`. Duplicitní články se zahazují a případně se o tom provede záznam v logu. Podobně se nakládá se starými články[160] nebo s články, které neobsahují požadované hlavičkové údaje (například údaj `Subject:`). Když **innd** považuje článek za přijatelný, podívá se na hlavičkový údaj `Newsgroups:` a zjistí, ve které skupině má být článek umístěn. Pokud se jedná o nějakou skupinu podle souboru `active`, uloží se článek na disk. V opačném případě bude zařazen do skupiny **junk**.

Jednotlivé články se ukládají pod adresářem `/var/spool/news`. Každá skupina používá vlastní adresář, každý článek se v něm ukládá jako samostatný soubor. Názvy těchto souborů jsou číselné, takže článek ve skupině `comp.risks` může být uložen například jako `comp/risks/217`. Pokud **innd** zjistí, že adresář v němž chce zprávu uložit neexistuje, vytvoří jej.

Kromě lokálního uložení zpráv je možné zprávy předávat vzdáleným systémům. To nastavuje soubor `neewsfeeds`, který obsahuje seznam „krmených“ systémů a skupiny, které se jim mají předávat.

Stejně jako příjem zpráv je i obsluha odesílaných zpráv obsluhována jedním rozhraním. Server **innd** neimplementuje jednotlivé přenosové mechanismy, namísto toho spoléhá na různé pomocné programy, které zajistí přenos zpráv na konkrétní cílové systémy. Možnosti odesílání zpráv se souhrně označují jako *kanály*. Podle svého účelu může mít kanál různé vlastnosti, které určují, jaké informace mu bude démon **innd** konkrétně předávat.

K odchodu zpráv protokolem NNTP spouští démon **innd** typicky program **innxmit**, kterému pak pro každý odesílaný článek předá na standardní vstup jeho identifikátor, velikost a název soubor v němž je článek uložen. K odesílání zpráv protokolem UUCP může místo toho zapisovat velikosti a názvy souborů jednotlivých článků do samostatného souboru, který bude obsluhován jiným procesem, jež v pravidelných interval podle tohoto souboru vytvoří dávky a předá je vzdálenému UUCP systému.

160 Které se poznávají podle hlavičky `Date:` a je typicky omezeno na dva týdny.

Kromě těchto dvou příkladů existují i další kanály, které nejsou určeny pouze pro odesílání zpráv. Používají se například pro archivaci určitých skupin, nebo k vytváření všeobecných informací. Tyto informace umožňují programům pro čtení zpráv efektivněji řadit témata článků. Starší programy musely přečíst všechny články a pak z nich mohly vytvořit skupiny debat. To představovalo velké zatížení serveru, zejména pokud se používal protokol NNTP, navíc byl tento postup poměrně pomalý[161]. Mechanismus všeobecných informací řeší tento problém přepsáním všech potřebných hlaviček do samostatného souboru `.overview`. Program pro čtení zpráv si pak může tento soubor vyzvednout buď přímo nebo prostřednictvím NNTP příkazu **XOVER**. Systém INN funguje tak, že démon **innd** předává všechny odesílané zprávy programu **overchan**, který je k démonu připojen jako kanál. Jak se to dělá, uvidíme později.

Čtení zpráv a INN

Programy pro čtení zpráv běžící na stejném počítači jako server (nebo na počítači, který má prostřednictvím NFS připojeny adresáře se zprávami) mohou číst zprávy přímo z jejich adresářů. K publikování zprávy napsané uživatelem spouštějí program **inews**, který doplní chybějící hlavičkové údaje a pak zprávu protokolem NNTP předá démonu INN.

Druhá možnost je, že programy přistupují ke zprávám protokolem NNTP. Tento typ spojení se obsluhuje odlišně od přijímání zpráv od jiných serverů, aby se předešlo zablokování démona. Kdykoliv se k NNTP serveru připojí klientský program, spustí **innd** program **nnrpd**, který danou relaci obslouží a démon **innd** se bude moci věnovat důležitějším úkonům (například příjmu nových zpráv)[162]. Možná vás zajímá, jak **innd** rozliší mezi připojením jiného serveru a připojením klienta. Odpověď je velmi jednoduchá: protokol NNTP vyžaduje, aby program pro čtení zpráv jako první příkaz po připojení zadal **mode reader**. Jakmile **innd** přijme takovýto příkaz, vytvoří proces **nnrpd**, předá mu stávající spojení a vrátí se k obsluze příchozích spojení. Pokud víme, existuje přinejmenším jeden dosovský program pro čtení zpráv, který tento příkaz nevysílal a proto při komunikaci s INN žalostně selhal – program **innd** totiž nerozpoznává příkazy pro čtení zpráv pokud nedetekoval, že se k němu připojil klientský program.

O spolupráci programů pro čtení zpráv a balíku INN budeme podrobněji hovořit v části *Řízení přístupu programů pro čtení zpráv* dále v této kapitole.

Instalace balíku INN

Než se začneme zabývat konfigurací balíku INN, popišme si napřed jeho instalaci. Doporučujeme vám si tento text přečíst i v případě, pokud jste INN nainstalovali jako součást některé distribuce Linuxu. Obsahuje totiž několik poznámek ohledně bezpečnosti a kompatibility.

Distribuce Linuxu obsahovaly poměrně dlouhou dobu verzu INN-1.4sec. Tato verze však v sobě skrývá dva méně závažné bezpečnostní problémy. Moderní verze těmito problémy netrpí a většina distribucí Linuxu obsahuje INN verze 2 a vyšší.

Pokud chcete, můžete si INN přeložit sami. Zdrojový kód lze získat na adrese *ftp.isc.org* v adresáři `/isc/inn`. Pro překlad musíte upravit konfigurační soubor, který balíku INN poskytuje informace o vašem operačním systému, kromě toho některé funkce mohou vyžadovat drobné zásahy přímo do zdrojového kódu.

161 Rozdělení 1000 článků podle předmětu debaty mohlo na zatíženém serveru trvat i pět minut, což není ve většině případů přijatelné.

162 Název programu pochází z NewNews Read & Post Daemon.

Samotný překlad je jednoduchý, celým procesem vás provede skript **BUILD**. Zdrojový kód navíc obsahuje podrobnou dokumentaci k instalaci a konfiguraci balíku.

Po instalaci všech binárních souborů mohou být nutné drobné úpravy jiných aplikací, které pracují s programy **rnews** a **inews**. UUCP například spouští **rnews** z adresáře `/usr/bin` nebo `/bin`, zatímco INN jej standardně instaluje do adresáře `/usr/lib/bin`. Zajistěte, aby se adresář `/usr/lib/bin` nacházel ve standardní vyhledávací cestě, případně vytvořte symbolické odkazy ukazující na skutečná umístění programů **rnews** a **inews**.

Konfigurace programu INN: Základní nastavení

Jednou z největších komplikací, na kterou řada začátečníků narazí, bude to, že INN potřebuje ke své činnosti funkční síťové prostředí dokonce i tehdy, pokud pracuje na samostatném počítači. Pro spuštění balíku INN je nutné, aby jádro podporovalo protokol TCP/IP a aby bylo vytvořeno zpětné zařízení tak, jak jsme to popisovali v kapitole 5.

Dále je třeba zajistit, aby se při startu systému spouštěl démon **innd**. Standardní instalace balíku INN obsahuje v adresáři `/etc/news` skript `boot`. Pokud používáte distribuce založené na inicializačních souborech System V, stačí vytvořit symbolický odkaz `/etc/init.d/inn` vedoucí na `/etc/news/boot`. Pokud používáte jiný systém inicializačních souborů, musíte zajistit, aby se `/etc/news/boot` spouštěl některým z `rc` skriptů. Protože INN vyžaduje funkční síť, spouštěcí skript se musí volat až *po* konfiguraci síťových rozhraní.

Konfigurační soubory balíku INN

Po splnění základních podmínek se můžeme pustit do té opravdu zajímavé části, do konfiguračních souborů balíku INN. Všechny jsou umístěny v adresáři `/etc/news`. Ve verzi 2 prodělaly tyto soubory některé změny oproti předchozím verzím, my si popíšeme konfiguraci právě verze 2. Pokud používáte starší verzi, měl by vám následující text pomoci s provedením nezbytných úprav. V dalším textu si postupně popíšeme jednotlivé konfigurační soubory, jako základ všech příkladů použijeme systém virtuálního pivovaru.

Pokud se budete zajímat o funkce jednotlivých konfiguračních souborů podrobněji, mohou vám pomoci manuálové stránky – distribuce INN obsahuje manuálové stránky pro každý z konfiguračních souborů.

Globální parametry

Řada parametrů balíku INN je ze své podstaty globálních, vztahují se ke všech skupinám.

Soubor inn.conf

Hlavním konfiguračním souborem balíku INN je soubor `inn.conf`. Kromě jiného obsahuje i název, pod kterým je váš počítač znám na Usenetu. Verze 2 umožňuje v tomto souboru nastavit neskutečné množství parametrů. Naštěstí implicitní hodnoty většiny parametrů budou v řadě případů vyhovovat. Stránka `inn.conf(5)` popisuje všechny podrobnosti a v případě problémů byste ji měli pečlivě přečíst.

Příklad soubor `inn.conf` může vypadat takto:

```
# Příklad souboru inn.conf ve Virtual Brewery
server:          vlager.vbrew.com
domain:          vbrew.com
```

```
fromhost:        vbrew.com
pathhost:        news.vbrew.com
organization:    The Virtual Brewery
mta:             /usr/sbin/sendmail -oi %s
moderatormailer: %s@uunet.uu.net
#
# Cesty ke komponentám a souborům
#
pathnews:               /usr/lib/news
pathbin:                /usr/lib/news/bin
pathfilter:             /usr/lib/news/bin/filter
pathcontrol:            /usr/lib/news/bin/control
pathdb:                 /var/lib/news
pathetc:                /etc/news
pathrun:                /var/run/news
pathlog:                /var/log/news
pathhttp:               /var/log/news
pathtmp:                /var/tmp
pathspool:              /var/spool/news
patharticles:           /var/spool/news/articles
pathoverview:           /var/spool/news/overview
pathoutgoing:           /var/spool/news/outgoing
pathincoming:           /var/spool/news/incoming
patharchive:            /var/spool/news/archive
pathuniover:            /var/spool/news/uniover
overviewname:           .overview
```

První řídek říká programům **rnews** a **inews** s jakým hostitelem se mají spojovat při doručování článků. Tento údaj je naprosto nezbytný, aby mohly tyto programy předávat články démonu **innd**, musí s ním navázat NNTP spojení.

Klíčové slovo *domain* definuje doménovou část plně kvalifikovaného jména hostitele. Řada programů potřebuje ke správné činnosti plně kvalifikované jméno. Pokud resolver vrátí pouze název hostitele, připojí se k němu zde nastavená doménová část jména. Položku není problém nastavit, takže je rozumné to udělat bez ohledu na nastavení resolveru.

Další řádek definuje název hostitele, který má program **inews** použít v hlavičce `From:` zpráv vytvořených lokálními uživateli. Řada programů pro čtení zpráv využívá této hlavičky k sestavení poštovní adresy autora článku. Pokud tento údaj neuvedete, bude použito plně kvalifikované jméno vašeho hostitele. To není vždycky ideální. Poštu a news mohou například obsluhovat dva odlišné systémy. V takovém případě byste jako hodnotu parametru *frombost* použili adresu vašeho poštovního serveru.

Údaj *pathbost* definuje název, který ma INN přidat do hlavičky `Path:` zprávy vždy když zprávu předává dále. Ve většině případů to bude plně kvalifikované jméno news serveru, takže tento parametr nemusíte uvádět, protože se tak chová implicitně. Při obsluze velkých domén můžete použít obecné jméno, například *news.vbrew.com*. Díky tomu budete moci v případě potřeby snadno přenést celý systém na jiného hostitele.

Další řádek definuje organizaci. Nastavujete zde, jaký text bude **inews** uvádět v hlavičce `Organization:`. Formálně vzato byste zde měli uvést název organizace nebo její popis. Pokud nechcete být formální, můžete zde projevit svůj smysl pro humor.

Klíčové slovo *mta* je povinné a definuje cestu pro poštovního agenta použitou ke zveřejňování moderovaných zpráv. Symbol `%s` bude nahrazen adresou moderátora.

Údaj *moderatormailer* definuje adresu použitou při přidání článku do moderované skupiny. Seznam adres moderátorů jednotlivých skupin je obvykle uveden v samostatném souboru, může však být problém všechny správně udržovat. Údaj *moderatormailer* je tedy poslední záchrana, pokud je definován, nahradí **inews** symbol `%s` (mírně upravenou) adresou konference a na vzniklou adresu pošle článek. Pokud budete posílat příspěvek na soc.feminism, pošle se při výše uvedené konfiguraci na soc-feminism@uunet.uu.net. Na UUNETu by měly být definovány aliasy pro všechny takové adresy a článek tak bude předán správnému moderátorovi.

Konečně všechny zbývající údaje definují umístění některých souborů a komponent balíku INN. Pokud jste instalovali INN jako součást distribuce, mělo by být všechno nastaveno správně. Pokud jste instalovali ze zdrojových kódů, musíte zkontrolovat, zda nastavení souhlasí.

Konfigurace skupin

Správce news na každém systému může řídit, ke kterým skupinám budou mít uživatelé přístup. INN obsahuje dva konfigurační soubory udávající, které skupiny podporovat a jaký je jejich popis.

Soubory active a newsgroups

Soubory `active` a `newsgroups` definují a popisují skupiny podporované daným hostitelem. Uvádějí, ze kterých skupin chceme přijímat a obsluhovat články a jaké jsou administrativní údaje těchto skupin. Soubory se nacházejí v adresáři `/var/lib/news`.

Soubor `active` říká, které skupiny server podporuje. Jeho syntaxe je velmi jednoduchá. Každý řádek souboru `active` obsahuje čtyři údaje oddělené mezerami:

```
name himark lowmark flags
```

Údaj *name* je název skupiny. Údaj *himark* je nejvyšší číslo článku použité v dané skupině. Údaj *lowmark* je nejnižší aktivní číslo článku ve skupině. Abychom si demonstrovali jejich funkci, představme si následující situaci. Mějme nově vytvořenou skupinu. *Himark* i *lowmark* jsou 0, protože skupina neobsahuje žádné články. Po zadání pěti článků budou mít hodnoty 1 a 5. Pokud smažeme článek s číslem 5, nic se nezmění – *himark* bude pořád 5 aby se zajistilo, že toto číslo nebude znovu použito a *lowmark* bude 1 – číslo nejnižšího aktivního článku. Když nyní zrušíme článek 1, zůstane *himark* stále 5, *lowmark* však bude 2, protože článek 1 už neexistuje. Pokud nyní vytvoříme nový článek, bude mít číslo 6 a hodnota *himark* se zvýší na 6. Článek číslo 5 kdysi existoval a jeho číslo proto nebude použito znovu. *Lowmark* bude stále 2. Tento mechanismus umožňuje snadno článkům alokovat jedinečná čísla a také jednoduše odhadnout, kolik článků konference přibližně obsahuje – *himark-lowmark*.

Poslední pole může obsahovat následující příznaky:

y Je povoleno přímé publikování článků na serveru.

n Přímé publikování na serveru není povoleno. Tím se zabrání klientským programům v umisťování článků na server. Nové články mohou být přijaty pouze z jiných serverů.

m Skupina je moderovaná. Články publikované ve skupině se pošlou moderátorovi, který je případně zveřejní. Většina skupin je nemoderovaných.

j Články v této skupině se neuchovávají, pouze předávají. Díky tomu server přijme článek pro danou skupinu, hned jej však předá okolním serverům. Klientskými programy nepůjde článek z tohoto serveru přečíst.

x V této skupině není možné publikovat články. Články se doručují na server pouze z jiných serverů, klientské programy nemohou přímo zapsat nový článek.

=*něco* Články se lokálně předávají do skupiny *něco*.

V naší jednoduché konfiguraci budeme uchovávat pouze několik skupin, proto bude soubor `/var/lib/news/active` vypadat takto:

```
control 0000000000 0000000001 y
junk 0000000000 0000000001 y
rec.crafts.brewing 0000000000 0000000001 y
rec.crafts.brewing.ales 0000000000 0000000001 y
rec.crafts.brewing.badtaste 0000000000 0000000001 y
rec.crafts.brewing.brandy 0000000000 0000000001 y
rec.crafts.brewing.champagne 0000000000 0000000001 y
rec.crafts.brewing.private 0000000000 0000000001 y
```

Hodnoty *himark* a *lowmark* odpovídají hodnotám při vytvoření nové konference. Jejich hodnoty budou vypadat úplně jinak pro konference, které jsou již nějakou dobu aktivní.

Soubor `newsgroups` je ještě jednodušší. Obsahuje jednořádkové popisy skupin. Některé klientské programy jej dokáží číst a zobrazují tyto údaje uživateli, aby se mohl snáze rozhodnout, do které skupiny se chce přihlásit.

Formát souboru `newsgroups` je jednoduchý:

```
name description
```

První údaj je název skupiny, druhý údaj její jednořádkový popis.

Protože chceme mít popsané skupiny podporované naším serverem, bude soubor `newsgroups` vypadat takto:

```
rec.crafts.brewing.ales         Home brewing Ales and Lagers
rec.crafts.brewing.badtaste     Home brewing foul tasting brews
rec.crafts.brewing.brandy       Home brewing your own Brandy
rec.crafts.brewing.champagne    Home brew your own Champagne
rec.crafts.brewing.private      The Virtual Brewery home brewers group
```

Předávání zpráv

Systém INN umožňuje administrátorovi nastavit, které skupiny budou předávány dalším news serverům a jak. Nejběžnější způsob používá dříve popsaný protokol NNTP, nicméně INN je schopen zajistit distribuci i jinými způsoby, například pomocí UUCP.

Soubor newsfeeds

Soubor `newsfeeds` říká, kam mají být články posílány. Normálně je umístěn v adresáři `/etc/news`.

Formát souboru `newsfeeds` je na první pohled složitý. Popíšeme si zde základy, podrobnosti naleznete na manuálové stránce `newsfeeds(5)`. Formát souboru vypadá takto:

```
# formát souboru newsfeeds
site:pattern:flags:param
site2:pattern2\
        :flags2:param2
```

Zásobování nějakého systému zprávami je vždy definováno na jednom řádku, případně může pokračovat na dalším řádku při použití symbolu \. Údaje na řadku jsou odděleny dvojtečkou. Znak # na začátku řádku představuje komentář.

Údaj *site* definuje systém, kterého se záznam týká. Název systému může být zadán jakkoliv, nemusí jít o doménové jméno systému. Toto jméno bude uvedeno později v jiné tabulce, která programu **innxmit** zajišťuje konverzi názvů systémů na názvy hostitelů. Tento program protokolem NNTP přenáší zprávy na vzdálené servery. Pro jednotlivé systémy může být uvedeno více záznamů, každý bude posuzován samostatně.

Údaj *pattern* definuje, které skupiny budou danému systému posílány. Implicitně se posílají všechny skupiny, takže pokud vám to vyhovuje, nemusíte tento údaj vyplňovat. Údaj bývá obvykle čárkami oddělený seznam masek, které popisují názvy skupin. Znaku * vyhovuje žádný nebo více libovolných znaků, znak . nemá zvláštní význam, znak ! na začátku výrazu znamená negaci a znak @ na začátku jména říká „nepředávej zprávy zveřejněné v této skupině“. Seznam se čte a zpracovává zleva doprava, takže méně přesná pravidla musí být uvedena dříve. Hodnota

```
rec.crafts.brewing*,!rec.crafts.brewing.poison,@rec.crafts.brewing.private
```

bude posílat celou hierarchii zpráv *rec.crafts.brewing* vyjma *rec.crafts.brewing.poison*. Nebude posílat zprávy, které byly na našem systému zveřejněny v konferenci *rec.crafts.brewinf.private*, ty budou přístupné pouze uživatelům našeho serveru. Pokud bychom prohodili pořadí prvního a druhého pravidla, bylo by první přepsáno druhým a přenášely by se i zprávy konference *rec.crafts.brewing.poison*. To samé platí i pro první a poslední pravidlo; obecně musíte přesnější pravidla umístit vždy až za méně přesná pravidla, jinak se neuplatní.

Údaj *flags* řídí a omezuje předávání zpráv na vzdálený systém. Jedná se o čárkami oddělený seznam položek podle následující tabulky:

`<`*`size`*	Článek musí být kratší než *size* bajtů.
`A`*`items`*	Kontrola článku. *items* může být `d` (musí obsahovat hlavičku `Distribution`) nebo `p` (neprovádí se kontrola hlavičky `Path`) nebo obojí.
`B`*`high`*`/`*`low`*	Velikost interního bufferu před zápisem na výstup.
`H[`*`count`*`]`	Článek musí mít méně než *count* skoků, implicitní hodnota je 1.
`I`*`size`*	Velikost interního bufferu (pro souborový přenos).
`M`*`pattern`*	Pouze moderované skupiny vyhovující masce.
`N`*`pattern`*	Pouze nemoderované skupiny vyhovující masce.
`S`*`size`*	Zahájí zpracování je-li ve frontě více než *size* bajtů.
`T`*`type`*	Typ předávání: `f` (soubor), `m` (nalévání, *param* udává kam budou zprávy nalévány), `p` (roura na program), `c` (poslání na stdin procesu uvedeného v *param*) a `x` (jako `c`, obsluhuje však příkazy na stdin).
`W`*`items`*	Co zapisovat: `w` (velikost článku v bajtech), `f` (úplnou cestu), `g` (první skupinu), `m` (identifikátor zprávy), `n` (relativní cestu), `t` (čas přijetí), * (název systémů, které článek dostaly), `N` (hlavička skupin), `D` (distribuční hlavička), `H` (všechny hlavičky), `o` (přehled dat) a `R` (data pro replikaci).

Údaj *param* má speciální význam podle typu předávání zpráv. Ve většině konfigurací bude obsahovat název souboru, do nějž se mají předávané zprávy zapisovat. V jiných případech jej můžete vynechat. Ještě v jiných případech bude mít úplně jiný význam. Pokud budete chtít nastavit něco neobvyklého, význam pole *param* je podrobně popsán na manuálové stránce `newsfeeds(5)`.

Jako speciální název systému může být uvedeno `ME` a tímto záznamem musí soubor začínat. Tímto záznamem definujeme příjem zpráv naším systémem. Pokud údaj `ME` obsahuje distribuční seznam, bude tento seznam připojen ke všem dalším údajům o systémech. Tím se umožní automa-

ticky předávat některé skupiny, automaticky zablokovat jiné a to bez nutnosti opakovat u každého systému potřebné masky.

Už jsme se zmínili, že je možné generovat data o skupině, která usnadní klientským programům orientaci. Zajišťuje to program **overchan**, který je součástí distribuce INN. Použijeme jej tak, že vytvoříme speciální lokální distribuci `overview`, která bude články předávat programu **overchan** a ten pak zajistí generování potřebných dat.

Náš systém bude předávat zprávy pouze jedinému externímu systému, tím je počítač na Groucho Marx University, a budeme mu posílat články ve všech skupinách kromě *control* a *junk*, *rec.crafts.brewing.private*, která se ukládá lokálně, a *rec.crafts.brewing.poison*, u níž nechceme, aby cizí lidé viděli naše příspěvky.

K přenosu zpráv protokolem NNTP na počítač *news.groucho.edu* použijeme program **nntpsend**. Tento program vyžaduje „souborovou" doručovací metodu a zápis cesty ke článku a identifikátoru článku. Všimněte si, že údaj *param* nastavujeme na název souboru. O příkazu **nntpsend** budeme podrobněji hovořit za chvíli. Výsledný konfigurační soubor `neewsfeeds` bude vypadat takto:

```
# /etc/news/newsfeeds pro Virtual Brewery
#
# Posílat vše kromě control a junk
ME:!control,!junk::
#
# Generovat přehledy pro všechny skupiny
overview::Tc,WO:/usr/lib/news/bin/overchan
#
# Předávat na Groucho Marx University vše kromě skupiny private
# a kromě příspěvků na rec.crafts.brewing.poison.
gmarxu:!rec.crafts.brewing.poison,@rec.crafts.brewing.private:\
        Tf,Wnm:news.groucho.edu
#
```

Soubor nntpsend.ctl

Program **nntpsend** řídí přenos článků protokolem NNTP tím, že volá program **innxmit**. Příklad použití příkazu **nntpsend** jsme už viděli, i on má však vlastní konfigurační soubor, pomocí nějž můžeme nastavit některé parametry předávání zpráv.

Příkaz **nntpsend** očekává, že pro systémy, jimž má předávat zprávy, bude mít k dispozici soubory s dávkami. Tyto soubory se jmenují `/var/spool/news/out.going/`*`sitename`*. Tyto soubory vytvoří **innd** jako reakci na záznamy v souboru `newsfeeds`, který jsme viděli před chvílí. V poli *param* jsme jako název souboru zadali název systému a tím splníme vstupní požadavky programu **nntpsend**.

Příkaz **nntpsend** používá konfigurační soubor `nntpsend.ctl`, který bývá obvykle uložen v adresáři `/etc/news`.

V souboru `nntpsend.ctl` můžeme systému jemuž předáváme poštu přiřadit plně kvalifikovaný název, omezení týkající se velikosti přenášených zpráv a celou řadu parametrů pro řízení přenosu. Název systému představuje logický identifikátor, který jednoznačně popisuje cílový systém. Obecný formát záznamů v tomto souboru je následující:

```
sitename:fqdn:max_size:[args]
```

Jednotlivé údaje mají následující význam:

sitename	Název systému podle souboru `neewsfeeds`.
fqdn	Plně kvalifikované doménové jméno news serveru, kterému budeme zprávy předávat.
max_size	Maximální velikost zpráv přenášených v jednom přenosu.
args	Nepovinné parametry předávané programu **innxmit**.

V naší konfiguraci používáme velmi jednoduchý soubor `nntpsend.ctl`. Předáváme zprávy pouze jednomu systému. Velikost přenosu omezujeme na 2 MB a kromě toho předáváme programu **innxmit** parametr, kterým nastavujeme 180 sekund dlouhý timeout. Pokud by byl náš systém větší a zásobovali bychom zprávami více serverů, vytvořili bychom pro každý z nich samostatný řádek, který by vypadal podobně jako tento:

```
# /etc/news/nntpsend.ctl
#
gmarxu:news.groucho.edu:2m:-t 180
#
```

Řízení přístupu programů pro čtení zpráv

Ještě před několika lety bylo obvyklé, že různé organizace poskytovaly veřejný přístup ke svým news serverům. Dnes už se dá veřejný news server najít jen těžko, většina organizací si řídí, kdo má k jejich serverům přístup, typicky jej omezuje na uživatele svého systému. Toto řízení přístupu podporuje i program INN.

Soubor incoming.conf

V počátečním popisu programu INN jsme říkali, že jeho efektivita a velikost je dána tím, že mechanismy distribuce zpráv a čtení zpráv jsou od sebe odděleny. V souboru `/etc/news/incoming.conf` se definuje, kteří hostitelé vám budou protokolem NNTP předávat zprávy a zároveň se zde definují některé parametry, které se tohoto předávání týkají. Hostitelé, kteří nejsou uvedeni v tomto souboru a připojí se k vašemu systému, nebudou obslouženi démonem **innd**, ale démonem **nnrpd**.

Syntaxe souboru `/etc/news/incoming.conf` je velice jednoduchá, nicméně bude užitečné si vyjasnit používanou terminologii. Může obsahovat tři platné typy údajů: dvojice klíč/hodnota, kterými se definují parametry a jejich hodnoty; partnery, definující hostitele, kteří nám mohou posílat zprávy protokolem NNTP; a skupiny, představující mechanismus, jak uplatnit určité dvojice klíč/hodnota na určité skupiny partnerů. Dvojice klíč/hodnota mohou mít tři různé rozsahy platnosti. Globální dvojice platí pro všechny partnery definované v souboru. Dvojice partnera platí pouze pro daného partnera. A konečně dvojice skupin platí pro všechny partnery v dané skupině. Specifičtější definice přepisují méně specifické definice: tedy nastavení partnera přepíše nastavení skupiny a to zase přepisuje globální nastavení.

Složené závorky (`{ }`) označují začátek a konec specifikace skupin a partnerů. Znak # znamená, že zbytek řádku za ním je komentář. Dvojice klíč/hodnota se oddělují dvojtečkou a uvádí se vždy jedna dvojice na jednom řádku.

Používaných klíčů je celá řada. Mezi nejčastější patří:

hostname	Klíč definuje čárkou oddělený seznam plně kvalifikovaných názvů nebo IP adres partnerů, kteří nám posílají články. Pokud není tento klíč uveden, je jméno hostitele nastaveno na hodnotu odpovídající názvu partnera.
streaming	Tento klíč udává, zda jsou od daného hostitele povoleny proudové příkazy. Jeho hodnota je booleovská a implicitně je `true`.

max-connections	Maximální počet spojení povolený od dané skupiny nebo hostitele. Nulová hodnota znamená neomezený počet spojení (můžete použít také hodnotu `none`).
password	Tento klíč umožňuje definovat heslo, kterým se musí vzdálený systém prokázat předtím, než vám začne posílat články. Implicitně se heslo nevyžaduje.
patterns	Tento klíč definuje diskusní skupiny, které od vzdáleného systému přijímáme. Hodnota klíče se zadává úplně stejně jako údaje v souboru `newsfeeds`.

V našem příkladu očekáváme, že nám zprávy bude posílat pouze jediný hostitel. Nebudeme po něm požadovat heslo, nicméně zajistíme si, aby nám zvenčí nemohly přijít žádné články do našich privátních skupin. Soubor `incoming.conf` bude vypadat takto:

```
# Soubor incoming.conf pro Virtual Brewery.

# Globální nastavení
streaming:       true
max-connections: 5

# Povolujeme příjem článků z lokálního hostitele.
peer ME {
    hostname: "localhost, 127.0.0.1"
}

# Všechno kromě privátní skupiny nám posílá groucho..
peer groucho {
    hostname: news.groucho.edu
    patterns: !rec.crafts.brewing.private
}
```

Soubor nnrp.access

Už jsme říkali, že programy pro čtení zpráv a vlastně všechny hostitele neuvedené v souboru `incoming.conf` po připojení na server INN obsluhuje program **nnrpd**. Tento program používá soubor `/etc/news/nnrp.access`, v němž je definováno, kdo může náš news server používat a jak.

Soubor `nnrp.conf` má podobnou strukturu jako ostatní konfigurační soubory. Obsahuje seznam masek, které se porovnávají proti názvu nebo IP adrese hostitele a údaje definující, jaká práva bude daný hostitel mít. Každý záznam se uvádí na samostatném řádku a jednotlivé údaje v něm se oddělují dvojtečkami. Použita bude vždy poslední vyhovující položka v souboru, takže se opět musí nejprve uvést obecné položky a až po nich konkrétnější záznamy. V každém záznamu se uvádějí tyto údaje:

Název nebo IP adresa	Tento údaj odpovídá pravidlům podle `wildmat(3)`. Jedná se o masku definující název hostitele nebo jeho IP adresu.
Oprávnění	Toto pole definuje práva daného hostitele. Je možno nastavit dvě práva: `R` povoluje čtení zpráv a `P` povoluje publikování zpráv.
Uživatelské jméno	Tento údaj je nepovinný a definuje uživatelské jméno pod nímž se musí NNTP klient přihlásit předtím, než bude moci publikovat zprávy. Pole může zůstat prázdné. Pro čtení zpráv se nepožaduje autentikace.
Heslo	Toto pole může být prázdné, jedná se o heslo vztahující se k uživatelskému jménu. Pokud bude pole prázdné, nebude se pro publikování zpráv požadovat autentikace.

Skupiny — Toto pole definuje masku udávající skupiny, k nimž má klient přístup. Maska se řídí stejnými pravidly jako v souboru `newsfeeds`. Implicitní hodnota jsou žádné skupiny, takže obvykle budete muset nějakou masku nastavit.

V příkladu virtuálního pivovaru povolujeme přístup všem NNTP klientům v naší doméně jak pro čtení, tak pro publikování zpráv. Kromě toho všem klientům povolujeme přístup jen pro čtení ke všem skupinám kromě naší privátní skupiny. Soubor `nnrp.access` bude vypadat takto:

```
# Virtual Brewery - nnrp.access
# Povolujeme veřejné čtení všech skupin kromě privátní.
*:R:::*,!rec.crafts.brewing.private

# Hostitelé v naší doméně mohou číst a publikovat cokoliv
*.vbrew.com:RP::*
```

Vyřazování článků

Když na server dorazí článek, uloží se na disk. Aby měl celý systém news smysl, musí být tyto zprávy po nějakou dobu dostupné uživatelům, takže větší servery mohou spotřebovávat značný objem diskového prostoru. Abychom zajistili efektivní využití diskového prostoru, je možné články po nějaké době automaticky mazat. Tomuto procesu říkáme *vyřazování článků*. Systém INN samozřejmě nabízí způsoby, jak zajistit automatické vyřazování článků.

Soubor expire.ctl

K mazání vyřazovaných článků slouží program **expire**. Ten se řídí souborem `/etc/news/expire.ctl`, v němž jsou definována pravidla pro vyřazování.

Syntaxe souboru `/etc/news/expire.ctl` je velmi jednoduchá. Stejně jako u většiny souborů se prázdné řádky a řádky začínající symbolem # ignorují. Obecně se zadává jedno pravidlo na jednom řádku. Každé pravidlo definuje, jak se bude řešit vyřazování u skupin, které odpovídají zadané masce. Syntaxe pravidel je následující:

pattern:*modflag*:*keep*:*default*:*purge*

Jednotlivé položky mají následující význam:

pattern — Čárkami oddělený seznam masek, který definuje vyhovující skupiny. Shoda se vyhodnocuje funkcí `wildmat`(3). Použije se poslední vyhovující pravidlo, takže pokud zadáte pravidlo pro všechny skupiny (*), musíte je uvést jako první.

modflag — Udává, jak pravidlo platí pro moderované skupiny. Může mít hodnotu `M` definující, že pravidlo platí pouze pro moderované skupiny, `U` říkající, že pravidlo platí pouze pro nemoderované skupiny, anebo `A`, kdy pravidlo platí pro všechny skupiny.

keep — Toto pole umožňuje definovat nejkratší dobu, po níž bude uchován článek obsahující v hlavičce údaj `Expires:`. Jednotkou je den a je možné zadat i desetinná čísla, takže můžete uvést například `7.5` s významem „sedm a půl dne“. Můžete také zadat hodnotu `never` pro případ, že chcete články skladovat věčně.

default — Tento údaj je nejdůležitější. Udává, jak dlouho bude uchováván článek bez hlavičky `Expires:`. Většina článků tuto hlavičku nemá. Hodnota se zadává stejně jako u údaje *keep* a i zde je povoleno nastavení `never`.

purge Toto pole udává nejdelší dobu, po níž bude uchováván článek s hlavičkou `Expires:`. Zadává se stejně jako údaje *keep* a *default*.

Naše požadavky jsou jednoduché. Všechny články ve všech skupinách budeme uchovávat 14 dní, články s nastavenou hlavičkou `Expires:` budeme uchovávat 7 až 21 dní. Skupina *rec.crafts.brewing.private* je naše privátní a nebudeme z ní žádné články vyřazovat:

```
# Soubor expire.ctl pro Virtual Brewery

# Vyřazení článků po 14 dnech, 7-21 pro články
# s hlavičkou Expires.
*:A:7:14:21

# Interní skupina, z té články nevyřazujeme.
rec.crafts.brewing.private:A:never:never:never
```

Uveďme si ještě jeden záznam, který se může v souboru `/etc/news/expires.ctl` objevit. Tento soubor může obsahovat právě jeden takovýto řádek:

```
/remember/:days
```

Tento údaj udává, jak dlouho mají být články evidovány v souboru `history` bez ohledu na to, zda samotný článek byl či nebyl vyřazen. Toto nastavení může být užitečné, pokud je některý ze systémů dodávajících vám zprávy „divný" a má ve zvyku vás znovu a znovu zásobovat starými zprávami. Nastavením údaje `/remember/` zabráníte v opakovaném přijetí článku i v případě, že byl na vašem serveru už vyřazen. Když si váš server pamatuje, že článek už měl, odmítne jej přijmout znovu. Toto nastavení nemá vliv na vyřazování článků, ovlivňuje pouze čas, po který jsou údaje o článcích uchovávány v databázi.

Obsluha řídících zpráv

Stejně jako C News, i INN automaticky zpracovává řídící zprávy. Zároveň nabízí mocný konfigurační mechanismus umožňující nastavit, co se má s kterou zprávou dělat, a kdo může různé řídící zprávy posílat.

Soubor control.ctl

Soubor `control.ctl` má velmi jednoduchou strukturu. Platí pro něj prakticky stejná syntaktická pravidla jako pro ostatní konfigurační soubory. Řádky začínající # se ignorují, záznam může pokračovat na více řádcích pomocí symbolu \ a položky se od sebe oddělují dvojtečkou.

Po přijetí řídící zprávy se zpráva postupně testuje proti jednotlivým pravidlům. Bude použito poslední nalezené vyhovující pravidlo, takže opět platí zásada umisťovat obecná pravidla na začátek a konkrétní na konec. Obecná syntaxe souboru je následující:

```
message:from:newsgroups:action
```

Jednotlivé položky mají následující význam:

message Název řídící zprávy. Typické řídící zprávy si popíšeme za chvíli.

from Maska definující emailovou adresu osoby, která zprávu poslala. Před porovnáváním se provede konverze adresy na malá písmena.

newsgroups Pokud jde o zprávu typu `newgroup` nebo `rmgroup`, tato maska definuje vytvářené nebo rušené skupiny.

action Operace, která se má provést. Provedených operací může být celá řada, popíšeme si je v samostatném seznamu.

Údaj *message* může mít jednu z následujících hodnot:

checkgroups Zpráva vyžaduje resynchronizaci databáze aktivních skupin pro seznam skupin uvedený v této zprávě.

newgroup Požadavek na vytvoření nové skupiny. Tělo zprávy by mělo obsahovat krátký popis nově vytvářené skupiny.

rmgroup Požadavek na zrušení skupiny.

sendsys Zpráva požaduje, aby se autorovi odeslal soubor `sys` daného serveru. Dokument RFC 1036 vyžaduje, aby byly tyto informace veřejně dostupné, protože se podle nich aktualizuje mapa Usenetu.

version Tato zpráva žádá o zaslání názvu hostitele a verze software news serveru.

all Tomuto údaji vyhovují všechny zprávy.

Pole *action* může definovat následující operace:

doit Příkaz se provede. V řadě případů bude navíc administrátorovi odeslána zpráva s oznámením, co bylo provedeno.

doit=*file* Stejné jako *doit*, navíc se do logovacího souboru *file* operace zaznamená. Pokud je jako název souboru uvedeno *mail*, pošle se oznámení poštou. Pokud název souboru není uveden, pošle se zpráva do souboru `/dev/null` a odpovídá samotné akci *doit*. Pokud název souboru začíná znakem /, bude cesta chápána absolutně, v opačném případě se bere relativně k `/var/log/news`.

doifarg Operace se provede pokud měla zpráva parametry. Pokud byla poslána bez parametrů, ignoruje se.

drop Příkaz se ignoruje.

log Na chybový výstup procesu **innd** se pošle zpráva. Tento výstup je obvykle přesměrován na `/var/log/news/errlog`.

log=*file* Stejné jako *log*, umožňuje však explicitně specifikovat soubor, do nějž se oznámení zapíše. Pro soubor platí stejná pravidla jako u *doit=file*.

mail Správci se pošle zpráva s oznámením o požadované operaci. Nestane se nic jiného.

verify-* Pokud operace začíná řetězcem „`verify-`“, provede se autentikace pomocí PGP (nebo GPG)[163].

Následující krátký příklad ilustruje, jak může soubor `control.ctl` vypadat v praxi:

```
## Příchod /etc/news/control.ctl
##
## UPOZORNĚNÍ: Tento soubor nepoužívejte, jde pouze o příklad.

## Obsluha řídících zpráv
all:*:*:mail
checkgroups:*:*:mail
```

163 PGP a GPG jsou nástroje určené k autentikaci nebo šifrování zpráv pomocí veřejného klíče. GPG je free GNU verze PGP. GPG najdete na adrese *http://www.gnupg.org/*, PGP na adrese *http://www.pgp.com*.

```
ihave:*:*:drop
sendme:*:*:drop
sendsys:*:*:log=sendsys
senduuname:*:*:log=senduuname
version:*:*:log=version
newgroup:*:*:mail
rmgroup:*:*:mail

##  Obsluha zpráv osmi nejdůležitějších hierarchií
##  COMP, HUMANITIES, MISC, NEWS, REC, SCI, SOC, TALK
checkgroups:*:comp.*|humanities.*|misc.*|news.*|rec.*|sci.*|soc.*|talk.*:drop
newgroup:*:comp.*|humanities.*|misc.*|news.*|rec.*|sci.*|soc.*|talk.*:drop
rmgroup:*:comp.*|humanities.*|misc.*|news.*|rec.*|sci.*|soc.*|talk.*:drop
checkgroups:group-admin@isc.org:*:verify-news.announce.newgroups
newgroup:group-admin@isc.org:comp.*|misc.*|news.*:verify-news.announce.new-
groups
newgroup:group-admin@isc.org:rec.*|sci.*|soc.*:verify-news.announce.newgroups
newgroup:group-admin@isc.org:talk.*|humanities.*:verify-news.announce.newgroups
rmgroup:group-admin@isc.org:comp.*|misc.*|news.*:verify-news.announce.newgroups
rmgroup:group-admin@isc.org:rec.*|sci.*|soc.*:verify-news.announce.newgroups
rmgroup:group-admin@isc.org:talk.*|humanities.*:verify-news.announce.newgroups

## GNU ( Free Software Foundation )
newgroup:gnu@prep.ai.mit.edu:gnu.*:doit
newgroup:news@*ai.mit.edu:gnu.*:doit
rmgroup:gnu@prep.ai.mit.edu:gnu.*:doit
rmgroup:news@*ai.mit.edu:gnu.*:doit

## LINUX (Newsfeed from news.lameter.com)
checkgroups:christoph@lameter.com:linux.*:doit
newgroup:christoph@lameter.com:linux.*:doit
rmgroup:christoph@lameter.com:linux.*:doit
```


Spuštění programu INN

Balík INN obsahuje skript vhodný pro spouštění systému při startu systému. Tento skript se obvykle jmenuje `/usr/lib/news/bin/rc.news`. Skript čte parametry z dalšího skriptu, obvykle pojmenovaného `/usr/lib/news/innshellvars`, ve kterém se definují názvy souborů a cesty, podle nichž bude **inn** hledat potřebné komponenty. Obecně je rozumné spouštět **inn** v režimu neprivilegovaného uživatele, například `news`.

Abyste zajistili, že se **inn** při startu systému skutečně spustí, zkontrolujte, že je správně nastaven skript `/usr/lib/news/innshellvars` a pak z nějakého skriptu spouštěného při startu systému zavolejte `/usr/lib/news/bin/rc.news`.

Kromě toho se musí pravidelně provádět administrativní operace. Obvykle jsou nakonfigurovány tak, aby je spouštěl démon **cron**. Nejlepší bude nastavit příslušné příkazy v souboru `/etc/crontab`, nebo ještě lépe vytvořit odpovídající soubor v adresáři `/etc/cron.d` v případě, že vaše distribuce tuto možnost podporuje. Příklad takového souboru může vypadat takto:

```
# Soubor /etc/cron.d/inn pro distribuci Debian.
#
SHELL=/bin/sh
```

```
PATH=/usr/lib/news/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Vyřazování zpráv a generování přehledů se dělá každou noc

15 0 * * *      news    news.daily expireover lowmark delayrm

# Každou hodinu se spouští rnews -U. Není to jen pro UUCP systémy,
# ale i pro obsluhu zpráv ve frontě, které tam umístil in.nnrpd
# v případě, že innd zprávy nepřijímal.

10 * * * *      news    rnews -U
```

Uvedený soubor zajistí, že se vyřazování zpráv bude provádět každou noc, a že zprávy ve frontě budou zpracovány každou hodinu. Všimněte si, že se všechny příkazy spouštějí pod uživatelem `news`.

Správa INN: Příkaz ctlinnd

Systém INN obsahuje příkaz sloužící ke každodenní správě. Pomocí příkazu **ctlinnd** je možné nastavovat skupiny a plnění skupin, zjišťovat status serveru, znovu nahrát, zastavit a spustit server.

Přehled syntaxe příkazu **ctlinnd** můžete získat takto:

```
# ctlinnd -h
```

Dále si popíšeme některé nejdůležitější operace prováděné příkazem **ctlinnd**, podrobnosti najdete na jeho manuálové stránce.

Přidání nové skupiny

Nová skupina se přidá následujícím příkazem:

```
ctlinnd newgroup group rest creator
```

Parametry jsou

group Název vytvářené skupiny.

rest Tento parametr se kóduje stejně jako údaj *flags* v souboru `active`. Není-li uveden, uvažuje se hodnota `y`.

creator Jméno osoby, která skupinu vytvořila. Pokud jméno obsahuje mezeru, uzavírá se do uvozovek.

Změna skupiny

Parametry skupiny se změní následujícím příkazem:

```
ctlinnd changegroup group rest
```

Parametry jsou

group Název měněné skupiny.

rest Tento parametr se kóduje stejně jako údaj *flags* v souboru `active`. Není-li uveden, uvažuje se hodnota `y`.

Tento příkaz se používá ke změně moderace skupiny.

Odstranění skupiny

Skupina se odstraní následujícím příkazem:

```
ctlinnd rmgroup group
```

Parametry jsou

group Název rušené skupiny.

Tento příkaz odstraní skupinu ze souboru `active`. Nemá vliv na samotné zprávy. Všechny články ve skupině budou vyřazeny podle běžných pravidel, nebudou však přijímány žádné nové články.

Přečíslování skupiny

Skupina se přečísluje následujícím příkazem:

```
ctlinnd renumber group
```

Parametry jsou

group Název přečíslovávané skupiny. Není-li uveden, přečíslují se všechny skupiny.

Tímto příkazem se vygeneruje správná hodnota ukazatele nejnižšího čísla zprávy.

Povolení/zákaz čtení

Tímto příkazem se povolí nebo zakáže přístup klientských programů pro čtení zpráv.

```
ctlinnd readers flag text
```

Parametry jsou

flag Hodnota `n` zakáže čtení zpráv. Hodnota `y` povolí čtení zpráv.

text Text, který se zobrazí klientskému programu při pokusu o připojení, obvykle vysvětluje důvod, proč byl přístup zakázán. Při opětném povolení přístupu je nutné tento text vymazat.

Tento příkaz neovlivňuje přijímání zpráv, nastavuje pouze možnost jejich čtení.

Zákaz příjmu zpráv

Tímto příkazem se zakáže příjem nových zpráv.

```
ctlinnd reject reason
```

Parametry jsou

reason Text zdůvodňující, proč nejsou zprávy přijímány.

Tento příkaz neovlivňuje příjem zpráv jdoucí přes **nnrpd** (například přes klientské programy), ovlivňuje pouze spojení obsluhované přímo démonem **innd**, například příjem zpráv od jiných serverů.

Povolení příjmu zpráv

Tímto příkazem se povolí příjem nových zpráv.

```
ctlinnd allow reason
```

Parametry jsou

reason Text musí být stejný jako u příkazu *reject*, nebo musí být prázdný.

Tento příkaz má opačný efekt než příkaz *reject*.

Vypnutí news serveru

Tímto příkazem se vypne news server.

```
ctlinnd throttle reason
```

Parametry jsou

reason Důvod vypnutí serveru.

Tento příkaz má stejný efekt jako příkazy `newsreaders no` a `reject`, je užitečný při havarijních zásazích do databáze. Zajistí, že po dobu práce s databází do ní nebude zasahováno.

Restart news serveru

Tímto příkazem se news server restartuje.

```
ctlinnd go reason
```

Parametry jsou

reason Důvod uvedený při vypnutí serveru. Pokud je zadán prázdný řetězec, bude spuštěn celý server. Je-li řetězec neprázdý, budou spuštěny pouze ty funkce, které byly ze stejného důvodu vypnuty.

Tento příkaz slouží k restartu serveru po příkazech **throttle**, **pause** nebo **reject**.

Stav příjmu zpráv

Tímto příkazem se zobrazí stav příjmu zpráv.

```
ctlinnd feedinfo site
```

Parametry jsou

site Název vzdáleného systému podle souboru `newsfeeds`.

Odmítnutí příjmu zpráv

Tímto příkazem se zakáže příjem zpráv od určitého systému.

```
ctlinnd drop site
```

Parametry jsou

site Název vzdáleného systému podle souboru `newsfeeds`. Pokud je parametr prázdný, bude zakázán příjem od všech systému.

Zakázání příjmu se vztahuje pouze k probíhajícímu příjmu zpráv, nejedná se o trvalou operaci. Tento příkaz je užitečný, pokud chcete změnit parametry příjmu zpráv od daného systému a právě z něj příjem zpráv probíhá.

Zahájení příjmu zpráv

Tímto příkazem se zahájí příjem zpráv od určitého systému.

```
ctlinnd begin site
```

Parametry jsou

site Název vzdáleného systému podle souboru `newsfeeds`. Pokud příjem právě probíhá, provede se nejprve příkaz *drop*.

Tento příkaz způsobí, že server znovu načte soubor `newsfeeds`, vyhledá zadaný údaj a zahájí příjem zpráv ze zadaného systému. Tímto příkazem můžete otestovat konfiguraci po změně v souboru `newsfeeds`.

Zrušení článku

Tímto příkazem zrušíte konkrétní článek.

```
ctlinnd cancel Message-Id
```

Parametry jsou

Message-Id Identifikátor rušeného článku.

Tímto příkazem dojde ke zrušení daného článku. Negeneruje zprávu *cancel*.

Konfigurace klienta pro čtení zpráv

Newsreader je program, kterým uživatel prohlíží, ukládá a vytváří články. Na Linuxu existuje několik takových programů. Popíšeme si základní konfiguraci tří nejoblíbenějších programů: **tin**, **trn** a **nn**.

Jeden z nejefektivnějších nástrojů pro čtení news získáte následujícím zápisem:

```
$ find /var/spool/news -name '[0-9]*' -exec cat { }  \ ; | more
```

Tímto způsobem čtou news unixoví fanatici.

Většina programů pro čtení news je však mnohem kultivovanější. Obvykle nabízí celoobrazovkové rozhraní s oddělenými úrovněmi nabízejícími zobrazení všech skupin, do kterých je uživatel přihlášen, zobrazení přehledu všech článků v jedné skupině a konečně i jednotlivých článků. Jako klienti news funguje i řada webových prohlížečů, pokud však chcete samostatný program pro čtení zpráv, dozvíte se zde, jak nakonfigurovat dva nejklasičtější: **trn** a **nn**.

Na úrovni diskusní skupiny zobrazuje většina programů seznam článků, ve kterém je uveden předmět článku a jeho autor. Ve velkých skupinách nemůže uživatel sledovat vzájemnou provázanost článků, i když je možné vystopovat odpovědi na dřívější články.

V odpovědi je většinou zopakován předmět původního článku, před nějž je předsunut řetězec „Re:“. Kromě toho by měla hlavička `References:` článku obsahovat identifikátor článku, na nějž se odpovídá. Seřazením článků podle těchto dvou kriterií vzniknou malé shluky (ve skutečnosti se jedná o stromy) článků, kterým se říká *vlákna*. Jedním z úkolů při psaní programu pro čtení news je vymyslet efektivní schéma pro tvorbu vláken, protože čas potřebný pro takovéto seřazení je úměrný druhé mocnině celkového počtu článků.

V této kapitole se však nebudeme hlouběji zabývat způsobem, jakým jsou vystavěna uživatelská rozhraní. Všechny programy pro čtení news, které jsou v současné době v Linuxu dostupné, disponují dobrou nápovědou, takže by vám práce s nimi neměla činit potíže.

Dále se budeme zabývat pouze administrativními úkoly. Většinou bude řeč o tvorbě databází vláken a evidencí.

Konfigurace programu tin

Nejvšestrannějším programem pro čtení news s ohledem na tvorbu vláken je program **tin**. Napsal ho Iain Leas a je volně odvozen od staršího programu jménem **tass** (jehož autorem je Rich Skren-

ta). Propočet vláken provádí v době, kdy uživatel vstoupí do diskusní skupiny, a kromě případu, kdy používáte spojení protokolem NNTP, je poměrně rychlý.

Na počítači 486DX50 zabere setřídění 1000 článků při čtení přímo z disku zhruba 30 sekund. Při spojení se zatíženým NNTP serverem tato operace může zabrat více než 5 minut[164]. Tuto dobu je možné zkrátit pravidelnou aktualizací indexového souboru, které dosáhnete spuštěním programu **tin** s parametrem `-u`, takže při příštím čtení zpráv bude už databáze k dispozici. Kromě toho můžete **tin** spustit s parametrem `-U`, pak provádí indexaci na pozadí při čtení zpráv.

Program **tin** obvykle ukládá svou databázi vláken do souboru `.tin/index` v domovském adresáři uživatele. Tato konfigurace však může příliš zatěžovat systém, takže možná budete chtít udržovat jedinou kopii těchto souborů na nějakém centrálním místě. Toho dosáhnete tak, že **tin** spustíte jako setuid **news**. Pak bude udržovat všechny databáze v souboru `/var/spool/news/.index`. Při každém přístupu k souboru nebo spuštění příkazového interpretu nastaví efektivní uid na skutečné uid uživatele, který ho spustil[165].

Program **tin** v některých distribucích Linuxu neobsahuje podporu protokolu NNTP, většina distribucí však používá verze s touto podporou. Pokud program spustíte jako **rtin** nebo s volbou `-r`, pokusí se **tin** spojit s NNTP serverem uvedeným v souboru `/etc/nntpserver` nebo v proměnné prostředí NNTPSERVER. Soubor `nntpserver` obsahuje na jediném řádku pouze název serveru.

Konfigurace programu trn

Program **trn** je také následníkem staršího programu pro čtení news, konkrétně programu **rn** (což znamená *read news*). Písmeno „t“ v jeho názvu znamená „threaded“. Napsal ho Wayne Davidson.

Na rozdíl od programu **tin** neumí program **trn** generovat svou databázi vláken za běhu. Místo toho využívá databázi vytvořenou programem **mthreads**, který je pravidelně spouštěn démonem **cron** a provádí aktualizaci indexových souborů.

Když nespustíte program **mthreads**, můžete články číst také, ovšem všechny zprávy typu „Vydělejte si $10000 za 10 minut“ budete mít rozházeny mezi ostatními užitečnými články a nepůjde je jednoduše přeskočit...

Budete-li chtít zapnout tvorbu vláken pro konkrétní diskusní skupinu, spustíte program **mthreads** a jako parametry mu na příkazové řádce předejte seznam diskusních skupin. Seznam vytvoříte stejným způsobem, jaký jste použili při tvorbě souboru `sys`:

```
mthreads 'comp,rec,!rec.games.go'
```

Výše uvedený zápis povolí vytváření vláken pro všechny skupiny **comp** a **rec** s výjimkou skupiny **rec.games.go** (lidé, kteří jsou schopni hrát Go, žádná vlákna nepotřebují). Potom spustíte stejný program bez parametrů, aby mohl zařadit nově příchozí články. Uspořádání všech skupin, které máte uvedeny v souboru `active`, lze zapnout pomocí řádkového paramteru **all**.

Pokud dostáváte zprávy jen přes noc, budete obvykle spouštět program **mthreads** jen jednou denně ráno, ale podle potřeby to může být i častěji. Systémy s velkým provozem možná budou používat program **mthreads** v režimu démona. Pokud ho totiž spustíte při zavádění systému s parametrem `-d`, umístí sám sebe na pozadí a každých deset minut bude zjišťovat, zda nedošly něja-

164 Výkon se významně zlepší, pokud roztřiďování do vláken provádí přímo NNTP server a umožní klientovi tuto databázi stáhnout. INN to například umožňuje.

165 Proto když spustíte **tin** jako root, dostanete hromadu chybových hlášení. Jako root byste ovšem takové věci stejně dělat neměli.

ké nové články, a pokud ano, pak je zařadí. Chcete-li spouštět program **mthreads** jako démona, umístěte do skriptu `rc.news` následující řádek:

```
/usr/local/bin/rn/mthreads -deav
```

Parametr `-a` zapíná automatickou tvorbu vláken pro nové skupiny, parametr `-v` povoluje zápis zpráv do logovacího souboru `mt.log` v adresáři, ve kterém máte nainstalován program **trn**.

Staré články, které již nejsou k dispozici, je třeba z indexových souborů pravidelně mazat. Implicitně jsou odstraňovány pouze články, jejichž číslo je pod nejnižším číslem článku ve skupině[166]. Články nad tímto číslem, jejichž doba platnosti už vypršela (protože nejstaršímu článku mohla být za pomoci hlavičkového pole `Expires:` přidělena dlouhá doba platnosti), lze odstranit pomocí parametru `-e`, který spustí takzvané pokročilé vyhodnocování doby platnosti. Pokud program **mthreads** běží jako démon, způsobí parametr `-e`, že přejde do tohoto režimu jednou denně krátce po půlnoci.

Konfigurace programu nn

Program **nn**, jehož autorem je Kim F. Storm, o sobě tvrdí, že je nástrojem, jehož hlavním cílem je nečíst news. Jeho název je zkratkou z „No News" a má následující moto: „No News is good news. **nn** is better." („Žádné zprávy – dobré zprávy. **nn** je lepší.")

Aby dosáhl tohoto smělého cíle, přichází program **nn** s velkou zásobou podpůrných nástrojů, které dovolují nejenom generovat vlákna, ale umožňují také důkladnou kontrolu konzistence databází, účtování, získávání statistik využití a omezení přístupů. Součástí balíku je i administrativní program **nnadmin**, který dovoluje provádět tyto úkoly interaktivně. Je velmi intuitivní, takže se jím zde nebudeme zabývat a zaměříme se pouze na vytváření indexových souborů.

Správce databáze vláken programu **nn** se nazývá **nnmaster**. Obvykle běží jako démon spuštěný z nějakého `rc` skriptu. Spouští se takto:

```
/usr/local/lib/nn/nnmaster -l -r -C
```

Tento zápis povolí vytváření vláken pro všechny diskusní skupiny, které jsou uvedeny v souboru `active`.

Jinou možností je pravidelně spouštět program **nnmaster** démonem **cron** a přitom mu předat seznam příslušných skupin. Tento seznam je podobný seznamu zapsaných skupin v souboru `sys`. Liší se jen v tom, že místo čárek používá jako oddělovače mezery. Namísto názvu **all** se k označení všech skupin používá prázdný řetězec `""`. Program lze spustit například takto:

```
# /usr/local/lib/nn/nnmaster !rec.games.go rec comp
```

Všimněte si, že v tomto zápisu záleží na pořadí prvků v seznamu. „Vítězí" vždy první záznam. Pokud bychom uvedli řetězec *!rec.games.go* až za řetězcem *rec*, byly by vždy zpracovány všechny články z této skupiny.

Program **nn** nabízí několik metod pro odstraňování článků, jimž vypršela doba platnosti. První z nich provádí aktualizaci databáze tak, že projde adresáře všech diskusních skupin a zruší ty záznamy, k nimž již neexistují odpovídající články. Tato operace je implicitní a lze ji aktivovat i pomocí parametru `-E`. Kromě případu, kdy ji provádíte prostřednictvím NNTP, je poměrně rychlá.

166 Pozor – C News (viz kapitola 21) tuto hodnotu automaticky neaktualizují, její aktualizaci musíte zajistit programem **updatemin**.

Druhá metoda se chová úplně stejně jako program **mthreads** při implicitním spuštění. Odstraní totiž pouze ty záznamy, které odkazují na články, jejichž číslo je nižší než nejnižší číslo článku v souboru `active`. Tuto metodu aktivujete pomocí parametru `-e`.

Konečně třetí strategie spočívá ve zrušení celé databáze a opětovném sesbírání všech článků. Aktivujete ji, když program **nnmaster** spustíte s parametrem `-E3`.

Seznam skupin, u nichž se má kontrola platnosti provést, se předává pomocí parametru `-F` stejně jako při spouštění programu **nnmaster**. Pokud program běží jako démon, musíte jej nejprve zabít (pomocí parametru `-k`) a pak znovu spustit s původními parametry. Příslušný příkaz pro spuštění kontroly vypršení platnosti všech skupin podle první metody tedy vypadá takto:

```
# nnmaster -kF ""
# nnmaster -lrC
```

Chování programu **nn** je možné upravit prostřednictvím mnoha dalších parametrů. Chcete-li vědět, jak odstraňovat špatné články nebo provádět výtahy z článků, pak si přečtěte manuálové stránky programu **nnmaster**.

Program **nnmaster** spoléhá na soubor `GROUPS` v adresáři `/var/lib/nn`. Pokud neexistuje, program si ho sám vytvoří. Pro každou diskusní skupinu obsahuje tento soubor řádek začínající názvem skupiny, za nímž může volitelně následovat časové označení a příznaky. Úpravou těchto příznaků je možné změnit nakládání programu s touto skupinou. Nelze však změnit pořadí, ve kterém se skupiny objevují[167]. Přípustné příznaky a jejich význam také najdete v manuálových stránkách programu.

167 Jejich pořadí je dáno pořadím skupin v binárním souboru `MASTER`.

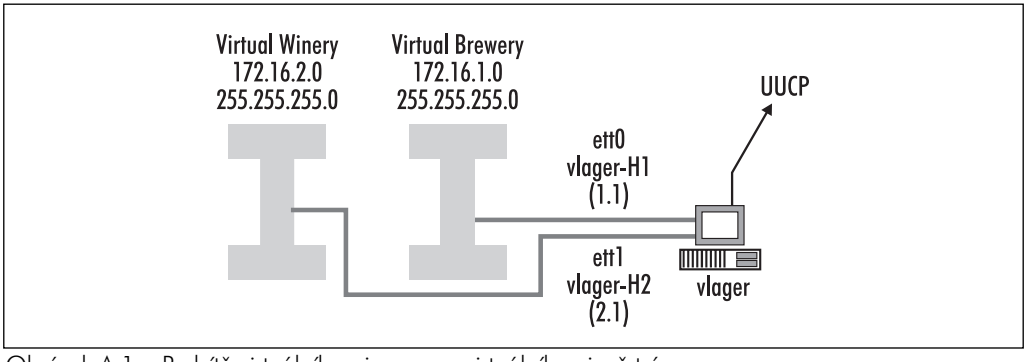
Příklad sítě virtuálního pivovaru

V knize jsme používali příklad sítě, která byla poněkud jednodušší než síť Groucho Marx University a zřejmě se bude více blížit vašim praktickým potřebám.

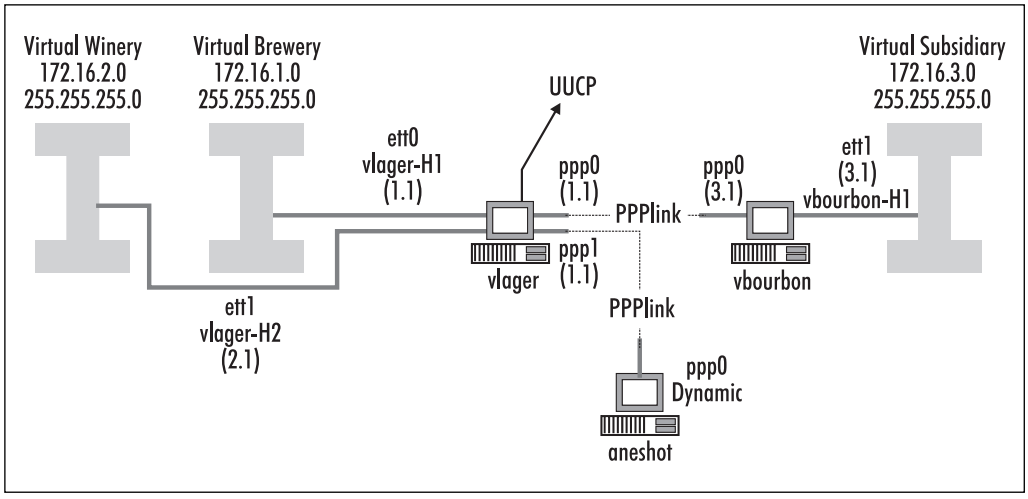
Virtuální pivovar je malá společnost, která, jak název napovídá, vaří virtuální pivo. Aby mohli vařit lépe, mají virtuální pivovarníci počítače propojeny v síti, přičemž všechny počítače používají to nejnovější a nejstabilnější linuxové jádro. Konfiguraci sítě ukazuje obrázek A.1.

Na stejném patře, jenom na druhé straně chodby, sídlí virtuální vinařství, které s virtuálním pivovarem úzce spolupracuje. Vinaři mají svou vlastní síť. Obě společnosti si své sítě propojily. Nejprve vytvořili bránu, která směruje datagramy mezi oběma podsítěmi. Pak si pořídili UUCP linku do okolního světa, kterou přenášejí poštu a news. A nakonec si obstarali i PPP spojení pro přístup na Internet.

Pivovar i vinařství používají podsítě třídy C v rámci třídy B pivovaru, a propojují je branou **vlager**, která zároveň slouží jako UUCP spoj. Tuto konfiguraci znázorňuje obrázek A.2.



Obrázek A.1 – Podsítě virtuálního pivovaru a virtuálního vinařství



Obrázek A.2 – Síť virtuálního pivovaru

Připojení sítě virtuální pobočky

Postupem času se virtuální pivovar rozrostl a otevřel si pobočku v jiném městě. V pobočce mají vlastní síť s IP adresou 172.16.3.0, což je třetí podsíť třídy B pivovaru. Počítač **vlager** funguje jako brána sítě pivovaru a podporuje PPP spojení. Její kolega na síti pobočky se jmenuje **vbourbon** a má IP adresu 172.16.3.1. Strukturu rovněž vidíte na obrázku A.2.

Užitečná zapojení kabelů

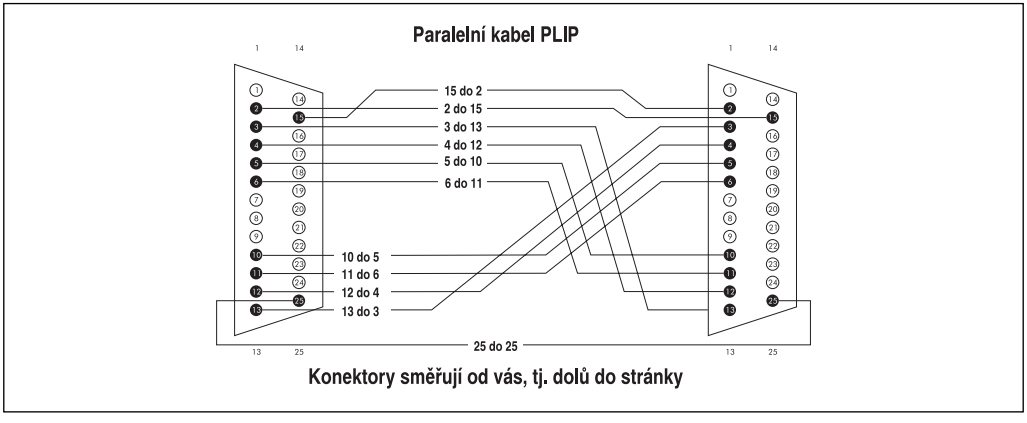
Pokud chcete propojit dva počítače a nemáte Ethernet, můžete použít buď null-modemový sériový kabel nebo paralelní kabel PLIP.

Kabely si můžete samozřejmě koupit, nicméně jejich výroba je velmi jednoduchá a vyjde vás podstatně levněji.

Paralelní kabel PLIP

K sestavení paralelního PLIP kabelu potřebujete dva 25pinové konektory (označení DB-25) a nějaký 11žilový drát. Ten může být dlouhý maximálně 15 metrů. Může a nemusí být stíněný, pokud se ale blížíte k limitu 15 metrů, bude lépe použít stíněný drát.

Podíváte-li se na konektor, měli byste vedle každého pinu přečíst drobná čísla, přičemž jedničku má pin vlevo nahoře (držíte-li konektor širší stranou nahoru) a pin v pravém dolním rohu má číslo 25. PLIP kabel vyrobíte tak, že propojíte vodiče podle následujícího obrázku.



Obrázek B.1 – Paralelní kabel PLIP

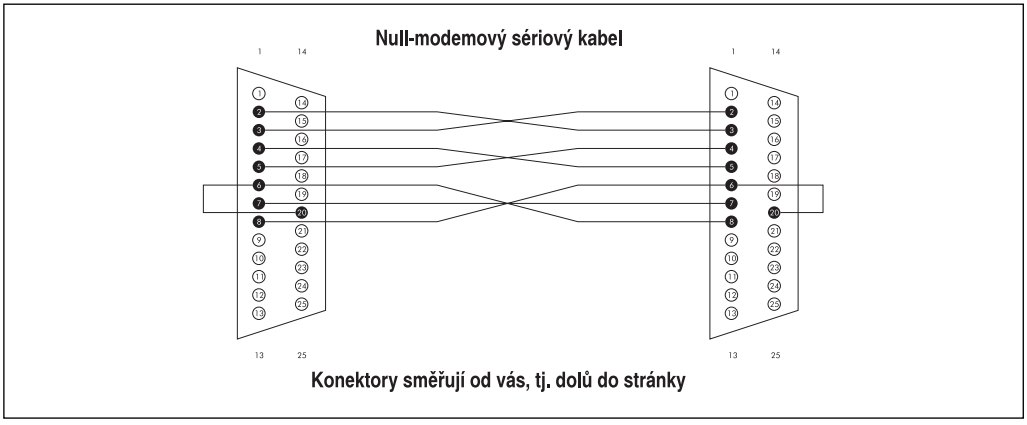
Všechny ostatní vodiče jsou nepropojeny. Pokud používáte stíněný kabel, stínění připojte ke kovovému masivu konektoru *pouze na jednom* konci kabelu.

Null-modemový sériový kabel

Sériový nullmodemový kabel funguje pro SLIP i PPP. Opět budete potřebovat dva DB-25 konektory, tentokrát vám ale postačí osmižilový vodič.

Existují i jednodušší (třívodičové) nullmodemové kabely, zde uvedené zapojení ale umožní používat hardwarové řízení toku – které je lepší než řízení protokolem XON/XOFF – nebo dokonce se úplně obejít bez řízení toku. Zapojení vidíte na obrázku B.2.

Pokud používáte stíněný kabel, stínění připojíte na první pin pouze jednoho konektoru.



Obrázek B.2 – Sériový nullmodemový kabel

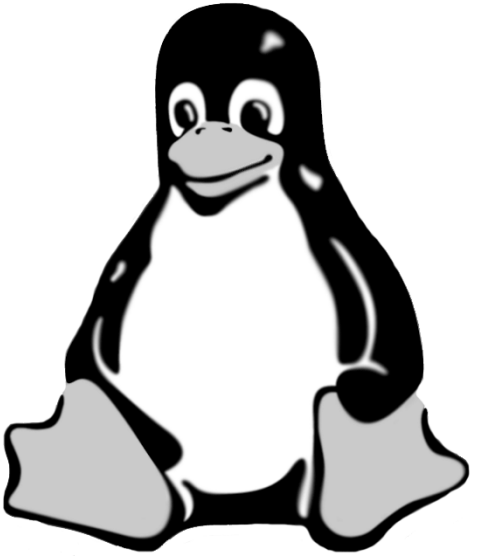
SAGE: The System Administrators Guild

Pokud se vám nepodaří dozvědět se všechno potřebné z příspěvků na *comp.os.linux.** ani v dokumentaci, možná stojí za to připojit se k projektu SAGE, System Administrators Guild, sponzorovanému USENIXem. Hlavním cílem SAGE je vzdělávat profesionální správce systémů. Propojuje správce systémů a síti, vývojáře hardware i software, slouží ke sdílení problémů a řešení, komunikaci s uživateli, dodavateli a managementem.

Mezi hlavní iniciativy SAGE patří:

- Společně s USENIXem pořádá ročně velmi oblíbenou konferenci System Administration Conference (LISA).
- Vydává knihu **Job Descriptions for System Administrators**, první ze série praktických příruček pokrývajících problémy a postupy správců systému.
- Vytváří archivy z přednášek na konferencích správců a z dalších zajímavých pramenů. Archiv najdete na adrese *ftp.sage.usenix.org*.
- Vytváří skupiny působící v různých důležitých oblastech, například publikování, strategie, distribuce informací, vzdělávání, dodavatelé nebo standardy.

Další informace o USENIX Association a o skupině SAGE získáte na americkém telefonním čísle (510) 528-8649, nebo elektronickou poštou na adrese *office@usenix.org*. V elektronické podobě si můžete informace vyžádat na adrese *info@usenix.org*. Roční členství v SAGE stojí 25 dolarů (zároveň musíte být členem USENIXu). Členství zahrnuje odběr časopisů *login:* a *Computing Systems*, slevy účastnických poplatků na konferencích, slevy na odebírané publikace a další.



ČÁST IV

Praktické návody

KAPITOLA 1

Joshua D. Drake (současný), ***Terry Dawson, Alessandro Rubini (předchozí)***

Linux v sítích

Úvod

Nové verze tohoto dokumentu najdete vždy na adrese *http://www.linuxports.com*.

Původně se tento dokument jmenoval Net3/4 Howto a Net3 Howto. Domníváme se, že tyto názvy nebyly řadě čtenářů jasné, proto jsme jej přejmenovali na Linux Networking HOWTO.

Tento dokument podporuje funkci automatických příspěvků na adrese LinuxPorts.Com (*www.linuxports.com/howto/networking/updates.phtml*). Pokud budete chtít do dokumentu něco přidat, můžete to udělat na této adrese.

Proč nehovoříme o problematice „XY“

Dostali jsme spoustu požadavků, abychom se zmínili o určitých tématech a i když jsme se snažili co nejvíce, pořád nám řada informací chybí. Pokud najdete něco, o čem bychom se měli zmínit, kontaktujte nás. Budeme se snažit.

Naše společnost CommandPrompt Inc. distribuuje tento dokument také v tištěné podobě za úplatu. Jistě, dokument je šiřitelný zdarma, nicméně distribuce něco stojí. Stále se snažíme, aby dokumentace byla přesná a aktuální, musíme ale nějak živit své děti. Pokud vás zajímá, co dalšího naše společnost nabízí, kontaktujte nás na adrese *info@commandprompt.com*.

Historie dokumentu

Původní dokument NET-FAQ napsali Matt Welsh a Terry Dawson, aby tak pokryli nejčastěji kladené otázky týkající se síťové problematiky ještě v době, kdy Linux Documentation Project nefungoval. Dokument popisoval první rané experimentální verze síťové podpory v jádře Linuxu. Nástupcem tohoto dokumentu se stal dokument NET-2-HOWTO, jeden z prvních dokumentů LDP, který hovořil o takzvané druhé a později třetí verzi síťové podpory v jádře. Dalším vývojovým krokem je pak právě tento dokument, který se týká pouze čtvrté verze síťové podpory, konkrétně jader 2.x a 2.2.x.

Předchozí verze tohoto dokumentu byly velice objemné vzhledem k jejich širokému záběru. Problém se omezil vznikem řady specializovanějších dokumentů HOWTO. My se proto budeme na takové dokumenty odkazovat a omezíme se pouze na témata, která zatím nejsou popsána jinde.

Odezva

Odezva nás velice zajímá. Můžete nám psát na adresu *poet@linuxports.com*.

Neváhejte nás kontaktovat, pokud narazíte na jakoukoliv chybu nebo na nějaké nepokryté téma.

Jak tento dokument používat

Dokument je organizován shora dolů. První části obsahují informativní údaje, které můžete případně přeskočit. Pak následuje obecný popis síťové problematiky a abyste se mohli věnovat specializovanějším částem, musíte mu rozumět. Zbytek – „specifické informace“ – se dělí na tři hlavní kategorie: informace týkající se Ethernetu a IP, informace týkající se rozšířeného PC hardware a informace o zřídka používaných technikách.

Doporučený postup čtení je následující:

Přečtěte si obecné části

Tyto části se vztahují ke každému nebo téměř každému postupu popisovanému později, takže je pro vás velmi důležité, abyste je pochopili. Na druhé straně předpokládám, že řada čtenářů bude popisovanou problematiku znát.

Berte v úvahu vaši síť

Měli byste vědět, jak je (nebo má být) navržena vaše síť a jaký typ hardwaru a technologií využívá.

Pokud jste připojeni k lokální síti nebo Internetu, přečtěte si část „Ethernet a IP“

V této části popisujeme základní konfiguraci Ethernetu a různé funkce, které Linux poskytuje pro sítě založené na IP – například firewally, složitější směrování a podobně.

Pokud vás zajímají levná lokální řešení a telefonická spojení, přečtěte si následující část

V této části popisujeme technologie PLIP, PPP, SLIP a ISDN, které se často používají na osobních počítačích.

Přečtěte si části týkající se vašich požadavků

Pokud máte i další požadavky než je jen IP a/nebo obvyklý hardware, v závěrečné části se hovoří o jiných protokolech než IP a o speciálním komunikačním hardware.

Proveďte konfiguraci

Měli byste se pokusit nakonfigurovat vaši síť a pečlivě se zaměřit na jakékoliv problémy, které vyvstanou.

Podle potřeby vyhledejte další pomoc

Jestliže se objeví problémy, se kterými vám tento dokument nepomůže, přečtěte si v příslušné části, kde je možné nalézt pomoc nebo nahlásit chyby.

Bavte se!

Používání sítí je zábava, tak si to užijte.

Konvence použité v dokumentu

Nepoužíváme žádné zvláštní konvence, snad jen dávejte pozor na to, jak zapisujeme příkazy. Držíme se klasické unixové dokumentace, a tak každý příkaz, který budete zapisovat, je uveden výzvou příkazového interpretu. V celém dokumentu používáme výzvu „`user%`“ pro příkazy, které nemusí spouštět superuživatel, a „`root#`“ pro příkazy, které musí spustit superuživatel. Zvolili jsme zápis „`root#`“ namísto prostého „#“, abychom předešli záměně s výpisy skriptů, kde řádky začínající znakem # představují komentář.

Pokud někde uvádíme volby jádra, používáme stejný formát jako *menuconfig*. Měli byste mu rozumět i v případě, že (stejně jako já) *menuconfig* nepoužíváte. Pokud si nebudete jisti, jak je co kde vnořeno, prostě si program spusťte, to rozhodně neuškodí.

Obecné informace o používání Linuxu na sítích

Informace o sítích v Linuxu

Informace o použití sítí v Linuxu můžete najít na řadě míst.

Existuje celá řada specializovaných konzultantů, jejich seznam najdete na adrese *http://www.linuxports.com*.

Alan Cox, který v současné době udržuje síťový kód jádra, provozuje také webové stránky, které obsahují spoustu užitečných informací. Najdete je na adrese *http://www.uk.linux.org*.

V hierarchii diskusních skupin najdete skupinu týkající se specificky síťové problematiky na Linuxu, je to skupina *comp.os.linux.networking*.

Pokud se chcete zeptat na něco, co se týká sítí v Linuxu, můžete se přihlásit do specializované poštovní skupiny. Stačí napsat mail

```
To: majordomo@vger.rutgers.edu
Subject: cokoliv vás napadne
Message:
subscribe linux-net
```

Pokud chcete popsat nějaký problém, uveďte co nejvíce relevantních informací. Měli byste zejména uvést verze používaných programů, zejména verzi jádra, verze nástrojů jako *pppd* a *dip* a rovněž přesnou podstatu popisovaného problému. Znamená to věnovat pozornost přesnému znění všech chybových hlášení a používaných příkazů.

Informace o sítích mimo Linux

Pokud sháníte nějaký základní obecný popis TCP/IP sítí, doporučujeme vám následující dokumenty:

TCP/IP Introduction v textové (*ftp://athos.rutgers.edu/runet/tcp-ip-intro.doc*) i postscriptové (*ftp://athos.rutgers.edu/runet/tcp-ip-intro.ps*) verzi.

TCP/IP Administration v textové (*ftp://athos.rutgers.edu/runet/tcp-ip-admin.doc*) i postscriptové (*ftp://athos.rutgers.edu/runet/tcp-ip-admin.ps*) verzi.

Pokud vás zajímají větší podrobnosti o sítících TCP/IP, doporučujeme

Douglas E. Comer: Internetworking with TCP/IP, Volume 1: principles, protocols and architecture, třetí vydání, Prentice Hall publications, 1995.

Pokud vás zajímá, jak vytvářet síťové aplikace v prostředí Unixu, vřele doporučujeme

W. Richard Stevens: Unix Network Programming, Prentice Hall publications, 1990

Momentálně je k dispozici druhé vydání, ale připravuje se nové, dokonce třísvazkové vydání. Podrobnosti najdete na stránkách nakladatelství Prentice Hall (*http://www.phptr.com*).

Dále vám může pomoci skupina *comp.protocols.tcp-ip*.

Dalším důležitým zdrojem informací týkajících se Internetu a protokolové rodiny TCP/IP jsou dokumenty RFC. RFC je zkratka z „Request for Comment“ a jedná se o obvyklý způsob publikování

a dokumentování internetových standardů. Dokumenty RFC najdete na řadě míst. Některé z nich jsou FTP systémy, jiné nabízejí WWW přístup a vyhledávací mechanismy k prohledávání dokumentů podle klíčových slov.

Jedním z míst, kde RFC dokumenty najdete, je adresa *http://www.nexor.com/index-rfc.htm*.

Informace k běžné konfiguraci sítě

Následující části potřebujete znát a pochopit ještě předtím, než se skutečně dostanete ke konfiguraci vaší sítě. Jedná se o základní principy, platné nezávisle na povaze použité sítě.

Co potřebuji na začátku?

Před vytvořením nebo konfigurací sítě budete potřebovat pár věcí. Nejdůležitější jsou:

Aktuální zdrojový kód jádra (nepovinné)

Upozornění:

Většina současných distribucí obsahuje podporu sítí, takže nebude nutné jádro překládat. Pokud používáte normální hardware, neměl by být v ničem problém. Typickým příkladem „normálního hardware“ jsou karty 3COM, NE2000 nebo Intel. Ocitnete-li se v situaci, že bude nutné jádro upravit, čtěte další text.

Protože jádro, které nyní používáte, nemusí mít podporu pro zamýšlené typy sítí nebo síťových karet, budete pravděpodobně potřebovat zdrojový text jádra, aby pak bylo možné jádro překompilovat s patřičnými volbami.

Pro uživatele hlavních distribucí jako je RedHat, Caldera, Debian nebo Su SE by tato situace nastat neměla. Pokud nepoužíváte exotický hardware, nemělo by být nutné jádro upravovat, ledaže byste potřebovali podporu nějaké velmi speciální funkce.

Nejnovější zdrojové kódy jádra můžete získat na adrese *ftp.cdrom.com*. Nejedná se o oficiální server, nicméně mají obrovskou přenosovou kapacitu. Oficiální server je na adrese *ftp.kernel.org*, ale můžete-li vyhněte se mu. Tento server je trvale přetížený, takže raději použijte nějaké zrcadlo.

Normálně bývá zdrojový text jádra rozbalen do adresáře `/usr/src/linux`. Informace o tom jak aplikovat různé úpravy a jak jádro přeložit, najdete v dokumentu Kernel-HOWTO. Informace o konfiguraci modulů jádra naleznete v dokumentu Modules mini-HOWTO. Odvážnému čtenáři doporučujeme také velmi vyčerpávající soubor `README` a adresář `Documentation` přímo ve zdrojovém textu jádra.

Pokud není explicitně řečeno jinak, doporučuji zůstat u standardních verzí jádra (mají druhou číslici ve verzi sudou). Vývojové verze jader (ty s druhou číslicí ve verzi lichou) mohou mít strukturální nebo jiné změny, které mohou při práci s jiným softwarem ve vašem systému způsobovat problémy. Jestliže si nejste jisti, že byste takovýto druh problémů zvládli (kromě dalších potenciálních softwarových chyb), tyto verze nepoužívejte.

IP adresy, vysvětlení

IP adresy se skládají ze čtyř bajtů. Je zvykem je zapisovat v takzvané „tečkové notaci“. Každý bajt je převeden na desítkové číslo (0-255), nuly na začátku se nepíší a čísla se oddělují tečkou. IP adresu má každé rozhraní hostitele nebo směrovače. Za určitých okolností je sice povoleno na jednom stroji u všech rozhraní použít stejnou IP adresu, ale obvykle má každé rozhraní svoji vlastní.

IP sítě jsou souvislé řady IP adres. Všechny adresy v rámci sítě mají určitou část čísel adresy společnou. Tato společná část se nazývá „síťová část“ adresy. Zbývající čísla jsou nazývána „hostitel-

ská část“ adresy. Počet bitů sdílených všemi adresami sítě se nazývá síťová maska a tato maska také určuje, které adresy do sítě patří a které ne. Vezměme si například:

Adresa hostitele	192.168.110.23
Síťová maska	255.255.255.0
Síťová část	192.168.110.
Hostitelská část	.23
Síťová adresa	192.168.110.0
Vysílací adresa	192.168.110.255

Každá adresa v dané síti po bitovém „and“ se síťovou maskou dává adresu sítě, do níž adresa patří. Adresa samotné sítě má proto mezi všemi adresami na dané síti nejnižší číslo a svoji hostitelskou část má nulovou.

Vysílací adresa je speciální adresa, na které poslouchají všichni hostitelé v síti (navíc, kromě své vlastní adresy). Na tuto adresu se odesílají datagramy, které má obdržet každý hostitel sítě. Odesílají se sem určité typy dat, jako směrovací informace a různá upozornění, takže se dostanou ke všem hostitelům v síti najednou. Existují dva standardy pro určení vysílací adresy. Nejčastěji se používá nejvyšší možná adresa v rámci dané sítě. V předchozím příkladu by to byla adresa 192.168.110.255. Někdy se jako vysílací používá síťová adresa. V praxi se nic nemění, ale všichni hostitelé sítě musí mít hlavně nakonfigurovánu stejnou vysílací adresu.

Z administrativních důvodů byly již v dobách vývoje protokolu IP vyčleněny určité skupiny adres do sítí, které byly seskupeny do tříd. Tyto třídy nabízejí různý rozsah sítí, které je možné alokovat. Rozsahy jsou následující:

Třída sítě	Síťová maska	Síťové adresy
A	255.0.0.0	0.0.0.0 - 127.255.255.255
B	255.255.0.0	128.0.0.0 - 191.255.255.255
C	255.255.255.0	192.0.0.0 - 223.255.255.255
Multicast	240.0.0.0	224.0.0.0 - 239.255.255.255

Které adresy použijete závisí na tom, co přesně děláte. Typicky můžete adresy potřebovat při některé z následujících operací:

Instalace linuxového stroje na již existující síti

Jestliže se chystáte linuxový stroj instalovat na již existující IP síť, musíte kontaktovat osobu, která ji spravuje, a zeptat se na následující informace:

- IP adresa hostitele
- Adresa IP sítě
- Vysílací IP adresa
- Síťová maska
- Adresa routeru
- Adresa DNS serveru

S takto získanými informacemi pak nakonfigurujete vaše linuxové síťové zařízení. Nemůžete si je vymyslet a pak očekávat, že by fungovaly.

Vytvoření zcela nové sítě, která se nebude připojovat k Internetu

Jestliže vytváříte soukromou síť a neočekáváte, že se kdy budete připojovat k Internetu, můžete si vybrat adresy, jaké chcete. Z důvodů jednotnosti (a také bezpečnostních) jsou zde však pro tyto účely určité adresy rezervovány. Specifikace podle RFC1597 je následující:

Adresy rezervované pro privátní sítě

Třída sítě	Síťová maska	Síťové adresy
A	255.0.0.0	10.0.0.0 - 10.255.255.255
B	255.255.0.0	172.16.0.0 - 172.31.255.255
C	255.255.255.0	192.168.0.0 - 192.168.255.255

Nejprve se rozhodněte, jak má být vaše síť velká. Potom vyberete tolik adres, kolik bude třeba.

Kam mám vložit konfigurační příkazy?

K procedurám zavádění systému Linuxu existují různé přístupy. Po zavedení jádra se vždy spouští program *init*. Tento program pak čte svůj konfigurační soubor `/etc/inittab` a začne proces zavádění. *init* má několik odrůd, i když v současnosti se vesměs přechází k verzi podle Systemu V, kterou napsal Miguel van Smoorenburg.

Navzdoru tomu, že program *init* může být pokaždé stejný, způsob řízení zavádění systému se v jednotlivých distribucích liší.

Soubor `/etc/inittab` obvykle obsahuje přibližně takovýto údaj:

```
si::sysinit:/etc/init.d/boot
```

Tento řádek určuje název skriptu příkazového interpretu, který spravuje zaváděcí sekvenci. Tento soubor je obdobou souboru AUTOEXEC.BAT v operačním systému MS-DOS.

Jsou zde obvykle i další skripty, volané zaváděcím skriptem, přičemž síť se často konfiguruje právě jedním z nich.

Jako průvodce vaším systémem může sloužit následující tabulka:

Distribuce	Konfigurace rozhraní a směrování	Inicializace serverů
Debian	`/etc/init.d/network`	`/etc/rc2.d/*`
Slackware	`/etc/rc.d/rc.inet1`	`/etc/rc.d/rc.inet2`
RedHat	`/etc/rc.d/init.d/network`	`/etc/rc.d/rc3.d/*`

Všimněte si, že Debian a RedHat používají ke spouštění systémových služeb celé adresáře skriptů (přičemž důležité informace nejsou typicky uloženy přímo v těchto souborech – RedHat například ukládá veškeré konfigurační údaje do adresáře `/etc/sysconfig`, odkud si je startovací skripty čtou). Pokud vás zajímají podrobnosti o startování systému, podívejte se do souboru `/etc/inittab` a na dokumentaci k programu *init*. Článek o inicializaci různých systémů připravuje také Linux Journal a jakmile se objeví, doporučujeme jej přečíst.

Většina současných distribucí zahrnuje program, který umožňuje konfiguraci mnoha běžných síťových rozhraní. Jestliže jednu z nich máte, před pokusem o ruční konfiguraci se podívejte, jestli to co chcete udělat ručně, nemůžete tímto programem automatizovat:

Distribuce	Konfigurační program sítě
RedHat	`/sbin/netcfg`
Slackware	`/sbin/netconfig`

Vytvoření síťových rozhraní

V mnoha unixových operačních systémech se síťová zařízení objevují v adresáři `/dev`. V Linuxu tomu tak není. V Linuxu jsou síťová zařízení vytvářena dynamicky softwarem, takže nevyžadují přítomnost souborů zařízení.

Ve většině případů je síťové zařízení automaticky vytvořeno ovladačem zařízení při inicializaci a nalezení hardwaru. Například ovladač ethernetového zařízení vytvoří rozhraní `eth[0..n]` postupně po nalezení ethernetového hardwaru. První nalezená ethernetová karta se stane `eth0`, druhá `eth1` a tak dále.

V některých případech, například u zařízení `slip` a `ppp`, jsou síťová zařízení vytvořena zásahem některého uživatelského programu. Použije se stejné postupné číslování zařízení, ale zařízení nejsou automaticky vytvářena v době zavádění systému. Důvod je v tom, že na rozdíl od ethernetových zařízení se počet aktivních zařízení `slip` a `ppp` za běhu počítače mění. Tyto případy si později ještě podrobněji popíšeme.

Konfigurace síťového rozhraní. Jádra 2.0 a 2.2

Jakmile máte všechny programy, které potřebujete, a informace o adrese a síti, můžete přistoupit ke konfiguraci vašich síťových rozhraní. Když mluvíme o konfiguraci síťového rozhraní, mluvíme o procesu přiřazení příslušných adres síťovému zařízení a o nastavení příslušných hodnot pro další nastavitelné parametry síťového zařízení. Nejčastějším programem, který se k tomuto účelu používá, je příkaz *ifconfig* (interface configure).

Většinou příkaz použijete v následující podobě:

```
root# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
```

V tomto případě konfiguruji ethernetové rozhraní `eth0` s IP adresou `192.168.0.1` a síťovou maskou `255.255.255.0`. Dodatek `up` za příkazem sděluje rozhraní, že se má zaktivovat, nicméně nemusíte jej zadat, protože se použije implicitně. Chcete-li rozhraní odpojit, zadáte jednoduše `ifconfig eth0 down`.

Jádro předpokládá při konfiguraci rozhraní určité implicitní hodnoty. Například pro rozhraní můžete určit síťové adresy a vysílací adresy, ale když je neurčíte (viz můj příklad), jádro je odvodí ze zadané síťové masky. Když nezadáte ani masku, jádro se rozhodne podle třídy nakonfigurované IP adresy. V mém případě by jádro předpokládalo na rozhraní konfiguraci sítě třídy C a u rozhraní by nastavilo adresu sítě `192.168.0.0` a vysílací adresu `192.168.0.255`.

Příkaz *ifconfig* má mnoho dalších voleb. Nejdůležitější jsou:

`up`	Tato volba aktivuje rozhraní (je implicitní).
`down`	Tato volba deaktivuje rozhraní.
`[-]arp`	Tato volba u rozhraní nastavuje nebo ruší využití protokolu ARP.
`[-]allmulti`	Tato volba nastavuje nebo ruší příjem všech hardwarových multicastových paketů. Hardwarový multicast umožňuje skupinám hostitelů obdržet pakety adresované na speciální místa určení. Tohle může být důležité v případě využití aplikací, jako jsou videokonference, které normálně nepoužíváte.
`mtu N`	Tento parametr umožňuje nastavit MTU tohoto zařízení.
`netmask <addr>`	Tento parametr umožňuje nastavit síťovou masku sítě, do které toto zařízení patří.

`irq <addr>`	Tento parametr je funkční pouze na určitých typech hardwaru a umožňuje nastavení IRQ tohoto zařízení.
`[-]broadcast <addr>`	Tento parametr umožňuje nastavit příjem datagramů určených na vysílací adresu, nebo jejich příjem zrušit.
`[-]pointopoint <addr>`	Tento parametr umožňuje nastavení adresy stroje na vzdáleném konci dvoubodového spoje, například typu *slip* nebo *ppp*.
`hw <type> <addr>`	Tento parametr umožňuje nastavení hardwarových adres určitých typů síťových zařízení. Na Ethernetu se obvykle nepoužívá, ale je užitečný na jiných sítích, jako je například AX.25.

Jádro 2.2 nabízí celou řadu dalších parametrů, které v předchozí tabulce nejsou uvedeny. Některé nejzajímavější se týkají tunelování a nastavení protokolu IPv6. Parametry programu *ifconfig* v jádrech 2.2 jsou uvedeny v následující tabulce:

`interface`	Název rozhraní. Typicky se uvádí název ovladače následovaný číslem zařízení, například `eth0` pro první ethernetové rozhraní.
`up`	Příznak způsobí aktivaci rozhraní. Použije se implicitně, pokud je rozhraní přiřazena adresa.
`down`	Tento parametr způsobí odpojení rozhraní.
`[-]arp`	Zapíná nebo vypíná použití protokolu ARP na daném rozhraní.
`[-]promisc`	Zapíná nebo vypíná promiskuitní režim rozhraní. Pokud je režim zapnut, rozhraní přijímá všechny pakety jdoucí po síti.
`[-]allmulti`	Zapíná nebo vypíná režim multicast. Je-li zapnut, rozhraní bude přijímat všechny multicastové pakety na síti.
`metric` *`N`*	Tento parametr nastavuje metriku rozhraní.
`mtu` *`N`*	Tento parametr nastavuje MTU rozhraní.
`dstaddr` *`addr`*	Nastavuje vzdálenou IP adresu na dvoubodovém spoji (například PPP). Tento parametr je zastaralý, použijte místo něj `pointopoint`.
`netmask` *`addr`*	Nastavuje síťovou masku rozhraní. Implicitně se použijí masky tříd A, B nebo C podle zadané adresy, je však možné nastavit libovolnou masku.
`add` *`addr prefixlen`*	Přidá rozhraní adresu protokolu IPv6.
`del` *`addr prefixlen`*	Odstraní z rozhraní adresu protokolu IPv6.
`tunnel` *`aa.bb.cc.dd`*	Vytvoří nové zařízení SIT (IPv6 na IPv4), které tuneluje na daný cíl.
`irq` *`addr`*	Nastavuje přerušení používané daným zařízením. Ne všechna zařízení umožňují dynamicky změnit používané přerušení.
`io_addr` *`addr`*	Nastavuje začátek adresového prostoru zařízení.
`mem_start` *`addr`*	Nastavuje počáteční adresu sdílené paměti zařízení. Toto nastavení potřebuje jen málo zařízení.
`media` *`type`*	Nastavuje fyzické médium používané zařízením. Ne všechna zařízení umožňují tuto hodnotu změnit a i u těch, která to dokážou, se liší podporované typy. Typické hodnoty jsou *10base2* (tenký Ethernet), *10baseT* (kroucený 10Mbps Ethernet), *AUI* (externí transceiver) a podobně. Kromě toho je možné zadat typ *auto*, pak se ovladač po-

kusí o automatickou detekci připojeného média. Ne všechny ovladače to podporují.

`[-]broadcast` *addr* — Je-li zadána adresa, nastavuje vysílací adresu rozhraní. V opačném případě nastavuje (nebo maže) příznak IFF_BROADCAST rozhraní.

`[-]pointopoint [`*addr*`]` — Tento příznak zapíná dvoubodový režim rozhraní, což znamená, že se jedná o přímý spoj na jiné zařízení, na kterém neposlouchá nikdo další. Pokud je zadána i adresa, nastavuje se tak adresa vzdáleného konce linky stejně, jako to dělal zastaralý parametr `dstaddr`. Bez adresy nastavuje nebo maže příznak IFF_POINTOPOINT rozhraní.

`hw` *class address* — Nastavuje hardwarovou adresu rozhraní, pokud tuto funkci podporuje ovladač zařízení. Za klíčovým slovem musí být uvedena třída hardwarové adresy a její ASCII ekvivalent. V současné době jsou podporovány třídy *ether* (Ethernet), *ax25* (AMPR AX.25), *ARCnet* a *netrom* (AMPR NET/ROM).

`multicast` — Nastavuje multicastový příznak rozhraní. Obvykle se nastavovat nemusí, protože jej správně nastaví ovladač zařízení.

`address` — IP adresa přiřazovaná zařízení.

`txqueuelen` *length* — Nastavuje délku odesílací fronty zařízení. Pro pomalá zařízení s velkým zpožděním (ISDN, modemy) je vhodné tuto hodnotu snížit, aby větší objemy přenosu (například FTP) nebrzdily interaktivní provoz typu *telnet*.

Příkaz *ifconfig* můžete použít na jakémkoliv síťovém rozhraní. Některé uživatelské programy, jako například *pppd* a *dip*, automaticky konfigurují síťová zařízení ve chvíli, kdy je vytváří, takže ruční použití příkazu *ifconfig* nebývá nutné.

Konfigurace resolveru

Resolver je částí standardní linuxové knihovny. Jeho základní funkcí je poskytnutí služeb převodu „lidských“ názvů, jako je *ftp.funet.fi*, na strojové IP adresy, jako je `128.214.248.6`.

Co je v názvu?

Se vzhledem jmen hostitelů Internetu jste se již asi seznámili, ale jak se sestavují a čtou možná nevíte. Názvy internetových domén jsou z podstaty hierarchické, to znamená, že jejich struktura se podobá stromu. *Doména* je rodina nebo skupina názvů. Tu můžeme rozdělit na *subdomény*. *Doménou nejvyšší úrovně* rozumíme doménu, která není subdoménou. Domény nejvyšší úrovně jsou specifikovány v RFC-920. Příklady nejběžnějších domén nejvyšší úrovně:

COM	komerční organizace
EDU	vzdělávací organizace
GOV	vládní organizace
MIL	vojenské organizace
ORG	jiné organizace
Geografické domény nejvyšší úrovně	dvoupísmenný kód reprezentující danou zemi

Z historických důvodů byla většina domén nejvyšší úrovně, které nejsou podle zemí, určena organizacím v USA, i když Spojené státy mají také vlastní kód země – `.us`. To už dnes neplatí pro řadu domén `.com` a `.org`, které se běžně používají i společnostmi mimo Spojené státy.

Každá z těchto domén nejvyšší úrovně má subdomény. Domény nejvyšší úrovně rozdělené podle zemí, jsou dále často rozděleny na subdomény podle domén `com`, `edu`, `gov`, `mil` a `org`. Například komerční a vládní organizace v Austrálii by měly `com.au` a `gov.au`. Nejedná se ovšem o obecné pravidlo, konkrétní rozdělení závisí na příslušné autoritativní instituci v daném státě.

Další úroveň rozdělení většinou reprezentuje název organizace. Ještě nižší subdomény se liší podle povahy. Často je vyjádřením členění organizace, ale může to být i jakékoliv jiné kritérium, které dává síťovým administrátorům organizace smysl.

Část názvu, která je nejvíce vlevo, je vždy jedinečný název hostitelského stroje – *název hostitele*. Část napravo je pak *název domény* a celý název je *plně kvalifikované doménové jméno*.

Když jako příklad využijeme počítač původního autora tohoto dokumentu, jeho plně kvalifikovaný název je `perf.no.itg.telstra.com.au`. To znamená, že název hostitele je `perf` a název domény je `no.itg.telstra.com.au`. Doménové jméno je odvozeno z domény nejvyšší úrovně, což je v tomto případě jeho stát, tedy Austrálie, adresa patří komerční organizaci, takže jako doménu následující úrovně máme `com`. Název společnosti je (byl) `telstra` a vnitřní pojmenovávací struktura je založena na organizační struktuře. V tomto případě počítač náležel do Information Technology Group, sekce Network Operations.

Typicky bývají názvy kratší, například současný autor používá poskytovatele `systemy.it` a působí v organizaci `linux.it`, bez nějakého toho `com` nebo `org`, takže autorův počítač se jmenuje `morgana.systemy.it` a používá poštovní adresu `rubini@linux.it`. Vlastník domény má právo registrovat hostitele i subdomény, takže například LUG I patří do domény `pluto.linux.it`, protože vlastník domény `linux.it` umožnil vytvořit pro LUG subdoménu.

Jaké informace budete potřebovat

Budete potřebovat vědět, do které domény náleží název vašeho hostitele. Tuto překladovou službu poskytuje resolver provedením žádosti na DNS server, takže budete muset znát IP adresu lokálního nameserveru, který můžete využívat.

Editovat musíte tři soubory a nyní je postupně všechny probereme.

/etc/resolv.conf

Jedná se o hlavní konfigurační soubor resolveru. Jeho formát je velmi jednoduchý. Jedná se o textový soubor s jedním klíčovým slovem na každém řádku. Většinou se zde používají tři klíčová slova:

domain	Toto klíčové slovo určuje název lokální domény.
search	Toto klíčové slovo určuje seznam dalších domén, kde je možné hledat název hostitele.
nameserver	Toto klíčové slovo může být použito vícekrát a určuje IP adresu DNS serveru, na který se můžete obrátit při zjišťování názvů.

Ukázkový `/etc/resolv.conf`:

```
domain maths.wu.edu.au
search maths.wu.edu.au wu.edu.au
nameserver 192.168.10.1
nameserver 192.168.12.1
```

Tento příklad říká, že implicitní název domény připojované za nekvalifikované názvy (názvy hostitelů bez domény) je `maths.wu.edu.au` nebo (v případě neúspěchu) přímo `wu.edu.au`. Jsou zde dva nameservery, přičemž při zjišťování názvu je možné využít oba.

/etc/host.conf

Zde se konfigurují některé položky ovlivňující chování kódu resolveru. Formát tohoto souboru je podrobně popsán na manuálové stránce `resolv+`. Téměř za všech okolností by vám měl fungovat následující příklad:

```
order hosts,bind
multi on
```

Tato konfigurace sdělí resolveru, aby před pokusem o žádost na nameserver kontroloval soubor `/etc/hosts` a aby podle tohoto souboru vracel všechny platné adresy, ne jenom první.

/etc/hosts

Sem se vkládají názvy a IP adresy lokálních hostitelů. Jestliže sem vložíte hostitele, nemusíte při zjišťování jeho IP adresy obtěžovat DNS server. Nevýhodou je to, že tento soubor musíte sami ručně aktualizovat (dochází-li ke změnám). V dobře spravovaném systému se zde objevují pouze údaje pro zpětnovazebné rozhraní loopback a název lokálního hostitele.

```
# /etc/hosts
127.0.0.1    localhost loopback
192.168.0.1  nazev.tohoto.pocitace
```

Můžete použít i více než jeden název hostitele na řádek, viz první údaj v příkladu.

Spuštění DNS serveru

Pokud chcete provozovat vlastní DNS server, lze to snadno provést. Přečtěte si dokument DNS-HOWTO a všechny dokumenty týkající se vaší verze programu BIND (Berkeley Internet Name Domain).

Konfigurace zpětnovazebného rozhraní

Zpětnovazebné rozhraní je speciálním typem rozhraní, které umožňuje provést připojení na sebe samého. K tomu existují různé důvody, například testování síťového softwaru bez ovlivnění provozu sítě. Podle konvence se tomuto rozhraní přiděluje adresa `127.0.0.1`. Takže pokud zkusíte *telnet* na `127.0.0.1`, vždy se dostanete na lokálního hostitele.

Konfigurace tohoto rozhraní je jednoduchá a rozhodně byste ji neměli vynechat (i když se rozhraní většinou nastavuje automaticky při instalaci systému).

```
root# ifconfig lo 127.0.0.1
root# route add -host 127.0.0.1 lo
```

O příkazu *route* si více povíme v následující části.

Směrování

Směrování je velké téma. Dalo by se o něm napsat opravdu hodně. Většina z vás má na směrování velmi jednoduché požadavky, někteří ale ne. Budeme se zde zabývat pouze základy. Jestliže se zajímáte o podrobnější informace, doporučuji vám odkazy ze začátku tohoto dokumentu.

Začněme definicí. Co je IP směrování? Nám se líbí tento popis:

„IP směrování je proces, kterým hostitel s více síťovými připojeními rozhodne, kam odeslat obdržené IP datagramy."

K tomu by se hodil příklad. Představte si typický kancelářský směrovač, mohl by mít linku PPP na Internet, několik ethernetových segmentů pro pracovní stanice a další linku PPP do jiného úřadu. Když směrovač na kterémkoliv ze svých síťových připojení obdrží datagram, bude směrování mechanismem, který rozhodne, na které rozhraní se datagram dále odešle. Směrování potřebují i jednodušší hostitelé. Všichni internetoví hostitelé mají alespoň dvě síťová zařízení – jedním je výše popsané zpětnovazebné rozhraní a druhým je rozhraní použité pro komunikaci se zbytkem sítě – snad Ethernet, PPP nebo sériové rozhraní SLIP.

Dobrá, tak jak tedy směrování funguje? Každý hostitel má speciální seznam směrovacích pravidel, nazývaný *směrovací tabulka*. Tabulka obsahuje řádky většinou s alespoň třemi políčky – adresa určení, název rozhraní, kam se má datagram směrovat a třetí je volitelná IP adresa dalšího stroje, který má datagram provést přes další krok na jeho cestě sítí. V Linuxu je možné tabulku zobrazit následujícím příkazem:

```
user% cat /proc/net/route
```

nebo použitím jednoho z následujících příkazů:

```
user% /sbin/route -n
user% netstat -r
```

Směrovací proces je velice jednoduchý: po příchodu datagramu se jeho cílová adresa (tedy komu je datagram určen) porovná s údaji v tabulce. Vybere se údaj tabulky, který adrese nejlépe odpovídá, a na takto získané rozhraní je datagram předán. Jestliže je vyplněno pole brány, datagram je předán přes specifické rozhraní této bráně, jinak se předpokládá, že adresa určení je uvedeným rozhraním dosažitelná přímo.

Pro manipulaci s touto tabulkou se používá speciální příkaz. Tento příkaz bere parametry příkazového řádku a převádí je na volání systému jádra, kde se žádá o přidání, smazání nebo úpravu údajů směrovací tabulky. Příkaz se nazývá *route*.

Jednoduchý příklad. Představte si ethernetovou síť. Bylo vám řečeno, že se jedná o síť třídy C s adresou 198.168.1.0. Pro vaše využití vám byla přidělena IP adresa 192.168.1.10 a řekli vám, že 192.168.1.1 je směrovač připojený na Internet.

Prvním krokem je konfigurace rozhraní (viz výše) – k tomu využijete příkaz podobný tomuto:

```
root# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
```

Nyní potřebujete přidat do směrovací tabulky údaj, aby jádro vědělo, že datagramy pro všechny hostitele s adresou odpovídající `192.168.1.*` by měly být odeslány na naše ethernetové zařízení. Použili byste přibližně takovýto příkaz:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
```

Všimněte si využití parametru `-net`, který sděluje směrovacímu programu, že tento údaj je síťový směr. Vaše další volba je zde směr `-host`, což je směr specifický pro jednu IP adresu.

Tento směr vám umožní vytvořit IP spojení se všemi hostiteli na vašem ethernetovém segmentu. Ale co pak se všemi IP hostiteli, kteří na vašem ethernetovém segmentu nejsou?

Nastavení směrů pro všechna možná místa určení na síti by bylo zhola nemožné, takže se zde využívá určitý trik. Jedná se o směr `default` – výchozí směr, který odpovídá všem možným místům určení, nicméně odpovídá jim „málo“, takže pokud se nalezne i jiný směr, bude použit tento jiný.

Smyslem je umožnit prohlásit „a všechno ostatní pošlete sem“. V tomto případě by tedy údaj vypadal takto:

```
root# route add default gw 192.168.1.1 eth0
```

Parametr `gw` sděluje příkazu *route*, že následujícím parametrem je IP adresa (nebo název) brány nebo směrovače, kterému budou všechny odpovídající datagramy předány pro další směrování.

Takže vaše kompletní konfigurace by vypadala takto:

```
root# ifconfig eth0 192.168.1.10 netmask 255.255.255.0 up
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add default gw 192.168.1.1 eth0
```

Jestliže se blíže podíváte na vaše síťové soubory `rc`, zjistíte, že alespoň jeden z nich vypadá dost podobně. Tato konfigurace je velmi běžná.

Nyní se zaměřme na trochu komplikovanější směrovací konfiguraci. Představme si konfiguraci dříve zmiňovaného směrovače, podporujícího linku PPP na Internet a segmenty LAN pro pracovní stanice na úřadě. Představme si, že směrovač má tři ethernetové segmenty a jednu linku PPP. Naše směrovací konfigurace by vypadala takto:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add -net 192.168.2.0 netmask 255.255.255.0 eth1
root# route add -net 192.168.3.0 netmask 255.255.255.0 eth2
root# route add default ppp0
```

Každá pracovní stanice by využívala jednodušší formu (prezentovanou výše), pouze směrovač potřebuje určit všechny síťové směry odděleně, protože směrovací mechanismus s využitím implicitní trasy na pracovních stanicích „odchytí“ všechny neznámé datagramy a nechá na směrovači, aby je rozdělil. Možná se divíte, proč zde implicitní směr neurčujeme parametrem `gw`. Důvod je jednoduchý – protokoly sériových linek (PPP a SLIP) mají na svých sítích pouze dva hostitele – jeden na každém konci. Určit hostitele z druhého konce linky jako bránu je zbytečné, protože nic jiného stejně nezbývá. U těchto typů síťových připojení tedy není nutné určit bránu. Jiné typy sítí, jako je Ethernet, arcnet nebo token ring, určení brány vyžadují, protože tyto sítě v sobě podporují velké množství hostitelů.

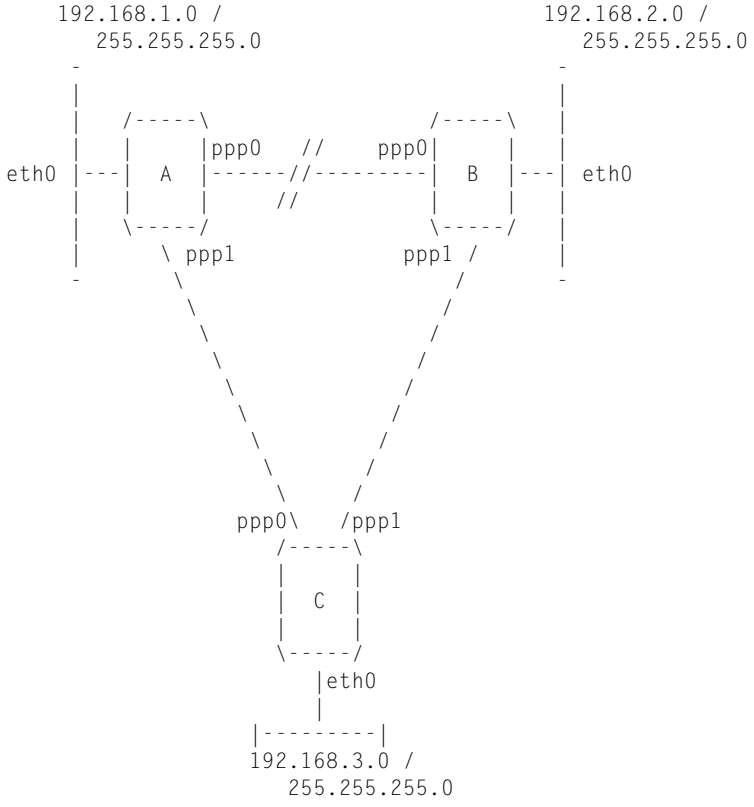
Takže co dělá program routed?

Výše popsaná směrovací konfigurace se nejlépe hodí pro jednodušší sítě, kde je většinou jen jedna možná cesta k cíli. Když je síť složitější, věci se hodně zkomplikují – naštěstí se to většiny z vás netýká a nemusíte se tím zabývat.

Velkým problémem „ručního směrování“ nebo „statického směrování“ (jak bylo popsáno) je situace, kdy selže stroj nebo linka a jediným způsobem směrování jinou cestou (existuje-li vůbec taková) je osobní zásah a provedení příslušných příkazů. To je samozřejmě hloupé, pomalé, nepraktické a riskantní. Je možné toto provádět i automaticky – existuje mnoho postupů, souhrnně nazývaných „dynamické směrovací protokoly“.

Možná jste o těch běžnějších slyšeli – nejznámější jsou RIP (Routing Information Protocol) a OSPF (Open Shortest Path First Protocol). RIP je velmi běžný na malých sítích, jako jsou sítě v malých až středních společnostech nebo v budovách. OSPF je modernější a schopnější zvládnout konfigurace velkých sítí. Je také vhodnější v případě, že existuje velké množství možností cesty sítí. Známé implementace těchto protokolů jsou: *routed* – RIP a *gated* – RIP, OSPF a další. Program *routed* se normálně dodává ve vaší distribuci Linuxu nebo je obsažen v balíku „NetKit“, viz výše.

Příklad kdy a jak můžete využít dynamický směrovací protokol vypadá následovně:



Máme tři směrovače A, B a C. Každý podporuje jeden ethernetový segment s IP sítí třídy C (síťová maska 255.255.255.0). Každý směrovač má také linku PPP ke zbylým směrovačům. Síť je ve tvaru trojúhelníku.

Mělo by být jasné, že směrovací tabulka na směrovači A by měla vypadat takto:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# route add -net 192.168.2.0 netmask 255.255.255.0 ppp0
root# route add -net 192.168.3.0 netmask 255.255.255.0 ppp1
```

Tohle by mělo spolehlivě fungovat až do chvíle, kdy by selhala linka mezi směrovači A a B. V takovém případě by hostitelé na ethernetovém segmentu A nemohli dosáhnout hostitele na ethernetovém segmentu B, protože jejich datagramy by byly směrovány na linku PPP směrovače A, která je nefunkční. Mohou ale stále komunikovat s hostiteli na segmentu C a ti zase s hostiteli na segmentu B, protože linka mezi B a C je funkční.

Když ale A komunikuje s C a C s B, proč by A nemohlo poslat svoje datagramy do B přes C? To je zhruba druh problémů, které řeší dynamické směrovací protokoly jako RIP. Kdyby měly všechny směrovače A, B a C puštěný směrovací démon, pak jsou jejich směrovací tabulky při poruše

některé linky automaticky upraveny, aby odrážely nový stav sítě. Konfigurace takové sítě je snadná – na každém směrovači potřebujete provést pouze dvě věci. V našem případě pro směrovač A:

```
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# /usr/sbin/routed
```

Směrovací démon *routed* při spuštění automaticky nalezne všechny aktivní síťové porty a odesílá a přijímá zprávy na každé síťové zařízení, aby bylo možné určit a aktualizovat směrovací tabulku na hostiteli.

Tohle bylo velice stručné vysvětlení dynamického směrování a jeho použití. Jestliže potřebujete více informací, měli byste se vrátit na začátek dokumentu a pročíst si odkazy.

Důležité postřehy vztahující se k dynamickému směrování jsou:

1. Démona pro dynamické směrování potřebujete na stroji s Linuxem spouštět pouze v případě, že máte na výběr více možných směrů na místo určení. Taková situace může nastat, pokud chcete použít IP maškarádu.
2. Směrovací démon automaticky upravuje vaši směrovací tabulku, čímž odráží změny v síti.
3. RIP je vhodný pro malé až středně velké sítě.

Konfigurace síťových serverů a služeb

Síťové servery a služby jsou ty programy, které umožňují vzdálenému uživateli, aby se stal uživatelem vašeho linuxového stroje. Serverové programy naslouchají na síťových portech. Síťové porty jsou prostředkem adresace určité služby na určitém hostiteli. Podle nich server pozná rozdíl mezi přicházejícím telnetovým a ftp spojením. Vzdálený uživatel vytvoří síťové spojení k vašemu stroji a serverovému programu. Program síťového démona, který na tomto portu naslouchá, spojení přijme a provede. Existují dva způsoby, jakými mohou síťoví démoni fungovat. Oba jsou běžně využívány v praxi:

samostatný

Síťový démon naslouchá na vyznačeném síťovém portu a jakmile je akceptováno příchozí spojení, spravuje samotné síťové spojení, aby poskytovalo svoje služby.

řízený démonem inetd

Server *inetd* je síťový démon, který se specializuje na příchozí síťová spojení. Má konfigurační soubor, kde se říká, který program má být spuštěn po navázání určitého spojení. Dokáže obsluhovat libovolné porty protokolů TCP nebo UDP. Porty jsou popsány v dalším textu, ke kterému se brzy dostaneme.

Musíme nakonfigurovat dva důležité soubory. Jedná se o `/etc/services` (přiřazuje názvy číslům portů) a `/etc/inetd.conf`, který je konfiguračním souborem síťového démona *inetd*.

`/etc/services`

Tento soubor je jednoduchou databází, která přiřazuje „lidské“ názvy strojovým portům služeb. Jeho formát je jednoduchý. Soubor je textový, přičemž každý řádek reprezentuje údaj databáze. Každý údaj se skládá ze tří polí, oddělených libovolným množstvím prázdného místa (tabulátory nebo mezery). Pole jsou:

```
jméno port/protokol     přezdívky # komentář
```

jméno	Jednoslovný název, který reprezentuje popisovanou službu.
port/protokol	Toto pole je rozděleno na dvě části
port	Číslo, které určuje číslo portu, na němž bude daná služba k dispozici. Většina běžných služeb má přiřazena standardní čísla. Jsou popsána v RFC-1340.
protokol	Toto pole je možné nastavit na `tcp` nebo `udp`. Je důležité vědět, že údaje `18/tcp` a `18/udp` znamenají něco úplně jiného a neexistuje žádný technický důvod, proč by stejná služba měla existovat na obou. Většinou převládá zdravý rozum a údaj pro oba se objeví pouze v případě, že je určitá služba také pro oba k dispozici.
přezdívky	Další názvy, které je možné použít při odkazu na údaje této služby.

Jakýkoliv text, který se na řádku objeví za znakem #, je ignorován a posuzován jako komentář.

Příklad souboru /etc/services

Všechny současné distribuce Linuxu obsahují kvalitní soubor `/etc/services`. Jen pro případ, že byste vytvářeli stroj úplně od základů, sem umisťuji kopii souboru `/etc/services`, který je v distribuci Debian (*www.debian.org*):

```
# /etc/services:
# $Id: Net-HOWTO.sgml,v 1.1 2000/07/19 16:04:26 gferg Exp $
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1340, ``Assigned Numbers'' (July 1992). Not all ports
# are included, only the more common ones.
tcpmux              1/tcp                       # TCP port service multiplexer
echo                7/tcp
echo                7/udp
discard             9/tcp     sink null
discard             9/udp     sink null
systat              11/tcp    users
daytime             13/tcp
daytime             13/udp
netstat             15/tcp
qotd                17/tcp    quote
msp                 18/tcp                      # message send protocol
msp                 18/udp                      # message send protocol
chargen             19/tcp    ttytst source
chargen             19/udp    ttytst source
ftp-data            20/tcp
ssh                 22/tcp                      # SSH Remote Login Protocol
ssh                 22/udp                      # SSH Remote Login Protocol
telnet              23/tcp
# 24 - private
```

```
smtp                25/tcp     mail
# 26 - unassigned
time                37/tcp     timserver
time                37/udp     timserver
rlp                 39/udp     resource          # resource location
nameserver          42/tcp     name              # IEN 116
whois               43/tcp     nicname
re-mail-ck          50/tcp                       # Remote Mail Checking Protocol
re-mail-ck          50/udp                       # Remote Mail Checking Protocol
domain              53/tcp     nameserver        # name-domain server
domain              53/udp     nameserver
mtp                 57/tcp                       # deprecated
bootps              67/tcp                       # BOOTP server
bootps              67/udp
bootpc              68/tcp                       # BOOTP client
bootpc              68/udp
tftp                69/udp
gopher              70/tcp                       # Internet Gopher
gopher              70/udp
rje                 77/tcp     netrjs
finger              79/tcp
www                 80/tcp     http              # WorldWideWeb HTTP
www                 80/udp                       # HyperText Transfer Protocol
link                87/tcp     ttylink
kerberos            88/tcp     kerberos5 krb5    # Kerberos v5
kerberos            88/udp     kerberos5 krb5    # Kerberos v5
supdup              95/tcp
# 100 - reserved
hostnames           101/tcp    hostname          # usually from sri-nic
iso-tsap            102/tcp    tsap              # part of ISODE.
csnet-ns            105/tcp    cso-ns            # also used by CSO name server
csnet-ns            105/udp    cso-ns
rtelnet             107/tcp                      # Remote Telnet
rtelnet             107/udp
pop-2               109/tcp    postoffice        # POP version 2
pop-2               109/udp
pop-3               110/tcp                      # POP version 3
pop-3               110/udp
sunrpc              111/tcp    portmapper        # RPC 4.0 portmapper TCP
sunrpc              111/udp    portmapper        # RPC 4.0 portmapper UDP
auth                113/tcp    authentication tap ident
sftp                115/tcp
uucp-path           117/tcp
nntp                119/tcp    readnews untp     # USENET News Transfer Protocol
ntp                 123/tcp
ntp                 123/udp                      # Network Time Protocol
netbios-ns          137/tcp                      # NETBIOS Name Service
netbios-ns          137/udp
netbios-dgm         138/tcp                      # NETBIOS Datagram Service
netbios-dgm         138/udp
netbios-ssn         139/tcp                      # NETBIOS session service
netbios-ssn         139/udp
imap2               143/tcp                      # Interim Mail Access Proto v2
```

```
imap2
143/udp
snmp                161/udp                     # Simple Net Mgmt Proto
snmp-trap           162/udp    snmptrap         # Traps for SNMP
cmip-man            163/tcp                     # ISO mgmt over IP (CMOT)
cmip-agent          164/tcp
cmip-agent          164/udp
xdmcp               177/tcp                     # X Display Mgr. Control Proto
xdmcp               177/udp
nextstep            178/tcp    NeXTStep NextStep # NeXTStep window
nextstep            178/udp    NeXTStep NextStep # server
bgp                 179/tcp                     # Border Gateway Proto.
bgp                 179/udp
prospero            191/tcp                     # Cliff Neuman's Prospero
prospero            191/udp
irc                 194/tcp                     # Internet Relay Chat
irc                 194/udp
smux                199/tcp                     # SNMP Unix Multiplexer
smux                199/udp
at-rtmp             201/tcp                     # AppleTalk routing
at-rtmp             201/udp
at-nbp              202/tcp                     # AppleTalk name binding
at-nbp              202/udp
at-echo             204/tcp                     # AppleTalk echo
at-echo             204/udp
at-zis              206/tcp                     # AppleTalk zone information
at-zis              206/udp
z3950               210/tcp    wais             # NISO Z39.50 database
z3950               210/udp    wais
ipx                 213/tcp                     # IPX
ipx                 213/udp
imap3               220/tcp                     # Interactive Mail Access
imap3               220/udp                     # Protocol v3
ulistserv           372/tcp                     # UNIX Listserv
ulistserv           372/udp
#
# UNIX specific services
#
exec                512/tcp
biff                512/udp    comsat
login               513/tcp
who                 513/udp    whod
shell               514/tcp    cmd              # no passwords used
syslog              514/udp
printer             515/tcp    spooler          # line printer spooler
talk                517/udp
ntalk               518/udp
route               520/udp    router routed    # RIP
timed               525/udp    timeserver
tempo               526/tcp    newdate
courier             530/tcp    rpc
conference          531/tcp    chat
```

```
netnews             532/tcp    readnews
netwall             533/udp                     # -for emergency broadcasts
uucp                540/tcp    uucpd            # uucp daemon
remotefs            556/tcp    rfs_server rfs   # Brunhoff remote filesystem
klogin              543/tcp                     # Kerberized `rlogin' (v5)
kshell              544/tcp    krcmd            # Kerberized `rsh' (v5)
kerberos-adm        749/tcp                     # Kerberos `kadmin' (v5)
#
webster             765/tcp                     # Network dictionary
webster             765/udp
#
# From ``Assigned Numbers'':
#
#> The Registered Ports are not controlled by the IANA and on most systems
#> can be used by ordinary user processes or programs executed by ordinary
#> users.
#
#> Ports are used in the TCP [45,106] to name the ends of logical
#> connections which carry long term conversations. For the purpose of
#> providing services to unknown callers, a service contact port is
#> defined. This list specifies the port used by the server process as its
#> contact port. While the IANA can not control uses of these ports it
#> does register or list uses of these ports as a convenience to the
#> community.
#
ingreslock          1524/tcp
ingreslock          1524/udp
prospero-np         1525/tcp                    # Prospero non-privileged
prospero-np         1525/udp
rfe                 5002/tcp                    # Radio Free Ethernet
rfe                 5002/udp                    # Actually uses UDP only
bbs                 7000/tcp                    # BBS service
#
#
# Kerberos (Project Athena/MIT) services
# Note that these are for Kerberos v4 and are unofficial. Sites running
# v4 should uncomment these and comment out the v5 entries above.
#
kerberos4           750/udp    kdc              # Kerberos (server) udp
kerberos4           750/tcp    kdc              # Kerberos (server) tcp
kerberos_master     751/udp                     # Kerberos authentication
kerberos_master     751/tcp                     # Kerberos authentication
passwd_server       752/udp                     # Kerberos passwd server
krb_prop            754/tcp                     # Kerberos slave propagation
krbupdate           760/tcp    kreg             # Kerberos registration
kpasswd             761/tcp    kpwd             # Kerberos „passwd“
kpop                1109/tcp
# Pop with Kerberos
knetd               2053/tcp                    # Kerberos de-multiplexor
zephyr-srv          2102/udp                    # Zephyr server
zephyr-clt          2103/udp                    # Zephyr serv-hm connection
zephyr-hm           2104/udp                    # Zephyr hostmanager
```

```
eklogin             2105/tcp                    # Kerberos encrypted rlogin
#
# Unofficial but necessary (for NetBSD) services
#
supfilesrv          871/tcp                     # SUP server
supfiledbg          1127/tcp                    # SUP debugging
#
# Datagram Delivery Protocol services
#
rtmp                1/ddp                       # Routing Table Maintenance Proto-
col
nbp                 2/ddp                       # Name Binding Protocol
echo                4/ddp                       # AppleTalk Echo Protocol
zip                 6/ddp                       # Zone Information Protocol
#
# Debian GNU/Linux services
rmtcfg              1236/tcp                    # Gracilis Packeten remote config
                                                # server
xtel                1313/tcp                    # french minitel
cfinger             2003/tcp                    # GNU Finger
postgres            4321/tcp                    # POSTGRES
mandelspawn         9359/udp   mandelbrot       # network mandelbrot
# Local services
```

Tento soubor se postupně rozrůstá tak, jak se objevují nové služby. Pokud byste měli pocit, že soubor ve vašem systému je neúplný, zkopírujte si jej z nějaké nové distribuce.

/etc/inetd.conf

Jedná se o konfigurační soubor pro serverového démona *inetd*. Jeho funkcí je sdělit *inetd* co dělat, když obdrží žádost o spojení na určitou službu. U každé služby, pro kterou chcete přijmout spojení, musíte démonu *inetd* sdělit, jak a který konkrétní démon síťového serveru spustit.

Jeho formát je také velice jednoduchý. Jedná se o textový soubor, v němž každý řádek popisuje jednu službu, kterou chcete poskytovat. Jakýkoliv text, který se na řádku objeví za znakem „#“, je ignorován a posuzován jako komentář. Každý řádek obsahuje sedm polí oddělených libovolným množstvím prázdného místa (tabulátory nebo mezery). Obecný formát vypadá takto:

```
služba  typ_socketu  protokol  volby  uživatel  cesta_k_serveru  argumenty_serveru
```

služba — Název služby odpovídající souboru `/etc/services`.

typ_socketu — Toto pole popisuje typ socketu. Možné hodnoty jsou *stream*, *dgram*, *raw*, *rdm* nebo *seqpacket*, což je sice trochu více technické, ale platí zde pravidlo, že skoro všechny služby založené na protokolu *tcp* používají *stream* a skoro všechny služby na protokolu *udp* používají *dgram*. Pouze velmi speciální typy serverových démonů využívají některé ze zbývajících hodnot.

protokol — Protokol platný pro tento údaj. Měl by odpovídat příslušnému údaji v souboru `/etc/services` a bývá *tcp* nebo *udp*. Servery Sun RPC budou využívat *rpc/tcp* nebo *rpc/udp*.

volby — Toto pole má jen dvě možná nastavení. Zde *inetd* zjistí, jestli program síťového serveru po spuštění uvolní socket a jestli má tedy *inetd* při následující žádosti o spojení spustit další instanci serveru, nebo jestli má *inetd*

čekat a předpokládat, že již běžící serverový démon se o žádost o nové připojení postará. Opět je zde chápání trochu zjednodušené, ale pravidlem je, že všechny servery *tcp* by měly mít nastaveno *nowait* a většina serverů *udp* by měla mít *wait*. Jsou zde však časté výjimky, takže pokud si nejste jisti, raději se řiďte podle příkladu.

uživatel — Toto pole popisuje, který uživatelský účet z `/etc/passwd` bude pro spuštěného síťového démona nastaven jako vlastník. Nastavení je užitečné zejména z hlediska bezpečnosti. Uživatele můžete nastavit jako `nobody`, takže pokud se naruší bezpečnost serveru, možné ztráty budou minimální. Většinou je zde ale nastaven `root`, protože mnohé servery ke své správné funkci vyžadují práva superuživatele.

cesta_k_serveru — Cesta k serverovému programu, který danou službu obsluhuje.

argumenty_serveru — Tohle pole tvoří zbytek řádku a je nepovinné. Zde se umisťují všechny parametry příkazového řádku, které chcete předat serverovému démonu při jeho spuštění.

Příklad souboru /etc/inetd.conf

Stejně jako u souboru `/etc/services` všechny současné distribuce Linuxu nabízejí správný soubor `/etc/inetd.conf`. Pro doplnění sem umisťuji kopii souboru `/etc/inetd.conf`, který je v distribuci Debian (*http://www.debian.org*).

```
# /etc/inetd.conf: see inetd(8) for further informations.
#
# Internet server configuration database
#
#
# Modified for Debian by Peter Tobias <tobias@et-inf.fho-emden.de>
#
# <service_name> <sock_type> <proto> <flags> <user> <server_path>
#   <args>
#
# Internal services
#
#echo    stream  tcp     nowait  root    internal
#echo    dgram   udp     wait    root    internal
discard stream  tcp     nowait  root    internal
discard dgram   udp     wait    root    internal
daytime stream  tcp     nowait  root    internal
daytime dgram   udp     wait    root    internal
#chargenstream  tcp     nowait  root    internal
#chargendgram   udp     wait    root    internal
time    stream  tcp     nowait  root    internal
time    dgram   udp     wait    root    internal
#
# These are standard services.
#
telnet  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.telnetd
ftp     stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.ftpd
#fsp    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.fspd
#
```

```
# Shell, login, exec and talk are BSD protocols.
#
shell   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd
login   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind
#exec   stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rexecd
talk    dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.talkd
ntalk   dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.ntalkd
#
# Mail, news and uucp services.
#
smtp    stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.smtpd
#nntp   stream  tcp     nowait  news    /usr/sbin/tcpd  /usr/sbin/in.nntpd
#uucp   stream  tcp     nowait  uucp    /usr/sbin/tcpd  /usr/lib/uucp/uucico
#comsat dgram   udp     wait    root    /usr/sbin/tcpd  /usr/sbin/in.comsat
#
#       Pop     et      al
#
#pop-2  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop2d
#pop-3  stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.pop3d
#
# `cfinger' is for the GNU finger server available for Debian.
# (NOTE: The current implementation of the `finger' daemon allows it
# to be run as `root'.)
#
#cfingerstream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.cfingerd
#finger stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.fingerd
#netstatstream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/netstat
#systat stream  tcp     nowait  nobody  /usr/sbin/tcpd  /bin/ps -auwwx
#
# Tftp service is provided primarily for booting. Most sites
# run this only on machines acting as „boot servers.“
#
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd
#tftp   dgram   udp     wait    nobody  /usr/sbin/tcpd  /usr/sbin/in.tftpd     /boot
#bootps dgram   udp     wait    root    /usr/sbin/bootpd bootpd -i -t 120
#
# Kerberos authenticated services (these probably need to be
# corrected)
#
#klogin stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind -k
#eklogin stream tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rlogind
#-k -x
#kshell stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/in.rshd -k
#
# Services run ONLY on the Kerberos server (these probably need to be
# corrected)
#
#krbupdate      stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/registerd
#kpasswd        stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/kpasswdd
#
# RPC based services
#
#mountd/1       dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.mountd
#rstatd/1-3     dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rstatd
```

```
#rusersd/2-3    dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rusersd
#walld/1dgram   rpc/udp wait    root    /usr/sbin/tcpd  /usr/sbin/rpc.rwalld
#
# End of inetd.conf.
ident  stream tcp nowait nobody /usr/sbin/identd identd -i
```

Další síťové konfigurační soubory

V Linuxu existuje ještě řada dalších síťových konfiguračních souborů, které by vás mohly zajímat. Nebudete je možná nikdy upravovat, ale stojí za to si je popsat, abyste věděli, co obsahují a k čemu slouží.

/etc/protocols

Tento soubor je databází, která mapuje identifikační čísla protokolů na názvy protokolů. Využívají jej programátoři, aby mohli ve svých programech specifikovat protokoly prostřednictvím názvů, a také některé programy (například *tcpdump*), aby na výstupu zobrazily názvy místo čísel. Obecná syntaxe je:

```
jméno   číslo   přezdívky
```

Soubor `/etc/protocols` v distribuci Debian vypadá takto:

```
# /etc/protocols:
# $Id: Net-HOWTO.sgml,v 1.1 2000/07/19 16:04:26 gferg Exp $
#
# Internet (IP) protocols
#
# from: @(#)protocols 5.1 (Berkeley) 4/17/89
#
# Updated for NetBSD based on RFC 1340, Assigned Numbers (July 1992).
ip              0       IP              # internet protocol, pseudo protocol num-
ber
icmp            1       ICMP            # internet control message protocol
igmp            2       IGMP            # Internet Group Management
ggp             3       GGP             # gateway-gateway protocol
ipencap                 4       IP-ENCAP        # IP encapsulated in IP (officia-
lly ``IP'')
st              5       ST              # ST datagram mode
tcp             6       TCP             # transmission control protocol
egp             8       EGP             # exterior gateway protocol
pup             12      PUP             # PARC universal packet protocol
udp             17      UDP             # user datagram protocol
hmp             20      HMP             # host monitoring protocol
xns-idp                 22      XNS-IDP                 # Xerox NS IDP
rdp             27      RDP             # „reliable datagram“ protocol
iso-tp4                 29      ISO-TP4                 # ISO Transport Protocol
class 4
xtp             36      XTP             # Xpress Tranfer Protocol
ddp             37      DDP             # Datagram Delivery Protocol
idpr-cmtp       39      IDPR-CMTP       # IDPR Control Message Transport
rspf            73      RSPF            # Radio Shortest Path First.
vmtp            81      VMTP            # Versatile Message Transport
ospf            89      OSPFIGP                 # Open Shortest Path First IGP
ipip            94      IPIP            # Yet Another IP encapsulation
encap           98      ENCAP           # Yet Another IP encapsulation
```

`/etc/networks`

Tento soubor má podobnou funkci, jako má `/etc/hosts`. Poskytuje jednoduchou databázi síťových názvů a síťových adres. Jeho formát se liší v tom, že na řádku jsou jen dvě pole a ty vypadají následovně:

```
jméno_sítě      adresa_sítě
```

Příklad může vypadat takto:

```
loopnet 127.0.0.0
localnet 192.168.0.0
amprnet 44.0.0.0
```

Když používáte příkazy jako je *route*, a místem určení je nějaká síť o níž existuje v `/etc/networks` záznam, příkaz *route* nebude vypisovat číselnou adresu sítě, ale její název.

Zabezpečení sítě a kontrola přístupu

Dovolte mi začít tuto část upozorněním, že zabezpečení počítače a sítě proti nežádoucím útokům je složité umění. Na tohle téma se necítím příliš povolaným odborníkem. I když následující mechanismy jsou jistě užitečné, pokud vám na bezpečnosti skutečně záleží, doporučujeme vám se rozhodně ohlédnout i po dalších informacích. Na dané téma je možné z Internetu získat mnoho dobrých informací, například v dokumentu Security-HOWTO.

Důležité pravidlo zní: *Nespouštějte servery, které nehodláte používat.* Mnoho distribucí má nakonfigurovány různé služby, které se automaticky spustí. Alespoň minimální úroveň zabezpečení zajistíte, když si projdete soubor `/etc/inetd.conf` a zakomentujete služby (umístěním znaku # na začátek řádku), které nehodláte používat. Dobrými kandidáty jsou služby jako `shell`, `login`, `exec`, `uucp`, `ftp` a informační služby jako `finger`, `netstat` a `systat`.

Existují různé druhy zabezpečovacích mechanismů a mechanismů pro řízení přístupu, popíšeme si však jenom ty základní.

`/etc/ftpusers`

Tento soubor je jednoduchým mechanismem, který umožňuje vybraným uživatelům zakázat přístup na váš stroj pomocí FTP. Soubor je čten ftp démonem *ftpd* při obdržení přicházejícího ftp spojení. Seznam je jednoduchým seznamem těch uživatelů, kterým není povolen přístup. Může vypadat asi takto:

```
# /etc/ftpusers - uživatelé, kterým není povoleno ftp-spojení
root
uucp
bin
mail
```

`/etc/securetty`

Soubor umožňující nastavení zařízení `tty`, na která se může přihlásit `root`. Tento soubor je čten programem *login* (obvykle */bin/login*). Obsahuje seznam názvů zařízení `tty`, na která se `root` může přihlásit, na všechna ostatní se přihlásit nemůže:

```
# /etc/securetty - zařízení tty, na která se může přihlásit root
tty1
tty2
tty3
tty4
```

Mechanismus řízení přístupu programem tcpd

Program *tcpd*, který můžete vidět v souboru `/etc/inetd.conf`, poskytuje logovací služby a služby řízení přístupu sloužící k ochraně služeb, pro něž je nastaven.

Po vyvolání programem *inetd* čte dva soubory obsahující přístupová pravidla, a umožňuje nebo zakazuje přístup na server, který chrání.

Prohledává soubory s pravidly, dokud nenalezne první shodu. Jestliže není nalezena žádná shoda, usoudí, že je možné povolit přístup každému. Soubory, které postupně prochází, jsou `/etc/hosts.allow` a `/etc/hosts.deny`. Oba dva ještě popíši. Kompletní popis naleznete na příslušných manuálových stránkách (dobrým začátkem je `host_access`).

/etc/hosts.allow

Konfigurační soubor programu */usr/sbin/tcpd*. Soubor `hosts.allow` obsahuje pravidla popisující, kteří hostitelé mohou přistupovat ke službám na vašem stroji.

Formát souboru je jednoduchý:

```
# /etc/hosts.allow
#
# <seznam služeb>: <seznam hostitelů> [: příkaz]
```

seznam služeb Čárkami oddělený seznam názvů serverů, na které se tohle pravidlo vztahuje. Příklady názvů serverů jsou: *ftpd*, *telnetd* nebo *fingerd*.

seznam hostitelů Čárkami oddělený seznam názvů hostitelů. Můžete zde také použít IP adresy. Názvy hostitelů nebo adresy můžete určit také pomocí zástupných znaků, aby odpovídaly celým skupinám. Příklady jsou: *gw.vk2ktj.ampr.org*, odpovídající jednomu hostiteli; *uts.edu.au*, odpovídající jakémukoliv názvu, končícímu tímto řetězcem; *44.*, odpovídající jakékoliv IP adrese začínající na toto číslo. Existuje pár speciálních slov, která zjednodušují konfiguraci – některá z nich jsou: *ALL* (odpovídá všemu), *LOCAL* (odpovídá všemu, co neobsahuje tečku, což znamená, že je ve stejné doméně jako váš stroj) a *PARANOID* (odpovídá všem hostitelům, jejichž název neodpovídá jejich adrese). Ještě je užitečné *EXCEPT*, umožňující nastavení seznamu s výjimkami. Později uvedeme příklad.

příkaz Operační parametr. Tento parametr určuje plnou cestu k příkazu spouštěnému vždy, když je pravidlo splněno. Může například spouštět příkaz, který se pokusí identifikovat, kdo je přihlášen na připojujícím se hostiteli, nebo generovat zprávu nebo jiné upozornění systémovému správci, že se někdo snaží připojit. Je možné využít množství rozšíření, nejběžnějšími příklady jsou: `%h` se expanduje na název připojovaného hostitele nebo adresu, jestliže nemá název, `%d` na název démona, který je volán.

Příklad:

```
# /etc/hosts.allow
#
# Poštu povolíme všem.
in.smtpd: ALL
# Telnet a ftp povolíme pouze počítačům z naší domény nebo počítači doma.
telnetd, ftpd: LOCAL, myhost.athome.org.au
# Službu finger povolíme všem, ale poznamenáme si je.
fingerd: ALL: (finger @%h | mail -s „finger from %h“ root)
```

/etc/hosts.deny

Konfigurační soubor programu */usr/sbin/tcpd*. Soubor `hosts.deny` obsahuje pravidla udávající, kteří hostitelé nesmí přistupovat ke službám na vašem stroji.

Jednoduchá ukázka vypadá následovně:

```
# /etc/hosts.deny
#
# Zakážeme všechny služby těm, kteří nemají v pořádku jméno hostitele
ALL: PARANOID
#
# Zakážeme vše
ALL: ALL
```

Údaj `PARANOID` je zde nadbytečný, protože následující záznam stejně vše zablokuje. Obě uvedená nastavení však mohou podle potřeby představovat rozumné výchozí nastavení každého počítače.

Nejbezpečnější konfigurací je mít v souboru `/etc/hosts.deny` implicitní nastavení `ALL: ALL` a následně v souboru `/etc/hosts.allow` vybraným hostitelům povolit vybrané služby.

/etc/hosts.equiv

Soubor se využívá k umožnění přístupu na váš stroj bez hesla pro některé hostitele nebo uživatele. To je užitečné v bezpečném prostředí, kde ovládáte všechny stroje, jinak je to velmi riskantní. Váš stroj je zabezpečen tak, jako nejméně zabezpečený hostitel, kterému důvěřujete. Aby se zabezpečení zvýšilo, nepoužívejte tento mechanismus a zařiďte, aby vaši uživatelé nevyužívali soubor `.rhosts`.

Řádně nakonfigurujte ftp démona

Množství hostitelů využívá anonymní server, umožňující vkládání a stahování souborů bez specifické uživatelské identifikace. Jestliže se rozhodnete tuto možnost nastavit, musíte pro anonymní přístup řádně nakonfigurovat ftp démona. Většina manuálových stránek pro `ftpd` popisuje, jak se s tím vypořádat. Vždy se řiďte doporučenými instrukcemi. Důležitým tipem je nepoužívat kopii vašeho souboru `/etc/passwd` v adresáři `/etc` anonymního účtu. Odstiňte všechny podrobnosti o účtech, kromě těch nejnutnějších, jinak budete zranitelní metodami rozkódování hesla hrubou silou.

Firewally

Skvělé bezpečnostní opatření je vůbec nedovolit datagramům, aby se k vašemu počítači dostaly. Podrobně je tato technika popsána v dokumentu Firewall-HOWTO a v některých dalších částech tohoto dokumentu.

Další návrhy

Zde je pár dalších návrhů, které můžete vzít v úvahu.

sendmail

Navzdory své popularitě se démon *sendmail* objevuje se železnou pravidelností v hlášeních o nedostatcích v zabezpečení. Je to na vás, ale já bych jej nepoužíval.

NFS a další služby Sun RPC

Pozor na ně. Tyto služby umožňují všechny možné druhy průniků. Ke službám jako je NFS se obtížně hledá náhrada, ale pokud je použijete, pečlivě kontrolujte, komu umožňujete právo připojení.

Informace vztahující se k Ethernetu

V této kapitole uvádíme údaje vztahující se k Ethernetu a ke konfiguraci ethernetových karet.

Podporované ethernetové karty

3Com

- 3Com 3c501 – (pozor na „kompatibilní“ plagiáty) (ovladač 3c501)
- 3Com 3c503 (ovladač 3c503), 3c505 (ovladač 3c505), 3c507 (ovladač 3c507), 3c509/3c509B (ISA) /
- 3c579 (EISA)
- 3Com Etherlink III Vortex Ethercards (3c590, 3c592, 3c595, 3c597) (PCI), 3Com Etherlink XL
- Boomerang (3c900, 3c905) (PCI) a Cyclone (3c905B, 3c980) Ethercards (ovladač 3c59x) a 3Com
- Fast EtherLink Ethercard (3c515) (ISA) (ovladač 3c515)
- 3Com 3ccfe575 Cyclone Cardbus (ovladač 3c59x)
- 3Com 3c575 série Cardbus (ovladač 3c59x) (všechno PCMCIA?)

AMD, ATT, Allied Telesis, Ansel, Apricot

- AMD LANCE (79C960) / PCnet-ISA/PCI (AT1500, HP J2405A, NE1500/NE2100)
- ATT GIS WaveLAN
- Allied Telesis AT1700
- Allied Telesis LA100PCI-T
- Allied Telesyn AT2400T/BT (moduly „ne*“)
- Ansel Communications AC3200 (EISA)
- Apricot Xen-II / 82596

Cabletron, Cogent, Crystal Lan

- Cabletron E21xx
- Cogent EM110
- Crystal Lan CS8920, Cs8900

Danpex, DEC, Digi, DLink

- Danpex EN-9400
- DEC DE425 (EISA) / DE434/DE435 (PCI) / DE450/DE500 (ovladač DE4x5)
- DEC DE450/DE500-XA (dc21x4x) (ovladač Tulip)
- DEC DEPCA a EtherWORKS
- DEC EtherWORKS 3 (DE203, DE204, DE205)
- DECchip DC21x4x „Tulip“
- DEC QSilver's (ovladač Tulip)
- Digi International RightSwitch
- DLink DE-220P, DE-528CT, DE-530+, DFE-500TX, DFE-530TX

Fujitsu, HP, ICL, Intel

- Fujitsu FMV-181/182/183/184
- HP PCLAN (27245 a série 27xxx)
- HP PCLAN PLUS (27247B a 27252A)
- HP 10/100VG PCLAN (J2577, J2573, 27248B, J2585) (ISA/EISA/PCI)
- ICL EtherTeam 16i / 32 (EISA)
- Intel EtherExpress
- Intel EtherExpress Pro

KTI, Macromate, NCR NE2000/1000, Netgear, New Media

- KTI ET16/P-D2, ET16/P-DC ISA (jak „jumperless“ tak s hardwarovou konfigurací)
- Macromate MN-220P (režimy PnP nebo NE2000)
- NCR WaveLAN
- NE2000/NE1000 (pozor na různé klony)
- Netgear FA-310TX (čip Tulip)
- New Media Ethernet

PureData, SEEQ, SMC

- PureData PDUC8028, PDI8023
- SEEQ 8005
- SMC Ultra / EtherEZ (ISA)
- SMC 9000 série
- SMC PCI EtherPower 10/100 (ovladač DEC Tulip)
- SMC EtherPower II (ovladač epic100.c)

Sun Lance, Sun Intel, Schneider, WD, Zenith, IBM, Enyx

- Sun LANCE série (jádro 2.2 a novější)
- Sun Intel série (jádro 2.2 a novější)
- Schneider a Koch G16
- Western Digital WD80x3
- Zenith Z-Note / IBM ThinkPad 300 vestavený adaptér
- Znyx 312 etherarray (ovladač Tulip)

Obecné informace o Ethernetu

Názvy ethernetových zařízení jsou `eth0`, `eth1`, `eth2` a tak dále. První detekované kartě je přiřazen název `eth0` a dalším pak názvy v tom pořadí, v jakém jsou detekovány.

Máte-li jádro sestaveno s podporou vaší ethernetové karty, je již vlastní konfigurace karty jednoduchá.

Typicky použijete následující zápis (většina distribucí to zařídí automaticky, pokud si nainstalujete podporu Ethernetu):

```
root# ifconfig eth0 192.168.0.1 netmask 255.255.255.0 up
root# route add -net 192.168.0.0 netmask 255.255.255.0 eth0
```

Většinu ethernetových ovladačů vytvořil Donald Becker, *becker@CESDIS.gsfc.nasa.gov*.

Dvě a více ethernetových karet v jednom počítači

Pokud používáte modulární ovladač (obvyklé v moderních distribucích)

Modulární ovladač typicky detekuje všechny nainstalované karty.

Informace o detekovaných kartách se nacházejí v souboru `/etc/conf.modules`.

Řekněme, že používáte tři karty NE2000 na adresách 0x300, 0x240 a 0x220. Do souboru `/etc/modules.conf` pak zapíšete následující údaje

```
alias eth0 ne
alias eth1 ne
alias eth2 ne
options ne io=0x220,0x240,0x300
```

Tím říkáte programu *modprobe*, aby vyzkoušel tři NE* karty na uvedených adresách. Definujete tím také, v jakém pořadí je má hledat a jaká zařízení jim mají být přiřazena.

Většina ISA modulů dokáže pracovat s více vstupně-výstupními adresami, například

```
alias eth0 3c501
alias eth1 3c501
options eth0 -o 3c501-0 io=0x280 irq=5
options eth1 -o 3c501-1 io=0x300 irq=7
```

Parametr `-o` umožní každému modulu přiřadit samostatný název. Děláme to proto, protože nelze mít nahráno více kopií stejnojmenného modulu.

Parametr `irq=` slouží k nastavení čísla přerušení, parametr `io=` nastavuje vstupně-výstupní adresu.

Implicitně hledá jádro Linuxu pouze jedno ethernetové zařízení, abyste si vynutili detekci i dalších zařízení, musíte si o to explicitně říct.

Informace o zprovoznění ethernetových karet pod Linuxem obsahuje dokument Ethernet-HOWTO.

Informace o protokolu IP

V této kapitole uvádíme informace vztahující se k protokolu IP.

Parametry na úrovni jádra

Tato část obsahuje informace o nastavení parametrů protokolu IP při startu jádra. Příkladem takových parametrů jsou **ip_forward** nebo **ip_bootp_agent**. Jednotlivé parametry se nastavují uvedením příslušných hodnot do souboru v

```
/proc/sys/net/ipv4
```

Pokud budete například chtít zapnout parametr **ip_forward**, zadáte příkaz

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Seznam obecných parametrů protokolu IP

`ip_forward` Pokud má parametr **ip_forward** hodnotu 0, je předávání IP paketů zakázáno. Jakákoliv jiná hodnota tuto funkci zapíná. Tento parametr se používá ve spolupráci s takovými technikami jako je směrování mezi rozhraními s IP maškarádou.

`ip_default_ttl`	Životnost IP paketu. Implicitní hodnota je 64 milisekund.
`ip_addrmask_agent`	Booleovský příznak udávající, zda se má reagovat na požadavek ICMP ADDRESS MASK. Implicitní hodnota je TRUE pro směrovače a FALSE pro stanice.
`ip_bootp_agent`	Booleovská hodnota. Povoluje příjem paketů se zdrojovou adresou typu *0.a.b.c*, které jsou určeny pro lokální systém, jsou broadcastovány nebo multicastovány. Není-li příznak nastaven, tento typ paketů se v tichosti zahazuje. Implicitní hodnota je FALSE.
`ip_no_pmtu_disc`	Vypíná funkci MTU Path Discovery. Implicitně FALSE.
`ip_fib_model`	0 (implicitní) – standardní model. Všechny trasy jsou ve třídě MAIN. 1 - implicitní trasy jsou ve třídě DEFAULT. Tento režim je velmi pohodlný pro vytváření směrovacích politik u menších poskytovatelů. 2 - model odpovídající RFC1812. Trasy přes rozhraní jsou třídy MAIN. Trasy přes brány jsou třídy DEFAULT.

EQL - distributor zátěže na více linek

EQL zařízení se jmenuje `eql`. Se standardním zdrojovým kódem jádra můžete mít v na jednom počítači pouze jedno EQL zařízení. EQL nabízí mechanismus, jak donutit více spojů bod-bod (jako jsou například linky PPP, SLIP nebo PLIP), aby se chovaly jako jediné zařízení přenášející protokol TCP/IP. Často bývá levnější použít více pomalejších linek, než instalovat jednu rychlou linku.

Volby pro překlad jádra:

```
Network device support  --->
    [*] Network device support
    <*> EQL (serial line load balancing) support
```

Aby celý mechanismus fungoval, musí být EQL podporováno i na druhém konci linek. Kompatibilní funkce nabízejí Linux, Livingstone Portmasters a novější dial-up servery.

Ke konfiguraci EQL potřebujete nástroje dostupné na adrese *ftp://metalab.unc.edu/pub/linux/system/Serial/eql-1.2.tar.gz*.

Konfigurace je poměrně přímočará. Nejprve nastavíte eql rozhraní. Toto rozhraní je stejné jako kterékoliv jiné síťové rozhraní. Příkazem *ifconfig* nastavíte jeho IP adresu a MTU. Takto vypadá příklad:

```
root# ifconfig eql 192.168.10.1 mtu 1006
```

Pak musíte ručně aktivovat všechny používané linky. Může se jednat o jakoukoliv kombinaci linek typu bod-bod. Způsob inicializace linky se liší podle toho, o jaký typ linky jde. Další informace naleznete v příslušné části této knihy.

Nakonec musíte linky přiřadit k zařízení EQL. Tomuto kroku se říká *enslaving* a provede se příkazem *eql_enslave* tak, jak to ukazuje následující příklad:

```
root# eql_enslave eql sl0 28800
root# eql_enslave eql ppp0 14400
```

Parametr *estimated speed*, který příkazu *eql_enslave* zadáváte, přímo nedělá nic. Používá jej ovladač EQL k určení, jakou část přenášených dat má to které zařízení obdržet. Nastavováním této hodnoty můžete doladit provoz na jednotlivých linkách.

Odpojení linky od EQL zařízení se provede příkazem *eql_emancipate* takto:

```
root# eql_emancipate eql sl0
```

Směrování se nastavuje stejně jako na jakékoliv lince bod-bod, trasa se ovšem vede přes zařízení EQL, nikoliv přes fyzické linky. Typicky se použije příkaz

```
root# route addthemselveseql
```

Ovladač EQL vytvořil Simon Janes, *simon@ncm.com*.

IP účtování (pro Linux 2.0)

Funkce IP účtování v jádře Linuxu umožňuje shromažďovat a analyzovat data týkající se síťového provozu. Shromažďují se údaje o počtu přenesených paketů a bajtů od posledního vynulování čítačů. Dále je možno nastavovat pravidla, která slouží k přesné definici typu přenášených dat, která chcete sledovat. Tato funkce byla odstraněna v jádře 2.1.102, protože starý firewall ipfwadm byl nahrazen novým, ipfwchains.

Volby překladu jádra:

```
Networking options  --->
    [*] IP: accounting
```

Po přeložení a instalaci jádra musíte účtování nastavit příkazem *ipfwadm*. Existuje mnoho způsobů, jak účtovací informace shromažďovat. Vybrali jsme několik užitečných příkladů. Další informace naleznete na manuálových stránkách příkazu *ipfwadm*.

Situace: Máte ethernetovou síť připojenou k Internetu přes PPP linku. Na ethernetu je umístěn počítač, který poskytuje celou řadu služeb. Zajímá vás provoz generovaný službami FTP a WWW a rovněž celkový TCP a UDP provoz.

Můžete použít přibližně následující příkazy (zapsané v shell skriptu):

```
#!/bin/sh
#
# Vymazání účtovacích pravidel
ipfwadm -A -f
#
# Nastavení zkratek
localnet=44.136.8.96/29
any=0/0
# Zavedení pravidel pro lokální ethernetový segment
ipfwadm -A in  -a -P tcp -D $localnet ftp-data
ipfwadm -A out -a -P tcp -S $localnet ftp-data
ipfwadm -A in  -a -P tcp -D $localnet www
ipfwadm -A out -a -P tcp -S $localnet www
ipfwadm -A in  -a -P tcp -D $localnet
ipfwadm -A out -a -P tcp -S $localnet
ipfwadm -A in  -a -P udp -D $localnet
ipfwadm -A out -a -P udp -S $localnet
#
# Pravidla implicitní trasy
ipfwadm -A in  -a -P tcp -D $any ftp-data
ipfwadm -A out -a -P tcp -S $any ftp-data
ipfwadm -A in  -a -P tcp -D $any www
ipfwadm -A out -a -P tcp -S $any www
ipfwadm -A in  -a -P tcp -D $any
ipfwadm -A out -a -P tcp -S $any
ipfwadm -A in  -a -P udp -D $any
ipfwadm -A out -a -P udp -S $any
#
```

```
# Výpis pravidel
ipfwadm -A -l -n
#
```

Názvy *ftp-data* a *www* se vztahují k údajům v souboru `/etc/services`. Poslední příkaz vypíše všechna účtovací pravidla a nashromážděné statistiky.

Důležité pro analýzu účtovacích dat je to, že se inkrementují počitadla u *všech vyhovujících pravidel*. Pokud potřebujete získat rozdílové hodnoty, musíte si je dopočítat sami. Pokud budeme například chtít vědět, kolik dat bylo přeneseno jinak než přes FTP a WWW, budeme muset hodnoty získané těmito pravidly odečíst od hodnot získaných pravidlem zahrnujícím veškerý provoz.

```
root# ipfwadm -A -l -n
 IP accounting rules
  pkts bytes dir prot source              destination         ports
     0     0 in  tcp  0.0.0.0/0           44.136.8.96/29      * -> 20
     0     0 out tcp  44.136.8.96/29      0.0.0.0/0           20 -> *
    10  1166 in  tcp  0.0.0.0/0           44.136.8.96/29      * -> 80
    10   572 out tcp  44.136.8.96/29      0.0.0.0/0           80 -> *
   252 10943 in  tcp  0.0.0.0/0           44.136.8.96/29      * -> *
   231 18831 out tcp  44.136.8.96/29      0.0.0.0/0            * -> *
     0     0 in  udp  0.0.0.0/0           44.136.8.96/29      * -> *
     0     0 out undp  44.136.8.96/29      0.0.0.0/0           * -> *
     0     0 in  tcp  0.0.0.0/0           0.0.0.0/0           * -> 20
     0     0 out tcp  0.0.0.0/0           0.0.0.0/0           20 -> *
    10  1166 in  tcp  0.0.0.0/0           0.0.0.0/0           * -> 80
    10   572 out tcp  0.0.0.0/0           0.0.0.0/0           80 -> *
   253 10983 in  tcp  0.0.0.0/0           0.0.0.0/0           * -> *
   231 18831 out tcp  0.0.0.0/0           0.0.0.0/0           * -> *
     0     0 in  udp  0.0.0.0/0           0.0.0.0/0           * -> *
     0     0 out udp  0.0.0.0/0           0.0.0.0/0           * -> *
```

IP účtování (pro Linux 2.2)

Nový účtovací kód je přístupný nástrojem „IP Firewall Chains“. Podrobnosti naleznete na domovské stránce tohoto nástroje *(http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html)*. Ke konfiguraci filtrů se namísto příkazu *ipchains* použije příkaz *ipfwadm*.

IP aliasing

V některých situacích je užitečné přiřadit jednomu síťovému rozhraní více IP adres. Například poskytovatelé internetových služeb tuto funkci využívají k „vlastním nastavením“ WWW a FTP přístupu svých klientů. Podrobnější informace naleznete v dokumentu „IP-Alias mini-HOWTO“.

Volby pro překlad jádra

```
Networking options  --->
    ....
    [*] Network aliasing
    ....
    <*> IP: aliasing support
```

Po překladu a instalaci jádra s podporou IP aliasů je samotná konfigurace velmi jednoduchá. Další adresy se přiřazují virtuálním síťovým zařízením, sdruženým s fyzickým síťovým zařízením. Tato zařízení používají jednoduchou konvenci názvů: <zařízení>:<číslo virtuálního zařízení>, například `eth0:0`, `ppp0:10` a podobně. Virtuální zařízení je možné vytvořit až po nastavení příslušného fyzického zařízení.

Předpokládejme například, že máte ethernetovou síť, která současně podporuje dvě IP podsítě. Chcete, aby měl nějaký počítač přístup k oběma podsítím. Můžete použít něco jako

```
root# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0
root# ifconfig eth0:0 192.168.10.1 netmask 255.255.255.0 up
root# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
```

Pokud chcete alias odstranit, přidejte na konec jeho názvu „-“. Například

```
root# ifconfig eth0:0- 0
```

Automaticky dojde i k vymazání všech tras, které přes toto zařízení vedou.

IP Firewall (pro Linux 2.0)

Problematika IP firewallů je podrobněji popsána v dokumentu Firewall-HOWTO. Firewall vám umožňuje zabezpečit počítač před neoprávněným síťovým přístupem tím, že povolíte nebo zakážete přenos IP datagramů z vybraných IP adres. Existují tři třídy pravidel: příchozí filtrování, odchozí filtrování a předávací filtrování. Příchozí pravidla se použijí na všechny datagramy přijaté síťovým rozhraním. Odchozí pravidla se použijí na datagramy odesílané na síťové zařízení. Předávací pravidla se používají pro datagramy přijaté tímto počítačem, které mu však nepatří (tedy na datagramy, které se budou předávat dále).

Volby překladu jádra:

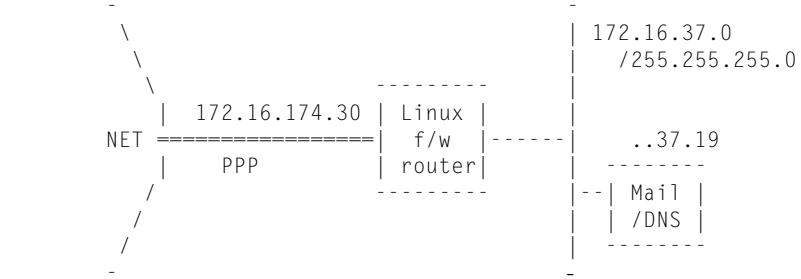
```
Networking options  --->
    [*] Network firewalls
    ....
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: firewalling
    [ ] IP: firewall packet logging
```

Nastavení firewallových pravidel se provádí příkazem *ipfwadm*. Jak už jsem říkal, nejsem specialista na otázky bezpečnosti. Ukážu příklad, který můžete použít. Nicméně byste měli tuto problematiku prostudovat sami a sestavit si vlastní pravidla.

Asi nejběžnější použití IP firewallu spočívá v použití linuxového počítače jako brány a filtru k ochraně vaší lokální sítě před neoprávněným přístupem zvenčí.

Následující konfigurace vychází z příspěvku Arnta Gulbrandsena, *agulbra@troll.no*.

Příklad popisuje konfiguraci firewallových pravidel na linuxové bráně v situaci podle následujícího obrázku



Následující příkazy budou typicky umístěny v `rc` souboru. Tím se zajistí jejich automatické provedení při spuštění systému. Kvůli maximální bezpečnosti by měly být provedeny po nastavení síťových zařízení, ale před jejich aktivací, aby se tak zabránilo napadení sítě v době spouštění firewallu.

```
#!/bin/sh
# Vymazání předávacích pravidel
# Nastavení implicitní politky na „accept“
#
/sbin/ipfwadm -F -f
/sbin/ipfwadm -F -p accept
#
# Stejně pro příchozí pravidla
#
/sbin/ipfwadm -I -f
/sbin/ipfwadm -I -p accept
# Nejprve nastavíme PPP rozhraní
# Raději bych namísto „-a reject -y“ použil „-a deny“, jenže
# pak by nebylo možné na rozhraní vytvářet spojení. Parametr -o
# způsobí zaznamenání odmítnutých datagramů. Tím je možné
# snáze odhalit chyby v konfiguraci.
#
/sbin/ipfwadm -I -a reject -y -o -P tcp -S 0/0 -D 172.16.174.30
# Zahazujeme klasické typy falešných paketů:
# Ignorujeme adresy anycast/multicast/broadcast
#
/sbin/ipfwadm -F -a deny -o -S 224.0/3 -D 172.16.37.0/24
#
# na síti se nesmí objevit nic z loopback rozhraní
#
/sbin/ipfwadm -F -a deny -o -S 127.0/8 -D 172.16.37.0/24
# povolujeme příchozí SMTP a DNS spojení, ovšem jen na
# DNS/SMTP server
#
/sbin/ipfwadm -F -a accept -P tcp -S 0/0 -D 172.16.37.19 25 53
#
# DNS používá TCP i UDP, takže to povolíme také, a» je možné
# se serveru ptát
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 -D 172.16.37.19 53
#
# neodpovídat na nebezpečných portech jako je NFS a
# Larry McVoy’s NFS extension.  Pokud používáte squid, přidejte sem
# jeho port.
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 53 \
        -D 172.16.37.0/24 2049 2050
# odpovědi na ostatní uživatelské porty jsou v pořádku
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 53 \
        -D 172.16.37.0/24 53 1024:65535
# Odmítáme příchozí spojení na identd
# Používáme „reject“ aby připojující se počítač hned věděl, že nemá
# pokračovat, jinak by docházelo ke zpoždění, než nastane timeout
# služby ident.
#
```

```
/sbin/ipfwadm -F -a reject -o -P tcp -S 0/0 -D 172.16.37.0/24 113
# Povolujeme některá běžná spojení ze sítí 192.168.64 a 192.168.65,
# jsou to kamarádi a věříme jim.
#
/sbin/ipfwadm -F -a accept -P tcp -S 192.168.64.0/23 \
        -D 172.16.37.0/24 20:23
# povolujeme cokoliv pocházející zevnitř
#
/sbin/ipfwadm -F -a accept -P tcp -S 172.16.37.0/24 -D 0/0
# zakazujeme většinu jiných příchozích TCP spojení a zaznamenáme je
# (máte-li problémy s FTP, doplňte 1:1023)
#
/sbin/ipfwadm -F -a deny -o -y -P tcp -S 0/0 -D 172.16.37.0/24
# ... totéž pro UDP
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 -D 172.16.37.0/24
```

Dobře nastavit firewall je dost obtížné. Předchozí příklad by měl představovat rozumný počáteční bod. Další podrobnosti o použití programu *ipfwadm* najdete na jeho manuálových stránkách. Pokud chcete nastavit firewall, rozhlédněte se kolem a snažte se nasbírat co nejvíce informací z důvěryhodných zdrojů. Požádejte někoho vně vaší sítě, aby konfiguraci firewallu ověřil.

IP Firewall (pro Linux 2.2)

Nový kód firewallu se obsluhuje nástrojem „IP Firewall Chains“. Další informace naleznete na domovských stránkách tohoto nástroje (*http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html*). Mimo jiné použijete k nastavení namísto příkazu *ipchains* příkaz *ipfwadm*.

Jsme si vědomi, že tento text je poněkud zastaralý, Momentálně pracujeme na novější verzi, která by se měla objevit v průběhu roku.

IPIP Zapouzdření

Proč bychom měli zapozdřovat IP datagramy do IP datagramů? Pokud jste nikdy nepotkali příklad, musí vám to připadat divné. Dvě typická nasazení jsou mobilní IP a IP multicast. Asi nejrozšířenějším (a nejméně známým) použitím je Amateur Radio.

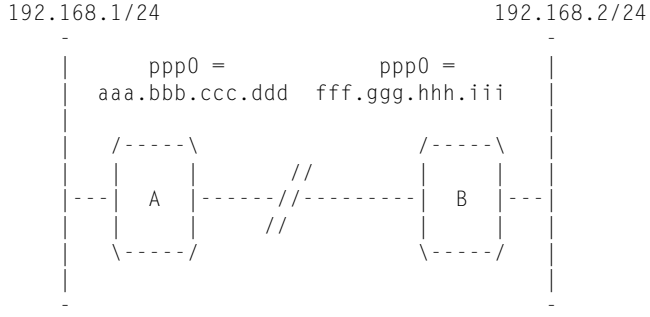
Volby překladu jádra

```
Networking options  --->
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    <*> IP: tunneling
```

Tunelovací zařízení se jmenují `tunl0`, `tunl1` a podobně.

Proč ale? No dobrá. Klasická pravidla směrování v IP sítích stojí na tom, že IP síť je definována síťovou adresou a síťovou maskou. Tím vzniká souvislá sekvence adres, které je možné směrovat jediným záznamem. Je to velice pohodlné. Znamená to, že při připojení k určité síti můžete použít kteroukoliv adresu této sítě. Ve většině případů je to fajn. Pokud jste ale mobilní uživatel, nepřipojujete se trvale z jednoho místa. IPIP zapouzdření (nebo IP tuneling) vám umožňuje obejít toto omezení tím, že se datagramy určené na vaši IP adresu zabalí a pošlou se na jinou IP adresu. Když víte, že budete pracovat na jiné IP síti, můžete si nějaký počítač na vaší „mateřské síti“ nastavit tak, aby přijímal datagramy pro vaši IP adresu. Pak je můžete přesměrovat na tu adresu, kterou dočasně používáte.

Konfigurace tunelované sítě



Obrázek ilustruje další aplikaci IP zapouzdření, totiž virtuální privátní sítě. Příklad předpokládá, že máte dva počítače, každý s jednoduchým vytáčeným připojením k Internetu. Každý počítač má přidělenu jednu IP adresu. Za těmito počítači jsou nějaké privátní lokální sítě. Ty jsou nastaveny tak, že používají privátní IP adresy. Řekněme, že chcete, aby kterýkoliv počítač na síti A mohl komunikovat s kterýmkoliv počítačem na síti B (tak jako kdyby byly plnohodnotně připojeny k Internetu). IP zapouzdření vám to umožní. Poznámka: zapouzdření neřeší problém toho, jak budou počítače na sítích A a B komunikovat s jinými počítači na Internetu. K tomu budete potřebovat nějaký jiný trik, jako je IP maškaráda. Zapouzdření typicky realizují počítače, které slouží zároveň jako brány.

Bránu na síti A je možné nakonfigurovat následujícím skriptem:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=fff.ggg.hhh.iii
#
# Konfigurace ethernetu
ifconfig eth0 192.168.1.1 netmask $mask up
route add -net 192.168.1.0 netmask $mask eth0
#
# konfigurace ppp0 (nastavení ppp linky a výchozí trasy)
pppd
route add default ppp0
#
# Konfigurace tunelovacího zařízení
ifconfig tunl0 192.168.1.1 up
route add -net 192.168.2.0 netmask $mask gw $remotegw tunl0
```

Brána sítě B bude nastavena podobným skriptem:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=aaa.bbb.ccc.ddd
#
# Konfigurace ethernetu
ifconfig eth0 192.168.2.1 netmask $mask up
route add -net 192.168.2.0 netmask $mask eth0
#
# Konfigurace ppp0 (nastavení ppp linky a výchozí trasy)
```

```
pppd
route add default ppp0
#
# Konfigurace tunelovacího zařízení
ifconfig tunl0 192.168.2.1 up
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

Příkaz

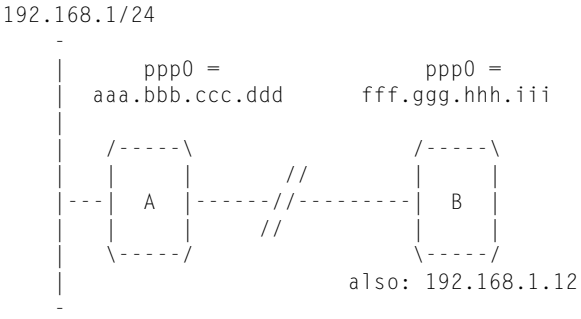
```
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

říká „pošli všechny datagramy určené na 192.168.1.0/24 jako zapouzdřené datagramy na adresu aaa.bbb.ccc.ddd“.

Konfigurace na obou koncích si musejí odpovídat. Tunelovací zařízení používá bránu jako cíl IP datagramů – tedy jako místo, kam bude posílat datagramy přišlé do tunelu. Cílová brána musí umět datagramy „rozbalit“ - jinými slovy, musí být rovněž nastavena jako tunelovací zařízení.

Konfigurace tunelovacího počítače

Tento mechanismus není nutné použít ke směrování celých sítí. Je možné směrovat pouze jedinou IP adresu. V takovém případě se tunelovací zařízení na „vzdáleném“ konci nastaví s vlastní IP adresou. Na „místním“ konci bude nastavena trasa na hostitele (a ARP proxy), namísto síťové trasy přes tunelovací zařízení. Zkusme celou konfiguraci příslušně upravit. Máme zde pouze hostitele B, který se má chovat jako přímo připojený k Internetu a zároveň jako součást vzdálené sítě poskytované hostitelem A:



Konfigurace brány A bude vypadat takto:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=fff.ggg.hhh.iii
#
# Konfigurace ethernetu
ifconfig eth0 192.168.1.1 netmask $mask up
route add -net 192.168.1.0 netmask $mask eth0
#
# Konfigurace ppp0 (nastavení ppp linky a výchozí trasy)
pppd
route add default ppp0
#
# Konfigurace tunelovacího zařízení
```

```
ifconfig tunl0 192.168.1.1 up
route add -host 192.168.1.12 gw $remotegw tunl0
#
# ARP proxy na vzdáleného hostitele
arp -s 192.168.1.12 xx:xx:xx:xx:xx:xx pub
```

Počítač B bude nastaven takto:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=aaa.bbb.ccc.ddd
#
# Konfigurace ppp0 (nastavení ppp linky a výchozí trasy)
pppd
route add default ppp0
#
# Konfigurace tunelovacího zařízení
ifconfig tunl0 192.168.1.12 up
route add -net 192.168.1.0 netmask $mask gw $remotegwtunl0
```

Tento příklad konfigurace je pro mobilní IP aplikace typičtější. Chceme, aby se jeden počítač připojoval z různých míst a zachovával si stejnou IP adresu. Další informace o této problematice naleznete v části Mobilní IP.

IP maškaráda

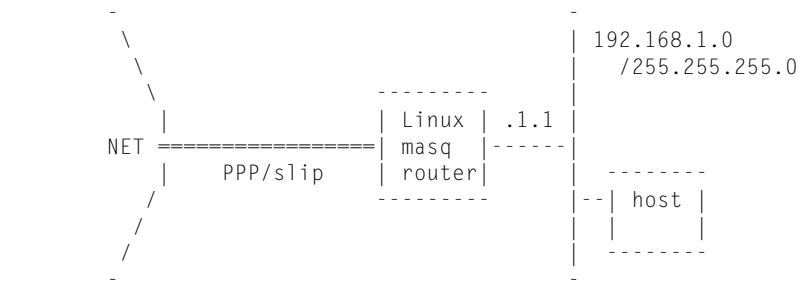
Řada lidí se k Internetu připojuje jednoduchým vytáčeným připojením. Prakticky každý uživatel takového připojení dostává od svého poskytovatele jednu IP adresu. Ta stačí k tomu, aby bylo možné k Internetu připojit jeden počítač. IP maškaráda představuje šikovný trik, jak prostřednictvím této jediné IP adresy připojit k Internetu více počítačů. Způsobí, že ostatní počítače se budou tvářit navenek stejně jako onen počítač s připojením k Internetu. Proto se používá termín maškaráda. Má to jednu nevýhodu: maškaráda typicky funguje pouze jedním směrem. Znamená to, že počítač za maškarádou může vytvářet spojení ven, nemůže však přijímat spojení ze vzdálených počítačů. Některé síťové služby (například *talk*) tak nebudou fungovat, jiné (například *ftp*) je nutné používat v pasivním režimu. Nicméně nejběžnější služby jako *telnet*, *www* a *irc* fungovat budou.

Volby překladu jádra

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
    [*] Network firewalls
    ....
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: masquerading (EXPERIMENTAL)
```

Typicky máte počítač podporující vytáčené připojení přes SLIP nebo PPP. Kromě toho je v něm další síťové zařízení (asi ethernet), které má přidělenu rezervovanou síťovou adresu. Na této síti se pak budou nacházet maškarádované počítače. Všechny tyto počítače budou jako implicitní bránu používat IP adresu ethernetového rozhraní maškarádovacího počítače.

Typická konfigurace bude vypadat takto:



Maškaráda pomocí IPFWADM (jádra 2.0.x)

Důležité příkazy týkající se této konfigurace jsou

```
# Trasa na ethernet
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Implicitní trasa do Internetu
route add default ppp0
#
# Maškaráda pro počítače na síti 192.168.1/24
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
```

Maškaráda pomocí IPCHAINS

Používá se podobně jako IPFWADM, struktura příkazu se ale změnila:

```
# Trasa na ethernet
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Implicitní trasa do Internetu
route add default ppp0
#
# Maškaráda pro počítače na síti 192.168.1/24
ipchains -A forward -s 192.168.1.0/24 -j MASQ
```

Další informace o maškarádě naleznete na adrese *http://ipmasq.cjb.net/*. Velmi podrobný popis maškarády je také uveden v dokumentu IP-Masquerade mini-HOWTO, kde se také dočtete, jak donutit jiné operační systémy ke spolupráci s maškarádovacím serverem.

O použití IP maškarády se dočtete na stránce *http://www.tsmservices.com/masq/*.

Transparentní proxy

Transparentní proxy je funkce, která umožňuje přesměrovat servery nebo služby určené jiným počítačům na stejné služby na místním počítači. Typicky je to užitečné v situaci, kdy používáte počítač s linuxem jako bránu – všechna spojení na určitou službu přesměrujete na lokální proxy server.

Volby překladu jádra

```
Code maturity level options  --->
        [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
```

```
[*] Network firewalls
....
[*] TCP/IP networking
....
[*] IP: firewalling
....
[*] IP: transparent proxy support (EXPERIMENTAL)
```

Konfigurace transparentního proxy se provádí příkazem *ipfwadm*.

Příklad vypadá třeba takto:

```
root# ipfwadm -I -a accept -D 0/0 telnet -r 2323
```

Tento příkaz způsobí, že všechna připojení na port služby *telnet* (23) na libovolný počítač budou přesměrována na port *2323* na lokálním počítači. Pokud na tomto portu budete provozovat příslušnou službu, můžete spojení předat dále, zaznamenat, nebo s nimi provést cokoliv jiného.

Zajímavější příklad je přesměrování veškerého HTTP provozu přes lokální cache server. Protokol používaný proxy serverem se liší od čistého protokolu *http* (kde se klient připojí na *www.server.com:80* a žádá */cesta/stránka*). Při připojování na proxy server se připojuje na *proxy.local.domain:8080* a žádá o *www.server.com/cesta/stránka*.

K filtrování HTTP požadavků přes lokální proxy server potřebujete protokol upravit. K tomu slouží malý server *transproxy*. Můžete jej spustit na portu 8081. Pak použijete příkaz

```
root# ipfwadm -I -a accept -D 0/0 80 -r 8081
```

Od této chvíle bude *transproxy* zachytávat všechna spojení na externí servery, upraví tvar požadavku a předá je na lokální proxy server.

IPv6

Když si konečně začnete myslet, že IP sítím rozumíte, pravidla se změní! IPv6 je zkratka pro šestou verzi protokolu IP. Tento protokol byl vyvinut hlavně k uklidnění internetové komunity. Uživatelé se obávali, že brzy dojdou volné IP adresy. Adresy v protokolu IPv6 jsou dlouhé 16 bajtů (128 bitů). Kromě toho obsahuje protokol IPv6 řadu dalších změn, hlavně různých zjednodušení, která by měla usnadnit správu sítí založených na tomto protokolu.

Linux obsahuje funkční (ne však úplnou) podporu protokolu IPv6 od verze jádra 2.2.*.

Informace o IPv6

http://www.bieringer.de/linux/IPv6/IPv6-HOWTO/IPv6-HOWTO.html
http://www.xelia.ch/Linux/IPng.html
http://v6rpm.jindai.net/
http://www.linuxhq.com/IPv6/linux-ipv6.faq.html

Mobilní IP

Termínem „mobilní IP“ se rozumí schopnost počítače přesouvat své připojení k Internetu z jednoho místa na druhé beze změny IP adresy a beze ztráty konektivity. Obvykle pokud se změní přípojné místo počítače, musí se změnit i jeho IP adresa. Mobilní IP tento problém obchází tím, že mobilnímu počítači přidělí pevnou IP adresu. Dále používá zapouzdření (tuneling) s automatickým směrováním, který zajistí, že všechny datagramy určené pro danou pevnou adresu budou správně doručeny na adresu, kterou mobilní počítač právě používá.

Cílem projektu je doplnit Linux o úplnou podporu mobilních počítačů. Informace o projektu a příslušných nástrojích najdete na adrese *http://anchor.cs.binghamton.edu/~mobileip/*.

Multicast

IP multicast umožňuje, aby více počítačů na různých IP sítích dostávalo společně doručované datagramy. Tento mechanismus je určen k poskytování „veřejně vysílaných" informací, jako jsou například audio a video přenosy.

Volby překladu jádra

```
Networking options  --->
         [*] TCP/IP networking
         ....
         [*] IP: multicasting
```

Jsou nutné drobné zásahy do konfigurace sítě a speciální konfigurační nástroje. O podpoře multicastingu v Linuxu hovoří dokumentu Multicast-HOWTO.

Shaper – omezení provozu

Shaper je ovladač, který vytvoří nová síťová zařízení. Tato zařízení pak mají na základě uživatelských nastavení omezenu šířku přenosového pásma. Samotný přenos se odehrává přes nějaké fyzické síťové zařízení a samotný shaper slouží ke směrování odchozího síťového provozu.

Shaper se poprvé objevil v jádře 2.1.15, posléze byl přenesen na jádro 2.0.36 (objevil se v patchi *2.0.36-pre-patch-2* Alana Coxe, který je autorem shaperu a zároveň udržuje jádro 2.0).

Shaper je možné přeložit pouze jako modul. Konfiguruje se programem *shapecfg*. Příkazy vypadají takto:

```
shapecfg attach shaper0 eth1
shapecfg speed shaper0 64000
```

Shaper dokáže omezit pouze odchozí provoz (protože podle nastaveného směrování tyto pakety přenáší). Při použití „směrování podle zdrojové adresy" jej lze použít i k omezení celkového provozu na konkrétním počítači.

Linux 2.2 tento typ směrování podporuje. Pokud byste jej potřebovali pro jádro 2.0, doporučujeme patch Mika McLagana na adrese *ftp.invlogic.com*. Podrobnosti o shaperu naleznete v *Documentation/networking/shaper.txt*.

Pokud chcete použít shaping i pro příchozí provoz, můžete použít *rshaper-1.01* (nebo novější) na adrese *ftp://ftp.systemy.it/pub/develop*.

DHCP a DHCPD

Zkratka DHCP znamená Dynamic Host Configuration Protocol. Protokol DHCP značně usnadňuje konfiguraci sítě s více hostiteli. Nemusíte totiž konfigurovat každého hostitele samostatně, nastavení všech společných parametrů zajistí na všech hostitelích DHCP server.

Vždy když hostitel nabootuje, vyšle na síť paket určený všem hostitelům v síti. Jedná se o volání požadavek na DHCP server, který se nachází na stejném segmentu a který pošle hostiteli konfigurační informace.

Protokol DHCP umožňuje snadno nastavovat IP adresy, síťové masky a brány.

Nastavení DHCP klienta programem LinuxConf

Jako `root` spusťte program *linuxconf*. Tento program je součástí všech distribucí RedHat a funguje jak v X Windows, tak na konzoli. Obsahují jej také distribuce SuSE a Caldera.

```
Select Networking
----------------->Basic Host Information
----------------->Select Enable
----------------->Set Config Mode DHCP
```

Nastavení DHCP serveru v Linuxu

Pokud na počítači DHCP server nemáte, můžete si jej stáhnout na adrese *ftp://ftp.isc.org/isc/dhcp/*.

Poznámka: V jádru musíte mít povolen multicast.

Pokud není k dispozici binární podoba pro vaši verzi Linuxu, budete si muset démona *dhcpd* přeložit sami.

V souboru `/etc/rc.d/rc.local` doplňte trasu 255.255.255.255.

Výňatek z README dokumentu k DHCPD

Aby mohl *dhcpd* korektně spolupracovat s tupými DHCP klienty (například Windows 95), musí být schopen posílat pakety s cílovou adresou 255.255.255.255. Linux bohužel automaticky převádí adresu 255.255.255.255 na vysílací adresu lokální sítě (například 192.168.1.255). Tím se zmate protokol DHCP a i když většina DHCP klientů si toho nevšimne, některé (například všichni DHCP klienti Microsoftu) to poznají. Takoví klienti pak „nevidí“ zprávy DHCPOFFER serveru.

Jako `root` zadejte následující:

```
route add -host 255.255.255.255 dev eth0
```

Pokud se objeví hlášení

```
255.255.255.255: Unknown host
```

doplňte do souboru `/etc/hosts` následující záznam

```
255.255.255.255 dhcp
```

Pak vyzkoušejte

```
route add -host dhcp dev eth0
```

Nastavení DHCP serveru

Démona *dhcpd* je nutné nakonfigurovat. Musíte vytvořit nebo upravit soubor `/etc/dhcpd.conf`. Program *linuxconf* nabízí grafické rozhraní ke konfiguraci démona *dhcpd*. Tím se konfigurace a správa serveru výrazně usnadňuje.

Pokud chcete nakonfigurovat server ručně, použijte následující postup. Doporučujeme vám nakonfigurovat jej ručně alespoň jednou. Získáte tak přesnější představu o jeho vlastnostech, které vám grafické konfigurační prostředí nemůže poskytnout. Bohužel Microsoft na tento přístup nevěří.

Nejjednodušší řešení je přiřazovat IP adresy náhodně. Následující příklad konfiguračního souboru ukazuje právě toto nastavení:

```
# Sample /etc/dhcpd.conf
# (add your comments here)
```

```
default-lease-time 1200;
max-lease-time 9200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-name „mydomain.org“;
subnet 192.168.1.0 netmask 255.255.255.0 {
range 192.168.1.10 192.168.1.100;
range 192.168.1.150 192.168.1.200;
}
```

Toto nastavení říká serveru, aby klientům přiděloval IP adresy z rozsahu 192.168.1.10 – 192.168.1.100 nebo 192.168.1.150 – 192.168.1.200.

Pokud si klient nevyžádá delší čas, pronajme adresu na 1200 sekund. Nejdelší možný čas pronájmu adresy je 9200 sekund. Server pošle klientovi následující nastavení:

Jako masku sítě použij 255.255.255.0, vysílací adresa 192.168.1.255, výchozí brána je 192.168.1.254 a DNS servery jsou 192.168.1.1 a 192.168.1.2.

Pokud chcete klientům Windows umožnit použití serveru WINS, přidejte do souboru `dhcpd.conf` následující řádek:

```
option netbios-name-servers 192.168.1.1;
```

Kromě toho můžete určitým klientům přidělovat pevné IP adresy v závislosti na jejich MAC adrese:

```
host haagen {
hardware ethernet 08:00:2b:4c:59:23;
fixed-address 192.168.1.222;
}
```

Toto nastavení přidělí klientovi s hardwarovou adresou `08:00:2b:4c:59:23` IP adresu `192.168.1.222`.

Spuštění serveru

Většina DHCP serverů si při instalaci nevytváří soubor `dhcp.leases`. Před spuštěním serveru proto musíte tento soubor vytvořit:

```
touch /var/state/dhcp/dhcp.leases
```

Server spustíte zadáním následujícího příkazu (můžete jej také definovat ve spouštěcích skriptech):

```
/usr/sbin/dhcpd
```

Tímto příkazem spustíte server na zařízení `eth0`. Pokud jej chcete spustit pro jiné zařízení, musíte to uvést na příkazovém řádku, například

```
/usr/sbin/dhcpd eth1
```

Budete-li chtít konfiguraci otestovat, zda v ní nejsou nějaké chyby, můžete démona spustit v ladicím režimu. Po zadání následujícího příkazu uvidíte na konzoli spoustu hlášení říkajících, co přesně server dělá:

```
/usr/sbin/dhcpd -d -f
```

EQL – distribuce zátěže na více linek

Zařízení EQL se jmenuje `eql`. Se standardním zdrojovým kódem jádra můžete mít na jednom počítači pouze jedno EQL zařízení. Toto zařízení představuje prostředek, jak spojit více fyzických dvoubodových linek typu PPP, SLIP nebo PLIP do jednoho logického zařízení, které bude přenášet TCP/IP provoz. Obvykle bývá levnější použít více pomalejších linek, než jednu linku rychlou.

Volby pro překlad jádra

```
Network device support --->
[*] Network device support
<*> EQL (serial line load balancing) support
```

Aby celý mechanismus fungoval, zařízení na druhé straně linky rovněž musí podporovat EQL. Kompatibilní funkci podporují kromě Linuxu i Livingstone Portmasters a novější telefonní servery.

Ke konfiguraci EQL potřebujete balík konfiguračních nástrojů, které můžete získat na adrese *http://www.cwareco.com/eqlplus.html*[1].

Konfigurace je velmi jednoduchá. Nejprve musíte nakonfigurovat `eql` rozhraní. Toto rozhraní se chová jako každé jiné rozhraní v Linuxu, takže mu programem *ifconfig* nastavíte IP adresu a MTU například takto:

```
root# ifconfig eql 192.168.10.1 mtu 1006
```

Pak musíte ručně nastavit každou linku, kterou bude zařízení používat. Jednotlivé linky jsou realizovány dvoubodovými síťovými zařízeními. Inicializace jednotlivých linek záleží na tom, jak jsou linky fyzicky provedeny, takže to musíte provést v závislosti na použitém spojení.

Nakonec přiřadíte linky zařízení EQL (tento proces se označuje jako *enslaving*) pomocí příkazu *eql_enslave* takto:

```
root# eql_enslave eql sl0 28800
root# eql_enslave eql ppp0 14400
```

Odhadovaná rychlost, definovaná jako poslední parametr, nemá přímo vliv na nic. Používá ji ovladač EQL k rozhodování, jaký podíl zátěže má každé zařízení dostávat, takže vyladěním těchto hodnot můžete optimalizovat chování spoje.

K odpojení fyzické linky od zařízení EQL použijete příkaz *eql_emancipate* takto:

```
root# eql_emancipate eql sl0
```

Trasy do směrovací tabulky přidáváte stejně jako pro jakékoliv jiné dvoubodové zařízení s tím rozdílem, že trasy se budou vztahovat k zařízení `eql` a nikoliv k jednotlivým sériovým zařízením. Typicky budete definovat trasu takto:

```
root# route add default eql
```

Ovladač EQL vyvinul Simon Janes, *simon@ncm.com*.

---

[1] Pozn. překladatele: Odkaz uvedený v originálním dokumentu do archivu *metalab.unc.edu* nebyl funkční, zmíněný software se v archivu již nenalézá. Zde uvedený odkaz vede na program *eqlplus*, který je založený na původním *eql* Simona Janese, nicméně je vylepšen o použití techniky IP maškarády a tím pádem nemusí být distribuce zatížení podporována vzdálenou stranou. Další popis se nicméně týká *původního* programu, takže pokud použijete *eqlplus*, nevěřte ničemu, co si dále přečtete.

IP účtování (pro jádro 2.0)

Funkce IP účtování v jádře Linuxu vám umožňuje shromažďovat a analyzovat některá data týkající se využití sítě. Mezi sbíraná data patří počet paketů a počet bajtů nahromaděných od posledního vynulování součtů. Ke kategorizaci součtů lze zadat množství nejrůznějších pravidel tak, aby výsledek vyhovoval vašim potřebám. Od jádra 2.1.102 byla tato funkce odstraněna, protože původní firewall založený na programu *ipfwadm* byl nahrazen novým programem *ipfwchains*.

Volby překladu jádra

```
Networking options  --->
    [*] IP: accounting
```

Po přeložení a instalaci jádra je třeba pomocí příkazu *ipfwadm* nastavit IP účtování. Existuje mnoho různých způsobů zavedení účtování podle různých požadavků. Vybrali jsme jednoduchý příklad zahrnující to podstatné. Další informace najdete v manuálových stránkách příkazu *ipfwadm*.

Scénář: Máte ethernetovou síť, která je připojena k Internetu prostřednictvím spojení PPP. Na Ethernetu máte počítač, který nabízí několik služeb, a zajímá vás, jaký provoz generují služby *ftp* a *www* a jaké jsou celkové objemy TCP a UDP provozu.

Můžete použít následující příkazy:

```
#!/bin/sh
#
# Vyprázdnit účetní pravidla
ipfwadm -A -f
#
# Nastavení aliasů
localnet=44.136.8.96/29
any=0/0
#
# Pravidla pro lokální ethernetový segment
ipfwadm -A in -a -P tcp -D $localnet ftp-data
ipfwadm -A out -a -P tcp -S $localnet ftp-data
ipfwadm -A in -a -P tcp -D $localnet www
ipfwadm -A out -a -P tcp -S $localnet www
ipfwadm -A in -a -P tcp -D $localnet
ipfwadm -A out -a -P tcp -S $localnet
ipfwadm -A in -a -P udp -D $localnet
ipfwadm -A out -a -P udp -S $localnet
ipfwadm -A in -a -P tcp -D 44.136.8.96/29 20
#
# Implicitní pravidla
ipfwadm -A in -a -P tcp -D $any ftp-data
ipfwadm -A out -a -P tcp -S $any ftp-data
ipfwadm -A in -a -P tcp -D $any www
ipfwadm -A out -a -P tcp -S $any www
ipfwadm -A in -a -P tcp -D $any
ipfwadm -A out -a -P tcp -S $any
ipfwadm -A in -a -P udp -D $any
ipfwadm -A out -a -P udp -S $any
#
# Výpis pravidel
ipfwadm -A -l -n
#
```

Názvy jako *ftp-data* a *www* se odkazují na soubor `/etc/services`. Poslední příkaz vypíše všechna pravidla účtování a zobrazí shromážděné souhrny.

Pro analýzu shromážděných dat je důležité vědět, že přenosy odpovídající více pravidlům se započítávají do každého z nich, takže některé výsledky si budete muset dopočítat. Pokud by nás například zajímal objem přenosů jiných než *www* a *ftp*, musíme od celkového objemu přenosů odečíst přenosy těchto dvou služeb.

```
root# ipfwadm -A -l -n
IP accounting rules
```

pkts	bytes	dir	prot	source	destination	ports
0	0	in	tcp	0.0.0.0/0	44.136.8.96/29	* -> 20
0	0	out	tcp	44.136.8.96/29	0.0.0.0/0	20 -> *
10	1166	in	tcp	0.0.0.0/0	44.136.8.96/29	* -> 80
10	572	out	tcp	44.136.8.96/29	0.0.0.0/0	80 -> *
252	10943	in	tcp	0.0.0.0/0	44.136.8.96/29	* -> *
231	18831	out	tcp	44.136.8.96/29	0.0.0.0/0	* -> *
0	0	in	udp	0.0.0.0/0	44.136.8.96/29	* -> *
0	0	out	udp	44.136.8.96/29	0.0.0.0/0	* -> *
0	0	in	tcp	0.0.0.0/0	0.0.0.0/0	* -> 20
0	0	out	tcp	0.0.0.0/0	0.0.0.0/0	20 -> *
10	1166	in	tcp	0.0.0.0/0	0.0.0.0/0	* -> 80
10	572	out	tcp	0.0.0.0/0	0.0.0.0/0	80 -> *
253	10983	in	tcp	0.0.0.0/0	0.0.0.0/0	* -> *

IP účtování (pro jádro 2.2)

Nová obsluha účtování je zajišťována mechanismem „IP Firewall Chains“. Podrobnější informace naleznete na domovské stránce na adrese *http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html*. Kromě jiných změn budete filtry konfigurovat programem *ipchains* a ne programem *ipfwadm*.

Přidělování IP aliasů

U některých aplikací je výhodné přidělit jednomu síťovému zařízení více IP adres. Poskytovatelé internetových služeb často využívají tuto vlastnost k poskytování zákaznických stránek na svém WWW nebo FTP serveru. Různé informace se můžete dozvědět v dokumentu „IP-Alias mini-HOWTO“.

Volby překladu jádra

```
Networking options  --->
    ....
    [*] Network aliasing
    ....
    <*> IP: aliasing support
```

Po přeložení a instalaci jádra s podporou IP aliasů je vlastní konfigurace velice jednoduchá. Aliasy se přiřazují virtuálním síťovým zařízením sdruženým se skutečným síťovým zařízením. Konven-

ce názvů těchto zařízení je velmi jednoduchá: `<fyzické_zařízení>:<číslo_virt_zařízení>`, například `eth0:0`, `ppp0:10` a podobně. Upozorňujeme, že virtuální zařízení typu *xxx:n* lze vytvořit až *poté*, co je aktivní odpovídající reálné zařízení.

Předpokládejme například, že máte ethernetovou síť, která současně podporuje dvě různé podsítě IP a chcete, aby měl váš počítač přímý přístup k oběma. V takovém případě můžete použít tento zápis:

```
root# ifconfig eth0 192.168.1.1 netmask 255.255.255.0 up
root# route add -net 192.168.1.0 netmask 255.255.255.0 eth0

root# ifconfig eth0:0 192.168.10.1 netmask 255.255.255.0 up
root# route add -net 192.168.10.0 netmask 255.255.255.0 eth0:0
```

Budete-li chtít odstranit nějaký alias, stačí přidat na konec názvu pomlčku, jako v následujícím příkazu:

```
root# ifconfig eth0:0- 0
```

Automaticky budou odstraněny všechny trasy sdružené s tímto zařízením.

IP firewall (pro jádro 2.0)

Problematika IP firewallů je podrobněji popsána v dokumentu Firewall-HOWTO. Firewally chrání váš počítač před neautorizovaným síťovým přístupem, což provádějí filtrováním nebo propouštěním datagramů z a na vámi nastavené IP adresy. Existují tři různé třídy pravidel: příchozí filtrování, odchozí filtrování a předávací filtrování. Příchozí pravidla jsou aplikována na datagramy, které obdrží síťové zařízení. Odchozí pravidla jsou aplikována na datagramy, které mají být síťovým zařízením odeslána. Předávací pravidla jsou aplikována na datagramy, které zařízení obdrží, ale nejsou určeny pro lokální počítač, tedy na datagramy, které je třeba směrovat.

Volby překladu jádra

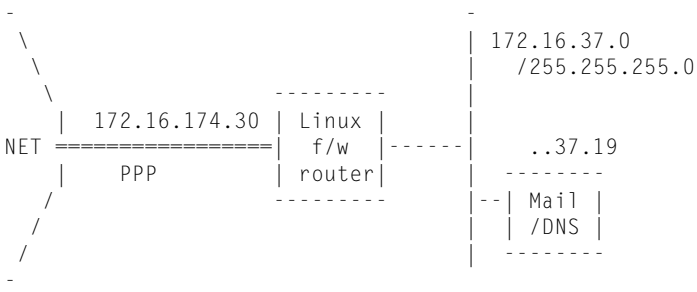
```
Networking options  --->
    [*] Network firewalls
    ....
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: firewalling
    [ ] IP: firewall packet logging
```

Konfigurace pravidel firewallu se provádí pomocí příkazu *ipfwadm*. Jak jsem již uvedl, nepatřím v otázkách bezpečnosti mezi experty, takže ačkoli vám zde představím jednoduchý příklad, který můžete použít, je-li pro vás bezpečnost prvořadá, měli byste zkoumat sami na vlastní pěst a vytvořit si svá vlastní pravidla.

Pravděpodobně nejběžnějším využitím firewallu je případ, kdy používáte linuxový počítač jako směrovač a firewallovou bránu k ochraně vaší lokální počítačové sítě před neautorizovaným přístupem z vnějšku.

Následující konfigurace vychází z příspěvku, jehož autorem je Arnt Gulbrandsen, *agulbra@troll.no*.

Příklad popisuje konfiguraci pravidel firewallu na bráně znázorněné na tomto diagramu:



Následující příkazy by byly normálně umístěny v souboru `rc`, aby došlo k jejich automatickému spuštění při startu systému. Pro zajištění maximální bezpečnosti by měly být tyto příkazy provedeny až po konfiguraci síťových rozhraní, ale před vlastním spuštěním těchto zařízení, aby nemohl nikdo získat přístup v době, kdy počítač startuje.

```
#!/bin/sh

# Flush the 'Forwarding' rules table
# Change the default policy to 'accept'
#
/sbin/ipfwadm -F -f
/sbin/ipfwadm -F -p accept
#
# .. and for 'Incoming'
#
/sbin/ipfwadm -I -f
/sbin/ipfwadm -I -p accept

# First off, seal off the PPP interface
# I'd love to use '-a deny' instead of '-a reject -y' but then it
# would be impossible to originate connections on that interface
# too.
# The -o causes all rejected datagrams to be logged. This trades
# disk space against knowledge of an attack of configuration error.
#
/sbin/ipfwadm -I -a reject -y -o -P tcp -S 0/0 -D 172.16.174.30

# Throw away certain kinds of obviously forged packets right away:
# Nothing should come from multicast/anycast/broadcast addresses
#
/sbin/ipfwadm -F -a deny -o -S 224.0/3 -D 172.16.37.0/24
#
# and nothing coming from the loopback network should ever be
# seen on a wire
#
/sbin/ipfwadm -F -a deny -o -S 127.0/8 -D 172.16.37.0/24

# accept incoming SMTP and DNS connections, but only
# to the Mail/Name Server
```

```
#
/sbin/ipfwadm -F -a accept -P tcp -S 0/0 -D 172.16.37.19 25 53
#
# DNS uses UDP as well as TCP, so allow that too
# for questions to our name server
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 -D 172.16.37.19 53
#
# but not „answers“ coming to dangerous ports like NFS and
# Larry McVoy's NFS extension.  If you run squid, add its port here.
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 53 \
              -D 172.16.37.0/24 2049 2050

# answers to other user ports are okay
#
/sbin/ipfwadm -F -a accept -P udp -S 0/0 53 \
              -D 172.16.37.0/24 53 1024:65535

# Reject incoming connections to identd
# We use 'reject' here so that the connecting host is told
# straight away not to bother continuing, otherwise we'd experience
# delays while ident timed out.
#
/sbin/ipfwadm -F -a reject -o -P tcp -S 0/0 -D 172.16.37.0/24 113

# Accept some common service connections from the 192.168.64 and
# 192.168.65 networks, they are friends that we trust.
#
/sbin/ipfwadm -F -a accept -P tcp -S 192.168.64.0/23 \
              -D 172.16.37.0/24 20:23

# accept and pass through anything originating inside
#
/sbin/ipfwadm -F -a accept -P tcp -S 172.16.37.0/24 -D 0/0

# deny most other incoming TCP connections and log them
# (append 1:1023 if you have problems with ftp not working)
#
/sbin/ipfwadm -F -a deny -o -y -P tcp -S 0/0 -D 172.16.37.0/24

# ... for UDP too
#
/sbin/ipfwadm -F -a deny -o -P udp -S 0/0 -D 172.16.37.0/24
```

Správná konfigurace firewallu je poměrně obtížná. Tento příklad by vám měl posloužit jako odrazový můstek. Manuálové stránky příkazu *ipfwadm* nabízí jistou pomoc při používání tohoto nástroje. Budete-li konfigurovat firewall, poptejte se kolem a snažte se získat co nejvíce rad ze zdrojů, které považujete za spolehlivé, a požádejte někoho, aby rozumně otestoval vaši konfiguraci z vnějšku.

IP firewall (pro jádro 2.2)

Nový kód firewallu poskytuje mechanismus „IP Firewall Chains“. Podrobnější informace naleznete na domovské stránce na adrese *http://www.adelaide.net.au/~rustcorp/ipfwchains/ ipfwchains.html*. Kromě jiných změn budete filtry konfigurovat programem *ipchains* a ne programem *ipfwadm*.

IPIP zapouzdření

Proč byste chtěli zapouzdřovat IP datagramy do IP datagramů? Pokud jste o tom zatím neslyšeli, bude vám to asi připadat zvláštní. Zde je tedy příklad využití: Mobile-IP a IP-Multicast. Pravděpodobně nejrozšířenější využití má tato technika v málo známé oblasti, kterou je amatérské rádio.

Volby překladu jádra

```
Networking options  --->
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    <*> IP: tunneling
```

Zařízení provádějící IP tunelování se nazývají `tunl0`, `tunl1` a podobně.

„Ale proč?“ No protože konvenční IP směrování nařizuje, že síťová část IP adresy je dána adresou sítě a síťovou maskou. Takto vzniká množství souvislých adresových oblastí, které mohou být všechny směrovány přes jeden směrovací záznam. To je poměrně výhodné, ale znamená to, že všechny konkrétní IP adresy můžete používat jen v případě, kdy jste připojeni ke konkrétní části sítě, které tyto adresy náleží. Ve většině případů je to v pořádku, ale patříte-li k mobilním uživatelům sítě, pak nejste celou dobu připojeni k jednomu místu. IPIP zapouzdření (nebo také IP tunelování) dovoluje toto omezení překonat tím, že dovolí IP datagramy určené pro vaši IP adresu zabalit a přesměrovat na jinou IP adresu. Víte-li dopředu, že budete nějakou dobu pracovat v jiné IP síti, můžete nastavit počítač ve vaší domovské síti tak, aby přijímal datagramy určené pro vaši IP adresu a přesměroval je na adresu, kterou budete ve skutečnosti dočasně používat.

Konfigurace tunelované sítě

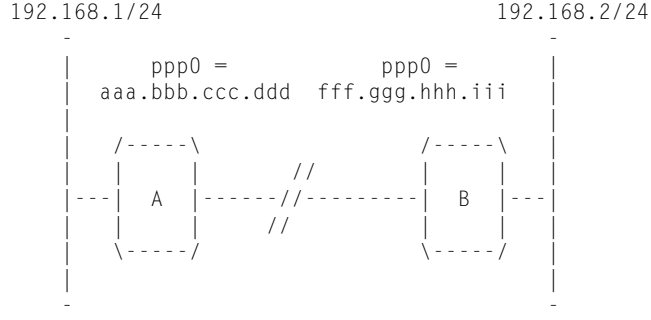


Diagram ilustruje další možný důvod použití IPIP zapouzdření, kterým je virtuální privátní síť. Tento příklad předpokládá, že máte dva počítače, každý s jedním vytáčeným internetovým připojením. Každému hostiteli je přidělena právě jedna IP adresa. Za těmito počítači je určitá soukromá lokální počítačová síť s privátními síťovými IP adresami. Dejme tomu, že chcete libovolnému uživateli v síti A povolit připojení k libovolnému hostiteli v síti B, stejně jako by byli řádně připoje-

ni k Internetu prostřednictvím síťového směrovače. Právě to vám umožní dosáhnout IPIP zapouzdření. Všimněte si, že zapouzdření neřeší problém ohledně způsobu, jakým hostitelé v sítích A a B komunikují s kterýmkoliv jiným hostitelem v Internetu. Stále k tomu budete potřebovat triky typu IP maškaráda. Zapouzdřování normálně provádí počítač fungující jako směrovač.

Linuxový směrovač A by byl nakonfigurován takto:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=fff.ggg.hhh.iii
#
# Ethernet configuration
ifconfig eth0 192.168.1.1 netmask $mask up
route add -net 192.168.1.0 netmask $mask eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.1.1 up
route add -net 192.168.2.0 netmask $mask gw $remotegw tunl0
```

Směrovač B by byl nakonfigurován takto:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=aaa.bbb.ccc.ddd
#
# Ethernet configuration
ifconfig eth0 192.168.2.1 netmask $mask up
route add -net 192.168.2.0 netmask $mask eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.2.1 up
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

Příkaz

```
route add -net 192.168.1.0 netmask $mask gw $remotegw tunl0
```

říká: „Všechny datagramy určené pro adresu `192.168.1.0/24`[2] pošli do IPIP zapouzdřeného datagramu s cílovou adresou `$remotegw`."

Všimněte si, že obě konfigurace si musí odpovídat. Tunelovací zařízení používá `gw` v trase jako cíl pro datagramy, do nichž umístí původní datagramy, jež se mají směrovat. Druhá strana musí vědět, jak došlé datagramy „odpouzdřit", takže musí být také nastavena jako tunelovací zařízení.

2 Pozn. překladatele: Toto je jiný formát zápisu síťové masky. Pokud ovládáte dvojkovou soustavu, víte, že maska `255.255.255.0` je 24 jedniček a 8 nul. Proto ji můžeme zapsat prostě jako 24 a namísto adresy `192.168.1.0/255.255.255.0` dostáváme `192.168.1.0/24`.

Konfigurace tunelovacího hostitele

Směrování nemusí pokrývat celou síť. Můžete třeba směrovat pouze jedinou IP adresu. V takovém případě stačí nastavit zařízení `tunl` na vzdáleném počítači s jeho domácí IP adresou a na konci A použít trasu na hostitele (prostřednictvím ARP proxy) a ne trasu na síť prostřednictvím tunelování. Pojďme tedy naši konfiguraci příslušným způsobem upravit. Nyní máme pouze hostitele B, u kterého chceme, aby se choval jako připojený k Internetu a zároveň byl součástí vzdálené sítě podporované hostitelem `A`:

```
192.168.1/24
    -
    |       ppp0 =                  ppp0 =
    |  aaa.bbb.ccc.ddd       fff.ggg.hhh.iii
    |
    |    /-----\                   /-----\
    |    |     |        //         |     |
    |---|  A  |------//----------|  B  |
    |    |     |      //           |     |
    |    \-----/                   \-----/
    |                         also: 192.168.1.12
    -
```

Směrovač A by byl nakonfigurován takto:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=fff.ggg.hhh.iii
#
# Ethernet configuration
ifconfig eth0 192.168.1.1 netmask $mask up
route add -net 192.168.1.0 netmask $mask eth0
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
# Tunnel device configuration
ifconfig tunl0 192.168.1.1 up
route add -host 192.168.1.12 gw $remotegw tunl0
#
# Proxy ARP for the remote host
arp -s 192.168.1.12 xx:xx:xx:xx:xx:xx pub
```

Počítač B pak nastavíme takto:

```
#!/bin/sh
PATH=/sbin:/usr/sbin
mask=255.255.255.0
remotegw=aaa.bbb.ccc.ddd
#
# ppp0 configuration (start ppp link, set default route)
pppd
route add default ppp0
#
```

```
# Tunnel device configuration
ifconfig tunl0 192.168.1.12 up
route add -net 192.168.1.0 netmask $mask gw $remotegwtunl0
```

Tento druh konfigurace je typičtější pro aplikace typu mobilní IP, kdy se chce jeden hostitel potulovat po Internetu a přitom si po celou dobu zachovat jedinou použitelnou IP adresu. Zajímá-li vás praktická stránka tohoto tématu, pak si přečtěte stať *Mobilní IP*, kde najdete více informací.

IP maškaráda

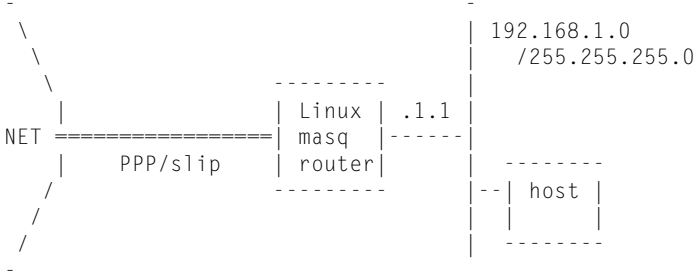
Spousta lidí vlastní jednoduchý vytáčený účet pro připojení k Internetu. Téměř každému, kdo používá tuto konfiguraci, přidělil poskytovatel internetových služeb jednu IP adresu. Za normálních okolností to stačí ke zprostředkování plného přístupu k síti pouze jednomu hostiteli. IP maškaráda je trik, díky němuž může jedinou IP adresu využívat více počítačů – protože se všichni tváří jako jediný hostitel, používá se termín maškaráda. Nevýhodou je, že tato maškaráda funguje skoro vždy jen jedním směrem, to znamená, že maškarádovaní hostitelé mohou volat ven, ale nemohou přijímat síťová spojení od vzdálených hostitelů. To znamená, že zde nefungují některé síťové služby, například *talk* a další, třeba *ftp*, je třeba nastavit na pasivní režim (PASV). Naštěstí nejpoužívanější síťové služby, jako je *telnet*, *www* a *irc* fungují dobře.

Volby překladu jádra

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
    [*] Network firewalls
    ....
    [*] TCP/IP networking
    [*] IP: forwarding/gatewaying
    ....
    [*] IP: masquerading (EXPERIMENTAL)
```

Za normálních okolností necháte počítač podporující protokol SLIP nebo PPP vytáčet stejně, jako by se jednalo o samostatný počítač Kromě toho bude mít nastaveno další síťové zařízení, možná Ethernet, s jednou z rezervovaných síťových adres. Hostitelé, kteří mají být maskováni, by byli na této druhé síti. Každý z těchto hostitelů by měl nastavenu IP adresu ethernetového portu linuxového počítače jako implicitní bránu nebo router.

Typická konfigurace by mohla vypadat asi takto:



Maškaráda programem IPFWADM (pro jádra 2.0.x)

Nejdůležitější příkazy této konfigurace jsou:

```
# Network route for ethernet
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Default route to the rest of the internet.
route add default ppp0
#
# Cause all hosts on the 192.168.1/24 network to be masqueraded.
ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
```

Maškaráda programem IPCHAINS

Princip je podobný programu IPFWADM, příkazy se ale trochu liší:

```
# Network route for ethernet
route add -net 192.168.1.0 netmask 255.255.255.0 eth0
#
# Default route to the rest of the internet.
route add default ppp0
#
# Cause all hosts on the 192.168.1/24 network to be masqueraded.
ipchains -A forward -s 192.168.1.0/24 -j MASQ
```

Více informací o IP maškarádě pod Linuxem získáte na stránce IP Masquerade Resource Page *http://www.huy401.com/achau/ipmasq*. Vynikající dokumentací je také IP-Masquerade mini-HOWTO (najdete zde i informace o tom, jak nakonfigurovat jiné operační systémy pro správnou práci s maškarádou).

Transparentní proxy

Transparentní proxy je funkce, která umožňuje přesměrovat servery nebo služby určené pro jiný počítač na tyto služby na daném počítači. Typicky se používá v případě linuxového počítače pracujícího jako směrovač a zároveň i proxy server. Všechna spojení určená pro danou službu byste měli vzdáleně přesměrovat na lokální proxy server.

Volby překladu jádra

```
Code maturity level options  --->
    [*] Prompt for development and/or incomplete code/drivers
Networking options  --->
    [*] Network firewalls
    ....
    [*] TCP/IP networking
    ....
    [*] IP: firewalling
    ....
    [*] IP: transparent proxy support (EXPERIMENTAL)
```

Konfigurace transparetní proxy se provádí pomocí příkazy *ipfwadm*.

Toto je jeden z užitečných příkladů:

```
ipfwadm -I -a accept -D 0/0 telnet -r 2323
```

Tento příklad způsobí přesměrování všech pokusů o spojení libovolného hostitele s portem 23 (telnet) na port 2323 na tomto hostiteli. Pokud na tomto portu budete nabízet nějakou službu, můžete postoupit telnetové spojení dalšímu počítači, můžete je zaznamenávat nebo můžete dělat cokoliv jiného, co uznáte za vhodné.

Zajímavější příklad je přesměrování veškerého *http* provozu na lokální cache server. Protokol používaný proxy servery se ale poněkud liší od klasického protokolu HTTP. Když se klient připojuje na `www.server.com:80` a žádá o stránku `/path/page`, při připojování přes proxy server by se připojoval na `proxy.local.domain:8080` a žádal by o `www.server.com/path/page`.

K filtrování HTTP požadavků přes lokální proxy potřebujete protokol upravit vložením malého serveru, který se jmenuje *transproxy* (je volně k dispozici na spoustě míst na webu). Server *transproxy* můžete spustit na portu 8081 a zadat následující příkaz:

```
root# ipfwadm -I -a accept -D 0/0 80 -r 8081
```

Program *transproxy* pak přijme veškerá spojení určená původně externím serverům, upraví protokolové rozdíly a předá je lokálnímu cache serveru.

IPv6

Když už máte pocit, že začínáte rozumět protokolu IP, změní se síťová pravidla! IPv6 je zkratka 6. verze protokolu IP. Tento protokol byl vyvinut zejména k vyřešení nedostatku IP adres v Internetu. Adresy protokolu IPv6 jsou dlouhé 16 bajtů (128 bitů). Protokol IPv6 obsahuje také několik dalších změn, většinou se jedná o zjednodušení, což přispěje k lepší ovladatelnosti sítí IPv6 ve srovnání se sítěmi IPv4.

Linux obsahuje funkční, nicméně ne úplnou implementaci protokolu IPv6 v jádrech *2.2.x*.

Informace o IPv6 v Linuxu

- IPv6-HOWTO (*http://www.bieringer.de/linux/IPv6/IPv6-HOWTO/IPv6-HOWTO.html*)
- IPv6 for Linux (*http://www.xelia.ch/Linux/IPng.html*)
- Linux IPv6 RPM Project (*http://v6rpm.jindai.net/*)
- IPv6 FAQ/HOWTO (*http://www.linuxhq.com/IPv6/linux-ipv6.faq.html*)

Mobilní IP

Termín „IP mobilita“ popisuje schopnost hostitele přesunout své síťové spojení z jednoho místa na Internetu do jiného bez změny své IP adresy nebo ztráty konektivity. Typicky platí, že když IP hostitel změní místo svého připojení, musí zpravidla změnit také svoji IP adresu. Mobilita tento problém překonává alokováním pevné IP adresy mobilnímu hostiteli a používáním IP tunelování s automatickým směrováním, čímž se zajistí, že datagramy určené pro tohoto hostitele jsou směrovány na skutečnou IP adresu, kterou právě používá.

V současné době se pracuje na projektu, jehož cílem je kompletní sada mobilních nástrojů pro Linux. Stav tohoto projektu a nástrojů zjistíte na stránce Linux Mobile IP Home Page, *http://gunpowder.stanford.edu/mip*.

Multicast

IP Multicast umožňuje simultánní směrování IP datagramů na libovolný počet IP hostitelů v různých sítích. Tento mechanismus je využíván k vysílání dat (například audia, videa a podobně) po Internetu.

Volby překladu jádra

```
Networking options  --->
    [*] TCP/IP networking
    ....
    [*] IP: multicasting
```

Je nutná sada nástrojů a některé malé úpravy sítě. Informace týkající se způsobu instalace a konfigurace pro Linux najdete v dokumentu Multicast-HOWTO.

Shaper – nastavení přenosové rychlosti

Přenosový shaper je ovladač, který vytvoří nové zařízení, přičemž toto zařízení bude mít uživatelem definovaným způsobem omezenu přenosovou rychlost. Shaper pracuje nad určitým fyzickým zařízením a lze jej použít jako odchozí směrovač síťového provozu.

Shaper se poprvé objevil v Linuxu 2.1.15 a byl zpětně přenesen na Linux 2.0.36 (objevil se v distribuci `2.0.36-pre-patch-2` Alana Coxe, který je autorem shaperu a zároveň udržoval jádro 2.0).

Shaper lze přeložit pouze jako modul a konfiguruje se programem *shapecfg* následujícími příkazy:

```
shapecfg attach shaper0 eth1
shapecfg speed shaper0 64000
```

Shaper dokáže řídit pouze odchozí provoz, protože pakety přes něj procházejí jen díky nastavení směrovací tabulky, takže „směrování podle zdrojové adresy“ by mohlo ve spolupráci se směrovačem vést k omezení celkového síťového provozu na určitém hostiteli.

Linux 2.2 obsahuje podporu tohoto typu směrování, pokud používáte Linux 2.0, použijte úpravu Mika McLagana na adrese *ftp.invlogic.com*. Další informace o shaperu můžete zjistit v dokumentaci `Documentation/networking/shaper`.

Pokud byste chtěli použít shaper regulující i příchozí provoz, vyzkoušejte program `rshaper-1.01` (nebo novější) na adrese *ftp://ftp.systemy.it/pub/develop*.

Pokročilé síťové funkce v jádře 2.2

V jádře 2.2 byly zásadně rozšířeny možnosti směrování. Bohužel dokumentace k těmto novým funkcím (pokud vůbec existuje) se hledá jen velmi obtížně.

Pokusili jsme se nějaké informace shromáždit. Jen co nám vyjde čas, pokusíme se doplnit víc a podrobněji vysvětlit některé funkce.

V jádře 2.0 a nižších používal Linux k práci se záznamy ve směrovací tabulce standardní příkaz *route*. Pokud byste zadali příkaz *netstat –rn*, uviděli byste, jak tabulka vypadá.

V novějších jádrech (2.1 a vyšších) máte jinou možnost. Tato možnost je založena na pravidlech a umožňuje vám vytvářet více směrovacích tabulek. Nově zavedená pravidla umožňují velmi pružně rozhodovat o tom, jak pakety zpracovávat. Můžete volit směrování založené nejen na cílové adrese, ale i na zdrojové adrese, TOS nebo zařízení.

Základy

Výpis směrovací tabulky:

```
ip route
```

Příklad výpisu vypadá takto:

```
207.149.43.62 dev eth0 scope link
207.149.43.0/24 dev eth0 proto kernel scope link src 207.149.43.62
default via 207.149.43.1 dev eth0
```

První řádek *207.149.43.62 dev eth0 scope link* definuje trasu pro příslušné zařízení.

Druhý řádek *207.149.43.0/24 dev eth0 proto kernel scope link src 207.149.43.62* říká, že vše co má jít na *207.149.43.0,* musí odejít přes *207.149.43.62*.

Třetí řádek *default via 207.149.43.1 dev eth0* definuje implicitní trasu.

Využití získaných informací

Viděli jsme, jak vypadá jednoduchá směrovací tabulka. Podívejme se nyní, jak se s ní dá pracovat. Nejprve si přečtěte text Policy routing (*http://nic.funet.fi/pub/Linux/PEOPLE/Linus/v2.1/patch-html/patch-2.1.16/linux_Documentation_networking_policy-routing.txt.html*). Pokud jste z toho zmatení, nevadí – on ten text je poměrně matoucí. Obsahuje informace o všem, co nový směrovací kód dokáže.

Přidání trasy pomocí nových ip nástrojů

V předchozím textu jsme si ukázali jak vypsat směrovací tabulku a co údaje v ní znamenají. Výpis tabulky se naštěstí velmi podobá tomu, co budete zadávat při implementaci vlastních směrovacích pravidel:

```
ip route add 207.149.43.62 dev eth0 scope link
ip route add 207.149.43.0/24 dev eth0 proto kernel scope link src 207.149.43.62
ip route add 127.0.0.0/8 dev lo scope link
ip route add default via 207.149.43.1 dev eth0
```

Můžete si všimnout, že vstup i výstup jsou prakticky stejné pouze s tím rozdílem, že při zadávání tras se na začátku každého řádku uvádí *ip route add*.

Poznámka: Jsme si vědomi toho, že dokumentace ke směrování v jádře 2.2 je nedostačující. Myslíme si, že si je toho vědom každý. Pokud máte s touto problematikou nějaké zkušenosti, napište nám na adresu *poet@linuxports.com*. Rádi přivítáme jakékoliv informace, které zde budeme moci prezentovat.

NAT v jádře 2.2

Funkce IP Network Address Translation (překlad síťových adres, NAT) je v podstatě jen standardizovaným větším bratrem IP maškarády. Podrobněji se o této funkci hovoří v RFC 1631. NAT nabízí funkce, které maškaráda neumožňuje, takže je mnohem výhodnější k nasazení na firewallech a na rozsáhlých instalacích.

Alfa implementaci NAT pro Linux 2.0.29 vytvořil Michael Hasenstein, *Michael.Hasenstein@informatik.tu-chemnitz.de*. Jeho dokumentaci a implementaci lze získat na adrese *http://www.suse.de/~mha/HyperNews/get/linux-ip-nat.html*.

Nová implementace TCP/IP v jádře 2.2 má již funkci NAT přímo vestavěnu. Zdá se, že díky této funkci je implementace Michaela Hasensteina zastaralá.

Aby všechno fungovalo, musíte mít jádro nakonfigurováno s volbami CONFIG_IP_ADVANCED_ROUTER, CONFIG_IP_MULTIPLE_TABLES a CONFIG_IP_ROUTE_NAT. Pokud chcete nastavovat složitější pravidla, potřebujete také CONFIG_IP_FIREWALL a CONFIG_IP_ROUTE_FWMARK. Aby tyto funkce jádra fungovaly, budete potřebovat program *ip* Alexeje Kuzněcova, který můžete získat na adrese *ftp://ftp.inr.ac.ru/ip-routing/*.

Pro překlad adres příchozích datagramů se použije příkaz

```
ip route add nat <ext-addr>[/<masklen>] via <int-addr>
```

Tím se v příchozích paketech určených pro *ext-addr* (adresu viditelnou zvenčí) přeloží cílová adresa na *int-addr* (adresa v interní síti, za branou či firewallem). Paket se pak dále směruje podle místní směrovací tabulky. Je možné překládat jak jednotlivé adresy, tak celé bloky:

```
ip route add nat 195.113.148.34 via 192.168.0.2
ip route add nat 195.113.148.32/27 via 192.168.0.0
```

První příklad zpřístupní interní adresu 192.168.0.2 jako 195.113.148.34. Druhý příklad ukazuje mapování bloku adres 192.168.0.0 – 31 na 195.113.148.32 – 63.

Přehled IP příkazů v jádře 2.2 (ve výstavbě)

ip

Máte-li nainstalovány nástroje *iproute2*, po zadání příkazu *ip* se zobrazí jeho základní syntaxe:

```
[root@jd Net4]# ip
Usage: ip [ OPTIONS ] OBJECT { COMMAND | help }
where OBJECT := { link | addr | route | rule | neigh | tunnel |
                  maddr | mroute | monitor }
      OPTIONS := { -V[ersion] | -s[tatistics] | -r[esolve] |
                   -f[amily] { inet | inet6 | dnet | link } | -o[neline] }13;
```

Můžete použít následující přepínače:

-V, *-Version*: vytiskne verzi používaného programu *ip* a skončí.

-s, *-stats*, *-statistics*: podrobnější výstup k danému zařízení. Tento parametr je možné zadat několikrát a vyvolat tak zobrazení ještě většího množství údajů.

-f, *-family*, následuje rodina protokolů zadaná identifikátorem jako *inet*, *inet6* nebo *link*: Specifikuje, která protokolová rodina se má použít. *inet* používá IPv4 (tedy současný standard), *inet6* používá IPv6 (nový standard), *link* je fyzická linka. Pokud tento údaj nezadáte, pokusí se systém odhadnout použitou rodinu a pokud k tomu nemá dostatek údajů, použije standardní nastavení.

-o, *-oneline*: výstup bude pro každé zařízení na jednom řádku.

-r, *-resolve*: použije resolver (například DNS) k převodu IP adres na názvy.

Údaj *OBJECT* definuje objekt nebo zařízení, které se má nastavit nebo pro něž se mají vypsat informace. Současná implementace zná následující objekty:

- *link* – síťové zařízení, například `eth0` nebo `ppp0`
- *address* – adresa (IP nebo IPv6) zadaného zařízení
- *neigh* – položka ARP nebo NDISC tabulky
- *route* – položka směrovací tabulky
- *rule* – pravidlo v databázi směrovacích politik
- *madress* – multicastová adresa
- *mroute* – záznam trasy o multicastové adrese
- *tunnel* – zda provozovat tunelling přes IP

Parametry zadávané u jednotlivých objektů se liší podle prováděné operace (*COMMAND*). Obecně je možné objekty přidávat (*add*), rušit (*delete*) a zobrazit (*show*), ne všechny objekty však umožňují i další operace. Příkaz samozřejmě obsahuje nápovědu, která zobrazuje povolenou syntaxi příkazů pro různé typy objektů.

Pokud operaci nezadáte, provede se implicitní operace. Tou je typicky vypsání seznamu objektů daného typu, případně, pokud objekty vypsat nelze, zobrazení nápovědy.

Konečně *ARGUMENTS* představuje parametry dané operace. Počet parametrů je dán typem operace a typem objektu. Parametry jsou dvojího typu:

Příznaky jsou tvořeny klíčovým slovem, za nímž následuje hodnota. Kvůli jednoduchosti má každý příkaz nějaké implicitní parametry, které není nutné udávat. Například parametr *dev*> implicitně znamená *ip link*.

Chyby: Všechny operace prováděné příkazem *ip* jsou dynamické. Pokud bude uvedena neplatná syntaxe příkazu, konfigurace systému se nezmění. I v tomto pravidle však existuje výjimka v příkazu *ip link*, který se používá ke změně parametrů zařízení.

Je těžké vypsat zde všechna chybová hlášení (zejména hlášení o syntaktických chybách). Jejich význam je ale obvykle z kontextu příkazu jasný. Nejčastějšími chybami jsou:

1. Netlink není v jádře nakonfigurován. Objeví se hlášení „Cannot open netlink socket: Invalid value“.
2. V jádře není spelling RTNETLINK. V takovém případě může být podle typu příkazu vypsáno hlášení „Cannot talk to rtnetlink: Connection refused“ nebo „Cannot send dump request: Connection refused“.
3. Při konfiguraci jádra nebyl zadán parametr CONFIG_IP_MULTIPLE_TABLES. V takovém případě selžou všechny pokusy o zadání příkazu *ip*, například

```
jd@home $ ip rule list
RTNETLINK error: Invalid argument dump terminated
```


Typický PC hardware

ISDN

ISDN (Intergrated Services Digital Network) je sada standardů, které specifikují přepínané digitální datové sítě pro obecné použití. „Volání“ ISDN vytvoří synchronní dvoubodovou datovou službu s daným cílem. ISDN je obecně posílán po vysokorychlostních linkách, které jsou rozděleny na několik samostatných kanálů. Rozeznáváme dva různé typy kanálů, takzvané B kanály slouží k vlastnímu pře-

nosu dat a jeden D kanál slouží k posílání řídicích informací o navázání spojení a dalších funkcích. V Austrálii například může být ISDN doručován po lince o rychlosti 2 Mb/s, která je rozdělena na 30 samostatných B kanálů o rychlosti 64 kb/s a jeden D kanál o rychlosti 64 kb/s. Současně může být využíván libovolný počet a kombinace kanálů. Je možné například navázat 30 samostatných hovorů o rychlosti 64 kb/s s 30 různými místy nebo 15 hovorů s 15 různými místy o rychlosti 128 kb/s (každý hovor bude využívat dva kanály) případně jen malý počet hovorů a zbytek kanálů nechat zahálet. Kanál může být využíván buď pro příchozí, nebo odchozí hovor. Původním záměrem ISDN bylo umožnit telekomunikačním společnostem poskytovat jedinou datovou službu, která by umožnila doručování buď telefonních (prostřednictvím digitalizace hlasu), nebo datových služeb do domácností nebo firem bez nutnosti provádění zvláštních změn v konfiguraci.

Existuje několik různých způsobů, jakými se může váš počítač připojit ke službě ISDN. První způsob spočívá ve využití zařízení zvaného „Terminal Adapter“, které se zapojí do jednoho ze sériových rozhraní „Network Terminating Unit“, které vám nainstalovali pracovníci telekomunikací při zavádění služby ISDN. Jedno z těchto rozhraní slouží k zadávání příkazů pro navázání hovorů a konfiguraci a ostatní jsou skutečně připojena k síťovým zařízením, která budou využívat datové obvody po jejich vytvoření. Linux bude v této konfiguraci fungovat bez jakýchkoliv úprav. S portem na terminálovém adaptéru se zachází stejným způsobem jako s kterýmkoliv jiným sériovým zařízením. Jiný způsob, pro který je navržena podpora jádra ISDN, spočívá v nainstalování karty ISDN do linuxového počítače a v nechání softwaru aby obsluhoval protokoly a prováděl vlastní hovory.

Volby překladu jádra

```
ISDN subsystem  --->
       <*> ISDN support
       [ ] Support synchronous PPP
       [ ] Support audio via ISDN
       < > ICN 2B and 4B support
       < > PCBIT-D support
       < > Teles/NICCY1016PC/Creatix support
```

Linuxová implementace ISDN podporuje několik různých typů interních karet ISDN. Následující karty nabízí přímo jádro:

- ICN 2B a 4B
- Octal PCBIT-D
- Karty Teles ISDN a kompatibilní

Některé z těchto karet vyžadují pro správnou funkci natažení softwaru. K tomu slouží samostatná utilita.

Všechny podrobnosti o konfiguraci linuxové podpory ISDN najdete v adresáři */usr/src/linux/Documentation/isdn/* a v dokumentu FAQ věnovaném programu *isdn4linux*, který je dostupný na adrese *http://www.lrz-muenchen.de/~ui161ab/www/isdn/*. (Anglickou verzi tohoto dokumentu získáte klepnutím na anglickou vlajku.)

Poznámka o PPP: Sada protokolů PPP bude fungovat po asynchronních i synchronních sériových linkách. Běžně dodávaný démon protokolu PPP pro Linux s názvem *pppd* podporuje pouze asynchronní režim. Pokud hodláte provozovat protokoly PPP přes službu ISDN, potřebujete speciálně upravenou verzi. Podrobnosti ohledně jejího získání najdete ve výše uvedené dokumentaci.

PLIP pro Linux 2.0

Názvy PLIP zařízení jsou `plip0`, `plip1` a tak dále.

Volby překladu jádra

```
Networking options  --->
       <*> PLIP (parallel port) support
```

Protokol PLIP (Parallel Line IP) je podobný protokolu SLIP v tom, že slouží k poskytování dvoubodových síťových spojení mezi dvěma počítači. Ale je navržen pro paralelní tiskový port místo portu sériového (schéma zapojení kabelu najdete v příslušné stati na konci dokumentu). Protože pomocí paralelního portu je možné přenášet více než jeden bit současně, dosahuje se u rozhraní PLIP vyšších rychlostí než u standardního sériového zařízení. Kromě toho i ten nejjednodušší paralelní port, totiž port tiskárny, lze použít místo poměrně drahého rozhraní 16550AFN UART pro sériové porty. Protokol PLIP využívá v porovnání se spojením přes sériový port velkou část času CPU a pokud můžete získat nějaké levné ethernetové karty, není dobrou volbou. Pokud však nemáte k dispozici nic jiného, můžete se na něj spolehnout. Při dobrém spojení můžete očekávat přenosové rychlosti kolem 20 kilobajtů za sekundu.

Ovladače zařízení protokolu PLIP se „perou“ s ovladačem paralelních zařízení o paralelní port. Pokud chcete používat oba tyto ovladače, pak byste je měli přeložit jako moduly, abyste mohli volit, který port chcete pro PLIP a které porty pro ovladač tiskárny. Podrobnější informace najdete v dokumentu *Modules mini-HOWTO*.

Některé přenosné počítače používají chipsety, které nebudou s protokolem PLIP fungovat, protože neumožňují některé kombinace signálů, na něž tento protokol spoléhá a tiskárny je nepoužívají.

Linuxové rozhraní PLIP je kompatibilní s Crynwyr Packet Driver PLIP, což znamená, že prostřednictvím PLIP můžete spojit váš linuxový počítač s dosovým počítačem, na kterém běží kterýkoliv jiný druh softwaru TCP/IP.

V jádru verzí 2.0.* jsou zařízením PLIP přiděleny následující vstupně-výstupní porty a přerušení:

Zařízení	Port	Přerušení
plip0	0x3bc	5
plip1	0x378	7
plip2	0x278	2

Pokud vašemu paralelnímu portu žádná z těchto kombinací nevyhovuje, můžete změnit přerušení pomocí příkazu *ifconfig*, jemuž předáte parametr `irq`. Pokud to váš BIOS umožňuje, nezapomeňte na tiskových portech povolit příslušná přerušení. Druhá možnost je, při použití modulů, zadat parametry `io=` a `irq=` příkazu *insmod*. Například.

```
root# insmod plip.o io=0x288 irq=5
```

Chování PLIP linky je řízeno dvěma timeouty, jejichž implicitní nastavení ve většině případů vyhovuje. Budete je muset prodloužit, pokud používáte extrémně pomalý počítač – pak ovšem prodlužujete timeouty na tom *druhém* počítači. Existuje program *plipconfig*, který umožňuje změnit tato nastavení bez nutnosti rekompilace jádra. Je dodáván jako součást většiny distribucí Linuxu.

Ke konfiguraci rozhraní PLIP musíte spustit následující příkazy (nebo je zadejte do inicializačních skriptů):

```
root# /sbin/ifconfig plip1 localplip pointopoint remoteplip
root3 /sbin/route add remoteplip plip1
```

V tomto případě se používá paralelní port s adresou 0x378, hodnoty *localplip* a *remoteplip* jsou názvy nebo IP adresy počítačů na obou koncích linky. Doporučuji je uložit v souboru `/etc/hosts`:

```
# plip entries
192.168.3.1    localplip
192.168.3.2    remoteplip
```

Parametr *pointopoint* má stejný význam jako u protokolu SLIP v tom, že specifikuje adresu počítače na druhém konci spojení.

Téměř ve všech ohledech můžete zacházet s rozhraním PLIP, jako by se jednalo o rozhraní SLIP, jen s tou výjimkou, že není zapotřebí ani není možné používat příkazy *dip* a *slattach*.

Další informace najdete v dokumentu PLIP-mini-HOWTO.

PLIP pro Linux 2.2

V době vývoje jádra 2.1 byla vylepšena podpora paralelních portů.

Volby překladu jádra:

```
General setup --->
    [*] Parallel port support
Network device support --->
    <*> PLIP (parallel port) support
```

Nový kód PLIP se chová podobně jako starý (používají se stejné příkazy *ifconfig* a *route* jako ty v předchozím textu), nicméně inicializace zařízení je odlišná vzhledem ke změněnému způsobu obsluhy paralelního portu.

„První“ PLIP zařízení se vždy jmenuje `plip0`, přičemž *první* znamená první zařízení detekované systémem, podobně jako tomu je u Ether zařízení. Jako paralelní port bude použit jeden z dostupných paralelních portů, které můžete vidět v `/proc/parport`. Pokud máte jen jeden paralelní port, uvidíte pouze adresář `/proc/parport/0`.

Pokud se jádru nepodaří detekovat přerušení používané paralelním portem, volání *insmod plip* skončí neúspěšně. V takovém případě přepište číslo přerušení v `/proc/parport/0/irq` a spusťte *insmod* znovu.

Podrobný popis správy paralelních portů je uveden v souboru `Documentation/parport.txt`, který je součástí zdrojového kódu jádra.

PPP

Vzhledem k chování, velikosti, složitosti a flexibilitě protokolu PPP mu byl věnován samostatný dokument HOWTO. Dokument PPP-HOWTO je stále součástí Linux Documentation Projectu (*http://www.linuxdoc.org/*), jeho oficiální umístění je ale na stránkách LinuxPorts.Com (*http://www.linuxports.com/howto/ppp/*).

SLIP klient (zastaralé)

Názvy zařízení SLIP jsou `sl0`, `sl1` a tak dále. První nalezené zařízení bude mít označení `0` a zbytek pak rostoucí čísla v pořadí, v jakém byly konfigurovány.

Volby překladu jádra

```
Network device support  --->
       [*] Network device support
       <*> SLIP (serial line) support
       [ ]  CSLIP compressed headers
       [ ]  Keepalive and linefill
       [ ]  Six bit SLIP encapsulation
```

Protokol SLIP (Serial Line Internet Protocol) dovoluje provozovat protokol TCP/IP po sériové lince, ať už se jedná o telefonní linku s vytáčeným modemem nebo nějaký druh pronajaté linky. Samozřejmě, že abyste mohli používat protokol SLIP, potřebujete mít ve vaší oblasti přístup k SLIP serveru. Spousta univerzit a společností po celém světě takovýto přístup poskytuje.

Zařízení SLIP používá sériové porty vašeho počítače k přenosu IP datagramů. K tomu potřebuje převzít kontrolu nad sériovým zařízením. Zařízení SLIP jsou pojmenována *sl0*, *sl1* a tak dále. Jak to koresponduje s vašimi sériovými zařízeními? Síťový kód pomocí volání *ioctl* (řízení vstupně-výstupních operací) změní sériová zařízení na SLIP zařízení. Tuto proceduru umějí dva programy *dip* a *slattach*.

dip

dip (Dialup IP) je chytrý program, který umí nastavit rychlost sériového zařízení, přikázat vašemu modemu, aby vytočil vzdálený konec linky, automaticky vás přihlásí ke vzdálenému serveru, hledá zprávy, které vám server poslal, a zjistí od nich informace, jako je IP adresa, a pak spustí volání *ioctl*, které je nutné pro přepnutí sériového portu do režimu SLIP. Program *dip* má mocnou podporu skriptů, což můžete využít k automatizaci přihlašovací procedury.

Tento program najdete na adrese *ftp://sunsite.unc.edu/pub/Linux/system/network/serial/dip/dip337o-uri.tgz*.

Při instalaci proveďte následující posloupnost příkazů:

```
user% tar xvzf dip337o-uri.tgz
user% cd dip-3.3.7o
user% vi Makefile
root# make install
```

Soubor `Makefile` předpokládá existenci skupiny jménem *uucp*, ale tento název můžete v závislosti na vaší konfiguraci změnit buď na *dip* nebo *SLIP*.

slattach

Program *slattach* je ve srovnání s programem *dip* velice jednoduchý, snadno se používá, ale není tak důmyslný jako *dip*. Nedisponuje podporou skriptů a veškerá jeho činnost spočívá v konfiguraci vašeho sériového zařízení jako zařízení SLIP. Předpokládá, že máte všechny potřebné informace a před jeho spuštěním je navázáno spojení po sériové lince. Program *slattach* je ideální tam, kde máte trvalé spojení se serverem, například fyzickým kabelem nebo vyhrazenou linkou.

Kdy který program použít?

Program *dip* byste měli použít, pokud se ke SLIP serveru připojujete prostřednictvím modemu nebo nějakého jiného dočasného spojení. Program *slattach* byste měli použít, je-li váš počítač spojen se serverem vyhrazenou linkou, třeba kabelem, a pro správnou funkci tohoto spojení není třeba provádět nějaké speciální činnosti. Více informací získáte v části *Trvalé spojení SLIP*.

Konfigurace protokolu SLIP je podobná konfiguraci ethernetového rozhraní (nahlédněte do části *Konfigurace ethernetového rozhraní* výše). Jsou zde však některé klíčové rozdíly.

Předně spojení SLIP se liší od ethernetových sítí tím, že zde jsou v síti vždy jen dva hostitelé, na každém konci jeden. Na rozdíl od Ethernetu, který lze používat ihned po připojení kabelů, u protokolu SLIP je třeba, v závislosti na typu spojení, určitým speciálním způsobem síťové spojení inicializovat.

Pokud používáte program *dip*, nedojde k tomu normálně při startu systému, ale o něco později, až budete připraveni navázat spojení. Tuto proceduru je možné automatizovat. Používáte-li program *slattach*, bude nutné přidat do souboru `rc.inet1` novou sekci, kterou si popíšeme za chvíli.

Existují dva základní typy SLIP serverů: servery s dynamickou IP adresou a servery se statickou IP adresou. Skoro každý SLIP server vás po vytočení vyzve k přihlášení za pomoci vašeho uživatelského jména a hesla. Program *dip* vás může přihlásit automaticky.

Statický SLIP server s vytáčenou linkou a programem DIP

Statický SLIP server vám poskytujete IP adresu, která je výlučně vaše. Pokaždé, když se připojíte k tomuto serveru, nastaví se SLIP port na tuto adresu. Statický SLIP server odpoví vašemu modemu, pravděpodobně vás vyzve k zadání vašeho uživatelského jména a hesla a potom vám přes toto spojení pošle veškeré datagramy určené pro vaši adresu. Máte-li server se statickou IP adresou, můžete název vašeho hostitele a IP adresu (pokud je dopředu znáte) umístit do souboru `/etc/hosts`. Také bude potřeba překonfigurovat některé další soubory: `rc.inet2`, `host.conf`, `resolv.conf`, `/etc/HOSTNAME` a `rc.local`. Nezapomeňte, že při úpravě souboru `rc.inet1` do něj nemusíte přidávat žádné speciální příkazy týkající se SLIP spojení, protože program *dip* provede veškerou práci související s konfigurací rozhraní za vás. Stačí, když programu *dip* poskytnete příslušné informace, a on nejprve přikáže modemu, aby navázal spojení a přihlásil se k vašemu SLIP serveru, a potom sám rozhraní nakonfiguruje.

Pokud váš SLIP server funguje tímto způsobem, pak přeskočte na další část *Používání programu DIP*, kde se dozvíte, jak jej nakonfigurovat.

Dynamický SLIP server s vytáčenou linkou a programem DIP

Dynamický SLIP server vám při každém přihlášení přidělí IP adresu náhodně ze skupiny adres. To znamená, že nelze zaručit, že budete mít pokaždé jednu konkrétní adresu a že tuto adresu může po vašem odhlášení využívat někdo jiný. Správce, který prováděl konfiguraci SLIP serveru, mu přidělil skupinu adres, které může používat. Když server obdrží nový hovor, vyhledá první nepoužívanou adresu, provede volajícího přihlašovacím procesem a potom zobrazí uvítací zprávu, která obsahuje přidělenou IP adresu a tuto adresu pak bude používat po celou dobu trvání hovoru.

Konfigurace tohoto typu serveru je podobná konfiguraci statického serveru. Pouze obsahuje jeden krok navíc, ve kterém je třeba získat IP adresu, kterou vám server vyhradil, a předat ji zařízení SLIP.

I v tomto případě za vás provede veškerou práci program *dip*, jehož novější verze vás nejenom přihlásí, ale umí také automaticky vytáhnout a uložit z uvítací zprávy IP adresu, kterou pak můžete použít při konfiguraci zařízení SLIP.

Pokud váš SLIP server funguje tímto způsobem, pak přeskočte na další část *Používání programu DIP*, kde se dozvíte, jak jej nakonfigurovat.

Používání programu DIP

Již jsme si řekli, že program *dip* je mocný nástroj, který vám může za pomoci příkazů *ifconfig* a *route* ulehčit a zcela automatizovat proces vytáčení SLIP serveru, přihlášení, navázání spojení a konfigurace SLIP zařízení.

V podstatě napíšete skript tvořený příkazy, kterým program *dip* rozumí a na jejichž základě provede příslušné akce. Představu o fungování programu *dip* si můžete utvořit podle následujícího vzorového skriptu `sample.dip`, který je dodáván společně s tímto programem. Program *dip* je poměrně výkonný a obsahuje množství voleb.

Nebudeme se jimi zde zabývat, ale v případě zájmu vás odkazuji na manuálové stránky, soubor `README` a vzorové soubory, které získáte současně s vaší verzí programu *dip*.

Možná jste si všimli, že skript `sample.dip` předpokládá, že používáte statický SLIP server, takže budete dopředu znát svou IP adresu. Pro dynamické SLIP servery obsahují novější verze programu *dip* příkaz, který slouží k automatickému přečtení IP adresy, kterou vám dynamický server přidělil, a následnému nakonfigurování zařízení SLIP. Následující ukázka je upravenou verzí souboru `sample.dip`, který je součástí balíku *dip337j-uri.tgz* a má vám posloužit jako jakýsi odrazový můstek. Soubor uložte pod názvem `/etc/dipscript` a upravte podle vaší konfigurace.

```
#
# sample.dip    Dialup IP connection support program.
#
#               This file (should show) shows how to use the DIP
#       This file should work for Annex type dynamic servers, if you
#       use a static address server then use the sample.dip file that
#       comes as part of the dip337-uri.tgz package.
#
#
# Version:      @(#)sample.dip  1.40    07/20/93
#
# Author:       Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
#

main:
# Next, set up the other side's name and address.
# My dialin machine is called 'xs4all.hacktic.nl' (== 193.78.33.42)
get $remote xs4all.hacktic.nl
# Set netmask on sl0 to 255.255.255.0
netmask 255.255.255.0
# Set the desired serial port and speed.
port cua02
speed 38400

# Reset the modem and terminal line.
# This seems to cause trouble for some people!
reset

# Note! „Standard“ pre-defined „errlevel“ values:
#  0 - OK
#  1 - CONNECT
#  2 - ERROR
#
# You can change those grep'ping for „addchat()“ in *.c...

# Prepare for dialing.
 send ATQ0V1E1X4\ r
wait OK 2
if $errlvl != 0 goto modem_trouble
```

```
dial 555-1234567
if $errlvl != 1 goto modem_trouble

# We are connected.  Login to the system.
login:
sleep 2
wait ogin: 20
if $errlvl != 0 goto login_trouble
send MYLOGIN\ n
wait ord: 20
if $errlvl != 0 goto password_error
send MYPASSWD\ n
loggedin:
# We are now logged in.
wait SOMEPROMPT 30
if $errlvl != 0 goto prompt_error

# Command the server into SLIP mode
send SLIP\ n
wait SLIP 30
if $errlvl != 0 goto prompt_error

# Get and Set your IP address from the server.
#   Here we assume that after commanding the SLIP server into SLIP
#   mode that it prints your IP address
get $locip remote 30
if $errlvl != 0 goto prompt_error

# Set up the SLIP operating parameters.
get $mtu 296
# Ensure „route add -net default xs4all.hacktic.nl“ will be done
default

# Say hello and fire up!
done:
print CONNECTED $locip ---> $rmtip
mode CSLIP
goto exit

prompt_error:
print TIME-OUT waiting for sliplogin to fire up...
goto error

login_trouble:
print Trouble waiting for the Login: prompt...
goto error

password:error:
print Trouble waiting for the Password: prompt...
goto error

modem_trouble:
print Trouble occurred with the modem...
```

```
error:
print CONNECT FAILED to $remote
quit

exit:
exit
```

Výše uvedený příklad předpokládá volání dynamického SLIP serveru. Voláte-li statický SLIP server, použijte soubor `sample.dip`, který je součástí balíku `dip337j-uri.tgz`.

Když předáte programu *dip* příkaz *get $local*, prohledá příchozí text, zda neobsahuje řetězec, který vypadá jako IP adresa, to znamená řetězec čísel oddělených tečkami. Tato úprava byla do programu přidána kvůli dynamickým SLIP serverům, aby mohl být proces čtení IP adresy poskytnuté serverem automatizován.

Výše uvedený příklad automaticky vytvoří implicitní trasu přes vaše SLIP spojení. Pokud si to nepřejete, může ponechat ethernetové spojení jako implicitní trasu, potom odstraňte příkaz *default* ze skriptu. Když po skončení tohoto skriptu spustíte příkaz *ifconfig*, zjistíte, že máte nové zařízení jménem `sl0`. To je vaše zařízení SLIP. Dle potřeby můžete manuálně upravit jeho konfiguraci po skončení příkazu *dip* pomocí příkazů *ifconfig* a *route*.

Všimněte si, že příkaz *mode* programu *dip* umožňuje výběr několika různých protokolů, nejpoužívanějším příkladem je CSLIP, což znamená SLIP s kompresí. Všimněte si také, že musí souhlasit oba konce spojení, takže cokoliv zvolíte, by mělo souhlasit s nastavením vašeho serveru.

Výše uvedený příklad je poměrně robustní a měl by odolat většině chyb. Podrobnější informace najdete v manuálových stránkách programu *dip*. Přirozeně byste mohli třeba vytvořit skript, který by prováděl opakované vytáčení serveru, pokud by nebylo navázáno spojení po uplynutí stanovené doby, nebo vyzkoušet skupinu serverů, pokud přistupujete k více než jednomu serveru.

Trvalé spojení SLIP s vyhrazenou linkou a programem slattach

Máte-li dva počítače propojeny kabelem, případně vyhrazenou linkou nebo nějakým jiným typem trvalého sériového spojení, nepotřebujete se zatěžovat všemi problémy kolem používání programu *dip* na sériové linky. Při konfiguraci spojení úplně vystačíte s jednoduchou utilitou jménem *slattach*.

Protože bude vaše spojení trvalé, bude nutné přidat určité příkazy do souboru `rc.inet1`. V podstatě je potřeba nastavit správnou rychlost sériového zařízení a přepnout ho do režimu SLIP. Díky programu *slattach* vám k tomu stačí jeden příkaz. Následující řádky přidejte do souboru `rc.inet1`.

```
#
# Attach a leased line static SLIP connection
#
#  configure /dev/cua0 for 19.2kbps and cslip
/sbin/slattach -p cslip -s 19200 /dev/cua0 &
/sbin/ifconfig sl0 IPA.IPA.IPA.IPA pointopoint IPR.IPR.IPR.IPR up
#
# End static SLIP.
```

Kde:

IPA.IPA.IPA.IPA je vaše IP adresa

IPR.IPR.IPR.IPR je IP adresa vzdáleného počítače

Program *slattach* přidělí první volné zařízení SLIP specifikovanému sériovému zařízení. Začne označením `sl0`. Takto první příkaz *slattach* přiřadí specifikovanému zařízení SLIP zařízení `sl0`, další pak `sl1` a tak dále.

Program *slattach* umožňuje pomocí parametru `-p` konfigurovat různé protokoly. Ve vašem případě použijete buď SLIP nebo CSLIP, podle toho, zda chcete nebo nechcete používat kompresi.

SLIP server

Pokud máte počítač, který je připojen k síti, a chcete, aby se k němu mohli připojovat i jiní uživatelé, budete muset tento počítač nakonfigurovat jako server. Chcete-li jako protokol sériové linky používat SLIP, máte, co se týče způsobu konfigurace vašeho linuxového počítače jako SLIP serveru, tři možnosti. Já osobně bych dal přednost první z nich, která využívá program *sliplogin* a připadá mi nejjednodušší co se konfigurace a pochopení týče. Nicméně představím vám všechny tři, abyste si mohli udělat vlastní úsudek.

SLIP server používající program sliplogin

Program *sliplogin* lze použít místo klasického přihlašovacího programu pro uživatele, kteří převedou terminálovou linku na SLIP linku. Umožňuje nastavit váš linuxový počítač jako statický server (uživatelé získají při každém zavolání vždy stejnou adresu) nebo jako dynamický server (uživatelům je přidělena adresa, která nutně nemusí být stejná jako při předchozím spojení).

Volající se přihlásí jako při normálním přihlašování, zadá své uživatelské jméno a heslo. Ovšem místo příkazového interpretu se po přihlášení spustí program *sliplogin*, který vyhledá v konfiguračním souboru (`/etc/slip.hosts`) záznam s uživatelským jménem, které odpovídá jménu volajícího. Pokud takový záznam najde, nastaví linku jako 8bitovou a pomocí volání *ioctl* ji převede na SLIP linku. Po skončení tohoto procesu začne poslední fáze konfigurace, kdy program *sliplogin* spustí skript příkazového interpretu, který přidělí rozhraní SLIP IP adresu, síťovou masku a nastaví směrování. Tento skript se většinou nazývá `/etc/slip.login`, ale vyžadují-li někteří volající speciální inicializaci, můžete vytvořit konfigurační skripty nazvané `/etc/slip.login.přihlašovacíjméno`, které budou spouštěny místo tohoto implicitního.

Pro správnou funkci programu sliplogin potřebujete tři nebo čtyři soubory. Podrobně vám popíši, jak a kde získáte příslušný software a jak ho nakonfigurujete. Jedná se o tyto soubory:

- `/etc/passwd` pro vytáčené uživatelské účty
- `/etc/slip.hosts` pro ukládání informací, které jsou pro každého uživatele jedinečné
- `/etc/slip.login` řídí konfiguraci směrování, kterou je třeba provádět pro každého uživatele
- `/etc/slip.tty` je nutný pouze v případě, že má server nastavenu dynamickou alokaci adres a obsahuje tabulku přidělovaných adres
- `/etc/slip.logout` obsahuje příkazy, které se provedou po zavěšení nebo odhlášení uživatele

Kde získat program sliplogin?

Je možné, že jste balík *sliplogin* získali jako součást vaší distribuce Linuxu. Pokud nikoliv, získáte program *sliplogin* na adrese *ftp://sunsite.unc.edu/pub/linux/system/network/serial/sliplogin-2.1.1.tar.gz*. Archiv ve formátu `tar` obsahuje zdrojový kód, předkompilované binární soubory a manuálovou stránku.

Aby mohli program sliplogin používat pouze autorizovaní uživatelé, měli byste do souboru `/etc/group` přidat záznam podobný tomu následujícímu:

```
..
slip::13:radio,fred
..
```

Po nainstalování balíku *sliplogin* změní soubor `Makefile` vlastnictví skupinu programu *sliplogin* na `slip`, což znamená, že ho mohou spouštět pouze uživatelé patřící do této skupiny. Výše uvedený zápis povolí spouštění tohoto programu pouze uživatelům `radio` a `fred`.

Následující posloupnost příkazů nainstaluje binární soubory do adresáře `/sbin` a manuálové stránky do sekce 8.

```
# cd /usr/src
# gzip -dc .../sliplogin-2.1.1.tar.gz | tar xvf -
# cd sliplogin-2.1.1
# <..editujte Makefile pokud nepoužíváte stínová hesla..>
# make install
```

Pokud budete chtít binární soubory ještě před instalací přeložit, spusťte před příkazem `make install` ještě příkaz `make clean`. Budete-li chtít nainstalovat binární soubory někam jinam, musíte upravit pravidlo install v souboru `Makefile`.

Konfigurace souboru `/etc/passwd` pro SLIP hostitele

Normálně byste pro tyto uživatele vytvořili v souboru `/etc/passwd` speciální účty. Běžně se před jméno volajícího hostitele přidává písmeno „S“. Takže když se například volající hostitel jmenuje `radio`, vytvoříte v souboru `/etc/passwd` následující záznam:

```
Sradio:FvKurok73:1427:1:radio SLIP login:/tmp:/sbin/sliplogin
```

Nijak zvlášť nezáleží na tom, jak se bude účet jmenovat, stačí, abyste se v účtech vyznali.

Poznámka: Volající nepotřebuje mít žádný speciální adresář, protože nebude mít přístup k příkazovému interpretu, takže postačí adresář `/tmp`. Nezapomeňte také, že místo normálního přihlašovacího příkazového interpretu se používá program *sliplogin*.

Konfigurace souboru `/etc/slip.hosts`

V souboru `/etc/slip.hosts` hledá program *sliplogin* záznamy vyhovující uživatelskému jménu, aby získal konfigurační informace pro příslušného volajícího. V tomto souboru specifikujete IP adresu a síťovou masku, která bude přidělena volajícímu. Následují vzorové záznamy pro dva hostitele, první je statická konfigurace pro hostitele `radio` a druhá dynamická pro uživatele `albert`:

```
#
Sradio   44.136.8.99   44.136.8.100  255.255.255.0  normal      -1
Salbert  44.136.8.99   DYNAMIC       255.255.255.0  compressed  60
#
```

Záznamy v souboru `/etc/slip.hosts` jsou složeny z následujících položek:

1. Přihlašovací jméno volajícího uživatele.
2. IP adresa serveru, to znamená tohoto počítače.
3. IP adresa, která bude přidělena volajícímu. Je-li v tomto poli uveden řetězec `DYNAMIC`, bude IP adresa přidělena v závislosti na informaci obsažené v souboru `/etc/slip.tty`, o kterém budeme mluvit za chvíli. Poznámka: Aby vše fungovalo tak, jak má, musíte používat program sliplogin alespoň verze 1.3.

4. Síťová maska přidělená volajícímu počítači v tečkové notaci, tedy 255.255.255.0 pro síťovou masku třídy C.
5. Nastavení režimu SLIP, kterým můžete povolit nebo zakázat kompresi. Povolené jsou hodnoty `normal` nebo `compressed`.
6. Časový údaj, který specifikuje dobu, po kterou může linka zůstat v nečinnosti (nepřijímá žádné datagramy), než dojde k automatickému zrušení spojení. Záporná hodnota tuto vlastnost ruší.
7. Nepovinné parametry.

Poznámka: Pole 2 a 3 mohou obsahovat buď názvy hostitelů, nebo IP adresy. Použijete-li názvy hostitelů, musí být váš počítač schopen zjistit příslušnou IP adresu, jinak skript selže. Můžete si to ověřit příkazem telnet, kterému předáte jako parametr název hostitele. Pokud se objeví zpráva „*Trying nnn.nnn.nnn...*“, podařilo se vašemu počítači nalézt IP adresu odpovídající tomuto názvu hostitele. Objeví-li se zpráva „*Unknown host*“, potom se to nepodařilo. V takovém případě buď použijte IP adresu převedenou na tečkovou desítkovou notaci, nebo upravte konfiguraci resolveru (viz *Konfigurace resolveru*).

Nejpoužívanější režimy SLIP jsou:

`normal` – zapíná normální nekomprimovaný SLIP

`compressed` – zapíná van Jacobsonovu kompresi hlaviček (CSLIP)

Samozřejmě, že se tyto režimy navzájem vylučují, takže musíte použít buď jeden, nebo druhý. Další informace o ostatních volbách najdete v manuálových stránkách.

Konfigurace souboru `/etc/slip.login`

Jakmile program *sliplogin* nalezl vyhovující záznam v souboru `/etc/slip.hosts`, pokusí se zpracovat soubor `/etc/slip.login` a dojde na vlastní konfiguraci rozhraní SLIP, kterému je předána IP adresa a síťová maska.

Vzorový soubor `/etc/slip.login`, který je dodáván s balíkem *sliplogin*, vypadá takto:

```
#!/bin/sh -
#
#       @(#)slip.login  5.1 (Berkeley) 7/1/90
#
# generic login file for a SLIP line.  sliplogin invokes this with
# the parameters:
#     $1       $2      $3    $4, $5, $6 ...
#   SLIPunit ttyspeed   pid   the arguments from the slip.host entry
#
/sbin/ifconfig $1 $5 pointopoint $6 mtu 1500 -trailers up
/sbin/route add $6
arp -s $6 <hw_addr> pub
exit 0
#
```

Jistě si všimnete, že tento skript používá ke konfiguraci SLIP zařízení příkazy *ifconfig* a *route*, kterým předá IP adresu, vzdálenou IP adresu a síťovou masku, a vytvoří směrování pro vzdálenou adresu prostřednictvím rozhraní SLIP. Stejného výsledku bychom dosáhli i pomocí programu *slattach*.

Nezapomeňte také použít Proxy ARP, aby ostatní hostitelé, kteří jsou připojeni ke stejnému Ethernetu jako server, věděli, jak se dostanou k volajícímu hostiteli. Pole `<hw_addr>` by mělo obsahovat hardwarovou adresu ethernetové karty v tomto počítači. Pokud váš server není součástí ethernetové sítě, můžete tento řádek vynechat.

Konfigurace souboru `/etc/slip.logout`

Jakmile volající zavěsí, budete chtít obnovit normální stav sériového zařízení, aby se mohli dovolat další volající. Tuto funkci plní soubor `/etc/slip.logout`. Jeho formát je poměrně jednoduchý a je spouštěn se stejnými parametry jako soubor `/etc/slip.login`.

```
#!/bin/sh -
#
#               slip.logout
#
/sbin/ifconfig $1 down
arp -d $6
exit 0
#
```

Jeho cílem je „shodit" rozhraní, což odstraní dříve vytvořené manuální směrování. Pomocí příkazu *arp* také smaže všechny záznamy v tabulce ARP. I zde platí, že pokud server neobsahuje ethernetový port, nemusí být ve skriptu příkaz *arp*.

Konfigurace souboru `/etc/slip.tty`

Používáte-li dynamické přidělování adres (máte u některých hostitelů nastaveno v souboru `/etc/slip.hosts` klíčové slovo `DYNAMIC`), musíte v souboru `/etc/slip.tty` uvést seznam adres přidělených jednotlivým portům. K dynamickému přidělování adres potřebujete pouze tento soubor.

Soubor `/etc/slip.tty` je vlastně tabulka obsahující zařízení `tty`, která budou podporovat vytáčené SLIP spojení a IP adresy, jež budou přiděleny uživatelům, kteří na příslušný port zavolají.

Formát souboru vypadá takto:

```
# slip.tty    tty -> IP address mappings for dynamic SLIP
# format: /dev/tty?? xxx.xxx.xxx.xxx
#
/dev/ttyS0      192.168.0.100
/dev/ttyS1      192.168.0.101
#
```

Tato tabulka říká, že volajícímu, který vytočí port `/dev/ttyS0` a má v souboru `/etc/slip.hosts` nastavené pole pro vzdálenou adresu na `DYNAMIC`, přiřadí adresu *192.168.0.100*.

Takto vám pro všechny uživatele, kteří nevyžadují vyhrazené adresy, stačí alokovat jednu adresu pro každý port. Vyhnete se tak plýtvání adresovým prostorem.

SLIP server používající program dip

Dovolte mi začít konstatováním, že některé níže uvedené informace pocházejí z manuálových stránek programu *dip*, kde je stručně popisována konfigurace SLIP serveru pod Linuxem. Předesílám též, že následující fakta vycházejí z balíku *dip337o-uri.tgz* a pravděpodobně se nebudou úplně shodovat s jinými verzemi tohoto programu.

Program *dip* může pracovat v takzvaném vstupním režimu, kdy automaticky vyhledá záznam uživatele, který ho spustil, a nakonfiguruje sériovou linku jako SLIP linku podle informací uložených v souboru `/etc/diphosts`. Tento vstupní režim aktivujete, když program *dip* spustíte jako *diplogin*. Tímto způsobem používáte program *dip* jako SLIP server, vytvoříte speciální účty, kde program *diplogin* funguje jako přihlašovací příkazový interpret.

Nejprve je třeba vytvořit následující symbolický odkaz:

```
# ln -sf /usr/sbin/dip /usr/sbin/diplogin
```

Potom doplníte určité záznamy do souborů `/etc/passwd` a `/etc/diphosts`. Tyto záznamy mají následující formát:

Při konfiguraci Linuxu coby SLIP serveru potřebujete vytvořit pro uživatele určité speciální SLIP účty, přičemž program *dip* (ve vstupním režimu) slouží jako přihlašovací příkazový interpret. Doporučuje se označovat všechny účty SLIP s počátečním písmenem „S“, např. „Sfredm“.

Takto vypadá vzorový záznam SLIP uživatele v souboru `/etc/passwd`:

```
Sfredm:ij/SMxiTlGVCo:1004:10:Fred:/tmp:/usr/sbin/diplogin
^      ^             ^    ^  ^    ^    ^
|      |             |    |  |    |    \___ diplogin jako přihlašovací příkazový interpret
|      |             |    |  |    \________ domovský adresář
|      |             |    |  \_____________ celé jméno uživatele
|      |             |    \________________ ID skupiny
|      |             \_____________________ ID uživatele
|      \__________________________________ zašifrované heslo
\_____________________________________ přihlašovací jméno
```

Jakmile se uživatel přihlásí, program *login*, pokud nalezne a ověří uživatele, spustí příkaz *diplogin*. Program *dip*, je-li spuštěn jako *diplogin*, ví, že má automaticky předpokládat, že je používán jako přihlašovací příkazový interpret. Pokud je spuštěn jako *diplogin*, zjistí nejprve pomocí funkce *getuid()* identifikátor uživatele, který jej spustil. Potom vyhledá v souboru `/etc/diphosts` první záznam, který odpovídá buď identifikátoru uživatele, nebo názvu zařízení `tty`, ze kterého hovor pochází a příslušným způsobem se nakonfiguruje. Moudrým rozhodnutím, zda vyhradit uživateli záznam v souboru `diphosts` nebo mu přidělit implicitní konfiguraci, získáte směs uživatelů s dynamicky a staticky přidělovanými adresami.

Program *dip*, je-li spuštěn ve vstupním režimu, automaticky přidá záznam Proxy-ARP, takže se nemusíte zabývat manuálním přidáváním těchto záznamů.

Konfigurace souboru `/etc/diphosts`

Soubor `/etc/diphosts` slouží programu *dip* k vyhledávání přednastavených konfigurací vzdálených hostitelů. Tito vzdálení hostitelé mohou být uživatelé, kteří se přihlašují k vašemu linuxovému počítači, nebo to mohou být stroje, ke kterým se přihlašujete z vašeho počítače.

Následuje obecný formát souboru `/etc/diphosts`:

```
..
Suwalt::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
ttyS1::145.71.34.3:145.71.34.2:255.255.255.0:Dynamic ttyS1:CSLIP,296
..
```

Toto jsou jednotlivá pole každého záznamu:

1. Přihlašovací jméno: vrácené funkcí getpwuid (getuid()) nebo název tty

2. Nevyužito: (kvůli kompatibilitě s /etc/passwd)
3. Vzdálená adresa: IP adresa volajícího hostitele, buď číselná, nebo název
4. Lokální adresa: IP adresa tohoto počítače, opět buď číselná, nebo název
5. Síťová maska: v tečkové notaci
6. Pole komentáře: sem můžete napsat cokoliv
7. Protokol: SLIP, CSLIP a podobně
8. MTU: desítkové číslo

Příklad záznamu v souboru `/etc/diphosts` pro vzdáleného uživatele může vypadat třeba takto:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:SLIP,296
```

což udává SLIP spojení se vzdálenou adresou 145.71.34.1 a MTU 296 nebo jiný příklad:

```
Sfredm::145.71.34.1:145.71.34.2:255.255.255.0:SLIP uwalt:CSLIP,1006
```

který specifikuje spojení CSLIP se vzdálenou adresou 145.71.34.1 a MTU 1006.

Proto všichni uživatelé, kterým chcete povolit staticky přidělovaný vytáčený IP přístup, musí mít záznam v souboru `/etc/diphosts`. Pro uživatele, kteří vytáčejí konkrétní port, s nímž si dynamicky domlouvají detaily spojení, potřebujete mít záznam pro zařízení `tty`. Nezapomeňte nakonfigurovat alespoň jeden záznam pro každé zařízení `tty`, které používají tito uživatelé, aby pro ně byla k dispozici vhodná konfigurace bez ohledu na modem, se kterým se spojí.

Když se uživatel přihlásí, odpoví na normální přihlašovací výzvu svým přihlašovacím jménem a heslem. Po ověření neuvidí uživatel žádné zprávy a zařízení se přepne do režimu SLIP. Uživatel by se potom měl bez potíží připojit a příslušnému zařízení budou nastaveny parametry ze souboru `diphosts`.

SLIP server používající balík *dSLIP*

Matt Dillon (*dillon@apollo.west.oic.com*) napsal balík, který umí obsluhovat nejen odchozí, ale i příchozí SLIP spojení. Jde o kombinaci malých programů a skriptů, které se starají o vaše spojení. Potřebujete mít nainstalovaný program *tcsh*, protože ho ke své činnosti vyžaduje minimálně jeden skript. Matt Dillon dodává také utilitu *expect* (v binárním tvaru), kterou také vyžaduje jeden ze skriptů. Ke zprovoznění tohoto balíku budete potřebovat určitou zkušenost s programem *expect*, ale tím se nenechte odradit.

Matt Dillon popsal instalaci v souboru `README`, takže se jí zde nebudu zabývat.

Balík *dSLIP* získáte na adrese *ftp://sunsite.unc.edu/pub/Linux/system/network/serial/dslip203.tgz*.

Před spuštěním příkazu *make install* si přečtěte soubor `README` a vytvořte příslušné záznamy v souborech `/etc/passwd` a `/etc/group`.

Další síťové technologie

Následující kapitoly se týkají konkrétních síťových technologií. Informace v nich uvedené se nemusí týkat jiných síťových technologií. Témata jsou řazena abecedně.

ARCNet

Názvy zařízení ARCNet jsou `arc0e`, `arc1e`, `arc2e` a tak dále, nebo `arc0s`, `arc1s`, `arc2s` a tak dále. První jádrem detekované kartě je přiřazen název `arc0e` nebo `arc0s` a zbylé názvy jsou přiřa-

zovány postupně v pořadí, v jakém jsou detekovány. Písmeno na konci názvu označuje, zda jste vybrali paket formátu ethernetové zapouzdření nebo paket formátu RFC1051.

Volby překladu jádra:

```
Network device support  --->
    [*] Network device support
    <*> ARCnet support
    [ ]   Enable arc0e (ARCnet „Ether-Encap“ packet format)
    [ ]   Enable arc0s (ARCnet RFC1051 packet format)
```

Jakmile máte sestaveno jádro s podporou ethernetové karty, je již vlastní konfigurace karty jednoduchá:

Zpravidla byste měli použít zápis podobný tomu následujícímu:

```
root# ifconfig arc0e 192.168.0.1 netmask 255.255.255.0 up
root# route add -net 192.168.0.0 netmask 255.255.255.0 arc0e
```

Podrobnější informace najdete v souborech `/usr/src/linux/Documentation/networking/arcnet.txt` a `/usr/src/linux/Documentation/networking/arcnet-hardware.txt`.

Autorem podpory ARCNet je Avery Pennarun, *apenwarr@foxnet.net*.

Appletalk (AF_APPLETALK)

Podpora protokolu Appletalk nepoužívá žádné speciální názvy zařízení, protože využívá existující síťová zařízení.

Volby překladu jádra:

```
Networking options  --->
    <*> Appletalk DDP
```

Podpora Appletalk umožňuje vašemu počítači spolupracovat se sítěmi Apple. Má důležité využití při sdílení zdrojů jako jsou tiskárny a disky mezi linuxovými počítači a počítači Apple. Potřebujete k domu dodatečný software, který se jmenuje *netatalk*. Wesley Craig (*netatalk@umich.edu*) reprezentuje tým nazvaný „Research Systems Unix Group“ na michiganské univerzitě, který tento balík vytvořil. Poskytuje software, který implementuje protokol Appletalk a některé užitečné utility. Balík *netatalk* bude buď součástí vaší distribuce Linuxu, nebo si ho musíte stáhnout pomocí FTP z adresy *ftp://terminator.rs.itd.umich.edu/unix/netatalk/*.

Balík sestavíte a nainstalujete pomocí následujících příkazů:

```
user% tar xvfz .../netatalk-1.4b2.tar.Z
user% make
root# make install
```

Před voláním příkazu *make* budete možná muset upravit soubor `Makefile`. Konkrétně budete možná chtít změnit proměnnou DESTDIR, která definuje, kam se mají soubory nainstalovat. Implicitní adresář `/usr/local/atalk` je ovšem poměrně bezpečný.

Konfigurace softwaru Appletalk

Nejprve je třeba zkontrolovat, zda jsou v souboru `/etc/services` příslušné záznamy. Jsou to tyto záznamy:

```
rtmp    1/ddp   # Routing Table Maintenance Protocol
nbp     2/ddp   # Name Binding Protocol
echo    4/ddp   # AppleTalk Echo Protocol
zip     6/ddp   # Zone Information Protocol
```

Dalším krokem je vytvoření konfiguračních souborů Appletalk v adresáři `/usr/local/atalk/etc` (nebo v adresáři, do kterého jste nainstalovali balík Appletalk).

První soubor má název `/usr/local/atalk/etc/atalkd.conf`. Z počátku stačí, když bude tento soubor obsahovat pouze jediný řádek, na kterém bude uveden název síťového zařízení, které podporuje síť, k níž jsou připojeny vaše počítače Apple:

```
eth0
```

Démon Appletalk do něj po svém spuštění přidá další podrobnosti.

Export souborových systémů prostřednictvím protokolu Appletalk

Souborové systémy z vašeho linuxového počítače lze exportovat do sítě, aby je mohl sdílet počítač Apple připojený k téže síti.

Aby to bylo možné, je třeba nejdříve upravit soubor `/usr/local/atalk/etc/AppleVolumes.system`. Existuje ještě jeden konfigurační soubor jménem `/usr/local/atalk/etc/AppleVolumes.default`, který má úplně stejný formát a popisuje souborové systémy, které budou mít k dispozici uživatelé přihlašující se na účet `guest`.

Podrobnosti týkající se způsobu konfigurace těchto souborů a popis různých voleb najdete v manuálových stránkách programu *afpd*.

Toto je jednoduchý příklad:

```
/tmp Scratch
/home/ftp/pub „Public Area“
```

Tyto příkazy způsobí export vašeho souborového systému `/tmp` jako svazku AppleShare „Scratch“ a váš veřejný ftp adresář jako svazek AppleShare „Public Area“. Názvy svazků nejsou povinné, démon si nějaké zvolí sám, nicméně když je zadáte, nic zlého se nestane.

Sdílení linuxové tiskárny prostřednictvím protokolu Appletalk

Sdílení linuxové tiskárny na počítači Apple je poměrně jednoduché. Potřebujete spustit program *papd*, což je Appletalk Printer Access Protocol Daemon. Tento program bude po spuštění přijímat požadavky počítačů Apple a předávat tiskové úlohy vašemu démonu tiskárny k vytištění.

Konfigurace démona se provádí v souboru `/usr/local/atalk/etc/papd.conf`. Syntaxe tohoto souboru je stejná jako souboru `/etc/printcap`. Název, který přidělíte definici, je registrován pomocí jmenného protokolu Appletalk, NBP.

Vzorový konfigurační soubor by mohl například obsahovat následující záznam:

```
TricWriter:\
:pr=lp:op=cg:
```

Ten by zpřístupnil tiskárnu jménem „TricWriter“ vaší síti Appletalk a všechny přijaté úlohy by byly vytištěny na linuxové tiskárně `lp` (dle definice v souboru `/etc/printcap`) pomocí *lpd*. Zápis `op=cg` říká, že operátorem tiskárny je uživatel `cg`.

Spuštění softwaru Appletalk

Nyní bychom mohli otestovat základní konfiguraci. S balíkem netatalk je dodáván soubor *rc.atalk*, který by nám měl vyhovovat, takže stačí přidat jeho cestu do některého startovacího souboru

```
# /usr/local/atalk/etc/rc.atalk
```

a vše by mělo fungovat tak, jak má. Neměly by se objevit žádné chybové zprávy a software by měl posílat zprávy o svém stavu na konzolu.

Otestování softwaru Aappletalk

Chcete-li otestovat správnou funkci softwaru, vyvolejte na některém z počítačů menu Apple, zvolte položku Chooser, klepněte na AppleShare a mělo by se objevit okno Linux.

Nevýhody softwaru Appletalk

- Možná bude nutné spustit podporu Appletalk před konfigurací sítě IP. Máte-li problémy se spouštěním programů Appletalk nebo po jejich spuštění nefunguje IP síť, zkuste spustit software Appletalk ještě před spuštěním souboru `/etc/rc.d/rc.inet1`.
- Démon *afpd* (Apple Filing Protocol Daemon) značně zaplní váš pevný disk. Pod přípojnými body vytvoří dvojici adresářů nazvaných `.AppleDesktop` a `Network Trash Folder`. Potom pro každý adresář, do kterého vstoupíte vytvoří adresář `.AppleDouble`, aby do něj mohl uložit zámky a podobně. Takže dříve než exportujete adresář `/`, pořádně si to rozmyslete, jinak budete mít spoustu práce s čištěním disku.
- Program *afpd* očekává od počítačů Apple hesla v čistě textové podobě. To může být vážný bezpečnostní problém, takže pokud budete spouštět tohoto démona na počítači připojeném k Internetu, můžete za případný průnik do systému vinit jen sami sebe.
- Existující diagnostické nástroje, jako je *netstat* a *ifconfig*, nepodporují Appletalk. Základní informace najdete v adresáři `/proc/net/`.

Další informace

Podrobnější popis způsobu konfigurace softwaru Appletalk pro Linux najdete na stránce Linux Netatalk-HOWTO Andrease Brownwothe na adrese *http://www.anders.com/projects/netatalk/*.

ATM

Werner Almesberger (*werner.almesberger@lrc.di.epfl.ch*) vede projekt, který by umožnil podporu režimu ATM (Asynchronous Transfer Mode) v Linuxu. Aktuální informace o stavu tohoto projektu získáte na adrese *http://lrcwww.epfl.ch/linux-atm/*.

AX25 (`AF_AX25`)

Názvy zařízení protokolu AX.25 jsou ve verzi 2.0 `sl0`, `sl1` a tak dále, ve verzích 2.1 a 2.2 pak `ax0`, `ax1` a tak dále.

Volby překladu jádra:

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
```

Protokol AX25, Netrom a Rose jsou popisovány v dokumentu AX25-HOWTO (*http://www.linuxdoc.org/HOWTO/AX25-HOWTO.html*). Tyto protokoly používají radioamatéři po celém světě k experimentům s paketovým rádiem.

Velký díl práce při implementaci těchto protokolů odvedl Jonathon Naylor, *jsn@cs.nott.ac.uk*.

DECNet

Na podpoře protokolu DECNet se v současné době pracuje. Předpokládá se, že se objeví ve verzi jádra č. 2.1.*.

FDDI

Názvy zařízení FFDI jsou `fddi0`, `fddi1`, `fddi2` a tak dále. První detekované kartě je přiřazen název `fddi0` a dalším pak postupně názvy v tom pořadí, v jakém jsou detekovány.

Larry Stefani, *lstefani@ultranet.com*, napsal ovladač pro karty Digital Equipment Corporation FDDI EISA a PCI.

Volby překladu jádra

```
Network device support  --->
    [*] FDDI driver support
    [*] Digital DEFEA and DEFPA adapter support
```

Jakmile máte sestavené jádro s podporou ovladače FDDI, je vlastní konfigurace rozhraní FDDI takřka identická s konfigurací ethernetového rozhraní. Stačí zadat příslušný název rozhraní FDDI příkazům *ifconfig* a *route*.

Frame Relay

Názvy zařízení Frame Relay jsou `dlci00`, `dlci01` a tak dále pro DLCI zařízení zapouzdření a `sdla0`, `sdla1` a tak dále pro FRAD.

Frame Relay je nová síťová technologie, která je navržena pro datové komunikace, jež mají „pulsní“ nebo dočasnou povahu. K síti Frame Relay se připojíte pomocí zařízení zvaného Frame Relay Access Device (FRAD). Frame Relay pro Linux podporuje IP přes Frame Relay, dle popisu v dokumentu RFC-1490.

Volby překladu jádra

```
Network device support  --->
    <*> Frame relay DLCI support (EXPERIMENTAL)
    (24)   Max open DLCI
    (8)   Max DLCI per device
    <*>   SDLA (Sangoma S502/S508) support
```

IPX (`AF_IPX`)

Protokol IPX je častěji využíván v lokálních počítačových sítích Novell NetWare. Linux obsahuje podporu tohoto protokolu a lze ho nakonfigurovat tak, aby se choval jako síťový koncový bod nebo router IPX.

Volby kompilace jádra

```
Networking options  --->
    [*] The IPX protocol
    [ ] Full internal IPX network
```

Protokol IPX a NCPFS jsou podrobněji rozebírány v dokumentu IPX-HOWTO (*http://www.linux-doc.org/HOWTO/IPX-HOWTO.html*).

NetRom (AF_NETROM)

Názvy zařízení NetRom jsou `nr0`, `nr1` a tak dále.

Volby překladu jádra

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    [*] Amateur Radio NET/ROM
```

Protokoly AX25, NetRom a Rose jsou popsány v dokumentu AX25-HOWTO (*http://www.linux-doc.org/HOWTO/HOWTO-INDEX/howtos.html*). Tyto protokoly používají amatérští radiooperátoři při experimentech s paketovým rádiem.

Většinu práce na implementaci těchto protokolů odvedl Jonathon Naylor, *jsn@cs.nott.ac.uk*.

Protokol Rose (AF_ROSE)

Názvy zařízení Rose jsou ve verzi jádra 2.1.* `rs0`, `rs1` atd. Protokol Rose je dostupný od verzí 2.1.*.

Volby překladu jádra

```
Networking options  --->
    [*] Amateur Radio AX.25 Level 2
    <*> Amateur Radio X.25 PLP (Rose)
```

Protokoly AX25, NetRom a Rose jsou popsány v dokumentu AX25-HOWTO (*http://www.linux-doc.org/HOWTO/HOWTO-INDEX/howtos.html*). Tyto protokoly používají amatérští radiooperátoři při experimentech s paketovým rádiem.

Většinu práce na implementaci těchto protokolů odvedl Jonathon Naylor, *jsn@cs.nott.ac.uk*.

SAMBA – podpora NetBEUI, NetBios a CIFS

SAMBA je implementací protokolu Session Management Block. Umožňuje systémům firmy Microsoft a jiným připojovat a používat disky a tiskárny.

Projekt SAMBA a jeho konfigurace jsou rozebírány v dokumentu SMB-HOWTO (*http://www.linux-doc.org/HOWTO/SMB-HOWTO.html*).

Podpora protokolu STRIP (Standard Radio IP)

Názvy zařízení STRIP jsou `st0`, `st1` a tak dále.

Volby překladu jádra

```
Network device support  --->
    [*] Network device support
    ....
    [*] Radio network interfaces
    < > STRIP (Metricom starmode radio IP)
```

Protokol STRIP byl navržen speciálně pro rádiové modemy Metricom v rámci výzkumného projektu vedeného na Stanford University a pojmenovaného MosquitoNet Project (*http://mosquito-*

net.stanford.edu/mosquitonet.html). Je to velice zajímavé čtení, a to i pro ty, kdo se o tento projekt přímo nezajímají.

Rádiové modemy Metricom se připojují k sériovému portu, využívají široké spektrum technologií a zvládají přenosové rychlosti 100 kb/s. Informace o rádiových modemech Metricom získáte na adrese *http://www.metricom.com/*.

V současné době nepodporují standardní síťové nástroje ovladač protokolu STRIP, takže si budete muset ze serveru MosquitoNet stáhnout nějaké upravené nástroje. Podrobnosti ohledně softwaru, který potřebujete, najdete na stránce *http://mosquitonet.stanford.edu/software/strip.html*.

Co se konfigurace týče, pomocí upraveného programu *slattach* nastavíte chování linky sériového zařízení *tty* na STRIP a potom nakonfigurujete výsledné zařízení `st[0-9]` jako pro Ethernet, jen s jednou výjimkou. Z technických důvodů nepodporuje protokol STRIP protokol ARP, takže je třeba ručně upravit záznamy ARP pro každého hostitele ve vaší podsíti. To by nemělo být obtížné.

Token Ring

Názvy zařízení token ring jsou `tr0`, `tr1` a tak dále. Token Ring je standardní LAN protokol vytvořený firmou IBM, který zabraňuje kolizím pomocí mechanismu dávajícího v určitou dobu pouze jedné stanici v síti LAN právo vysílat. „Token" vlastní vždy pouze jedna stanice a pouze ona může přenášet data. Po skončení přenosu předá token další stanici. Token putuje dokola po všech aktivních stanicích, odtud tedy název „Token Ring".

Volby překladu jádra

```
Network device support  --->
    [*] Network device support
    ....
    [*] Token Ring driver support
    < > IBM Tropic chipset based adaptor support
```

Konfigurace protokolu Token Ring je kromě názvu síťového zařízení identická s konfigurací Ethernetu.

X.25

X.25 je kruhový protokol založený na přepínání paketů definovaný organizací C.C.I.T.T. (Sdružení, které navrhuje standardy, jež používají telekomunikační společnosti téměř na celém světě). Na implementaci protokolů AX.25 a LAPB se pracuje a poslední jádra verzí 2.1.* je již obsahují.

Vývoj vede Jonathon Naylor (*jsn@cs.nott.ac.uk*) a za účelem řešení problémů kolem protokolu AX.25 byla zřízena diskusní skupina. Chcete-li se do ní přihlásit, pošlete zprávu na adresu `majordomo@vger.rutgers.edu` a do jejího těla napište text `subscribe linux-x25`.

Dřívější verze konfiguračních nástrojů získáte na Jonathonově anonymním FTP na adrese *ftp://ftp.cs.nott.ac.uk*.

Karta WaveLan

Názvy zařízení WaveLan jsou `eth0`, `eth1` a tak dále.

Volby překladu jádra

```
Network device support  --->
    [*] Network device support
    ....
```

```
[*] Radio network interfaces
....
<*> WaveLAN support
```

Karta WaveLan je bezdrátová síťová karta. Při používání je velmi podobná ethernetové kartě a stejným způsobem se i konfiguruje.

Informace o kartě WaveLan získáte na adrese *http://www.wavelan.com/*.

Kabely a kabeláž

Ti z vás, kteří mají zkušenosti s pájením, si možná budou chtít vyrobit vlastní kabely, kterými by propojili své linuxové počítače. Následující diagramy kabelů by vám měly pomoci.

Sériový null-modemový kabel

Ne všechny null-modemové kabely jsou stejné. Většina těchto kabelů umí jen simulovat přítomnost všech příslušných signálů a přehodit vstupní a výstupní data. To je sice pěkné, ale znamená to, že musíte používat softwarovou kontrolu toku dat (XON/XOFF), která je méně efektivní než hardwarová. Následující kabel poskytuje nejlepší možnou signalizaci mezi počítači a umožňuje používat hardwarovou (RTS/CTS) kontrolu toku dat.

```
Název Pinu   Číslo pinu                          Číslo pinu
Tx Data           2 ---------------------------------  3
Rx Data           3 ---------------------------------  2
RTS               4 ---------------------------------  5
CTS               5 ---------------------------------  4
zem               7 ---------------------------------  7
DTR              20 -\-------------------------------  8
DSR               6 -/
RLSD/DCD          8 ------------------------------/- 20
                                                  \-  6
```

Kabel pro paralelní port (kabel PLIP)

Pokud hodláte ke komunikaci dvou počítačů používat protokol PLIP, můžete použít tento kabel bez ohledu na typ instalovaného paralelního portu.

```
Název pinu   Číslo pinu                     Číslo pinu
STROBE            1*
D0->ERROR         2 --------------------------- 15
D1->SLCT          3 --------------------------- 13
D2->PAPOUT        4 --------------------------- 12
D3->ACK           5 --------------------------- 10
D4->BUSY          6 --------------------------- 11
D5                7*
D6                8*
D7                9*
ACK->D3          10 ---------------------------  5
BUSY->D4         11 ---------------------------  6
PAPOUT->D2       12 ---------------------------  4
SLCT->D1         13 ---------------------------  3
FEED             14*
```

```
ERROR->DO       15 --------------------------  2
INIT            16*
SLCTIN          17*
zem             25 -------------------------- 25
```

Poznámky:

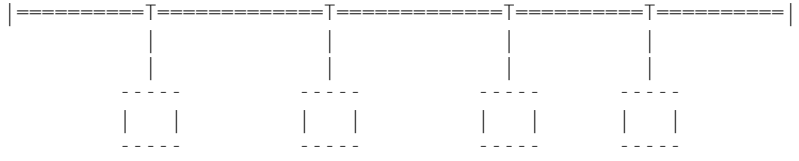
- Nespojujte piny označené hvězdičkou.
- Zem je i na pinech 18 až 24.
- Má-li vámi použitý kabel kovové stínění, měl by být opatřen kovovou zástrčkou DB-25 jen na *jednom* konci.

Upozornění: Špatně zapojený kabel PLIP může zničit kartu řadiče. Při zapojování buďte proto velice opatrní a dvakrát zkontrolujte každé spojení, abyste se vyhnuli problémům.

I když můžete kabel PLIP používat na velké vzdálenosti, měli byste se tomu, je-li to možné, vyhnout. Specifikace tohoto kabelu umožňuje délku kolem 1 metru. Při používání dlouhých kabelů tohoto typu dávejte proto pozor na zdroje elektromagnetických polí, jako je blesk, vodiče elektrického proudu a rádiové vysílače, které je mohou rušit a někdy též zničit váš řadič. Pokud opravdu potřebujete spojit dva počítače na velkou vzdálenost, měli byste si opatřit dvojici ethernetových karet a použít raději koaxiální kabel.

Ethernetový kabel 10base2 (tenký koaxiál)

10base2 je standard ethernetového kabelu, který specifikuje použití 50ohmového koaxiálního kabelu o průměru kolem 5 milimetrů. Při spojování počítačů kabelem 10base2 je třeba dodržet dvě důležitá pravidla. Zaprvé musíte použít terminátory na obou koncích kabelu. Terminátor je rezistor 50 ohmů, který zajišťuje, aby byl signál po dosažení konce absorbován a nikoliv odražen. Bez terminátoru na každém konci kabelu může být Ethernet nespolehlivý nebo nemusí vůbec fungovat. Normálně se používají k propojení počítačů konektory ve tvaru T, takže získáte následující zapojení:



Délka kabelu mezi T konektorem a vlastní ethernetovou kartou by měla být co nejmenší, v ideálním případě je T konektor zapojen přímo do ethernetové karty.

Ethernetový kabel twisted pair

Pokud máte pouze dvě ethernetové karty twisted pair a chcete je propojit, nepotřebujete rozbočovač. Tyto karty můžete propojit přímo. Schéma zapojení najdete v dokumentu Ethernet-HOWTO (*http://www.linuxdoc.org/HOWTO/Ethernet-HOWTO.html*).

Některé termíny používané v tomto dokumentu

ARP	Zkratka názvu Address Resolution Protocol; jedná se o způsob, jakým síťový počítač sdružuje IP adresy se síťovými adresami.
ATM	Zkratka názvu Asynchronous Transfer Mode. Síť ATM balí data do bloků standardních velikostí, které může efektivně dopravovat z jednoho místa na druhé. ATM je síťová technologie založená na kruhovém přepínání paketů.
datagram	Datagram je vyhrazený balík dat a hlaviček, který obsahuje adresy. Jedná se o základní přenosovou jednotku v síti IP. Někdy se také datagramům říká pakety.
DLCI	DLCI znamená Data Link Connection Identifier a slouží k identifikaci jedinečného virtuálního spojení typu point to point v síti Frame Relay. Identifikátor DLCI normálně přiděluje poskytovatel sítě Frame Relay.
Frame Relay	Frame Relay je síťová technologie, která je ideální pro občasný nebo ojedinělý přenos. Sdílí-li stejnou kapacitu sítě mnoho zákazníků, kteří používají síť v jinou dobu, mohou se značně zredukovat síťové náklady.
hardwarová adresa	Jedná se o číslo, které slouží k jedinečné identifikaci hostitele v rámci fyzické sítě na úrovni přístupu k médiu. Příkladem jsou ethernetové adresy a adresy AX.25.
IP adresa	Jedná se o číslo, které slouží k jedinečné identifikaci TCP/IP hostitele v síti. Adresa je 4 bajty dlouhá a zpravidla se zapisuje v takzvané tečkové notaci, kde je každý bajt uveden jako desítkové číslo a ta jsou vzájemně oddělená tečkami.
ISDN	Zkratka Integrated Services Digital Network. ISDN je standard, s jehož pomocí mohou telekomunikační společnosti doručovat hlasové nebo datové informace do domovů svých zákazníků. Z technického hlediska je ISDN kruhově přepínaná datová síť.
ISP	Zkratka Internet Service Provider (poskytovatelé internetových služeb). Jsou to organizace nebo společnosti, které lidem poskytují připojení k Internetu.
klient	Většinou se jedná o software na straně uživatele. Existují výjimky z tohoto tvrzení, například v okenním systému X11 běží na straně uživatele server a na vzdáleném počítači klient. Klient je program, který přijímá nějakou službu poskytovanou serverem. V případě systému peer to peer, jako je SLIP a PPP, navazuje klient spojení a vzdálený konec, ten, který je volán, se nazývá server.
MSS	Maximum Segment Size (maximální velikost segmentu) je největší objem dat, který lze najednou přenést. Chcete-li zabránit lokální fragmentaci, měla by být MSS rovna velikosti MTU minus velikost IP hlavičky.
MTU	Maximum Transmission Unit (maximální přenosová jednotka) je parametr, který určuje největší datagram, který lze přenést přes IP rozhraní, aniž by ho bylo nutné rozdělit na menší jednotky. Hodnota MTU by měla být větší než největší datagram, který chcete přenášet nefragmentován. Uvědomte si, že tak zabráníte pouze místní fragmentaci. Některá jiná linka totiž může mít nastavenu nižší hodnotu MTU a k fragmentaci datagramu pak dojde zde. Typické hodnoty jsou pro Ethernet 1500 bajtů a 576 bajtů pro rozhraní SLIP.

okno	Okno je největší množství dat, které je přijímací strana schopna v určitém okamžiku přijmout.
server	Zpravidla se jedná o software na vzdáleném konci spojení. Server poskytuje určité služby jednomu nebo více klientům. Mezi servery patří FTP, NFS nebo DNS. V případě systémů peer to peer, jakým je například SLIP nebo PPP, je server na straně, která přijímá spojení.
trasa	Trasa je cesta, po které putují datagramy sítí ke svému cíli.

Autorská práva

KAPITOLA 2

Mark Grennan, mark@grennan.com

Linux Intranet Server

Tento dokument obsahuje základní informace o firewallových systémech a poskytuje některé podrobnější informace o instalaci filtrovacích a proxy firewallů na linuxových systémech. HTML verzi tohoto dokumentu můžete najít na adrese *http://www.grennan.com/Firewall-HOWTO.html*.

Úvod

Původní verzi dokumentu Firewall-HOWTO napsal před delším časem David Rudder a rád bych mu poděkoval za to, že mi umožnil jeho práci aktualizovat.

Dále bych chtěl poděkovat Ianu Goughovi za ochotnou pomoc dyslektickému pisateli.

Firewally získaly velkou oblibu jako základní kámen bezpečnosti Internetu. Stejně jako u celé řady jiných populárních témat i kolem nich panuje velké množství nepochopení. Tento dokument uvádí základní informace o tom, co to firewally jsou a jak je konfigurovat.

Při vytváření tohoto dokumentu bylo použito jádro 2.2.13 a distribuce RedHat 6.1, všechny příklady se tedy vztahují k této distribuci. Pokud byste narazili na nějaké odlišnosti oproti své distribuci, kontaktujte mne a já dokument upravím.

Odezva

Jakákoliv odezva je vítána. **Oznamte mi prosím jakékoliv nepřesnosti, které v tomto textu naleznete.** Jsem jenom člověk a přiznávám se, že dělám chyby. Pokud budete chtít něco opravit, kontaktujte mne. Pokusím se odpovědět na všechnu došlou poštu, jsem však dost zaneprázdněn, takže se nezlobte, pokud se mi to nepodaří.

Moje poštovní adresa je *mark@grennan.com*.

Prohlášení

Nezodpovídám za jakékoliv škody, k nimž by došlo při provádění činností popsaných v tomto dokumentu. Tento dokument má sloužit jako základní informace o tom, jak firewally a proxy servery fungují. Nejsem a ani se netvářím jako specialista na bezpečnostní problematiku. Jsem jenom člověk, který toho hodně přečetl a má počítače radši než spousta jiných lidí. Tento dokument jsem napsal pro účely seznámení se s problematikou a nedávám ruku do ohně za to, že je v něm všechno správně.

Autorská práva

chováno toto prohlášení o autorských právech. Komerční distribuce jsou možné a vítané, autor si však přeje být o takovýchto distribucích informován.

Překlady, odvozené práce a souhrnné texty založené na dokumentech HOWTO musí být uvedeny tímto prohlášením o autorských právech. Nemůžete z dokumentu HOWTO vytvořit vlastní materiál a jeho distribuci zatížit dalšími omezeními. Výjimky z tohoto pravidla jsou za určitých okolností možné, v případě potřeby kontaktuje administrátora dokumentů HOWTO.

Stručně řečeno, snažíme se umožnit distribuci těchto materiálů všemi cestami. Přejeme si však zachovat na dokumentech prohlášení o autorských právech a budeme rádi, když nás budete informovat o jakýchkoliv plánech na jejich redistribuci.

Máte-li nějaké dotazy, obraťte se na autora dokumentu (adresa viz výše).

Důvod vzniku tohoto textu

Před několika lety jsem pracoval jako „Správce Internetu" pro stát Oklahoma a byl jsem požádán, abych bez jakéhokoliv rozpočtu zavedl „ve státě Internetu pořádek". (Všimněte si, že v té době nebylo pro podobnou činnost žádné zařazení, byl jsem prostě ten, co to měl udělat.) Nejlepší metoda jak toho dosáhnout, bylo použít co nejvíce zdarma distribuovaných programů a zastaralého technického vybavení. Vše, co jsem měl k dispozici, byl Linux a několik starých 486ek.

Komerční firewally jsou **velice** předražené a dokumentace o tom jak pracují bývá považována za velké tajemství. Zjistil jsem, že vytvořit vlastní firewall bylo prakticky nemožné.

V další fázi jsem byl požádán, abych firewall instaloval. Takže jsem opět prakticky bez rozpočtu začal vytvářet firewall na Linuxu. O šest měsíců později byl firewall hotov a tento dokument byl aktualizován.

Doporučená literatura

- Linux Networking Overview HOWTO
- Ethernet HOWTO
- IPchains Firewalling made Easy!
- NET-3 HOWTO
- NET-PPP HOWTO
- The easiest way to create Virtual Tunnels over TCP/IP networks

Vysvětlení činnosti firewallů

Firewall v původním slova smyslu představuje zařízení, které má zabránit šíření požáru. V domech jsou firewally tvořeny zdmi, které zcela oddělují jednotlivé části budovy. Firewall v autě je kovová přepážka oddělující motor od prostoru pro cestující.

Internetové firewally mají udržet plameny internetového pekla mimo vaši privátní síť. Mohou také zachovat uživatele vaší privátní sítě čisté a neposkvrněné tím, že je odstaví od všeho pokušení, které Internet nabízí.

První počítačový firewall byl nesměrující unixový počítač s připojením na dvě různé sítě. Jedna síťová karta je připojena na Internet, druhá na privátní síť. Pokud se chcete z privátní sítě dostat na Internet, musíte se nejprve přihlásit na firewall – unixový server. K přístupu na Internet pak využíváte prostředků tohoto systému. Můžete například použít X Windows System, na firewallu

spustit prohlížeč Netscape a zobrazit jej na své stanici. Prohlížeč spuštěný na firewallu bude mít přístup k oběma sítím.

Takovýto duální systém (systém se dvěma síťovými připojeními) je vynikající, pokud můžete důvěřovat **všem** jeho uživatelům. Můžete jednoduše vytvořit linuxový systém a na něm účty pro všechny, kdo potřebují přístup k Internetu. Při takovém uspořádání je jediný počítač s přístupem k Internetu na celé privátní síti právě firewall. Nikdo nemůže nic nahrát na vlastní pracovní stanici. Nejprve musí soubor nahrát na firewall a odtud pak na svou stanici.

Důležitá poznámka: 99 % všech průniků do systému začíná získáním přístupu k napadenému systému. Vzhledem k tomu nemůžu tento typ firewallu doporučit. Navíc je také velice omezující.

Politiky firewallu

Nemůžete věřit tomu, že firewall je všechno, co potřebujete. *Nejprve musíte vytvořit jeho politiku.*

Firewally se používají ke dvěma účelům:

1. Udržet lidi (červy, útočníky) venku
2. Udržet lidi (zaměstnance, děti) uvnitř

Když jsem začal firewall instalovat, překvapilo mě, že společnosti pro niž jsem pracoval šlo více o to „špehovat“ své zaměstnance než o to, zabránit útokům proti jejich síti.

Přinejmenším v zemi kde pracuji (v Oklahomě) má zaměstnavatel právo sledovat telefonní hovory a internetové aktivity zaměstnanců. Stačí, když to oznámí.

Nechápejte mě špatně. Souhlasím s tím, že lidi mají v práci pracovat a ne si hrát. A mám pocit, že pracovní morálka docela upadá. Všiml jsem si ale také toho, že management obvykle nejčastěji porušuje pravidla, která sám zavedl. Zažil jsem, jak byl obyčejný zaměstnanec postihován za to, že si na Internetu zjistil, jaké má nejlepší spojení do práce, zatímco manažer stejného podniku trávil hodiny pracovní doby tím, že na Internetu hledal luxusní restaurace a noční kluby, kam by mohl pozvat prominentní zákazníky.

Osobně se domnívám, že nejlepší způsob jak předcházet takovýmto způsobům zneužívání Internetu spočívá v tom, že záznamy firewallu budou zveřejněny na webu tak, aby je mohl vidět kdokoliv.

Bezpečnostní problematika je nebezpečná věc. Pokud spravujete firewall, hlídejte si záda.

Jak vytvořit bezpečnostní politiku

Četl jsem řadu dlouhých dokumentů o tom, jak bezpečnostní politiku vytvořit. Po letech zkušeností mohu říct, nevěřte jim ani slovo. Vytvoření bezpečnostní politiky je nesmírně jednoduché.

1. Definujte, jaké služby chcete nabízet.
2. Definujte, komu tyto služby budete nabízet.
3. Definujte, jaké služby potřebují jaké skupiny uživatelů.
4. Pro každou skupinu služeb definujte, jak má být zajištěna bezpečnost služby.
5. Všechny ostatní metody přístupu prohlašte za nežádoucí.

Postupem času se politika může komplikovat, nicméně pro začátek se nesnažte zacházet do velkých detailů. Snažte se, ať je jednoduchá a jasná.

Typy firewallů

Existují dva typy firewallů:

1. Filtrovací firewally – které blokují určené síťové pakety.
2. Proxy servery (někdy také označované jako firewally) – které přistupují k síti za vás.

Filtrovací firewally

Filtrování paketů je metoda činnosti firewallu, která je vestavěna přímo v jádře Linuxu.

Filtrovací firewall funguje na síťové úrovni. Data mohou přes firewall projít pouze tehdy, pokud to nastavení firewallu povoluje. Veškeré přicházející pakety se filtrují na základě jejich typu, zdrojové adresy, cílové adresy a portu.

Řada směrovačů umožňuje částečně fungovat jako firewall. Filtrovací firewally je naopak možné chápat jako jistý druh směrovače. Vzhledem k tomu musíte pro správné nastavení firewallu přesně rozumět problematice struktury IP paketů.

Protože firewall analyzuje a zaznamenává jenom malý objem dat, filtrovací firewally jsou nenáročné na procesor a vyvolávají jen malé síťové zpoždění.

Filtrovací firewally nezajišťují ochranu hesly. Uživatelé se jim nijak neidentifikují. Jedinou identifikací je IP adresa, přiřazená jejich stanici. To může způsobovat problémy v případě, že používáte protokol DHCP (dynamické přidělování IP adres). Je to dáno tím, že pravidla jsou založena na IP adresách a budete je muset upravovat pokaždé, když dojde ke změně přiřazení IP adres. Nevím, jak tento proces automatizovat.

Filtrovací firewally jsou pro uživatele transparentní. Uživatel nemusí v síťových aplikacích nastavovat žádné speciální volby. U proxy serverů to většinou neplatí.

Proxy servery

Proxy servery se často používají k řízení a sledování odchozího provozu. Některé aplikační proxy servery ukládají přenášená data. Tím se snižuje objem přenášených dat a zrychluje se přístup uživatelům, kteří budou chtít stejná data příště. Představují také neoddiskutovatelný doklad toho, co bylo přeneseno.

Existují dva typy proxy serverů:

1. Aplikační proxy servery – ty pracují za vás
2. SOCKS proxy servery – ty propojují porty

Aplikační proxy servery

Nejlepším příkladem je osoba, která se telnetem připojí na jiný počítač a z tohoto počítače se telnetem připojuje k nějakému vzdálenému počítači. Při použití aplikačního proxy serveru se tento proces automatizuje. Když se chcete telnetem připojit k nějakému vzdálenému počítači, klient telnetu vás nejprve připojí k proxy serveru. Proxy server se pak připojí k tomu počítači, k němuž jste se chtěli připojit, a vrací vám příslušná data.

Protože proxy server zajišťuje veškerou komunikaci, může zaznamenávat vše co dělá(te). U HTTP proxy serverů to například může znamenat zaznamenání každé navštívené adresy. U FTP proxy serverů mohou být zaznamenány všechny přenesené soubory. Tyto proxy servery mohou dokonce z přenášených dokumentů odfiltrovat „neslušná“ slova nebo je mohou kontrolovat na výskyt virů.

Aplikační proxy servery mohou zajišťovat autentikaci uživatelů. Než server provede připojení ke vzdálenému počítači, může uživatele nejprve požádat o přihlášení. Při prohlížení webových stránek to může vypadat tak, že každá navštívená stránka je chráněna heslem.

SOCKS proxy

SOCKS server funguje jako stará telefonní ústředna. Jednoduše propojuje vaše připojení k SOCKS serveru s propojením na vnější počítač.

Většina SOCKS serverů funguje pouze pro TCP spojení. Stejně jako filtrovací firewally nezajišťují autentikaci uživatelů. Mohou však zaznamenávat, kam se uživatelé připojují.

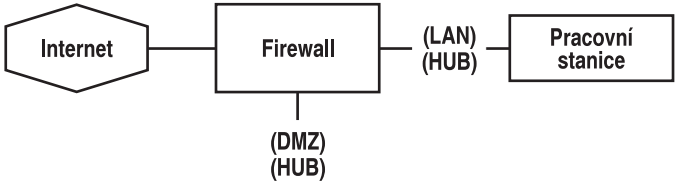
Architektura firewallu

Existuje řada možností jak strukturovat síť při zavedení ochrany firewallem.

Pokud máte pevné připojení k Internetu prostřednictvím směrovače, můžete směrovač přímo připojit k firewallu. Můžete také použít rozbočovač a umožnit tak plný přístup k vašim serverům, které jsou umístěny vně firewallu.

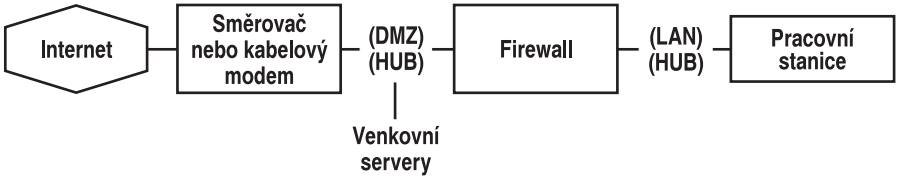
Vytáčené připojení

Můžete být připojeni nějakým vytáčeným připojením jako je ISDN. V takovém případě můžete pomocí třetí síťové karty zajistit filtrované DMZ[1]. Tím umožníte plné internetové služby a přitom je máte odděleny od interní sítě.



Architektura s jedním směrovačem

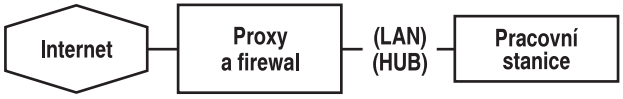
V případě, že je mezi vámi a Internetem směrovač nebo kabelový modem a pokud směrovač spravujete sami, můžete na něm nastavit přísná filtrační pravidla. Pokud směrovač spravuje poskytovatel internetového připojení, nemusíte mít k nastavení pravidel dostatečná oprávnění. Můžete však požádat poskytovatele, ať vám potřebnou filtraci zavede.



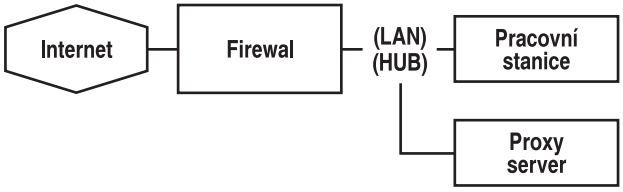
Firewall s proxy serverem

Pokud potřebujete sledovat, co uživatelé vaši sítě navštěvují a síť je poměrně malá, můžete do firewallu integrovat proxy server. Dělají to někteří poskytovatelé internetového připojení a vytvářejí si tak přehledy o zájmech svých klientů, které pak prodávají marketingovým agenturám.

1 Pozn. překladatele: DMZ znamená *demilitarizovanou zónu*. Tímto termínem se označuje část sítě, která je částečně chráněna, není však chráněna úplně. Typicky máte u velké sítě první („hodný“) firewall, který odděluje Internet od demilitarizované zóny. V té se nacházejí veřejně přístupné servery, například webový server a DNS server. Pak následuje druhý („zlý“) firewall a za ním už je vaše kompletně zabezpečená interní síť.

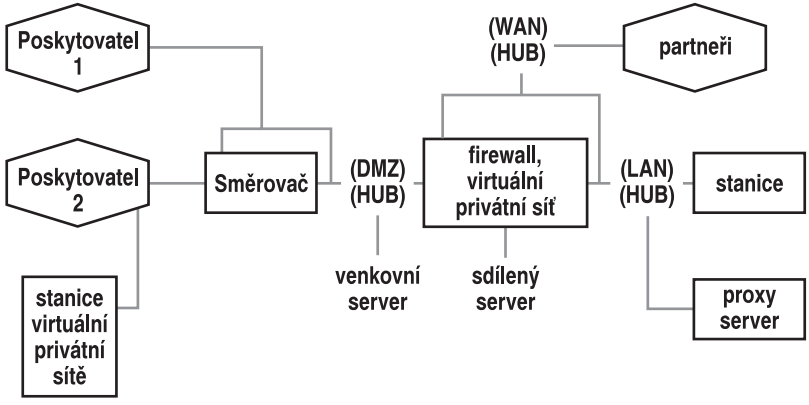


Pokud chcete, můžete proxy server umístit i na lokální síti. V takovém případě je nutné na firewallu nastavit taková pravidla, aby umožnil připojení ke službám, které proxy server poskytuje, právě jenom proxy serveru. Pak se uživatelé mohou dostat na Internet pouze prostřednictvím proxy serveru.



Redundantní internetové připojení

Pokud provozujete službu jako je Yahoo nebo SlashDot, pravděpodobně budete chtít zvýšit spolehlivost vašeho systému použitím redundantních směrovačů a firewallů. (Viz dokument High Availability HOWTO.)



Použitím round-robin DNS kolotoče můžete zajistit přístup k více serverům z jedné URL adresy a budete-li mít více poskytovatelů připojení, směrovačů a firewallů s využitím technologie vysoké dostupnosti, můžete vytvořit službu, která bude téměř 100% spolehlivá.

Konfigurace sítě vám může snadno přerůst přes hlavu. Nezapomínejte prověřovat jakákoliv připojení. K narušení celé privátní sítě stačí uživatel s modemem.

Vytvoření filtrujícího firewallu

Hardwarové požadavky

Filtrující firewall nemá vysoké hardwarové nároky. Nejde téměř o nic víc než o jednoduchý směrovač.

Vše co potřebujete je:

- 486 DX66 s 32 MB paměti
- pevný disk 250 MB (lépe 500 MB)
- síťové připojení (síťovou kartu, sériový port, bezdrátovou přípojku, ...)
- monitor a klávesnice

U některých systémů s využitím konzoly na sériovém portu se můžete obejít i bez monitoru a klávesnice.

Pokud vytváříte proxy server s velkým provozem, použijte co nejlepší vybavení. Jde o to, že pro každého uživatele připojeného k systému se vytváří nový proces. Pokud bude systém současně používat 50 nebo více uživatelů, doporučuji

- Pentium II s 64 MB paměti
- dva gigové disky k ukládání přístupových logů
- dvě síťová připojení
- monitor a klávesnici

Síťová připojení mohou být libovolného typu (síťové karty, ISDN i modemy).

Požadavky na software

Volba jádra

Chcete-li vytvořit filtrující firewall, nepotřebujete žádný speciální software. Linux vše zařídí. V době vzniku tohoto textu používám RedHat 6.1.

Firewall vestavěný v Linuxu se několikrát měnil. Pokud používáte staré jádro Linuxu (verzi 1.0.x nebo starší), přejděte na nové. Tato stará jádra používala software ipfwadm z *http://www.xos.nl/linux/ipfwadm/*, který již není podporován.

Používáte-li jádro 2.2.13 nebo novější, budete používat software ipchaims z adresy *http://www.rustcorp.com/linux/ipchains/*.

Používáte-li nové jádro 2.4, obsahuje nový firewallový nástroj s lepšími funkcemi. Budu o něm brzy hovořit.

Volba proxy serveru

Pokud chcete provozovat proxy server, musíte si nainstalovat některý z následujících balíků:

1. Squid
2. The TIS Firewall Toolkit (FWTK)
3. SOCKS

Squid je vynikající software a využívá linuxovou funkci Transparent Proxy. Popíšeme si instalaci právě tohoto programu.

V době vzniku tohoto textu došlo ke sloučení společností Network Associates a Trusted Information System (TIS). Nové informace o změnách naleznete na jejich webových stránkách. V současné době je možné FWTK stále získat na adrese *http://www.tis.com/research/software/*.

Společnost Trusted Information System nabízí kolekci programů k vybudování firewallu. Pomocí těchto programů můžete vytvořit samostatné démony pro jednotlivé používané služby (WWW, Telnet a podobně).

Příprava linuxového systému

Nainstalujte co nejméně softwaru. Já jsem instalaci zahájil v konfiguraci serveru a pak jsem v souboru /etc/inetd.conf vypnul všechny nepotřebné služby. Pro vyšší bezpečnost byste měli nepotřebné služby odinstalovat.

Protože většina distribucí neobsahuje jádro vhodné pro naše účely, budete si muset přeložit vlastní jádro. Nejlepší je překládat je na jiném počítači než na firewallu. Pokud na firewall nainstalujete překladač C a další utility, po dokončení konfigurace jádra je odstraňte.

Překlad jádra

Začněte s čistou minimální distribucí Linuxu. Čím méně programů je nahraných, tím méně děr, zadních vrátek a/nebo chyb může bezpečnost vašeho systému ohrozit.

Zvolte si stabilní jádro. Já používám jádro 2.2.13. Další dokumentace je založena na tomto jádře.

Budete muset jádro Linuxu přeložit s potřebnými parametry. Pokud jste ještě nikdy jádro nevytvářeli, přečtěte si dokumenty Kernel HOWTO, Ethernet HOWTO a NET-2 HOWTO.

Následuje seznam síťových nastavení o nichž vím, že fungují. Některá z nich jsou označena otazníkem. Pokud je budete používat, zapněte je také.

K editaci nastavení jádra používám `make menuconfig`.

```
<*> Packet socket
[ ] Kernel/User netlink socket
[*] Network firewalls
[ ] Socket Filtering
<*> Unix domain sockets
[*] TCP/IP networking
[ ] IP: multicasting
[*] IP: advanced router
[ ] IP: kernel level autoconfiguration
[*] IP: firewalling
[?] IP: always defragment (required for masquerading)
[?] IP: transparent proxy support
[?] IP: masquerading
--- Protocol-specific masquerading support will be built as modules.
[?] IP: ICMP masquerading
--- Protocol-specific masquerading support will be built as modules.
[ ] IP: masquerading special modules support
[*] IP: optimize as router not host
< > IP: tunneling
< > IP: GRE tunnels over IP
[?] IP: aliasing support
[*] IP: TCP syncookie support (not enabled per default)
--- (it is safe to leave these untouched)
< > IP: Reverse ARP
[*] IP: Allow large windows (not recommended if <16Mb of memory)
< > The IPv6 protocol (EXPERIMENTAL)
```

```
---
< > The IPX protocol
< > Appletalk DDP
< > CCITT X.25 Packet Layer (EXPERIMENTAL)
< > LAPB Data Link Driver (EXPERIMENTAL)
[ ] Bridging (EXPERIMENTAL)
[ ] 802.2 LLC (EXPERIMENTAL)
< > Acorn Econet/AUN protocols (EXPERIMENTAL)
< > WAN router
[ ] Fast switching (read help!)
[ ] Forwarding between high speed interfaces
[ ] PU is too slow to handle full bandwidth
QoS and/or fair queueing --->
```

Po zavedení všech potřebných nastavení byste měli jádro přeložit, znovu nainstalovat a restartovat systém.

K provedení všech těchto operací najednou používám příkaz

```
make dep;make clean;make bzlilo;make modules;make modules_install;init 6
```

Nastavení dvou síťových karet

Pokud máte v počítači dvě síťové karty, budete možná muset do souboru /etc/lilo.conf přidat příkaz append, kterým specifikujete přerušení a porty obou karet. Já používám následující příkaz append:

```
append="ether=12,0x300,eth0 ether=15,0x340,eth1"
```

Nastavení síťových adres

Nyní se dostáváme k zábavné části konfigurace. Nebudu se zabývat podrobnostmi o vytvoření lokální sítě. K tomu si přečtěte dokument Networking HOWTO.

Naším cílem je na filtrující firewall zavést dvě síťová připojení. Jedno vede na Internet (nezabezpečená strana), druhé na lokální síť (zabezpečená strana).

Musíte rozhodnout následující body:

1. Budete v lokální síti používat skutečné IP adresy, nebo si zavedete vlastní?
2. Poskytuje vám IP adresy poskytovatel, nebo používáte statické?

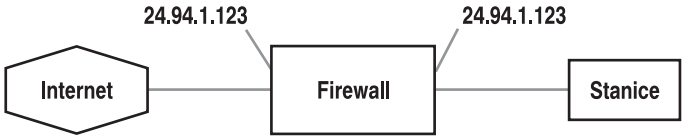
Protože nechcete aby byl z Internetu přístup na vaši privátní síť, nemusíte v ní používat „opravdové“ IP adresy. Můžete si v ní přidělit jakékoliv adresy chcete. Toto řešení se ovšem nedoporučuje. Pokud by totiž data „utekla“ mimo vaši síť, byla by směrována na nějaký úplně jiný systém.

Určité rozsahy IP adres jsou určeny pro přiřazení v privátních sítích. Jednou z rezervovaných oblastí je 192.168.1.*xxx*, kterou budeme používat v našich příkladech.

Aby celý systém fungoval, musíte zavést IP maškarádu. Tento proces zajistí, že firewall bude pakety předávat a překládat jejich adresy na „opravdové“ IP adresy, které mohou po Internetu cestovat.

Použitím rezervovaných nesměrovatelných adres zvyšujete bezpečnost vaší sítě. Internetové směrovače totiž pakety s těmito adresami nepropouštějí.

V tomto okamžiku byste si měli přečíst dokument IP Masquerading HOWTO.



Síťová karta na straně Internetu musí mít přiřazenu „opravdovou“ IP adresu. Tuto adresu můžete mít trvale přidělenu (statická adresa), nebo ji můžete dostávat přidělovánu vždy při připojení k síti protokolem PPP.

Síťové kartě na straně lokální sítě přiřadíte nějakou interní adresu, například 192.168.1.1. Tuto adresu budete používat jako adresu brány. Všem ostatním počítačům na chráněné lokální síti přiřadíte další adresy z oblasti 192.168.1.*xxx* (tedy 192.168.1.2 až 192.168.1.254).

Já používám RedHat Linux. Pro nastavení sítě v době startu systému jsem v adresáři /etc/sysconfig/network-scripts přidal soubor ifcfg-eth1. V tomto adresáři můžete najít také soubory ifcfg-ppp0 nebo ifcfg-tr0. Jsou pojmenovány podle používaného typu připojení.

Jako příklad uvádím soubor ifcfg-eth1 konfigurující druhou ethernetovou kartu:

```
DEVICE=eth1
IPADDR=192.168.1.1
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
GATEWAY=24.94.1.123
ONBOOT=yes
```

Pokud používáte vytáčené připojení, hledejte soubory ifcfg-ppp0 a chat-ppp0. Ty nastavují připojení protokolem PPP.

Tento soubor může vypadat například takto:

```
DEVICE="ppp0"
ONBOOT="yes"
USERCTL="no"
MODEMPORT="/dev/modem"
LINESPEED="115200"
PERSIST="yes"
DEFABORT="yes"
DEBUG="yes"
INITSTRING="ATZ"
DEFROUTE="yes"
HARDFLOWCTL="yes"
ESCAPECHARS="no"
PPPOPTIONS=""
PAPNAME="LoginID"
REMIP=""
NETMASK=""
IPADDR=""
MRU=""
MTU=""
DISCONNECTTIMEOUT=""
RETRYTIMEOUT="5"
BOOTPROTO="none"
```

Testování sítě

Začněte příkazy ifconfig a route. Pokud používáte dvě síťové karty, měl by ifconfig vypadat přibližně takto:

```
#ifconfig
lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        UP LOOPBACK RUNNING MTU:3924 Metric:1
        RX packets:1620 errors:0 dropped:0 overruns:0
        TX packets:1620 errors:0 dropped:0 overruns:0
        collisions:0 txqueuelan:0

eth0    Link encap:10Mbps Ethernet HWaddr 00:00:09:85:AC:55
        inet addr:24.94.1.123 Bcast:24.94.1.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:1000 errors:0 dropped:0 overruns:0
        TX packets:1100 errors:0 dropped:0 overruns:0
        collisions:0 txqueuelan:0
        Interrupt:12 Base address:0x310

eth1    Link encap:10Mbps Ethernet HWaddr 00:00:09:80:1E:D7
        inet addr:192.168.1.1 Bcast:192.168.1.255 Mask:255.255.255.0
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:1110 errors:0 dropped:0 overruns:0
        TX packets:1111 errors:0 dropped:0 overruns:0
        collisions:0 txqueuelan:0
        Interrupt:15 Base address:0x350
```

Směrovací tabulka by měla vypadat asi takto.

```
#route -n
Kernel routing table
Destination     Gateway         Genmask         Flags MSS    Window Use Iface
24.94.1.0       *               255.255.255.0   U     1500   0       15 eth0
192.168.1.0     *               255.255.255.0   U     1500   0        0 eth1
127.0.0.0       *               255.0.0.0       U     3584   0        2 lo
default         24.94.1.123     *               UG    1500   0       72 eth0
```

Poznámka: 24.94.1.0 je internetová strana tohoto firewallu, 192.168.1.0 strana privátní sítě.

Nejprve byste měli ověřit, zda lze ze všech stanic na lokální síti pingnout na interní adresu firewallu (v našem případě 192.168.1.1). Pokud ne, přečtěte si dokument NET HOWTO a dejte síť do pořádku.

Pak zkuste z firewallu pingnout na nějaký internetový systém. Já používám k testování adresu *www.internic.net*. Pokud to nefunguje, zkuste server vašeho poskytovatele. Pokud nefunguje ani to, nemáte v pořádku připojení k Internetu. Z firewallu byste měli být schopni připojit se kamkoliv na Internetu. Zkontrolujte nastavení výchozí brány. Pokud používáte vytáčené připojení, zkontrolujte, zda máte správné ID a heslo. Znovu si přečtěte dokument NET HOWTO a zkuste to znovu.

Dále zkuste z vnitřního počítače pingnout na vnější adresu firewallu (24.94.1.123). To by nemělo jít. Pokud to funguje, máte nastavenou maškarádou nebo IP forwarding nebo nějaká filtrační pravidla. Všechno vypněte a zkuste to znovu. Musíte mít ověřeno, že filtrace funguje.

Pro jádra novější než 2.1.102 můžete použít příkaz

```
echo "0" > /proc/sys/net/ipv4/ip_forward
```

Pokud používáte starší jádro (**proč?**), musíte je znovu přeložit s vypnutím funkce IP forwarding. (Lepší je aktualizovat jádro.)

Znovu zkuste ping na vnější adresu firewallu (24.94.1.123). Nemělo by to jít.

Nyní zapněte IP forwarding a/nebo maskování. Z kteréhokoliv systému na lokální síti byste měli dopingnout kamkoliv na Internet.

```
echo "1" > /proc/sys/net/ipv4/ip_forward
```

Poznámka: Pokud na lokální síti používáte „opravdové“ IP adresy (ne adresy ze skupiny 192.168.1.* apod.) a nemůžete se dopingnout na Internet, ale *můžete* se dopingnout na vnější stranu firewallu, ověřte si, zda váš poskytovatel správně směruje pakety pro vaše privátní adresy.

Tento test můžete provést s využitím někoho jiného na Internetu (například kamaráda s jiným poskytovatelem), který vyzkouší traceroute na vaši síť. Pokud se trasování zastaví na směrovači poskytovatele, pak nezajišťuje směrování vašeho provozu.

Funguje to? Výborně. Ta složitá část je hotova.

Zabezpečení firewallu

Firewall není k ničemu, pokud běží na napadnutelném systému. „Zlí hoši“ mohou získat přístup nějakou nefirewallovou službou a změní konfiguraci firewallu podle svých potřeb. Musíte vypnout všechny nepotřebné služby.

Podívejte se do souboru /etc/inetd.conf. Tento soubor slouží ke konfiguraci superserveru inetd. Řídí celou řadu serverových démonů, které spouští po přijetí požadavku na různé známé porty.

Pokud je máte zapnuty, měli byste vypnout služby echo, discard, daytime, chargen, ftp, gopher, shell, login, exec, talk, ntalk, pop-2, pop-3, netstat, systat, tftp, bootp, finger, cfinger, time, swat a linuxconfig.

Službu vypnete tak, že na první pozici příslušného řádku přidáte znak #. Až změníte vše potřebné, pošlete procesu inetd signál HUP zadáním příkazu `kill -HUP <pid>`, kde <pid> je číslo procesu inetd. Tím proces znovu načte svůj konfigurační soubor (inetd.conf) a restartuje se, aniž by bylo nutné restartovat celý systém.

Otestujte nové nastavení telnetem na port 15 (netstat) firewallu. Pokud dostanete nějakou odezvu, nepodařilo se vám služby vypnout:

```
telnet localhost 15
```

Dále můžete vytvořit soubor /etc/nologin. Napište do něj nějaký text jako třeba *Běž pryč*. Pokud takový soubor existuje, nedovolí proces login přihlášení uživatele. Zobrazí pouze obsah tohoto souboru a odmítne přihlášení. Přihlásit se může jenom root.

Dále můžete upravit soubor /etc/securetty. Pokud se má přihlásit uživatel root, musí se přihlásit na některém z tty uvedených v tomto souboru. Jiné zamítnuté pokusy zaznamenává služba syslog. Zavedete-li obě tyto ochrany, jediná možnost přihlášení k firewallu bude z konzoly jako uživatel root.

Nikdy se službou telnet nikam nepřihlašujte jako root. Pokud potřebujete vzdálený přístup, použijte službu SSH (Secure Shell). Nejlepší bude telnet úplně vypnout.

Pokud jste skutečně paranoidní, použijte lids (Linux Intrusion Detect System). Jedná se o doplněk jádra sloužící k detekci průniků, který dokáže chránit důležité systémové soubory. Pokud pracuje, nemůže nikdo (ani root) změnit chráněné soubory a adresáře (a jejich podadresáře). Chcete-li chráněné soubory změnit, musíte systém restartovat s nastavením LILO security=1 (v tomto případě doporučuji zároveň systém startovat v jednouživatelském režimu).

Nastavení IP filtrování (IPFWADM)

Pokud používáte jádro 2.1.102 nebo novější, přejděte na další část IPCHAINS.

Ve starších jádrech je IP forwarding standardně zapnutý. Proto byste měli začít zakázáním všeho a zrušením všech pravidel, která mohou platit od minula. Následující fragment kódu patří do spouštěcího síťového skriptu (/etc/rc.d/init.d/network)

```
#
# setup IP packet Accounting and Forwarding
#
# Forwarding
#
# By default DENY all services
ipfwadm -F -p deny
# Flush all commands
ipfwadm -F -f
ipfwadm -I -f
ipfwadm -O -f
```

Teď máme dokonalý firewall. Nepropustí vůbec nic.

Nyní vytvoříme soubor /etc/rc.d/rc.firewall. Tento skript povolí poštu, web a DNS provoz.

```
#! /bin/sh
#
# rc.firewall
#
# Source function library.
. /etc/rc.d/init.d/functions
# Get config.
. /etc/sysconfig/network
# Check that networking is up.
if [ ${NETWORKING} = "no" ]
then
exit 0
fi
case "$1" in
start)
echo -n "Starting Firewall Services: "
# Allow email to got to the server
/sbin/ipfwadm -F -a accept -b -P tcp -S 0.0.0.0/0 1024:65535 -D 192.1.2.10 25
# Allow email connections to outside email servers
/sbin/ipfwadm -F -a accept -b -P tcp -S 192.1.2.10 25 -D 0.0.0.0/0 1024:65535
# Allow Web connections to your Web Server
```

```
/sbin/ipfwadm -F -a accept -b -P tcp -S 0.0.0.0/0 1024:65535 -D 192.1.2.11 80
# Allow Web connections to outside Web Server
/sbin/ipfwadm -F -a accept -b -P tcp -S 192.1.2.* 80 -D 0.0.0.0/0 1024:65535
# Allow DNS traffic
/sbin/ipfwadm -F -a accept -b -P udp -S 0.0.0.0/0 53 -D 192.1.2.0/24
;;
stop)
echo -n "Stopping Firewall Services: "
ipfwadm -F -p deny
;;
status)
echo -n "Now do you show firewall stats?"
;;
restart|reload)
$0 stop
$0 start
;;
*)
echo "Usage: firewall {start|stop|status|restart|reload}"
exit 1
esac
```

Poznámka: V tomto skriptu povolujeme poštovní (SMTP) server na počítači 192.1.2.10, který musí umět přijímat a odesílat na portu 25. Webový server běží na počítači 192.1.2.11. Kdokoliv na lokální síti má přístup k externím webovým a DNS serverům.

Toto řešení není úplně bezpečné. Port 80 nemusí být používán jako port webového serveru, takže šikovný hacker jej může využít k tomu, že si jeho prostřednictvím vytvoří přes náš firewall virtuální privátní síť. Ochrana spočívá v instalaci webového proxy serveru a na firewallu povolíme průchod pouze tomuto serveru. Uživatelé na lokální síti budou muset k externím webovým serverům přistupovat přes proxy server.

Možná budete chtít provoz přes firewall zaznamenávat. Následující skript bude zaznamenávat všechny pakety. Můžete jej lehce upravit a sledovat pouze pakety z určitého systému.

```
# Flush the current accounting rules
ipfwadm -A -f
# Accounting
/sbin/ipfwadm -A -f
/sbin/ipfwadm -A out -i -S 192.1.2.0/24 -D 0.0.0.0/0
/sbin/ipfwadm -A out -i -S 0.0.0.0/0 -D 192.1.2.0/24
/sbin/ipfwadm -A in -i -S 192.1.2.0/24 -D 0.0.0.0/0
/sbin/ipfwadm -A in -i -S 0.0.0.0/0 -D 192.1.2.0/24
```

Pokud nepotřebujete nic více než filtrující firewall, můžete tímto skončit. Otestujte jej a užijte si ho.

Nastavení IP filtrování (IPCHAINS)

Linux ipchains je přepsaná verze linuxového firewalového kódu IPv4 a kódu ipfwadm, který je sám přepsanou verzí BSD kódu ipfw – aspoň myslím. Je potřebný k administraci paketových IP filtrů v linuxovém jádře 2.1.102 a vyšších.

Původní kód nedokázal pracovat s fragmenty, používal 32bitová počitadla (přinejmenším na platformě Intel), neumožňoval specifikovat jiné protokoly než TCP, UDP a ICMP, nedovoloval atomické provedení větších změn, neumožňoval specifikaci inverzních pravidel, nebyl úplně spolehlivý a obtížně se spravoval (takže byl náchylný na chyby uživatele). Aspoň to tvrdí autoři.

Nehodlám se pouštět do podrobnějšího popisu toho, jak nastavit firewall na bázi IPChains, protože k tomu slouží **vynikající** HOWTO dokument *http://www.rustcorp/linux/ipchains/HOWTO.html*. Skončilo by to tak, že bych ho musel opsat. Proto uvádím jenom základy.

Už podle názvu pracujete s řetězci. Na začátku máte tři vestavěné řetězce – vstupní, výstupní a forwardovací, které nemůžete zrušit. Kromě toho můžete vytvářet vlastní řetězce. Řetězec představuje skupinu pravidel, která můžete doplňovat a rušit.

S celými řetězci můžete provádět následující operace:

1. Vytvořit nový řetězec (-N).
2. Vymazat prázdný řetězec (-X).
3. Změnit politiku vestavěného řetězce (-P).
4. Vypsat pravidla v řetězci (-L).
5. Vymazat pravidla v řetězci (-F).
6. Vynulovat paketová a bajtová počitadla všech pravidel v řetězci (-Z).

S jednotlivými pravidly v řetězci můžete provádět další operace:

1. Připojit k řetězci nové pravidlo (-A).
2. Vložit nové pravidlo na určité místo řetězce (-I).
3. Nahradit pravidlo na určitém místě řetězce (-R).
4. Smazat pravidlo na určitém místě řetězce (-D).
5. Vymazat první vyhovující pravidlo v řetězci (-D).

Kromě toho existuje i několik operací týkajících se maškarádování:

1. Vypsání momentálně maškarádovaných spojení (-M -L)
2. Nastavení timeoutu maškarádovaných (-M -S)

Při změně pravidel firewallu mohou vzniknout problémy s načasováním. Pokud byste nebyli opatrní, mohlo by se stát, že někdy uprostřed provádění změn umožníte průchod nechtěným paketům. Velmi jednoduché řešení vypadá takto:

```
# ipchains -I input 1 -j DENY
# ipchains -I output 1 -j DENY
# ipchains -I forward 1 -j DENY
```

... teď provedete změny ...

```
# ipchains -D input 1
# ipchains -D output 1
# ipchains -D forward 1
#
```

Tím dojde k odmítnutí všech paketů, které přijdou v době provádění změn.

A takto bude vypadat nastavení výše zavedených pravidel v programu IPChains.

```
#!/bin/sh
#
# rc.firewall
#
## Flush everything, start from scratch
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
## Redirect for HTTP Transparent Proxy
#$IPCHAINS -A input -p tcp -s 192.1.2.0/24 -d 0.0.0.0/0 80 -j REDIRECT 8080
## Create your own chain
/sbin/ipchains -N my-chain
# Allow email to got to the server
/sbin/ipchains -A my-chain -s 0.0.0.0/0 smtp -d 192.1.2.10 1024:-j ACCEPT
# Allow email connections to outside email servers
/sbin/ipchains -A my-chain -s 192.1.2.10 -d 0.0.0.0/0 smtp -j ACCEPT
# Allow Web connections to your Web Server
/sbin/ipchains -A my-chain -s 0.0.0.0/0 www -d 192.1.2.11 1024: -j ACCEPT
# Allow Web connections to outside Web Server
/sbin/ipchains -A my-chain -s 192.1.2.0/24 1024: -d 0.0.0.0/0 www -j ACCEPT
# Allow DNS traffic
/sbin/ipchains -A my-chain -p UDP -s 0.0.0.0/0 dns -d 192.1.2.0/24 -j ACCEPT
## If you are using masquerading
# don't masq internal-internal traffic
/sbin/ipchains -A forward -s 192.1.2.0/24 -d 192.1.2.0/24 -j ACCEPT
# don't masq external interface direct
/sbin/ipchains -A forward -s 24.94.1.0/24 -d 0.0.0.0/0 -j ACCEPT
# masquerade all internal IP's going outside
/sbin/ipchains -A forward -s 192.1.2.0/24 -d 0.0.0.0/0 -j MASQ
## Deny everything else
/sbin/ipchains -P my-chain input DENY
```

Tím jsme ovšem neskončili. Zatím nemáte nastavený dokonalý firewall a jsem si jistý, že budete chtít povolit i jiné služby. Opět doporučuji přečíst si dokument IPCHAINS-HOWTO.

Instalace transparentního SQUID proxy serveru

Proxy server SQUID je dostupný na adrese *http://squid.nlanr.net/*.

Program je dostupný pro distribuce RedHat a Debian. Pokud to jde, použijte jednu z nich.

Instalace proxy serveru TIS

Získání programu

Program TIS FWTK je dostupný na adrese *http://ww.tis.com/research/software*.

Neudělejte stejnou chybu, jakou jsem udělal já. Když si soubory z TIS stáhnete, **přečtěte si dokument README**. Program TIS FWTK je schován ve skrytém adresáři jejich serveru.

TIS požaduje, abyste si přečetli licenční smlouvu na adrese *http://www.tis.com/research/software/fwtk_readme.html* a poté poslali mail na adresu *fwtk-request@tislabs.com* s jediným slovem **accepted** v těle zprávy. Předmět zprávy nemusí být vyplněn. Jejich systém vám pak pošle zpátky název skrytého adresáře (platný 12 hodin), odkud si budete moci program stáhnout.

V době vzniku tohoto textu je poslední verzí programu FWTK verze 2.1.

Instalace TIS FWTK

Spusťte `make install`.

Standardní instalační adresář je /usr/local/etc. Můžete jej změnit na nějaký bezpečnější (já jsem to neudělal). Rozhodl jsem se nastavit práva na tento adresář na 'chmod 700'.

Poslední co zbývá je konfigurace firewallu.

Konfigurace TIS FWTK

Teď začíná ta správná legrace. Musíme systém naučit volat nově nainstalované služby a musíme vytvořit tabulky, které je řídí.

Nehodlám zde opisovat manuál k programu TIS FWTK. Ukážu vám nastavení, které mi funguje a upozorním vás na problémy, na které jsem narazil společně s popisem toho, jak jsem je vyřešil.

Řízení firewallu obstarávají tři soubory:

- **/etc/services** – říká systému, na kterých portech služby pracují
- **/etc/inetd.conf** – říká programu inetd, jakou službu má volat, když někdo požaduje určitou službu
- **/usr/local/etc/netperm-table** – říká službám FWTK, kdo má jaké služby povoleny

Při konfiguraci FWTK budete tyto soubory měnit postupně odspodu nahoru. Změna souboru services bez předchozího správného nastavení v souborech inetd.conf a netperm-table může způsobit nedostupnost systému.

Soubor netperm-table

Tento soubor určuje, kdo má přístup ke službám TIS FWTK. Na provoz procházející firewallem musíte pohlížet z obou stran. Uživatelé vně chráněné sítě by se měli před získáním přístupu identifikovat, uživatelé uvnitř sítě mohou mít povolen průchod přímo.

Aby byla možná identifikace uživatelů, používá firewall program nazvaný **authsrv**, který udržuje databázi uživatelských ID a hesel. Autentikační část souboru netperm-table udává, kde je tato databáze uložena a kdo k ní má povolen přístup.

Měl jsem problémy zakázat přístup k této službě. Všimněte si, že na řádku permit-hosts mám nastavenu hodnotu '*', takže přístup má kdokoliv. Správné nastavení (pokud se vám podaří je zprovoznit) by mělo být `authsrv: permit-hosts localhost`.

```
#
# Proxy configuration table
#
# Authentication server and client rules
authsrv: database /usr/local/etc/fw-authdb
authsrv: permit-hosts *
authsrv: badsleep 1200
authsrv: nobogus true
# Client Applications using the Authentication server
*: authserver 127.0.0.1 114
```

K inicializaci databáze musíte být root a spustit program **./authsrv** v adresáři /var/local/etc, čímž se vytvoří záznam o administrátorovi. Takto vypadá příklad vytvoření záznamu.

V dokumentaci k FWTK zjistíte, jak vytvářet uživatele a skupiny.

```
#
# authsrv
authsrv# list
authsrv# adduser admin "Auth DB admin"
ok - user added initially disabled
authsrv# ena admin
enabled
authsrv# proto admin pass
changed
authsrv# pass admin "plugh"
Password changed.
authsrv# superwiz admin
set wizard
authsrv# list
Report for users in database
user group longname ok? proto last
------ ------ ------------------ ----- ------ -----
admin Auth DB admin ena passw never
authsrv# display admin
Report for user admin (Auth DB admin)
Authentication protocol: password
Flags: WIZARD
authsrv# ^D
EOT
#
```

Nastavení telnetové brány (tn-gw) je nejjednodušší a je to první věc, kterou byste měli začít.

V uvedeném příkladu povoluji počítačům z interní sítě přístup ven bez nutnosti autentikace (`permit-hosts 192.1.2.* -passok`). Ostatní uživatelé se však musejí autentikovat (`permit-hosts * -auth`).

Kromě toho jsem jednomu systému (192.1.2.202) povolil přímý přístup k firewallu bez nutnosti procházet přes firewall. Zajišťují to dva řádky inetacl-in.telnetd. Později si uvedeme, jak se těmto řádkům říká.

Timeout pro službu telnet by měl být krátký.

```
# pravidla telnetové brány:
tn-gw: denial-msg /usr/local/etc/tn-deny.txt
tn-gw: welcome-msg /usr/local/etc/tn-welcome.txt
tn-gw: help-msg /usr/local/etc/tn-help.txt
tn-gw: timeout 90
tn-gw: permit-hosts 192.1.2.* -passok -xok
tn-gw: permit-hosts * -auth
# pouze administrátor má na firewall přímý přístup telnetem na portu 24
netacl-in.telnetd: permit-hosts 192.1.2.202 -exec /usr/sbin/in.telnetd
```

r-příkazy fungují stejně jako telnet.

```
# pravidla rlogin brány:
rlogin-gw: denial-msg /usr/local/etc/rlogin-deny.txt
rlogin-gw: welcome-msg /usr/local/etc/rlogin-welcome.txt
```

```
rlogin-gw: help-msg /usr/local/etc/rlogin-help.txt
rlogin-gw: timeout 90
rlogin-gw: permit-hosts 192.1.2.* -passok -xok
rlogin-gw: permit-hosts * -auth -xok
# pouze administrátor má na firewall přímý přístup rloginem
netacl-rlogind: permit-hosts 192.1.2.202 -exec /usr/libexec/rlogind -a
```

Nikomu byste neměli umožnit přímý přístup k firewallu, což platí i pro službu FTP – proto na firewall neinstalujte FTP server.

I zde řádek permit-hosts umožňuje komukoliv z chráněné sítě přímý přístup na Internet, všichni ostatní se musejí autentikovat. Kromě toho je nastaveno zaznamenávání všech přijatých a odeslaných souborů (`-log {retr stor}`).

Nastavení timeoutu říká, za jak dlouho budou uzavřena chybná připojení i to, jak dlouho bude spojení udržováno bez aktivity.

```
# pravidla ftp brány:
ftp-gw: denial-msg /usr/local/etc/ftp-deny.txt
ftp-gw: welcome-msg /usr/local/etc/ftp-welcome.txt
ftp-gw: help-msg /usr/local/etc/ftp-help.txt
ftp-gw: timeout 300
ftp-gw: permit-hosts 192.1.2.* -log { retr stor }
ftp-gw: permit-hosts * -authall -log { retr stor }
```

FTP přenosy zprostředkované přes web, gopher a prohlížeče jsou řízeny nastavením http-gw. První dva řádky nastavují adresář pro ukládání dokumentů, které firewallem přicházejí. Tyto soubory vlastní uživatel root a jsou uloženy v adresáři, kam má přístup pouze root.

Webová připojení by měla být krátká. Délka spojení určuje, jak dlouho bude neplatné spojení trvat.

```
# pravidla www a gopher brány:
http-gw: userid root
http-gw: directory /jail
http-gw: timeout 90
http-gw: default-httpd www.afs.net
http-gw: hosts 192.1.2.* -log { read write ftp }
http-gw: deny-hosts *
```

ssl-gw je jednoduchá brána propouštějící cokoliv. Pozor na ni. V tomto případě povoluji komukoliv z chráněné sítě připojit se na jakýkoliv server vně sítě s výjimkou adres 127.0.0.* a 192.1.1.* a to pouze na portech 443 až 563. Tyto porty jsou používány pro službu SSL.

```
# pravidla ssl brány:
ssl-gw: timeout 300
ssl-gw: hosts 192.1.2.* -dest { !127.0.0.* !192.1.1.* *:443:563 }
ssl-gw: deny-hosts *
```

Další příklad ukazuje, jak pomocí plug-gw povolit přístup na server news. Povolujeme komukoliv z interní sítě připojení pouze na jediný server a na jeho konkrétní port.

Následující řádek povoluje přenos dat zpět na chráněnou síť.

Protože většina klientů počítá s tím, že zůstávají připojeni v době, kdy si uživatel zprávy čte, timeout této služby by měl být nastavený na dlouhou dobu.

```
# brána NetNews
plug-gw: timeout 3600
plug-gw: port nntp 192.1.2.* -plug-to 24.94.1.22 -port nntp
plug-gw: port nntp 24.94.1.22 -plug-to 192.1.2.* -port nntp
```

Brána služby finger je jednoduchá. Kdokoliv uvnitř chráněné sítě se nejprve musí autentikovat a potom mu povolíme použít program finger na firewallu. Kdokoliv jiný jenom obdrží nastavené hlášení.

```
# povolení služby finger
netacl-fingerd: permit-hosts 192.1.2.* -exec /usr/libexec/fingerd
netacl-fingerd: permit-hosts * -exec /bin/cat /usr/local/etc/finger.txt
```

Nenastavoval jsem poštovní služby a službu X-windows, proto neuvádím příklady konfigurace. Pokud jste někdo tyto služby konfigurovali, pošlete mi prosím mail.

Soubor /etc/services

V tomto souboru všechno začíná. Když se klient připojí k firewallu, připojí se na nějaký známý port (menší než 1024). Například služba telnet se připojuje na port 23. Démon inetd toto spojení zaregistruje a v souboru /etc/services zjistí název příslušné služby. Pak volá program, který je této službě přiřazen v souboru /etc/inetd.conf.

Některé služby které vytváříme normálně v souboru /etc/services nejsou. Můžete je přiřadit kterémukoliv portu chcete. Například administrátorskou službu telnet (telnet-a) jsem přiřadil na port 24. Můžete ji přiřadit klidně třeba na port 2323. Když se administrátor (tedy *vy*) bude chtít připojit k firewallu, musí provést telnet na port 24 a ne na 23 a pokud máte soubor netperm-table nastaven tak jako já, budete se moci připojit pouze z jednoho konkrétního počítače uvnitř chráněné sítě.

```
telnet-a  24/tcp
ftp-gw    21/tcp            # this named changed
auth      113/tcp   ident   # User Verification
ssl-gw    443/tcp
```

SOCKS proxy server

Instalace proxy serveru

Proxy server SOCKS můžete získat na adrese *http://www.socks.nec.com/*.

Tento soubor dekomprimujte a rozbalte do nějakého adresáře na vašem systému a držte se pokynů pro jeho sestavení. Já jsem s tím měl různé problémy. Zkontrolujte si, zda máte v pořádku soubory Makefile.

Důležitá věc je to, že tento proxy server je nutné přidat do souboru /etc/inetd.conf. Musíte přidat následující řádek:

```
socks  stream tcp nowait nobody /usr/local/etc/sockd sockd
```

Pak se bude server spouštět, když bude potřeba.

Konfigurace proxy serveru

Program SOCKS potřebuje dva samostatné konfigurační soubory. Jeden říká, jaké přístupy jsou povoleny, druhý říká, kam jednotlivé požadavky směrovat. Přístupový soubor je umístěn na serveru. Směrovací soubor musí být na každé unixové stanici. Dosovské počítače a zřejmě i Macy si zajišťují vlastní směrování.

Přístupový soubor

V programu socks4.2Beta se tento soubor jmenuje „sockd.conf". Měl by obsahovat dva řádky, jeden typu permit, druhý typu deny. Každý řádek obsahuje tři údaje:

- Identifikátor (permit nebo deny)
- IP adresa
- Modifikátor adresy

Identifikátor je buď permit nebo deny. Měli byste mít oba dva typy řádku.

IP adresa obsahuje čtyřbajtovou adresu v klasické IP notaci, například 192.168.1.0.

Modifikátor adresy je opět klasická čtyřbajtová IP adresa. Funguje podobně jako maska sítě. Chápejte uvedené číslo jako 32bitové binární, skládající se z jedniček a nul. Pokud je nějaký bit jedničkový, pak musí odpovídající bit v kontrolované adrese odpovídat hodnotě stejného bitu v IP adrese v řádku. Řekněme, že máme řádek.

```
permit 192.168.1.23 255.255.255.255
```

Tento řádek povolí pouze adresy, v nichž všechny bity odpovídají adrese 192.168.1.23, tedy pouze právě adresu 192.168.1.23. Řádek

```
permit 192.168.1.0 255.255.255.0
```

povolí všechny adresy v rozsahu 192.168.1.0 až 192.168.1.255, tedy celou jednu třídu C adres. Rozhodně byste neměli zadat řádek

```
permit 192.168.1.0 0.0.0.0
```

který povolí jakoukoliv adresu.

Nejprve tedy povolte všechny potřebné adresy a pak zakažte zbytek. Pokud chcete povolit všechny počítače v doméně 192.168.1.*, bude fungovat nastavení

```
permit 192.168.1.0 255.255.255.0
deny 0.0.0.0 0.0.0.0
```

Všimněte si prvního údaje 0.0.0.0 na řádku deny. Máme-li nastaven modifikátor 0.0.0.0, na adrese vůbec nezáleží a pravidlu vyhovuje vše. Obvykle se zadává právě adresa 0.0.0.0, protože ta se snadno zapisuje.

V souboru samozřejmě můžete mít více řádků jednotlivých typů.

Přístup je možné povolovat nebo zakazovat i určitým uživatelům. Dělá se to pomocí autentikace ident. Ne všechny systémy však ident podporují, například Trumpet Winsock, proto o tomto typu autentikace nebudeme mluvit. Potřebné informace získáte v dokumentaci k programu SOCKS.

Směrovací soubor

Směrovací soubor v programu SOCKS se nešťastně jmenuje „socks.conf“. Říkám nešťastně, protože jeho název je natolik blízký přístupovému souboru, že je jednoduché si tyto soubory splést.

Směrovací soubor říká klientům programu SOCKS kdy tuto službu používat a kdy ne. V našem příkladu například nebudeme potřebovat službu SOCKS pro komunikaci s počítačem 192.168.1.1 – firewallem. S tím máme přímé propojení Ethernetem. Samozřejmě SOCKS nepotřebujeme pro přístup na adresu 127.0.0.1, protože sami se sebou se můžete bavit jak chcete. Soubor obsahuje tři typy položek:

- deny
- direct
- sockd

Údaj deny říká, které požadavky se mají zamítnout. Obsahuje stejné údaje jako v souboru sockd.conf, tedy identifikátor, adresu a modifikátor. Protože je řízení přístupu obsluhováno i souborem sockd.conf, nastavuje se modifikátor na 0.0.0.0. Pokud si chcete zakázat přístup kamkoliv, můžete to udělat zde.

Údaj direct říká, pro které adresy se nemá služba SOCKS používat. Zde by měly být uvedeny všechny adresy, které je možné používat bez proxy serveru. I zde jsou tři údaje, identifikátor, adresa a modifikátor. V našem příkladu použijeme

```
direct 192.168.1.0 255.255.255.0
```

Tím povolujeme přímý přístup k čemukoliv na chráněné síti.

Údaj sockd definuje počítač, na kterém běží server SOCKS. Syntaxe je následující:

```
sockd @=<seznamserverů> <IP adresa> <modifikátor>
```

Všimněte si údaje @=. Ten nám umožňuje definovat seznam proxy serverů. V našem příkladu používáme pouze jediný proxy server. Můžete jich však mít více například pro zvýšení přenosové kapacity nebo jako zálohu pro případ výpadku.

IP adresa a modifikátor fungují stejně jako v ostatních příkladech. Nastavujete zde, kam mohou které adresy jít.

DNS za firewallem

Nastavení služby Domain Name Service za firewallem je poměrně jednoduché. Stačí pouze nainstalovat DNS server na firewallu. Pak každému počítači za firewallem řeknete, aby používal tento server.

Práce s proxy serverem

Unix

Aby mohla aplikace pracovat s proxy serverem, musí být „sockifikovaná“. Budete potřebovat dva různé telnetové klienty, jednoho pro přímou komunikaci, druhého pro komunikaci přes proxy server. Program SOCKS se dodává s popisem, jak programy sockifikovat i s některými připravenými běžnými klienty. Pokud použijete sockifikovanou verzi pro přímý přístup, SOCKS vás automaticky přepne na přímou verzi programu. Proto je nutné všechny programy na chráněné síti přejmenovat a nahradit je sockifikovanými verzemi. Z programu „finger“ tak dostaneme „finger.orig“, z „telnet“ bude „tel-

net.orig“ a podobně. O těchto souborech musíte program SOCKS informovat v souboru include/socks.h.

Některé programy zvládají směrování a komunikaci se SOCKS serverem samy. Umí to například Netscape. Chcete-li používat proxy server v programu Netscape, stačí zadat adresu proxy serveru (v našem případě 192.168.1.1) do údaje SOCK v konfiguračním poli Proxies. U každé aplikace bude nutný nějaký typ nastavení bez ohledu na to, jak s proxy serverem funguje.

MS Windows s Trumpet Winsockem

Trumpet Winsock má vestavěnou podporu proxy serverů. V nabídce „setup“ zadáte adresu proxy serveru a adresy všech počítačů, které jsou přístupné přímo. Trumpet bude všechen odcházející provoz správně obsluhovat.

Proxy server s podporou UDP paketů

Program SOCKS pracuje pouze s TCP pakety, nikoliv s UDP pakety. Tím jeho použitelnost poněkud klesá. Protokol UDP používá řada užitečných programů, například Archie. Existuje program, který má fungovat jako proxy server pro UDP pakety, jmenuje se UDPrelay a vytvořil jej Tom Fitzgerald (*fitz@wang.com*). V době vzniku tohoto textu však nebyl k dispozici pro Linux.

Nevýhody proxy serverů

Proxy server je v první řadě *bezpečnostní zařízení*. Pokud jej používáte k zajištění přístupu na Internetu pro počítače, pro něž nemáte řádné IP adresy, přináší to s sebou nevýhody. Proxy server sice umožňuje poměrně rozsáhlý přístup zevnitř sítě ven, neumožňuje však jakkoliv přistupovat zvenku dovnitř. Znamená to, že na interních počítačích nemohou pracovat žádné servery, nelze s nimi navázat žádné přímé poštovní, chatovací nebo archie spojení. Tyto nevýhody vypadají nepodstatně, ale:

- Důležitý dokument, na němž pracujete, jste si nechali na počítači v chráněné síti. Jste doma a chtěli byste na něm dále pracovat. Nemůžete. Nemůžete se ke svému počítači dostat, protože je za firewallem. Mohli byste se sice nejprve přihlásit na firewall, jenomže protože veškerý přístup zevnitř ven zajišťuje proxy server, nemáte na firewallu účet.
- Dcera je na vysoké a chcete jí poslat poštu. Potřebujete jí říct něco důvěrného a nejraději byste jí poštu poslali přímo na její počítač. Ne že byste nevěřili správci vašeho systému, ale přece jenom, jde o důvěrné věci.
- Zásadní nevýhodou proxy serverů je nemožnost použití UDP paketů. Předpokládám však, že podpora protokolu UDP bude brzy realizována.

Další problémy s proxy servery má služba FTP. Když stahujete nějaká data nebo jen vypisujete obsah adresáře na FTP serveru, server otevře na klientském počítači soket a přes něj vám posílá informace. Proxy server to nedovolí, takže FTP nebude fungovat.

Navíc jsou proxy servery pomalé. Vzhledem k velkému množství operací, které musejí provádět, je prakticky jakékoliv řešení rychlejší.

V zásadě platí, že pokud máte dost IP adres a nezáleží vám příliš na bezpečnosti, nepoužívejte ani firewall, ani proxy server. Pokud nemáte dost IP adres, nicméně na bezpečnosti vám pořád příliš nezáleží, můžete použít nějaký IP emulátor, například Term, Slirp nebo TIA. Term získáte na adrese *ftp://sunsite.unc.edu*, Slirp na adrese *ftp://blitzen.canbera.edu.au/pub/slirp*, TIA je k dispozici na *marketplace.com*. Tyto programy jsou rychlé, umožňují lepší spojení a zajišťují větší míru přístupu zvenčí dovnitř. Proxy servery jsou výhodné pro sítě, kde je hodně počítačů, které všechny potřebují častý přístup na Internet. V takovém případě se vyplatí jednorázově nakonfigurovat proxy server.

Složitější konfigurace

Existuje ještě jeden příklad konfigurace, o kterém bych se chtěl zmínit dříve, než tento dokument uzavřu. Zatím popsaná konfigurace bude pravděpodobně většině uživatelů vyhovovat. Následující text však popisuje složitější konfiguraci, na níž si můžeme vyjasnit některá problémová místa. Pokud vás zajímá více než to, o čem jsme doposud hovořili, nebo pokud se prostě jenom zajímáte o možnosti proxy serverů a firewallů, čtěte dále.

Velká síť s vysokým zabezpečením

Řekněme, že vedete teroristickou skupinu a chcete si vytvořit počítačovou síť. Máte 50 počítačů a podsítě 32 (5bitových) IP adres. V rámci sítě potřebujete různé úrovně přístupu, protože různým lidem říkáte různé věci. Proto musí být jednotlivé části sítě chráněny od ostatních.

Úrovně ochrany jsou:

1. Externí úroveň. Tato úroveň je přístupná všem. Zde inzerujete a získáváte nové dobrovolníky.
2. Dobrovolníci. Na této úrovni se pohybují různí nevýznamní dobrovolníci. Vysvětlujete jim, jak jsou vlády nepřátelské a jak vyrábět bomby.
3. Žoldnéři. Tady máte *opravdové* plány. Na této úrovni jsou uloženy informace o převzetí moci nad zbytkem světa, o plánech likvidace nepřátelských vlád, seznamy členů a tak.

Nastavení sítě

IP adresy jsou rozděleny takto:

- jedna adresa 192.168.1.255 je vysílací a tedy nepoužitelná
- 23 z 32 adres je přiřazeno 23 počítačům, které budou přístupné z Internetu
- jedna další IP adresa je přiřazena jednomu linuxovému stroji
- druhá další IP adresa je přiřazena druhému linuxovému stroji
- dvě IP adresy potřebuje směrovač
- čtyři adresy jsou nepoužité, ale jako kamufláž mají přiřazena jména paul, ringo, john a george
- obě chráněné sítě používají adresy 192.168.1.xxx

Dále vytvoříme dvě samostatné sítě, každou v jedné místnosti. Jsou propojeny infračerveným spojem, takže zvnějšku nejsou odposlouchávatelné. Infračervené propojení se naštěstí chová jako klasický Ethernet.

Obě samostatné sítě jsou připojeny každá k jednomu vyhrazenému linuxovému stroji.

K oběma chráněným sítím je připojen souborový server. Musí to tak být, protože v plánech na ovládnutí světa figurují i někteří výše postavení dobrovolníci. Tento server má adresu 192.168.1.17 na síti dobrovolníků a 192.168.1.23 na síti žoldnéřů. IP forwarding je vypnut.

I na obou linuxových strojích je vypnut IP forwarding. Směrovač nebude předávat pakety pro síť 192.168.1.xxx, pokud na to nebude explicitně nastaven, takže obě sítě budou z Internetu dostupné. Důvodem vypnutí forwardingu je to, aby se pakety ze sítě dobrovolníků nemohly dostat na síť žoldnéřů a naopak.

NFS server je možné nastavit tak, aby na různé sítě nabízel různé soubory. To může být velmi užitečné a pomocí pár triků se symbolickými odkazy jde zařídit i to, aby společné soubory byly sdíleny na všech sítích. Pomocí tohoto nastavení a další síťové karty můžeme zajistit, že server bude sloužit všem třem sítím.

Protože na všech třech úrovních potřebujeme získávat různé informace, všechny musejí mít přístup na Internet. Externí síť je na Internet připojena přímo, takže zde se nemusíme s proxy serverem zatěžovat. Sítě dobrovolníků a žoldnéřů jsou za firewallem, a tak pro ně musíme proxy server nastavit.

Obě sítě budou nastaveny velmi podobně. Obě mají přiřazeny stejné IP adresy. Definujme si nyní pár pravidel, ať se nám nastavení trochu zkomplikuje.

- Souborový server nemůže mít přístup na Internet – tím by byl vystaven virům a dalším odpornostem, a protože je pro nás velmi důležitý, tak to raději vůbec nepovolíme.
- Dobrovolníkům nepovolíme přístup k WWW. Procházejí právě výcvikem a možnost získávání různých informací by jim v této fázi mohla uškodit.

Soubor sockd.conf na linuxovém stroji sítě dobrovolníků tedy bude obsahovat následující řádek:

```
deny 192.168.1.17 255.255.255.255
```

A v síti žoldnéřů

```
deny 192.168.1.123 255.255.255.255
```

Kromě toho na stroji dobrovolníků nastavíme

```
deny 0.0.0.0 0.0.0.0 eq 80
```

Tím říkáme, že zakazujeme přístup všem počítačům na port rovný (equal) 80, tedy na port HTTP. Tím budou možné všechny ostatní služby, WWW přístup však bude zakázán.

Kromě toho v obou souborech nastavíme

```
permit 192.168.1.0 255.255.255.0
```

čímž všem počítačům na síti 192.168.1.xxx povolíme použití proxy serveru (kromě těch, které už ho mají zakázán – tedy kromě souborového serveru a kromě HTTP přístupu ze sítě dobrovolníků).

Soubor sockd.conf na síti dobrovolníků bude nastaven takto:

```
deny 192.168.1.17 255.255.255.255
deny 0.0.0.0 0.0.0.0 eq 80
permit 192.168.1.0 255.255.255.0
```

a na síti žoldnéřů takto:

```
deny 192.168.1.23 255.255.255.255
permit 192.168.1.0 255.255.255.0
```

Tím by mělo být všechno správně nastaveno. Jednotlivé sítě jsou izolovány s nastavením potřebné míry propojení. Všichni budou spokojení.

Usnadnění správy

Nástroje firewallů

Existuje několik softwarových balíků, které usnadňují správu firewallu.

Buďte opatrní a nepoužívejte je, pokud s nimi neumíte pracovat. Tak jak vám mohou usnadnit práci, stejně jednoduše vám dovolí i udělat chybu.

Pro nastavení filtračních pravidel existují grafická i webová rozhraní. Některé společnosti dokonce dodávají vlastní komerční firewally založené na Linuxu, nainstalovaném ve speciálním počítači se speciálním obslužným programem. (Milé.)

Nemám příliš rád grafická rozhraní, nicméně jsem nějakou dobu používal firewally s grafickým rozhraním. Takové rozhraní je užitečné, protože nabídne přehledný obrázek nastavení všech filtračních pravidel.

gfcc (GTK+ Firewall Control Center) je aplikace umožňující nastavovat firewallové politiky a pravidla, založená na balíku ipchains. Můžete ji získat na adrese *http://icarus.autostock.co.kr*. Jedná se o velmi pěkný nástroj.

V příloze A uvádím konfigurační RC skripty. Ty fungují s i bez gfcc.

Existuje řada skriptů pro nastavení firewallu. Jeden velmi dokonalý můžete najít na adrese *http://www.jasmine.org.uk/~simon/bookshelf/papers/instant-firewall/instant-firewall.html*. Další dobrý skript je na adrese *http://www.pointman.org/*.

Kfirewall je grafické rozhraní k balíkům ipchains nebo ipfwadm (v závislosti na verzi jádra). Najdete je na adrese *http://megaman.ypsilonia.net/kfirewall/*.

FCT je HTML nástroj pro konfiguraci firewallu. Umožňuje automatické generování skriptů obsahujících filtrovací příkazy (ipfwadm) pro firewall s více rozhraními a libovolnými službami. Nachází se na adrese *http://www.fen.baynet.de/~ft114/FCT/firewall.htm*.

Obecné nástroje

WebMin je nástroj pro obecnou správu systému. Neumožňuje sice nastavovat pravidla firewallů, umí však zapínat a vypínat démony a procesy. Jde o velmi dobrý program a věřím, že jeho autor J. Cameron jej doplní i o modul administrace IPCHAINS. Nachází se na adrese *http://www.webmin.com/*.

Pokud poskytujete přístup k Internetu, bude vás zajímat program IPFA (IP Firewall Accounting) na adrese *http://www.soaring-bird.com/ipfa/*. Umožňuje vytváření měsíčních, denních nebo minutových záznamů a obsahuje webové grafické rozhraní pro správu.

httptunel vytváří obousměrnou virtuální datovou cestu tunelem přes HTTP požadavky. HTTP požadavky je pak možné odesílat prostřednictvím firewallu. Umožňuje tak vytvoření virtuální privátní sítě. Adresa programu je *http://sunsite.auc.dk/vpnd/*.

Příklady skriptů

RC skript používající GFCC

```
#!/bin/bash
#
# Firewall Script - Version 0.9.1
#
# chkconfig: 2345 09 99
# popis: firewallový skript pro jádro 2.2.x
# Nastavit pro testování:
# set -x
#
# POZNÁMKY:
#
# Tento skript je napsán pro RedHat 6.1 a vyšší.
```

```
#
# Opatrně při poskytování veřejných služeb jako web nebo ftp.
#
# INSTALACE:
# 1. nahrejte tento soubor do /etc/rc.d/init.d (budete muset být root..)
# pojmenujte jej tak nějak jako "firewall" :-)
# ať jej vlastní root --> "chown root.root (název)"
# ať je spustitelný --> "chmod 755 (název)"
#
# 2. použijte GFCC k vytvoření pravidel firewallu a exportujte je do
# souboru /etc/gfcc/rules/firewall.rule.sh.
#
# 3. přidejte firewall do inicializačních struktur RH --> "chkconfig --add (ná-
zev)"
# při příštím startu systému by to mělo naběhnout automaticky!
# spěte sladce, protože máte systém O NĚCO MÉNĚ zranitelný...
#
# Poznámky k verzím:
# 30 Jan, 2000 - Změněno na GFCC skript
# 11 Dec, 1999 - aktualizoval Mark Grennan <mark@grennan.com>
# 20 July, 1999 - původní verze od Anthony Ball <tony@LinuxSIG.org>
#
######################################################
# Knihovna zdrojových funkcí.
. /etc/rc.d/init.d/functions
# Konfigurace zdrojové sítě.
. /etc/sysconfig/network
# Kontrola zda je síť nastavena.
[ ${NETWORKING} = "no" ] && exit 0
# Podíváme se, jak nás volali
case "$1" in
start)
# Začínáme poskytovat přístup
action "Spouštím firewall: " /bin/true
/etc/gfcc/rules/firewall.rule.sh
echo
;;
stop)
action "Zastavuji firewall: " /bin/true
echo 0 > /proc/sys/net/ipv4/ip_forward
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
echo
;;
restart)
action "Restartuji firewall: " /bin/true
$0 stop
$0 start
echo
;;
status)
# Výpis nastavení
```

```
/sbin/ipchains -L
;;
test)
action "Testovací režim: " /bin/true
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
echo 1 > /proc/sys/net/ipv4/ip_forward
/sbin/ipchains -A input -j ACCEPT
/sbin/ipchains -A output -j ACCEPT
/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -i $PUBLIC -j MASQ
echo
;;
*)
echo "Spuštění: $0 {start|stop|restart|status|test}"
exit 1
esac
```

GFCC skript

Tento skript byl vygenerován programem Graphical Firewall (GFCC). Nejedná se o funkční sadu pravidel, jde pouze o exportovanou sadu.

```
#!/bin/sh
# Vygenerováno programem Gtk+ firewall control center
IPCHAINS=/sbin/ipchains
localnet="192.168.1.0/24"
firewallhost="192.168.1.1/32"
localhost="172.0.0.0/8"
DNS1="24.94.163.119/32"
DNS2="24.94.163.124/32"
Broadcast="255.255.255.255/32"
Multicast="224.0.0.0/8"
Any="0.0.0.0/0"
mail_grennan_com="192.168.1.1/32"
mark_grennan_com="192.168.1.3/32"
$IPCHAINS -P input DENY
$IPCHAINS -P forward ACCEPT
$IPCHAINS -P output ACCEPT
$IPCHAINS -F
$IPCHAINS -X
# vstupní pravidla
$IPCHAINS -A input -s $Any -d $Broadcast -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any netbios-ns -j DENY
$IPCHAINS -A input -p tcp -s $Any -d $Any netbios-ns -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any netbios-dgm -j DENY
$IPCHAINS -A input -p tcp -s $Any -d $Any netbios-dgm -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any bootps -j DENY
$IPCHAINS -A input -p udp -s $Any -d $Any bootpc -j DENY
$IPCHAINS -A input -s $Multicast -d $Any -j DENY
$IPCHAINS -A input -s $localhost -d $Any -i lo -j ACCEPT
$IPCHAINS -A input -s $localnet -d $Any -i eth1 -j ACCEPT
```

```
$IPCHAINS -A input -s $localnet -d $Broadcast -i eth1 -j ACCEPT
$IPCHAINS -A input -p icmp -s $Any -d $Any -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any -j ACCEPT ! -y
$IPCHAINS -A input -p udp -s $DNS1 domain -d $Any 1023:65535 -j ACCEPT
$IPCHAINS -A input -p udp -s $DNS2 domain -d $Any 1023:65535 -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any ssh -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any telnet -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any smtp -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any pop-3 -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any auth -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any www -j ACCEPT
$IPCHAINS -A input -p tcp -s $Any -d $Any ftp -j ACCEPT
$IPCHAINS -A input -s $Any -d $Any -j DENY -l
# předávací pravidla
$IPCHAINS -A forward -s $localnet -d $Any -j MASQ
# výstupní pravidla
```

RC skript bez GFCC

Toto nastavení pravidel jsem vytvořil sám. Nepoužívá GFCC.

```
#!/bin/bash
#
# Firewall Script - Version 0.9.0
# chkconfig: 2345 09 99
# popis: firewallový skript pro jádro 2.2.x
# Nastavit pro testování:
# set -x
#
# POZNÁMKY:
#
# Tento skript je určen pro RedHat 6.0 a vyšší.
#
# Nastavení by mělo vyhovovat většině směrovačů, vytáčených připojení a kabelo-
vých
# modemů. Je určeno pro distribuci RedHat.
#
# Pozor při nabízení veřejných služeb jako web nebo ftp.
#
# INSTALACE:
# 1. Tento soubor je určen pro systém RedHat. Měl by pracovat
# i na jiných distribucích, ale...
# Kdo ví?!!? Pokyny platí pro RedHat.
#
# 2. nahrejte tento soubor do /etc/rc.d/init.d (budete muset být root..)
# pojmenujte jej tak nějak jako "firewall" :-)
# ať jej vlastní root --> "chown root.root (název)"
# ať je spustitelný --> "chmod 755 (název)"
#
# 3. nastavte hodnoty pro vaši síť, rozhraní a DNS servery
# odkomentujte řádky povolující potřebné volitelné služby
# zajistěte, aby karta v počítači byla "eth0" (nebo změňte tuto hodnotu ve
skriptu)
```

```
# otestujte skript --> "/etc/rc.d/init.d/<název> start"
# pravidla si můžete vypsat --> "ipchains -L -n"
# opravte co nefunguje... :-)
#
# 4. přidejte firewall do inicializačních struktur RH --> "chkconfig --add (ná-
zev)"
# při příštím startu systému by to mělo naběhnout automaticky!
# spěte sladce, protože máte systém O NĚCO MÉNĚ zranitelný...
#
# Poznámky k verzím:
# 20 July, 1999 - původní verze - Anthony Ball <tony@LinuxSIG.org>
# 11 Dec, 1999 - aktualizoval Mark Grennan <mark@grennan.com>
#
#########################################################
# Doplňte následující hodnoty aby platily pro vaši síť
PRIVATENET=xxx.xxx.xxx.xxx/xx
PUBLIC=ppp0
PRIVATE=eth0
# vaše DNS servery
DNS1=xxx.xxx.xxx.xxx
DNS2=xxx.xxx.xxx.xxx
#########################################################
# některé obecné užitečné hodnoty
ANY=0.0.0.0/0
ALLONES=255.255.255.255
# Knihvona zdrojových funkcí
. /etc/rc.d/init.d/functions
# Konfigurace zdrojové sítě
. /etc/sysconfig/network
# Kontrola zda je síť nastavena
[ ${NETWORKING} = "no" ] && exit 0
# Podíváme se, jak nás spustili
case "$1" in
start)
# Povolujeme přístup
action "Spouštím firewall: " /bin/true
##
## Nastavení prostředí
##
# Smazání všech pravidel
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
# Všechno zakázat
/sbin/ipchains -I input 1 -j DENY
# nastavíme politiku na deny (implicitně je ACCEPT)
/sbin/ipchains -P input DENY
/sbin/ipchains -P output ACCEPT
/sbin/ipchains -P forward ACCEPT
# Zapneme předávání paketů
echo 1 > /proc/sys/net/ipv4/ip_forward
##
## Instalace modulů
```

```
##
# Přidáme modul aktivního ftp. Tím se povolí aktivní ftp na počítače v lokální
# síti (ale ne na směrovač, protože nemá maškarádu)
if ! ( /sbin/lsmod | /bin/grep masq_ftp > /dev/null ); then
/sbin/insmod ip_masq_ftp
fi
##
## Nějaká bezpečnostní nastavení
##
# zapneme ověřování zdrojových adres a ochranu před spoofingem na všech
# aktivních i budoucích rozhraních.
if [ -e /proc/sys/net/ipv4/conf/all/rp_filter ]; then
for f in /proc/sys/net/ipv4/conf/*/rp_filter; do
echo 1 > $f
done
else
echo
echo "POTÍŽE PŘI AKTIVACI OCHRANY PŘED SPOOFINGEM! POZOR! "
echo
fi
# vypneme broadcast na zbývajících rozhraních
/sbin/ipchains -A input -d 0.0.0.0 -j DENY
/sbin/ipchains -A input -d 255.255.255.255 -j DENY
# toto vypínáme bez logování, protože toho bývá hodně...
/sbin/ipchains -A input -p udp -d $ANY 137 -j DENY # NetBIOS přes IP
/sbin/ipchains -A input -p tcp -d $ANY 137 -j DENY # ""
/sbin/ipchains -A input -p udp -d $ANY 138 -j DENY # ""
/sbin/ipchains -A input -p tcp -d $ANY 138 -j DENY # ""
/sbin/ipchains -A input -p udp -d $ANY 67 -j DENY # bootp
/sbin/ipchains -A input -p udp -d $ANY 68 -j DENY # ""
/sbin/ipchains -A input -s 224.0.0.0/8 -j DENY # Multicastové adresy
##
## Povolíme privátní síti ven
##
# povolíme pakety na zpětném rozhraní
/sbin/ipchains -A input -i lo -j ACCEPT
# povolíme pakety z interního "prověřeného" rozhraní
/sbin/ipchains -A input -i $PRIVATE -s $PRIVATENET -d $ANY -j ACCEPT
/sbin/ipchains -A input -i $PRIVATE -d $ALLONES -j ACCEPT
##
## Povolíme vnější služby na firewall (jestli si troufáte)
##
# povolíme ICMP
/sbin/ipchains -A input -p icmp -j ACCEPT
# povolíme TCP
/sbin/ipchains -A input -p tcp ! -y -j ACCEPT
# povolíme DNS dotazy (na firewall)
/sbin/ipchains -A input -p udp -s $DNS1 domain -d $ANY 1023: -j ACCEPT
/sbin/ipchains -A input -p udp -s $DNS2 domain -d $ANY 1023: -j ACCEPT
# nebo (LEPŠÍ ŘEŠENÍ) spustíme na směrovači caching DNS server a místo před-
chozích
# dáme následující dva řádky...
# /sbin/ipchains -A input -p udp -s $DNS1 domain -d $ANY domain -j ACCEPT
```

```
# /sbin/ipchains -A input -p udp -s $DNS2 domain -d $ANY domain -j ACCEPT
# odkomentováním následujícího se povolí příchozí ssh
/sbin/ipchains -A input -p tcp -d $ANY 22 -j ACCEPT
# odkomentováním následujícího se povolí telnet (ŠPATNÝ NÁPAD!!)
/sbin/ipchains -A input -p tcp -d $ANY telnet -j ACCEPT
# odkomentováním následujícího se povolí (network time protocol) na směrovač
# /sbin/ipchains -A input -p udp -d $ANY ntp -j ACCEPT
# odkomentováním následujícího se povolí SMTP (ne pro klienty, pouze server)
/sbin/ipchains -A input -p tcp -d $ANY smtp -j ACCEPT
# odkomentováním následujícího se povolí POP3 (pro klienty)
/sbin/ipchains -A input -p tcp -d $ANY 110 -j ACCEPT
# povolíme auth pro posílání pošty a ftp
/sbin/ipchains -A input -p tcp -d $ANY auth -j ACCEPT
# povolíme HTTP (pouze pokud na směrovači běží www server)
/sbin/ipchains -A input -p tcp -d $ANY http -j ACCEPT
# povolíme FTP
/sbin/ipchains -A input -p tcp -d $ANY ftp -j ACCEPT
##
## Nastavení maškarády
##
# maškaráda paketů z interní sítě
/sbin/ipchains -A forward -s $PRIVATENET -d $ANY -j MASQ
##
## zakážeme COKOLIV jiného a zalogujeme to do /var/log/messages
##
/sbin/ipchains -A input -l -j DENY
# Zrušení plošného zákazu
/sbin/ipchains -D input 1
;;
stop)
action "Zastavuji firewall: " /bin/true
echo 0 > /proc/sys/net/ipv4/ip_forward
/sbin/ipchains -F input
/sbin/ipchains -F output
/sbin/ipchains -F forward
echo
;;
restart)
action "Restartuji firewall: " /bin/true
$0 stop
$0 start
echo
;;
status)
# Výpis nastavení
/sbin/ipchains -L
;;
test)
##
## Tak jednoduché, jak to jenom jde
## (Není to VŮBEC bezpečné)
action "POZOR! Testování firewallu: " /bin/true
/sbin/ipchains -F input
```

```
/sbin/ipchains -F output
/sbin/ipchains -F forward
echo 1 > /proc/sys/net/ipv4/ip_forward
/sbin/ipchains -A input -j ACCEPT
/sbin/ipchains -A output -j ACCEPT
/sbin/ipchains -P forward DENY
/sbin/ipchains -A forward -i $PUBLIC -j MASQ
echo
;;
*)
echo "Spuštění: $0 {start|stop|restart|status|test}"
exit 1
esac
```

VPN RC skript pro RedHat

```
#!/bin/sh
#
# vpnd
# Tento skript zapíná a vypíná vpnd (Virtual Privage Network connections).
#
# chkconfig: - 96 96
# popis: vpnd
#
# Knihovna zdrojových funkcí.
. /etc/rc.d/init.d/functions
# Konfigurace zdrojové sítě.
. /etc/sysconfig/network
# Kontrola zda je síť nastavena.
[ ${NETWORKING} = "no" ] && exit 0
[ -f /usr/sbin/vpnd ] || exit 0
[ -f /etc/vpnd.conf ] || exit 0
RETVAL=0
# Podíváme se, jak nás volali.
case "$1" in
start)
# Start daemons.
echo -n "Spouštím vpnd: "
daemon vpnd
RETVAL=$?
[ $RETVAL -eq 0 ] && touch /var/lock/subsys/vpnd
echo
;;
stop)
# Stop daemons.
echo -n "Zastavuji vpnd: "
killproc vpnd
RETVAL=$?
[ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/vpnd
echo
;;
restart)
```

```
$0 stop
$0 start
;;
*)
echo "Spuštění: vpnd {start|stop|restart}"
exit 1
esac
exit $RETVAL
```

KAPITOLA 3

Guylhem Aznar, guylhem@metalab.unc.edu

Pošta z pohledu správce

Tento dokument popisuje nastavení, údržbu a provoz elektronické pošty pod operačním systémem Linux. Je primárně určen pro správce poštovních systémů, ne pro jejich uživatele. (Informace o uživatelské problematice a uživatelských programech naleznete v dokumentu HOWTO uživatele elektronické pošty.) Tento dokument byste si měli pročíst v případě, že budete lokálně nebo přes síť posílat elektronickou poštu. Pokud elektronickou poštu nepoužíváte (ať už mezi uživateli jednoho systému nebo mezi uživateli různých systémů), dokument si číst **nemusíte**.

Úvod, autorská práva a zodpovědnost za správnost

Zaměření

Účelem tohoto dokumentu je zodpovědět některé často se vyskytující otázky a komentáře (FAQ) obecně k aplikacím elektronické pošty pro Linux a specificky k verzím pro distribuce Debian GNU/Linux a RedHat.

Nové verze

Nové verze tohoto dokumentu budou pravidelně posílány do diskusních skupin *comp.os.linux.announce*, *comp.answers* a *mail.answers*. Objeví se také na různých anonymních ftp-serverech, které archivují takový typ informací, včetně ftp://www.ibiblio.org/pub/Linux/docs/HOWTO.

Kromě toho by mělo být vždy možné nalézt tento dokument na webových stránkách Linux Documentation Project na adrese *http:// www.linuxdoc.org*.

Ohlasy

Zajímají mě jakékoliv vaše ohlasy (elektronickou poštou), týkající se tohoto dokumentu, kladné i záporné. Určitě mi dejte vědět, pokud objevíte chyby nebo zjevné překlepy.

Všechny dopisy obdržené elektronickou poštou si přečtu, ale ne na všechny odpovídám. Žádosti budou zvažovány a vyřizovány v závislosti na mém volném čase, na naléhavosti žádosti a na mém krevním tlaku :-).

Co se mi nebude líbit, to skončí v /dev/null, takže můžete být klidní.

Ohlasy na formát dokumentu by měly být odesílány koordinátorovi projektu HOWTO: Timu Bunymu (*howto@wallybox.cei.net*).

Autorská práva

Vlastníkem autorských práv k tomuto dokumentu je (c) 1998 Guylhem Aznar. Distribuce se provádí v licenci LDP. Všechny dotazy prosím směřujte na koordinátora Linux HOWTO - Tima Bunyho (*howto@wallybox.cei.net*).

Omezení záruky

Samozřejmě se distancuji od jakékoliv zodpovědnosti za obsah tohoto dokumentu. Použití návodů, příkladů a jiného obsahu dokumentu provádíte zcela na vlastní riziko.

Další zdroje informací

Viz HOWTO uživatele elektronické pošty.

USENET

Na konfiguraci a provozu aplikací elektronické pošty pro Linux není rozhodně nic „specifického“. Vzhledem k tomu *NEPOSÍLEJTE* dotazy, týkající se využití elektronické pošty, do skupin **comp.os.linux**.*

Do hierarchie **comp.os.linux** posílejte pouze dotazy, specifické pro Linux, jako jsou: „Se kterými volbami byl kompilován Debian 1.2 sendmail?“ nebo „RedHat 5.0 smail se po spuštění zhroutí.“

Dovolte mi to ještě zopakovat.

Pro posílání čehokoliv, co se týká pošty, do hierarchie **comp.os.linux** není prakticky žádný důvod. Na *VŠECHNY* vaše otázky by měly stačit skupiny v hierarchii comp.mail.*

> **JESTLIŽE POSÍLÁTE NA COMP.OS.LINUX.* OTÁZKY, KTERÉ SE NETÝKAJÍ KONKRÉTNĚ LINUXU, HLEDÁTE POMOC NA ŠPATNÉM MÍSTĚ. ODBORNÍCI NA ELEKTRONICKOU POŠTU VYŘIZUJÍ DOTAZY NA MÍSTECH, ZMÍNĚNÝCH VÝŠE, PŘIČEMŽ NEMUSÍ NUTNĚ POUŽÍVAT LINUX.**
>
> **POSÍLÁNÍM OTÁZEK, KTERÉ SE NETÝKAJÍ KONKRÉTNĚ LINUXU, DO LINUXOVÉ HIERARCHIE JEDINĚ MARNÍTE SVŮJ A CIZÍ ČAS, NAVÍC SE TÍM PRODLOUŽÍ VAŠE ČEKÁNÍ NA ODPOVĚĎ.**

PATŘIČNÁ MÍSTA jsou:

comp.mail.elm	poštovní klient ELM
comp.mail.mh	systém Rand Message Handling
comp.mail.mime	rozšíření MIME
comp.mail.misc	všeobecná diskuse o počítačové poště
comp.mail.multi-media	multimediální pošta
comp.mail.mush	Mail User's Shell (MUSH)
comp.mail.sendmail	BSD sendmail
comp.mail.smail	smail
comp.mail.uucp	pošta v prostředí UUCP

Emailové konference

Pro sendmail, smail a qmail existuje velké množství e-mailových konferencí.

Adresy je možné nalézt v */usr/doc/tenkterýjstesizvolili*.

Další dokumenty z LDP

V jiných dokumentech Linux HOWTO a v projektu Linux Documentation Project je k dispozici množství zajímavého materiálu.

Konkrétně se můžete podívat na následující:

- na vašem vlastním počítači je adresář /usr/doc/ :-)
- Příručka správce sítě
- HOWTO uživatele elektronické pošty
- Serial Communications HOWTO (komunikace po sériové lince)
- Ethernet HOWTO
- UUCP HOWTO, jestliže používáte UUCP

Knihy

Následuje sada knih, které by vám mohly být užitečné:

- **„Managing UUCP and USENET“** od O'Reilly and Associates je podle mého názoru nejlepší knihou o programech a protokolech, používaných při provozu serveru pro USENET.
- **„Unix Communications“** od The Waite Group obsahuje perfektní popis všech součástí a způsobu jejich spojení.
- **„Sendmail“** od O'Reilly and Associates je nejlepší příručkou pro sendmail-v8 a sendmail+IDA. Nutná koupě pro každého, kdo chce využívat sendmail a nehodlá se učit až z vlastních chyb při jeho provozu.
- **„The Internet Complete Reference“** od Osborne je dobrou příručkou, vysvětlující různé služby, dostupné na Internetu, a současně je skvělým zdrojem informací o různých zdrojích Internetu, jako jsou news, a elektronická pošta.
- **„Příručka správce sítě“** od Olafa Kircha z Linux Documentation Project je k dispozici na síti a určitě ji vydává O'Reilly a SSC.

Jak elektronická pošta funguje

Nyní si popíšeme informační tok, ke kterému typicky dochází, když spolu dvě osoby komunikují elektronickou poštou. Řekněme, že Alice na svém počítači *wonderland.com* chce poslat poštu Bobovi na počítači *dobbs.com*. Oba počítače jsou připojeny k Internetu.

Je dobré vědět, že poštovní zprávy na Internetu se skládají ze dvou částí – hlaviček a těla, oddělených od sebe prázdným řádkem. Hlavičky obsahují adresu odesilatele a adresáta, uživatelem zadaný předmět zprávy, datum odeslání zprávy a celou řadu dalších užitečných informací. Tělo pak představuje vlastní obsah zprávy. Podívejme se na příklad:

```
From: "Alice" <alice@wonderland.com>
Message-Id: <199711131704.MAA18447@wonderland.com>
Subject: Neviděls mého bílého králíka?
To: bob@dobbs.org (Bob)
```

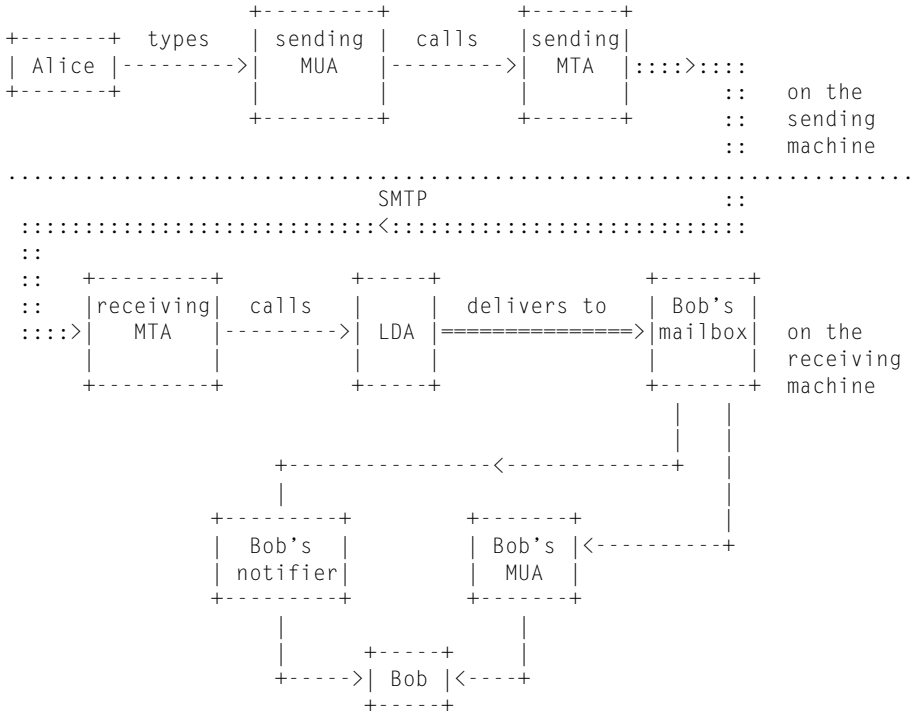
```
Date: Thu, 13 Nov 1997 12:04:05 -0500 (EST)
Content-Type: text

Jsem docela vyděšená. Mám strach, jestli nespadl do nějaké díry.
--
>>alice>>
```

Uspořádání a význam poštovních hlaviček je definován internetovým standardem RFC822.

Pošta mezi trvale připojenými počítači

Takto vypadá diagram celého procesu. Jednotlivé stavy a terminologii objasním dále.



types – píše; sending MUA – odesílající uživatelský agent; calls – volá; sending MTA – odesílající přenosový agent; on the sending machine – na odesílajícím počítači; receiving MTA – přijímající přenosový agent; calls – volá; LDA – lokální agent; delivers to – doručuje do; Bob's mailbox – Bobova schránka; Bob's notifier – Bobův notifikátor; Bob's MUA – Bobův uživatelský agent; on the receiving machine – na přijímajícím počítači

K odeslání pošty Alice spustí program pojmenovaný mail user agent (MUA, uživatelský agent). Uživatelský agent je to, co se obvykle označuje termínem „poštovní program" – tedy program, který jí pomůže napsat zprávu, typicky pomocí zvoleného textového editoru. Jakmile Alice klepne na tlačítko „Odeslat" uživatelského agenta, její úloha skončila. Později budeme v tomto dokumentu hovořit o některých oblíbených uživatelských agentech.

Alicí používaný uživatelský agent okamžitě předá její zprávu programu označovanému jako mail transfer agent (MTA, přenosový agent). Obvykle půjde o program *sendmail*, i když oblibu si zís-

kavají i jiné přenosové agenty, které se možná v příštích distribucích Linuxu objeví. Později budeme v tomto dokumentu hovořit i o přenosových agentech.

Úkolem přenosového agenta je předat zprávu přenosovému agentovi na Bobově počítači. Který je Bobův počítač zjistí analýzou hlavičky `To:`, kde na pravé straně Bobovy adresy najde text *dobbs.com*. Pomocí této adresy otevře spojení s Bobovým počítačem. Mechanismus vytvoření tohoto spojení představuje úplně jinou problematiku, pro naše potřeby nám stačí vědět, že toto spojení představuje pro přenosového agenta na Alicině počítači možnost jak na Bobův počítač posílat textové příkazy a jak přijímat výsledky o jejich provedení.

Na portu 25 Bobova počítače poslouchá jeho vlastní přenosový agent (pravděpodobně opět program sendmail). Alicin přenosový agent si bude s Bobovým povídat pomocí protokolu SMTP (Simple Mail Transfer Protocol). Následuje příklad toho, jak jejich dialog bude vypadat. Řádky odesílané Aliciným agentem jsou označeny A:, řádky odesílané Bobovým B:.

```
A: MAIL FROM:<alice@wonderland.com>
B: 250 OK
A: RCPT TO:<bob@dobbs.com>
B: 250 OK
A: DATA
B: 354 Start mail input; end with <CRLF>.<CRLF>
A: From: "Alice" <alice@wonderland.com>
A: Message-Id: <199711131704.MAA18447@wonderland.com>
A: Subject: Neviděls mého bílého králíka?
A: To: bob@dobbs.org (Bob)
A: Date: Thu, 13 Nov 1997 12:04:05 -0500 (EST)
A: Content-Type: text
A:
A: Jsem docela vyděšená. Mám strach, jestli nespadl do nějaké díry.
A: --
A:                                    >>alice>>
A: .
B: 250 OK
```

Příkaz protokolu SMTP je typicky jeden řádek textu a obdobně vypadá i odpověď na příkaz. Příkaz DATA představuje výjimku, jakmile přijímající agent narazí na tento příkaz, veškerý další text bude chápat jako samotnou zprávu až do doby, kdy narazí na řádek, obsahující samotnou tečku. (Protokol SMTP je definován standardem RFC821.)

Nyní má Bobův přenosový agent Alicinu zprávu. Do zprávy přidá hlavičku, která bude vypadat takto nějak:

```
Received: (from alice@wonderland.com)
    by mail.dobbs.com (8.8.5/8.8.5) id MAA18447
    for bob@dobbs.com; Thu, 13 Nov 1997 12:04:05 -0500
```

Tento údaj slouží pro potřeby sledování pošty v případě nějakých chyb (někdy totiž může pošta procházet přes více počítačů a bude takovýchto hlaviček obsahovat více). Bobův přenosový agent předá takto upravenou zprávu lokálnímu agentovi (local delivery agent, LDA). Na linuxových systémech slouží coby lokální agent obvykle program *procmail*, existují však i jiné.

Úkolem lokálního agenta je přidat poštu do Bobovy schránky. Tato operace je oddělena od doručovacího agenta, takže oba programy mohou být jednodušší a doručovací agent se může zabývat internetovou částí přenosu aniž by se staral o lokální podrobnosti jako je například to, kde je Bobova schránka uložena.

Bobovu schránku bude normálně představovat soubor */usr/spool/mail/bob* nebo */var/mail/bob*. Když Bob poštu čte, spouští svého vlastního uživatelského agenta, ve kterém si bude moci uložený soubor prohlédnout a upravit.

Notifikátory

V řetězci doručení pošty je ještě jeden důležitý program, i když sám o sobě poštu nečte ani nepřenáší. Je to *poštovní notifikátor*, program, který sleduje aktivity odehrávající se ve vaší poštovní schránce a upozorňující vás na příchod nové pošty.

Původně jako notifikátor sloužila dvojice unixových programů *biff(1)* a *comsat(8)*. Program *biff* představuje rozhraní, které vám umožňuje zapnout službu *comsat*. Když je tato služba zapnutá, vytiskne se na vašem terminálu hlavička nově přišlé poštovní zprávy. Toto chování bylo navrženo pro uživatele používající řádkové programy na jednoduchých terminálech, v dnešních prostředích není vyhovující.

Většina unixových příkazových interpretů má vestavěnou funkci kontroly poštovní schránky a fungují tak jako notifikátory méně drastickým způsobem (například vypíší informaci o došlé poště pod výzvou příkazového interpretu). Toto chování můžete typicky zapnout nastavením proměnných, které jsou popsány v manuálových stránkách shellu. Pokud používáte příkazový interpret z rodiny *sh/ksh/bash*, zajímají vás proměnné *MAIL* a *MAILPATH*.

Systémy s podporou X-Windows obsahují jeden nebo více malých prográmků na ploše, které periodicky kontrolují schránku a dojde-li pošta, akusticky i opticky vás na to upozorní. Nejstarší a nejpoužívanější program této kategorie je *xbiff*, pokud máte Linux nainstalován s podporou X-Windows, určitě tam tento program najdete. Podrobnosti o něm můžete zjistit na manuálových stránkách *xbiff(1)*.

Pošta mezi počítači s občasným připojením

Pokud jste četli pozorně, jistě jste si všimli, že právě popsaná komunikace je založena na tom, že Alicin počítač bude schopen přímo komunikovat s Bobovým počítačem. Co se ale bude dít, pokud bude Bobův počítač vypnutý, nebo pokud bude zapnutý, ale nebude připojen k Internetu?

Pokud nebude Alicin přenosový agent schopen spojit se s Bobovým agentem, uloží Alicinu zprávu do fronty na systému *wonderland.com*. Pak se bude opakovaně v určitých intervalech pokoušet poštu doručit až do doby, než vyprší časový limit a v takovém případě pošle Alici zprávu o tom, že její poštu nebylo možné doručit. Ve výchozím nastavením nejoblíbenějšího přenosového agenta (*sendmail*) je interval opakování 15 minut a limit pro doručení je nastaven na 4 dny.

Vzdálená pošta a protokoly vzdálené pošty

V dnešní době je řada uživatelů Linuxu připojena k Internetu prostřednictvím poskytovatele a nemají vlastní internetovou doménu. Namísto toho mají vytvořen účet na počítači poskytovatele. Jejich pošta se doručuje do schránky na tomto počítači poskytovatele. Typicky si ale uživatel čte svou poštu ze svého počítače, který je k Internetu nepravidelně připojován protokoly jako SLIP nebo PPP. Linux nabízí vzdálené poštovní protokoly, které toto uspořádání podporují.

Všimněte si, že jde o jiný mechanismus, než o jakém jsme hovořili v předchozím textu. Pošta uložená ve frontě a čekající na opakované odeslání není to samé jako pošta uložená v poštovní schránce na serveru poskytovatele. Pošta ve frontě je chápána jako nedoručená a po určité době bude zahozena, pošta uložená na serveru poskytovatele *je* doručená a může na něm čekat věčně.

Protokoly vzdálené pošty umožňují převzetí pošty ze serveru klientským programem prostřednictvím síťové linky (je to opačný proces než při klasickém doručení, kdy odesílající přenosový agent předá poštu přijímajícímu přenosovému agentovi). Existují dva běžně používané protokoly pro vzdálené přebírání pošty, POP3 (definovaný standardem RFC1939) a IMAP (definovaný standar-

dem RFC2060). Prakticky všichni poskytovatelé podporují protokol POP3, stále větší počet jich podporuje i protokol IMAP (který je výkonnější).

Takto vypadá příklad relace vedené protokolem POP3 (K znamená klienta, S je server):

```
K: <klient se připojí na port 110>
S: +OK POP3 server ready <1896.697170952@mailgate.dobbs.org>
K: USER bob
S: +OK bob
K: PASS redqueen
S: +OK bob's maildrop has 2 messages (320 octets)
K: STAT
S: +OK 2 320
K: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
K: RETR 1
S: +OK 120 octets
S: <server POP3 posílá 1. zprávu>
S: .
K: DELE 1
S: +OK message 1 deleted
K: RETR 2
S: +OK 200 octets
S: <server POP3 posílá 2. zprávu>
S: .
K: DELE 2
S: +OK message 2 deleted
K: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
K: <klient ukončí spojení>
```

Relace protokolem IMAP používá jiné příkazy a jiné odpovědi, logika je však velmi podobná.

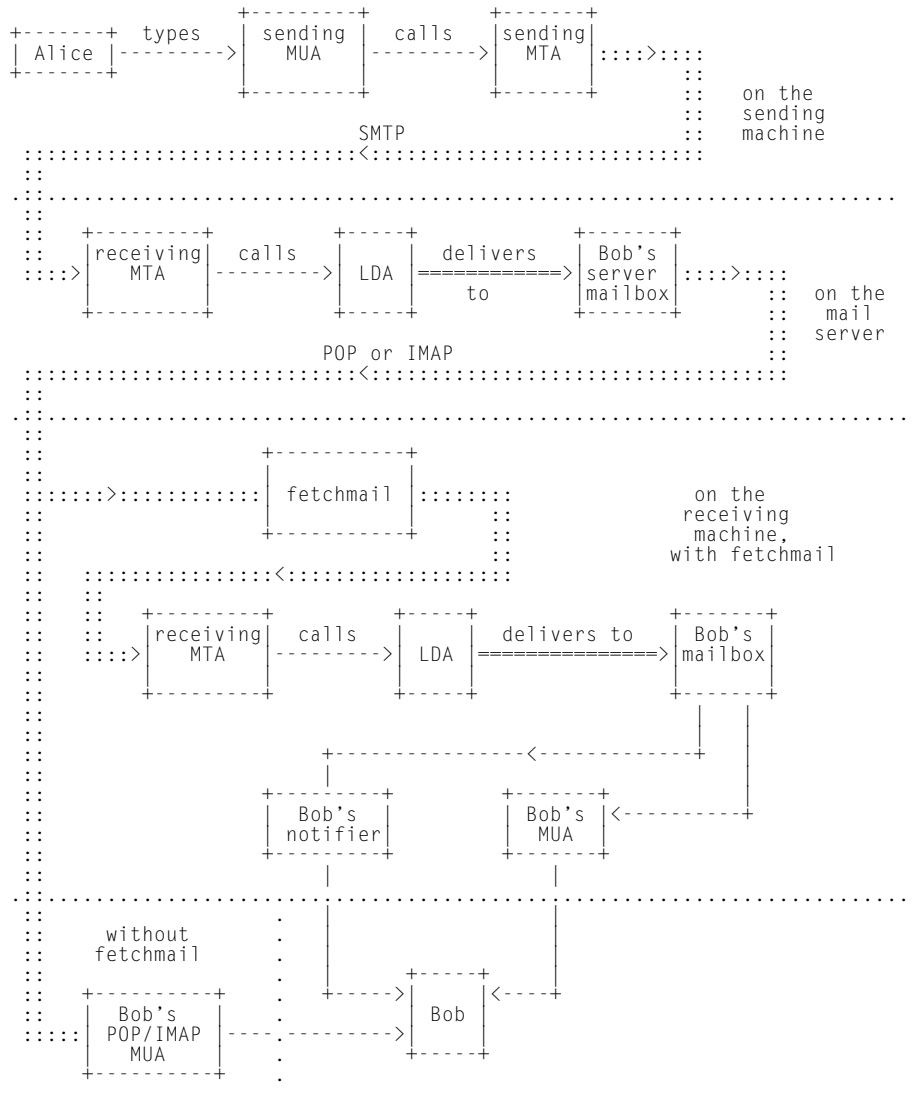
Abyste mohli protokol POP3 nebo IMAP použít, potřebujete vzdáleného poštovního klienta, který bude vaši poštu vybírat. Některé uživatelské agenty mají vestavěného i vzdáleného klienta (které se dozvíme dále) a program Netscape přímo podporuje protokoly POP i IMAP.

Hlavní nevýhodou POP klientů v uživatelských agentech je to, že musíte svému poštovnímu programu explicitně říct, aby se serveru na novou poštu dotázal, nedozvíte se o ní programem *xbiff(1)* nebo ekvivalentním tak jako v případě lokální pošty nebo pošty klasicky doručené protokolem SMTP. Navíc ne všichni poštovní agenti podporují protokoly POP/IMAP, takže se možná budete muset vzdát svého oblíbeného programu.

Linux obvykle obsahuje program *fetchmail* určený přímo pro komunikaci se vzdálenými servery, který se periodicky serveru dotazuje, automaticky poštu stahuje a doručí ji normálním postupem tak, že se protokolem SMTP spojí s vaším přenosovým agentem.

Pokud potřebujete poštu na serveru zachovávat (například protože pracujete na více různých počítačích), představuje program *fetchmail* pravděpodobně lepší volbu než cokoliv, co vám může nabídnout váš uživatelský agent. Program *fetchmail* je možné nakonfigurovat tak, aby běžel na pozadí a periodicky poštu stahoval, takže notifikátory jako *xbiff(1)* a podobné budou funogovat stejně jako pro poštu doručenou protokolem SMTP. Navíc je *fetchmail* podstatně tolerantnější k různým specifikům a nestandardním odpovědím serverů než klienti v uživatelských agentech a má také lepší zotavení z chyb.

Následující diagram ukazuje, jak obě varianty (s i bez programu *fetchmail*) fungují:



type – píše; sending MUA – odesílající uživatelský agent; calls – volá; sending MTA – odesílající přenosový agent; on the sending... – na odesílajícím počítači; receiving MTA – přijímající přenosový agent; LDA – lokální agent; delivers to – doručuje do; Bob's server ... – Bobova schránka na serveru; on the mail server – na poštovním serveru; on the receiving machine witch fetchmail – na přijímajícím počítači s programem fetchmail; receiving MTA – přijímající přenosový agent; calls – volá; LDA – lokální agent; delivers to – doručuje do; Bob's mailbox – Bobova schránka; Bob's notifier – Bobův notifikátor; Bob's MUA – Bobův uživatelský agent; without fetchmail – bez programu fetchmail; Bob's POP/IMAP MUA – Bobův uživatelský agent s podporou POP/IMAP

Formáty poštovní schránky

Když se došlá pošta přidává do poštovní schránky, je problémem přenosového agenta, aby zajistil nějaké oddělení toho, kde jedna zpráva končí a kde druhá začíná.

V Unixu platí pro prakticky všechny poštovní programy konvence, že řádek začínající textem „From „ (mezera je důležitá) označuje novou zprávu. Pokud by se slovo „From „ objevilo na za-

čátku řádku textu, přenosový agent je obecně uvede prefixem „větší než“, tedy „>From „. Za takovýmto řádkem From následují další hlavičky dle RFC822 (typicky následují jméno odesílatele a čas přijetí).

Tato konvence pochází z Unixu verze 7, proto se tento typ schránek někdy označuje jako „schránky V7“, nebo také jako „formát mbox“. Nebude-li řečeno jinak, všechny programy zmíněné v tomto dokumentu očekávají tento formát schránky. Nejedná se nicméně o univerzální formát a nástroje očekávající nebo generující jiné formáty mohou vést ke spoustě zmatků.

Další čtyři formáty o nichž je třeba vědět (a dávat si na ně pozor) jsou BABYL, MMDF, MH a qmail maildir. Z těchto formátů je nejjednodušší MMDF, který jako oddělovač zpráv používá sekvenci čtyř znaků Ctrl-A (ASCII 001) za nimiž následuje CR-LF. MMDF byl jeden z prvních a velmi neohrabaných poštovních programů na Internetu, jeho následníci se stále používají v systémech SCO.

BABYL je další přežitek, pocházející ze starého poštovního systému používaného na MITu. Stále jej používá Emacs v režimu čtení pošty.

MH a qmail maildir jsou formáty „schránky“, které jako schránku používají adresář souborů a každou zprávu ukládají v samostatném souboru. Pokud na takové „schránce“ spustíte *grep*, nedostanete se nikam, protože *grep* uvidí pouze kusy adresářů.

Schránky programu Microsoft Outlook (.mbx) je možné zkonvertovat do formátu RFC822 programem *mbx2mbox*.

Požadavky

Hardware

Používání pošty v Linuxu nemá žádné specifické hardwarové požadavky.

Budete potřebovat nějaký druh „transportních“ programů, používaných pro připojení ke vzdáleným systémům, což znamená buď TCP/IP nebo UUCP.

Znamená to, že podle konfigurace systému budete potřebovat modem nebo ethernetovou kartu.Ve většině případů je vhodné mít nejrychlejší dostupný modem, dnes například V90 57600 bps. Obecně je vhodné mít na sériovém portu nebo přímo v modemu 16550 UART, aby bylo možné zvládnout rychlosti nad 9600 baudů.

Jestliže nechápete smysl poslední věty, informace si zjistěte ve skupině **comp.dcom.modems** nebo využijte USENET a jeho FAQ a periodicky zasílané informace, týkající se komunikace pomocí modemu a sériové komunikace.

Volba transportního agenta

Transportní agent je program, který doručuje poštu z vašeho lokálního systému na vzdálený systém. Jen velmi zřídka je na současných Linuxech nutné na vestavěný transportní agent sahat nebo jej měnit, takže bude lépe nesahat na něco co funguje. Nicméně uvádíme základní přehled abyste si mohli porovnat různé systémy pro případ, že se rozhodnete pro něco bezpečnějšího a výkonnějšího, než váš systém standardně nabízí.

(Existují i jiné přenosové agenty než dále uvedené, ale je velmi nepravděpodobné, že byste se s nimi na Linuxu setkali.)

Každý z nich má své výhody, nejlepším kompromisem je *qmail*. Je vysoce bezpečný (i když *vmail* je bezpečnější), je rychlý (i když *smail* je pro lokální poštu rychlejší) a snadno se konfiguruje. Samozřejmě si zvolte kterýkoliv program budete chtít. Následující informace vám mají pouze pomoci zvolit si dobře.

Sendmail může vyhovovat pro sítě s velmi komplikovanými nastaveními, nicméně se domnívám, že jeho konfigurace je pro začátečníka zbytečně komplikovaná a není ani nijak rychlý ani nijak bezpečný, takže následující popis programu *sendmail* je **opravdu** zastaralý.

Pokud opravdu víte co děláte, použijte *sendmail* (a pak nečtěte tento dokument!), jinak vám doporučuji *qmail*.

Následuje podrobný popis jednotlivých programů.

sendmail

BSD *sendmail* je pradědeček přenosových agentů. Přežil už několik různých „následníků". Většina linuxových systémů jej dnes používá a má jej předinstalován.

Sendmail je proslulý jako noční můra administrátorů – těžko se chápe, zmateně se konfiguruje a obsahuje spousty bezpečnostních děr. S vývojem internetových technologií a standardů se řada voleb a nastavení tohoto programu, které daly vzniknout jeho hrozivé reputaci, naštěstí stala pro většinu systémů zbytečnou (hodně tomu pomohlo opuštění různých síťových vrstev jiných než TCP/IP, například UUCP). Navíc mají novější verze programu *sendmail* vylepšený konfigurační systém, který vás efektivně chrání před pověstným pralesem souboru *sendmail.cf*. A konečně, *sendmail* se normálně dodává předkonfigurovaný a neměli byste na něj raději vůbec sahat, pokud nepotřebujete nějaká skutečně speciální nastavení (například doručování pošty přes jinou síť než je TCP/IP).

Domovskou stránku programu *sendmail* najdete na adrese *http://www.sendmail.com*. Obsahuje odkazy na rozsáhlou dokumentaci, kterou budete jistě potřebovat, pokud se rozhodnete program sami konfigurovat.

Jiné přenosové agenty, pokud jsou označovány názvem „*sendmail*" obvykle napodobují sémantiku příkazových řádků tohoto programu. Je to výhodné pro uživatelské agenty, které typicky předpokládají, že se baví s programem *sendmail*.

smail verze 3.2

Smail byl první seriózní pokus nahradit program *sendmail*. Má podstatně jednodušší a výstižnější konfigurační systém než *sendmail* a je poměrně bezpečný. V některých distribucích Linuxu je předinstalován místo *sendmail*u.

Jednu dobu představoval důvod velké oblíbenosti programu *smail* jeho vynikající podpora pro kombinaci prostředí TCP/IP a UUCP, s ústupem standardu UUCP klesala i obliba programu *smail*. Kromě toho je na vysoce zatížených systémech méně efektivní než *sendmail*.

Stejně jako u programu *sendmail* i zde platí, že s největší pravděpodobností nebudete muset měnit přednastavenou konfiguraci.

(Občas můžete někde narazit na zmínku o programu *smail 2.5*. Tento program je už dlouhou dobu zastaralý. Nestarejte se o něj.)

qmail

Program *qmail* je přenosový agent kompatibilní se *sendmail*em navržený zejména s ohledem na vysokou bezpečnost. Autor vyhlásil odměnu 500 dolarů za první odhalenou bezpečnostní slabinu, od března 1997 nebyla tato odměna vyplacena.

Domovská stránka programu *qmail* je na adrese *http://pobox.com/~djb/qmail.html*.

exim

Program *exim* se podobá *smail*u 3, má však větší možnosti. Údajně je velmi silný v blokování spamů a podpoře více virtuálních počítačů (více DNS domén) na jednom systému.

Domovská stránka programu *exim* je na adrese *http://www.exim.org*.

Zkoušel jsem ho na svém počítači, tváří se jako příjemné spojení konfiguračního systému programu *smail* a bezpečnosti programu *qmail*, navíc je to GPL.

V blízké době by měla být doplněna sekce věnovaná náhradě stávajícího přenosového agenta programem *exim*.

Instalace transportního agenta

Qmail verze 1.03

Bezpečný, rychlý a snadno použitelný – můj oblíbený přenosový agent.

V současné době neobsahuje žádná distribuce předinstalovaný program *qmail*. Zaměříme se na kompilaci a instalaci programu *qmail*, protože to je jediné úskalí, jeho konfigurace je velmi jednoduchá.

Získání programu qmail

Nejnovější verzi můžete získat na adrese *www.qmail.org*.

Rozbalení zdrojových kódů

Zdrojový kód rozbalíte příkazy

```
mv qmail.tar.gz /usr/local/src
cd /usr/local/src ; tar -zxvf qmail.tar.gz
```

Pokud narazíte na verzi ve formátu bz2 (což je novější a účinnější formát komprese), nahraďte *tar* tímto:

```
bunzip2 qmail.tar.bz2
tar -xvf qmail.tar
```

Příprava na překlad

Přejděte do adresáře s rozbaleným programem a podívejte se na standardní konfiguraci:

```
cd qmail; more conf-*
```

Nemělo by být nutné cokoliv měnit, můžete ale (například) nastavit jiný adresář pro instalaci nebo zvolit lepší parametry překladu.

Nyní spusťte

```
mkdir /var/qmail
```

čímž vytvoříte cílový adresář.

Pokud nemáte nainstalovánu distribuci Debian, budete muset vytvořit několik nových uživatelských ID pro potřeby programu *qmail* – jeho vysoká bezpečnost stojí právě na těchto účtech.

Fakt, že *qmail* je rozdělen na samostatné moduly z nichž každý běží pod svým UID znamená, že pro případného útočníka bude daleko obtížnější probourat se do celého poštovního systému nebo jeho zneužitím získat práva superuživatele.

Takže spusťte:

```
# groupadd nofiles
# useradd -g nofiles -d /var/qmail/alias alias
# useradd -g nofiles -d /var/qmail qmaild
# useradd -g nofiles -d /var/qmail qmaill
# useradd -g nofiles -d /var/qmail qmailp
# groupadd qmail
# useradd -g qmail -d /var/qmail qmailq
# useradd -g qmail -d /var/qmail qmailr
# useradd -g qmail -d /var/qmail qmails
```

případně ručně změňte soubory */etc/passwd* a */etc/group* a přidejte příslušné uživatele sami.

Evan E. hlásil, že pro čistou verzi *qmail*u (Caldera 1.2) použil příkaz *–g groupid*, jinak hlásil *groupadd* chybu „A group with that name already exists“.

Můžete například dále přidat

```
qmail:*:2107:
nofiles:*:2108:
```

a

```
alias:*:7790:2108::/var/qmail/alias:/bin/true
qmaild:*:7791:2108::/var/qmail:/bin/true
qmaill:*:7792:2108::/var/qmail:/bin/true
qmailp:*:7793:2108::/var/qmail:/bin/true
qmailq:*:7794:2107::/var/qmail:/bin/true
qmailr:*:7795:2107::/var/qmail:/bin/true
qmails:*:7796:2107::/var/qmail:/bin/true
```

Nyní můžete spustit

```
make setup check
```

čímž se provede kontrola konfigurace a

```
./config
```

čímž se *qmail* nakonfiguruje.

Pozor, server musí být přístupný v DNS, jinak to *./config* nezvládne.

Pokud nemáte záznam v DNS, můžete server pojmenovat přímo příkazem

```
./config-fast foo.bar.com
```

Nyní musíte vytvořit několik aliasů, protože *qmail* nepoužívá soubor */etc/aliases* pokud nepřeložíte a nenainstalujete doplňkový balík.

Moje konfigurace vypadá takto:

```
File : ".qmail-MAILER-DAEMON"
&postmaster
File : ".qmail-bin"
&root
Soubor : ".qmail-daemon"
&root
```

```
Soubor: ".qmail-decode"
&root
Soubor: ".qmail-dumper"
&root
Soubor: ".qmail-games"
&root
Soubor: ".qmail-ingres"
&root
Soubor: ".qmail-mailer-daemon"
&postmaster
Soubor: ".qmail-manager"
&root
Soubor: ".qmail-news"
&root
Soubor: ".qmail-nobody"
&root
Soubor: ".qmail-operator"
&root
Soubor: ".qmail-postmaster"
&root
Soubor: ".qmail-root"
&guylhem
Soubor: ".qmail-system"
&root
Soubor: ".qmail-toor"
&root
Soubor: ".qmail-uucp"
&root
Soubor: ".qmail-uucp-default"
|preline -dr /usr/bin/uux - -r -gC -a"${SENDER:-MAILER-DAEMON}" lm!rmail
"($DEFAULT@$HOST)"
```

Tyto soubory musíte vytvořit v *~alias* a nahradit *&guylhem* v *.qmail-root* vašímm přihlašovacím jménem pro získání superuživatelské pošty.

> **UPOZORNĚNÍ PRO UŽIVATELE UUCP!** NEVĚŘTE TEXTU „QMAIL FAQ FOR UUCP“ – POUŽIJTE MŮJ DOKUMENT `.qmail-uucp-default`! V OPAČNÉM PŘÍPADĚ NEBUDETE SCHOPNI PŘES UUCP PŘIPOJENÍ ODESÍLAT ŽÁDNOU POŠTU.

Nyní se musíte rozhodnout, v jakém formátu budou vaši uživatelé dostávat poštu.

Navrhuji toto:

- Pro domovské adresáře připojované přes NFS použijte formát MAILDIR s patchem pro lokální čtení pošty (patche jsou dostupné na *www.qmail.org*).
- Pokud patch není k dispozici, použijte formát MAILFILE: soubor obsahující poštu dokáže přečíst jakýkoliv program, uživatelé si budou muset pouze vytvořit alias (pro *bash*) nebo setenv (pro *csh*) na svůj program pro čtení pošty.
- Nepoužívejte formát */var/spool/mail/$USER*, je málo bezpečný.

Ke změně standardního formátu si přečtěte všechny soubory */var/qmail/boot* a pak zkopírujte ten který se vám nejvíce líbí do */var/qmail/rc*.

home a *proc* jsou bezpečné volby, kvůli vyšší bezpečnosti upřednostněte *home*.

Konfigurace programu qmail

V */var/qmail/control* upravte:

defaultdomain, me, plusdomain

- *me* je FQDN (plně kvalifikované doménové jméno) vašeho počítače, například na mém počítači to je *barberouge.linux.lmm.com*
- *defaultdomain* se přidává za názvy všech počítačů bez tečky včetně *defaulthost*, můžete tuto hodnotu nastavit například na *localnetwork* a jakákoliv pošta odesílaná na adresu *joe@hisbox* bude doplněna a adresována na *joe@hisbox.localnetwork*.
- *plusdomain* je výjimka: přidává se za názvy počítačů končící znakem plus včetně *defaulthost* (nastaveného hodnotou *me*), pokud končí plusem.

Tyto tři příklady by vám měly názorně ukázat, jak snadno se *qmail* konfiguruje.

locals, rcpthosts

Pokud chcete podporovat virtuální názvy domén, uveďte je v těchto souborech. Pošta došlá pro zde definovaná jména bude obsluhována lokálně.

Rozdíl mezi soubory *locals* a *rcpthosts* je v tom, že druhý se nechápe jako lokální alias, což je užitečné pokud přijímáte poštu z nějakých volných poštovních adres jako jsou *yahoo.com* nebo *lemel.fr* a zároveň odesíláte poštu jiným uživatelům těchto nelokálních služeb – tedy pokud nechcete lokálně obsluhovat poštu odeslanou někomu na adresu *someone@yahoo.com*.

virtualdomains

Zde můžete nastavit implicitní odchozí mód, například

```
#:alias-uucp
```

pokud nechcete poštu odesílat přes uucp ale přes smtp (což je standardní nastavení), nebo

```
:alias-uucp
```

pokud chcete poštu odesílat přes uucp.

Testování qmailu

Po nakonfigurování programu vyzkoušejte

```
sh -cf '/var/qmail/rc &'
```

čímž spustíte *qmail* (nebude se hádat s lokálním přenosovým agentem) a potom

```
echo to: mylogin | /var/qmail/bin/qmail-inject
```

Tuto zprávu byste měli obdržet ve formátu, který jste zvolili v */var/qmail/boot/*.

Odstranění jiných přenosových agentů

Pokud byl test úspěšný, zrušte předchozího přenosového agenta:

```
killall -STOP název_démona
```

pokud běží nějaké synovské procesy:

```
killall -CONT jejich_jména
```

počkat, znovu

```
kilall -STOP
```

a tak dokud nezrušíte všechno.

Pokud žádné synovské procesy neběží,

```
killall -TERM
```

a potom

```
killall -CONT
```

Odstraňte příslušný program (jak, to záleží na instalované distribuci, například rpm –e –-nodeps pro RedHat, Calderu a SuSE, nebo dpkg –r –force-depends na Debianu). Pak spusťte:

```
# ln -s /var/qmail/bin/sendmail /usr/lib/sendmail
# ln -s /var/qmail/bin/sendmail /usr/sbin/sendmail
```

Nyní nastavte *qmail-smtpd* v */etc/inetd.conf* (vše na jednom řádku):

```
smtp stream tcp nowait qmaild /var/qmail/bin/tcp-env tcp-env
/var/qmail/bin/qmail-smtpd
```

Pokud používáte nějakou starší distribuci bez podpory SYSV-init (například starší RedHat), doplňte do bootovacích skriptů:

```
sh -cf '/var/qmail/rc &'
```

Typicky by to mělo být v souboru */etc/rc.local*, váš systém ale může chtít něco jiného.

Pro distribuce kompatibilní s SYSV-init (RedHat, Caldera, SuSE, Debian) doplňte do */etc/init.d/* nebo */etc/rc.d/init.d/* následující skrip:

```
Verze pro Debian:
#!/bin/sh

test -x /var/qmail/rc || exit 0

case "$1" in
  start)
    echo -n "Spouští se mta: "
    sh -cf '/var/qmail/rc &'
    echo "qmail."
    ;;
  stop)
    echo -n "Zastavuje se mta: "
    killall qmail-lspawn
    echo "qmail."
    ;;
  restart)
    echo -n "Restartuje se mta: "
    killall -HUP qmail-lspawn
```

```
    killall -ALRM qmail-lspawn
    echo "qmail."
    ;;
  *)
    echo "Použití: /etc/init.d/qmail {start|stop|restart}"
    exit 1

esac
exit 0

Verze pro RedHat:
#!/bin/sh
#
# qmail Tento skript spouští a zastavuje qmail.
#
# popis: qmail je transportní agent, což je program, který
#  předává poštu z jednoho počítače na druhý.
# processname: qmail
# config: /var/qmail/control/

# Načti definice funkcí.
. /etc/rc.d/init.d/functions

# Načti síťovou konfiguraci.
. /etc/sysconfig/network

export PATH=$PATH:/var/qmail/bin

# Kontrola zda síť funguje.
[ ${NETWORKING} = "no" ] && exit 0

[ -f /usr/sbin/sendmail ] || exit 0

# Podíváme se, jak nás volali.
case "$1" in
  start)
    # Spuštění démona.
    echo -n "Spouští se qmail: "
    qmail-start '|preline procmail' splogger qmail &
    touch /var/lock/subsys/qmail
    echo
    ;;
  stop)
    # Zastavení démona.
    echo -n "Zastavuje se qmail: "
    killproc qmail-lspawn
    echo
    rm -f /var/lock/subsys/qmail
    ;;
  restart)
    $0 stop
    $0 start
    ;;
```

```
  status)
    status qmail
    ;;
  *)
    echo "Použití: qmail {start|stop|restart|status}"
    exit 1

esac
exit 0
```

Dále vytvořte symbolické odkazy pro všechny */etc/rc.d/rcN.d/*, například

```
ln -sf /etc/init.d/qmail /etc/rc1.d/K19qmail
```

Pokud je první písmeno K, bude se *qmail* na této úrovni zastavovat (1 pro jednouživatelský režim a 6 pro boot), pokud je první písmeno S, bude se *qmail* na této úrovni spouštět (pro všechny ostatní úrovně běhu).

- Jak poznat, kam patří S a kam K? Udělejte to, co dělá většina démonů na stejné úrovni.
- Jaké číslo patří za S nebo K? Číslo následující za číslem síťového démona. Znamená to, že se přenosový agent bude spouštět nebo zastavovat hned po (respektive před) síťovým démonem. Pokud byste jej spouštěli dřív, síť by ještě nefungovala a agent by s ní počítal.

RedHat, Caldera a SuSE používají */etc/rc.d/* namísto pouhého */etc/* v distribuci Debian, tedy například */etc/rc.d/rc1.d/* nebo */etc/rc.d/init.d*.

A to je vše, přátelé!

Nemusíte rebootovat (nezapomeňte, používáte Linux a ne jiné systémy). Aby se změny aktivovaly, zadejte

```
killall inetd
init 1
```

čímž přejdete do jednouživatelské úrovně a pak

```
init 2
```

čímž se vrátíte na standardní úroveň (nastavenou v */etc/inittab* návěštím *initdefault*).

Můžete spouštět *qmail* i ručně, ale spouštění metodou „init“ vám zajistí spuštění skriptu ve správném okamžiku – tedy po síťových skriptech a před jakýmikoliv programy, které potřebují poštovní systém (jako je například *inn*).

Smail verze 3.1

Smail 3.1 je de-facto standard pro hostitele UUCP i pro některé hostitele SMTP.. Snadno se konfiguruje, kompiluje se bez nutnosti upravovat zdrojové kódy a je dostatečně bezpečný.

Konfigurování smailu

Nainstalujte si binární soubor *smailu* z vaší distribuce Linuxu (doporučuji) nebo si obstarejte zdrojový kód programu a sestavte jej. Jestliže *smail* vytváříte ze zdrojových kódů, musíte mít v souboru *os/linux* následující řádek (aby *sed* dával skripty příkazového interpretu, které správně fungují).

```
CASE_NO_NEWLINES=true
```

Po nainstalování se do */etc/smail* zapíší konfigurační soubory (pokud používáte starší verze Linuxu, situace se může lišit). Můžeme začít s jejich editací!

Soubor „config“

```
# Odkud
smart_path=polux
smart_transport=uux

# Kam
hostname=barberouge
domains=linux.lmm.com

visible_name=barberouge.linux.lmm.com
uucp_name= barberouge.linux.lmm.com
# max_message_size=512k
# auth_domains=foo.bar
# more_hostnames=barberouge.polux.freenix.fr
```

Takže zaprvé, kdo vám poštu doručuje? U mě je to *polux* přes UUCP (tedy transport pomocí *uux*); tento soubor si přirozeně změňte podle vlastní situace. Poštu vám může doručovat například *bargw.bar.foobar.com* přes *smtp*, přičemž zde nepotřebujete transportní soubor, což definujete pomocí *-transport_file*.

Můžete také nastavit *postmaster_address* = *vaše_jméno*, pomocí *visible_name* skrýt síťovou topologii v odesílaných adresách (pokud systém funguje jako brána) a pomocí *more_hostnames* nastavit, které aliasy mají být rovněž využity pro doručovanou poštu.

Více podrobností naleznete v dokumentaci programu *smail* případně si prohlédněte příklady z */usr/doc/smail/examples*, jestli vám náhodou některý z nich nevyhovuje.

Soubor „directors“

```
# aliasinclude -  expandovat ":include:filename"  adresy, vytvořené
# aliasovými soubory. Tento a následující údaj je v podstatě
# dost běžný. K provedení větších úprav je jen málo důvodů, jediný smysl
# je nalézt a expandovat adresy ve tvaru:
#   :include:pathname
# které se mohou objevit v aliasových souborech nebo poštovních
# seznamech/předávacích souborech (vyrobených direktorem s ovladačem
# forwardfile).
aliasinclude:
  driver = aliasinclude,      # použít tuto obsluhu speciálních stavů
  nobody;                     # přiřazení uživatele nobody (nikdo)
                              # v případě narušení přístupových práv
  copysecure,                 # získání práv z alias directoru
  copyowners,                 # získání vlastníků z alias directoru

# forwardinclude - expandovat ":include:filename" adresy, vytvořené
# předávacími soubory
forwardinclude:
  driver = forwardinclude,    # použít tuto obsluhu speciálních stavů
```

```
  nobody;
  copysecure,                  # získání práv z předávacího directoru
  copyowners,                  # získání vlastníků z předávacího
                               # direktoru

# aliases - vyhledání aliasových expanzí, uložených v databázi.
# Jedná se o standardní soubor aliases. Používá se pro běžné věci,
# jako je mapování roota, postmastera, MAILER-DAEMONa a uucp
# na správce sítě, vytváření některých aliasových expanzí
# menších systémů a podobně. V konfiguraci tohoto systému je soubor
# aliases využit zejména pro aliasové a předávací informace,
# specifické pro jednotlivé stroje. Celkové předávací informace
# se vkládají do databáze "forward".
aliases:
  driver=aliasfile,    # univerzální aliasový director
  -nobody,             # všechny adresy jsou s nobody spojeny
                       # implicitně, takže toto nastavení není
                       # nutné
  sender_okay,         # neodstraňujte sender (odesilatele)
                       # z výrazů
  owner=owner-$user;   # problémy se směrují na vlastníkovu adresu
  file=/etc/aliases,
  modemask=002,        # nemělo by to být globálně zapisovatelné
  optional,            # ignoruj, pokud soubor neexistuje
  proto=lsearch,       # nesetříděný ASCII-soubor

# forward - vyhledání expanzí, uložených v předávací databázi.
# Jedná se o předávací databázi celé subdomény. Údaje jsou
# udržovány pro aktuální nebo minulé uživatele, kterým se pošta
# předává na preferované místo určení. Předávací databáze se
# po provedení změn distribuuje po síti TCP/IP, aby byla síť jednotná.
# forward:
# driver = aliasfile,          # víceúčelový aliasový director
# -nobody,                     # všechny adresy jsou s nobody spojeny
                               # implicitně, takže toto nastavení není
                               # nutné
# owner = real-$user;          # problémy se směrují na vlastníkovu adresu
# file = /etc/forward,
# modemask = 002,
# proto = dbm,                 # k přístupu využijte knihovnu dbm(3X)

# dotforward - expanduje soubory .forward v domovských adresářích
# uživatelů
# Pro uživatele, kteří mají údaj v databázi "forward", se soubor
# ".forward" využije pouze tehdy, pokud je na "domovském" stroji,
# definovaném v předávací databázi. Je-li použit, bere se jako
# seznam adres, na které má být pošta doručena, a ne jako uživatel,
# určený v lokální adrese (nebo přídavek k němu).
dotforward:
  driver = forwardfile,        # víceúčelový předávací direktor
  owner = postmaster, nobody, sender_okay;
  file = ~/.forward,           # soubor .forward v domovských adresářích
  checkowner,                  # uživatel může tento soubor vlastnit
```

```
  owners = root,              # nebo root může tento soubor vlastnit
  modemask = 002,             # nemělo by to být globálně zapisovatelné
  caution = daemon:root,      # nespouštět nic jako root nebo daemon
  # věnovat zvláštní pozornost vzdáleně přístupným domovským adresářům
  unsecure = "~uucp:/tmp:/usr/tmp:/var/tmp"

# forwardto - expanduje "Forward to " (předat na) z uživatelských
# souborů v mailboxu
# Emuluje předávací mechanismus V6/V7/System-V, který využívá řádek
# předávacích adres, uložený na začátku uživatelských souborů
# mailboxu s předponou "Forward to".
forwardto:
  driver = forwardfile,
  owner = postmaster, nobody, sender_okay;
  file = /var/spool/mail/${ lc:user} ,       # ukazuje na uživatelské
                                             # soubory mailboxu
  forwardto,                  # umožňuje funkci "Forward to "
  checkowner,                 # uživatel může tento soubor vlastnit
  owners = root,              # nebo root může tento soubor vlastnit
  modemask = 0002,            # pod System V může zapisovat group mail
  caution = daemon:root,      # nespouštět nic jako root nebo daemon
# user - odevzdá uživatelům na jejich lokálních strojích do mailboxu
user:   driver = user;        # ovladač pro zjištění uživatelského jména
  transport = local           # lokální transport směřuje do mailboxů

# real_user - nalezne uživatelská jména s předponou "real-"
# Toto je užitečné při umožnění adres, které explicitně dodávají
# do uživatelova souboru mailboxu. Sem mohou být dodány například
# chyby při expanzi souboru .forward nebo tak mohou být rozřešeny
# předávací smyčky mezi více stroji. Uživatelé mohou poštou také
# na svůj "nedomovský" stroj předávat data, přičemž využijí adresu
# ve tvaru real-přihlašovací_jméno@vzdálený.hostitel.
real_user:
  driver = user;
  transport = local,
  prefix = "real-"            # odpovídá například real-root

# lists - expanduje poštovní seznamy, uložené v adresáři list
# poštovní seznamy mohou být vytvořeny jednoduchým vytvořením
# souboru v adresáři /etc/smail/lists.
lists:  driver = forwardfile,
  caution,                    # všechny adresy označit upozorněním
  nobody,                     # a potom přiřadit uživatele nobody
  owner = owner-$user;        # stroje se system V mohou využít o-$user
                              # - owner-$user by bylo moc dlouhé
                              # na 14 znakové názvy souborů
  file = lists/${ lc:user}    # lists je pod $smail_lib_dir

# owners - expanduje poštovní seznamy, uložené v adresáři list owner
# seznamy vlastníků mohou být vytvořeny vytvořením souboru
# v adresáři/etc/smail/lists/owner. Vlastníkům seznamů jsou posílány
# lokálně vzniklé chyby při práci se seznamy stejného názvu. Seznam
# vlastníků pro poštovní seznam vytvoříte souborem s názvem seznamu
```

```
# v /etc/smail/lists/owner. Tak se vytvoří seznam adres vlastníků
# daného seznamu, jak jej výše používá direktor "lists".
owners: driver = forwardfile,
  caution,                    # všechny adresy označit upozorněním
  nobody,                     # a potom přiřadit uživatele nobody
  owner = postmaster;         # stroje se system V mohou využít o-$user
                              # - owner-$user by bylo moc dlouhé
                              # na 14 znakové názvy souborů
  prefix = "owner-",
  file = lists/owner/${ lc:user}    # lists je pod $smail_lib_dir

# request - expanduje poštovní seznamy, uložené v adresáři list
# request seznamy požadavků mohou být vytvořeny vznikem souboru
# v adresáři /etc/smail/lists/request. Zde vypsané adresy se
# využívají jako standardní adresy pro dotazy k poštovním seznamům.
# Například žádosti o přidání nebo smazání ze seznamu budou odesílány
# na "list-request", což je nutné nastavit na předání k příslušným
# osobám.
request: driver = forwardfile,
  caution,                    # všechny adresy označit upozorněním
  nobody,                     # a potom přiřadit uživatele nobody
  owner = postmaster;         # stroje se system V mohou využít o-$user
                              # - owner-$user by bylo moc dlouhé
                              # na 14 znakové názvy souborů
  suffix = "-request",
  file = lists/request/${ lc:user}  # lists je pod $smail_lib_dir
```

Tady nemusíte nic měnit, snad jen volbu pro poštovní konference, jestliže ji v *smail*u budete využívat, nebo volbu forwards, jestliže budete chtít například znemožnit přeposílání pošty.

Soubor „fidopaths“

```
.f105.n324.z2.fidonet.org f105.n324.z2.fidonet.org!%s
.n324.z2.fidonet.org f105.n324.z2.fidonet.org!%s
.z2.fidonet.org f105.n324.z2.fidonet.org!%s
.fidonet.org f105.n324.z2.fidonet.org!%s
```

Tento soubor vytvořte pouze pokud používáte *ifmail* a FIDO.

Soubor „routers“

```
# forces - nastavit určité cesty
# Tato databáze existuje z důvodu pevného nastavení cest na různé stroje
# nebo domény. Využívá se při vytváření dočasných převodů k dalším
# směrovacím databázím. Změny provedete editací souboru
# maps/force.path a zadáním make v adresáři maps/.
forces:
  driver = pathalias,               # směrovač, prohledávající
                                    # soubor paths
  method = /etc/smail/maps/table;   # přenosy jsou v tomto souboru
  file = forcepaths,                # soubor, obsahující informace
                                    # o pevných cestách
  proto = lsearch,                  # použití setříděného souboru
  optional,
```

```
  reopen                                # zavřít, když se nepoužívá

uucp_neighbors:
  driver=uuname,                        # použít program, který vrací sousedy
  transport=uux;
  cmd="/usr/bin/uuname -a",             # konkrétně program uuname
# domain=uucp                           # oddělit koncovku .uucp

# smart_host - částečně určený direktor smarthost
# Jestliže je atribut smart_path v souboru config definován jako
# cesta z lokálního stroje na vzdálený, budou všechny stroje, které
# neodpovídají ničemu jinému, odeslány na zadaný vzdálený stroj.
# Atribut smart_transport je možné použít k určení jiného
# transportu. Jestliže není určen atribut smart_path, tento router
# je ignorován.
smart_host:
  driver = smarthost,                   # ovladač speciálních stavů
  transport = uux             # implicitně doručovat přes UUCP
# path=phreak

# ifmail - posílání pošty na fidonet a obráceně
ifmail:
  driver=pathalias,
  transport=ifmail;
  file=fidopaths,
  proto=lsearch
```

Část *ifmail* byste měli uvést pouze v případě, že pro přenos pošty z FIDO používáte *ifmail*. Všimněte si, že je možné změnit přenosový režim z *uux* (přes UUCP) například na *smtp* nebo je dokonce možné vypsat cesty k různým strojům nebo doménám v */etc/smail/maps/table*.

Soubor „transports“

```
# local - dodání pošty lokálním uživatelům
# Smail se má připojit přímo na soubory uživatelských mailboxů
# ve /var/spool/mail
#local: driver = appendfile,          # připojit zprávu k souboru
#       -return_path,         # začlenit pole Return-Path:
#       local,                        # při předání využít lokální formy
#       from,                         # dodat řádek obálky From_
#       unix_from_hack;               # v těle vložit před From >
#
#       file = /var/spool/mail/${ lc:user}, # použít toto umístění pro Linux
#                                           # Všimněte si, že mail spool musí
být 1777
#       file = ~/mailfile,            # toto umístění zajišťuje lepší zabezpe-
čení
#       group = mail,                 # skupina vlastnící soubor v System V
#       mode = 0660,                  # pod System V má skupina mail přístup
#       suffix = "\ n",       # připojení nového řádku navíc
#       append_as_user,

# Takto může mít každý uživatel soubor ~/.procmailrc, kterým
# řídí filtrování pošty a její ukládání z poštovních seznamů
```

```
# v oddělených mailboxech (podle potřeby).
local:  +inet,
        -uucp,
        driver = pipe,                  # připojit zprávu do souboru
        return_path,                    # začlenit pole Return-Path:
        local,                          # při předání využít lokální formy
        from,                           # dodat řádek obálky From_
        unix_from_hack;                 # v těle vložit před From >

        cmd = "/usr/bin/procmail",      # použít procmail k lokálnímu
                                        # předání
        parent_env,                     # info o prostředí z nadřazeného
                                        # adresáře
        pipe_as_user,                   # použít user-id, asociovaný
                                        # s adresou
        umask = 0022,                   # umask pro dceřiný proces
#       -ignore_status,                 # návratový kód by měl být akceptován
#       -ignore_write_errors,           # navazovat na broken pipes

# pipe - dodání pošty příkazům shellu
# To se využívá implicitně, když smail zjistí adresy, začínající
# svislou čarou, jako "|/usr/lib/news/recnews talk.bizarre".
# Svislá čára se z adresy odstraňuje před předáním k transportu.
# pipe:  driver = pipe,                 # předání zprávy jinému programu
#        return_path, local, from, unix_from_hack;
#
#        cmd = "/bin/sh -c $user",      # odeslání adresy do Bourne shellu
#        parent_env,                    # info o prostředí z nadřazeného
                                        # adresáře
#        pipe_as_user,                  # použít user-id, asociovaný s adresou
#        umask = 0022,                  # umask pro dceřiný proces
#        -log_output,                   # nelogovat stdout/stderr
#        ignore_status,                 # na návratový kód exit se nehledí
#        ignore_write_errors,           # ignorování broken pipes

# file - dodání pošty souborům
# To se využívá implicitně, když smail zjistí adresy, začínající
# lomítkem nebo tildou, jako "/usr/info/list_messages" nebo
# "~/Mail/inbox".
# file:  driver = appendfile,
#        return_path, local, from, unix_from_hack;
#
#        file = $user,                  # soubor se vezme z adresy
#        append_as_user,                # použít user-id, asociovaný s adresou
#        expand_user,                   # expandovat ~ a $ v adrese
#        check_path,
#        suffix = "\ n",
#        mode = 0644

# uux - předání programu rmail na vzdáleném hostiteli UUCP
#
# Při jednom UUCP-převodu bude na vzdálenou lokaci předáno až
# 5 adres.
```

```
uux:    driver = pipe,
        -uucp,
        inet,
#       uucp,                   # použít adresu ve formátu UUCP
        from,                   # dodat řádek From_
        max_addrs = 5,          # maximálně 5 adres na jedno vyvolání
        max_chars = 200;        # maximálně 200 znaků adres
# přepínač -r zabraňuje okamžitému dodání, závorky kolem proměnné
# $user zabraňují speciální interpretaci v uux.
        cmd = "/usr/bin/uux - -r -g$grade $host!rmail $(($${ strip:user} )$)",
#       cmd="/usr/bin/uux - $host!rmail $(($user)$)",
        ignore_write_errors,          # ignorování broken pipes
        umask = 0022,
#       pipe_as_sender,

# uux_one_addr - dodání pošty přes UUCP na vzdálenou lokaci, která
# zvládá v jednom okamžiku jednu adresu
# Toto je často nutné při dodávání na lokaci s neupravenou verzí
# 4.1BSD.

uux_one_addr:
     driver = pipe,
     uucp,                      # použít adresu ve formátu UUCP
     from;                      # dodat řádek From_
# přepínač -r zabraňuje okamžitému dodání
     cmd = "/usr/bin/uux - -r -g$grade $host!rmail ($${ strip:user} )",
     umask = 0022,
     pipe_as_sender

queueonly:
    driver = pipe;              # odeslat zprávu na rouru
    cmd = "/usr/lib/sendmail -Q -f $sender -bm $user",
                                # použít getmail pro lokální doručení
    user=root,                  # spustit getmail jako "root"
    group=mail,                 # spustit getmail jako "mail"
    parent_env,                 # info o prostředí z nadřazeného adresáře
    -pipe_as_user,              # použít user-id, asociovaný s adresou
    umask = 0007,               # umask pro dceřiný proces

# dodání zprávy. smtp transport se začlení, pouze pokud existuje síť
# BSD. Pro přenosy v zóně UUCP je možné určit atribut uucp. Při
# přenosech na Internetu je nutné nastavit atribut inet.
# POZNÁMKA: Toto je optimální, ke zvládnutí více zpráv v jednom
# připojení by měla existovat nástavba.
# TAKÉ: Možná bude nutné omezit max_addrs na 100, protože toto je
# nejmenší počet, který standard SMTP vyžaduje implementovat.
smtp:   driver=tcpsmtp,
        inet,                           # jestliže UUCP_ZONE není definováno
#       uucp,                           # jestliže UUCP_ZONE je definováno
        -max_addrs, -max_chars;         # žádné omezení v počtu adres

        short_timeout=5m,               # timeout pro krátké operace
```

```
        long_timeout=2h,              # timeout pro delší operace smtp
        service=smtp,                 # připojení na tento servisní port
# Při použití na Internetu: zrušte komentář u následujících 4 řádků
       use_bind,              # rozlišení MX a násobných A záznamů
       defnames,              # použití standardního vyhledávání domén
       defer_no_connect,      # nový pokus, je-li nameserver vypnutý
       local_mx_okay,         # selhat MX na lokálního hostitele

ifmail:
       from,received,max_addrs=5,max_chars=200,
       driver=pipe;
       pipe_as_sender,
       cmd="/usr/local/bin/ifmail -x9 -r$host $(($( strip:user} )$)"
```

Část *ifmail* můžete zahrnout pouze pokud pro doručování pošty z FIDO používáte *ifmail*. Mimo toho by nemělo být zapotřebí v souboru jakkoliv měnit konfiguraci přenosových agentů (*uux*, *smtp*...). Tyto parametry určíte v jiných konfiguračních souborech.

Povšimněte si, že některé části (jako *pipe* nebo *file*) jsou zakomentované kvůli zvýšení bezpečnosti.

Adresář „maps/“

Obsahuje soubory *map* a *table*:

Nejprve soubor *map*

```
#N   foo.bar foo2.bar2
#S   AT 486/RedHat Linux 1.2.13
#O   organizace
#C   kontakt
#E   administrace (email)
#T   telefon
#P   adresa
#R
#U   hostitelé, připojení přes uucp
#W   vytvořeno/editováno kým
#
hname polux

hname linux.eu.org

hname = polux
hname = polux.linux.eu.org
```

Tento soubor si opět upravte podle vlastní situace (já poštu dostávám z *polux.linux.eu.org*).

Nyní soubor *table*:

```
*      uux
```

Zde můžete určit různý způsob transportu do různých cílů. Například pro lokální síť určíte „smtp“ a pro zbytek světa „uux“ (přes UUCP) nebo obráceně (já využívám UUCP pro veškerou odesílanou poštu, proto jsem použil „*“).

Další dobré příklady

Předchozí soubory skutečně využívám, takže by neměly nastat problémy s jejich využitím jako základ pro vlastní soubory.

Následující soubory nabízím pouze jako příklady konfigurace *smail*u jiným způsobem.

```
#ident "@(#) transports,v 1.2 1990/10/24 05:20:46 tron Exp"

# Kompletní popis obsahu tohoto souboru viz smail(5).

# local - předání pošty lokálním uživatelům
#
# Smail se má připojit přímo na soubory uživatelských mailboxů
# v /usr/mail
local: driver = appendfile,          # připojit zprávu do souboru
       return_path,                  # začlenit pole Return-Path:
       local,                        # při předání využít lokální formy
       from,                         # dodat řádek From_
       unix_from_hack;               # v těle vložit > před From
       file = /usr/mail/${ lc:user} ,# použít toto umístění pro System V
       group = mail,                 # skupina, vlastnící soubor pro System V
       mode = 0660,                  # pod System V má skupina mail přístup
       suffix = "\ n",        # připojení nového řádku navíc
       append_as_user,

# pipe - dodání pošty příkazům shellu
#
# Využívá se implicitně, když smail zjistí adresy, začínající
# svislou čarou, jako "|/usr/lib/news/recnews talk.bizarre".
# Svislá čára se z adresy odstraňuje před předáním k transportu.
pipe:  driver = pipe,                # předání zprávy jinému programu
  return_path, local, from, unix_from_hack;

  cmd = "/bin/sh -c $user",   # odeslání adresy do Bourne Shell
  parent_env,                 # info o prostředí z nadřazeného adresáře
  pipe_as_user,               # použít user-id, asociovaný s adresou
  umask = 0022,               # umask pro dceřiný proces
  -log_output,                # nelogovat stdout/stderr
  ignore_status,              # na návratový kód se nehledí
  ignore_write_errors,        # ignorování broken pipes

# file - dodání pošty souborům
#
# Využívá se implicitně, když smail zjistí adresy, začínající
# lomítkem nebo tildou, jako "/usr/info/list_messages" nebo
# "~/Mail/inbox".

file:  driver = appendfile,
       return_path, local, from, unix_from_hack;

       file = $user,          # soubor se vezme z adresy
       append_as_user,        # použít user-id, asociovaný s adresou
       expand_user,           # expandovat ~ a $ v adrese
       suffix = "\ n",
       mode = 0644
```

```
# uux - předání programu rmail na vzdálenou UUCP lokaci
#
# Při jednom UUCP přenosu bude na vzdálenou lokaci předáno až
# 5 adres.
uux:    driver = pipe,
        uucp,                   # použít adresové formuláře ve stylu UUCP
        from,                   # dodat řádek From_
        max_addrs = 5,          # maximálně 5 adres na jedno volání
        max_chars = 200;                # maximálně 200 znaků adres

        # přepínač -r zabraňuje okamžitému dodání, závorky kolem proměnné
        # $user zabraňují speciální interpretaci v uux.
        cmd = "/usr/bin/uux - -r -g$grade $host!rmail $(($${ strip:user} )$)",
        umask = 0022,
        pipe_as_sender

# uux_one_addr - dodání pošty přes UUCP na vzdálenou lokaci, která
# zvládá v jednom okamžiku jednu adresu
#
# Toto je často nutné při dodávání na lokaci s neupravenou verzí
# 4.1BSD.
uux_one_addr:
        driver = pipe,
        uucp,                   # použít adresové formuláře ve stylu UUCP
        from;                   # dodat řádek From_

        # přepínač -r zabraňuje okamžitému dodání
        cmd = "/usr/bin/uux - -r -g$grade $host!rmail ($${ strip:user} )",
        umask = 0022, pipe_as_sender

# demand - dodat vzdálenému programu rmail, který podává žádost
demand: driver = pipe,
        uucp, from, max_addrs = 5, max_chars = 200;

        # bez přepínače -r se bude vzdálená lokace kontaktovat okamžitě
        cmd = "/usr/bin/uux - -g$grade $host!rmail $(($user)$)",
        umask = 0022, pipe_as_sender

# uusmtp - dodat programu rsmtp na vzdálené hostitele UUCP
#
# Dodat pomocí jednoduchého dávkového protokolu SMTP na vzdálený stroj.
# Umožňuje využít více libovolných adres. Odstraňuje také omezení
# adresátů na jedno vyvolání uux.
uusmtp: driver = pipe,
        bsmtp,                          # odeslat dávku SMTP příkazů
        -max_addrs,                     # není zde žádné omezení počtu nebo
        -max_chars;                     # celkové velikosti adres adresátů

        # přepínač -r zamezuje okamžitému dodání, adresáti jsou uloženi
        # v datech, odesílaných na standardní výstup rsmtp.
        cmd = "/usr/bin/uux - -r -g$grade $host!rsmtp",
        umask = 0022, pipe_as_sender
```

```
# demand_uusmtp - dodat vzdálenému programu rsmtp, který podává
# žádost
demand_uusmtp:
        driver = pipe,
        bsmtp, -max_addrs, -max_chars;

        # bez přepínače -r se bude vzdálený hostitel kontaktovat okamžitě
        cmd = "/usr/bin/uux - -g$grade $host!rsmtp",
        umask = 0022, pipe_as_sender

# smtp - dodání pomocí SMTP přes TCP/IP
#
# Připojení ke vzdálenému hostiteli pomocí TCP/IP a inicializace
# SMTP-konverzace o dodání zprávy. SMTP transport je začleněn pouze
# s existující sítí BSD.

# POZNÁMKA: Možná bude nutné omezit max_addrs na 100, protože toto
# je nejnižší počet, který musí každá implementace SMTP
# v jedné relaci obsloužit.
smtp:   driver = smtp,
        -max_addrs,
        -max_chars

#ident "@(#) table,v 1.2 1990/10/24 05:20:31 tron Exp"

# Tento soubor vyjmenovává transporty, které se při dodání na
# specifické lokace z bargw využijí.

#lokace       transport
#--------     ---------
curdsgw       demand_uusmtp   # pomocí dávkového SMTP
oldbsd        uux_one_addr    # lokace s 4.1BSD nemohou vzít > 1 adresu
sun           demand          # když odesíláte poštu na sun, volejte jej
*             uux             # u všeho ostatního intervalové zpracování
```

Restart démona inetd

Aby *smail* fungoval jako SMTP démon, přidejte do */etc/inetd.conf* jeden z následujících řádků:

```
smtp stream tcp nowait root /usr/bin/smtpd smtpd
```

nebo

```
smtp stream tcp nowait root /usr/sbin/tcpd /usr/sbin/in.smtpd
```

Odesílaná pošta se bude při použití *elm*u odesílat automaticky.

Smail a SMTP

Poskytovatelé většinou používají SMTP, takže s odesíláním pošty byste neměli mít žádné problémy. Jestliže je vaše připojení na Internet v době odeslání pošty nefunkční, pošta se uschová ve */var/spool/smail/input*. Při obnovení spojení se spouští příkaz *runq*, který poštu odešle. S obdržením pošty je ale v takovém případě problém, protože váš poskytovatel Internetu nemá na starosti jen vás, ale i mnoho jiných klientů!

Vaši poštu je obvykle možné zachránit pomocí protokolu POP, viz níže.

Sendmail 8.x

Sendmail 8.7.x z Berkeley byl poslední větší inovací od dob sendmail5. Měl nádhernou zabudovanou podporu pro kompilaci v Linuxu: „make linux“ a všechno je nastavené.

Nejlépe vám pravděpodobně poslouží nějaká binární podoba programu, kterou si přetáhnete z některých linuxových archivních serverů. Je to lepší než se potýkat například s Berkeley dbm.

Na `ftp://ftp.ibiblio.org/pub/Linux/system/Mail/delivery/sendmail-8.6.12-bin.tgz` naleznete skvělou distribuci sendmailu 8.6.12 od Jasona Haara (*j.haar@lazerjem.demon.co.uk*), která obsahuje dokumentaci ke zdrojovým textům a pěkný popis rychlého nastavení sendmailu v8 při běžných konfiguracích.

Ukázkový soubor mc pro 8.7.x

Podobně jako sendmail+IDA využívá sendmail v8 m4 pro převod konfiguračního souboru na plný sendmail.cf, využívaný sendmailem. Následuje můj aktuální mc soubor, využívaný na mém hostiteli (PPP na Internet pro odesílanou poštu, UUCP pro doručenou poštu).

```
dnl divert(-1)
#-----------------------------------------------------------------
#
# tohle je soubor .mc pro linuxového hostitele, nastaveného následovně:
#
#      - připojen na Internet pro odchozí poštu přes ppp
#      - připojen přes UUCP pro příchozí poštu
#      - doménové hlavičky
#      - žádný lokální doručovatel (místo toho 'deliver')
#      - žádný běžící DNS, takže nic se nekanonizuje
#      - veškerá nelokální výchozí pošta jde na RELAY_HOST přes SMTP
#        (používáme ppp a necháváme našeho poskytovatele, aby se staral)
#
#                                        vds 3/31/95
#
#-----------------------------------------------------------------

include('../m4/cf.m4')
VERSIONID('linux nodns relays to slip service provider smarthost')dnl
Cwmyhostname.myprimary.domain myhostname.UUCP localhost
OSTYPE(linux)
FEATURE(nodns)dnl
FEATURE(always_add_domain)dnl
FEATURE(redirect)
FEATURE(nocanonify)
dnl MAILER(local)dnl
MAILER(smtp)dnl
MAILER(uucp)dnl
define('RELAY_HOST', smtp:my.relay.host.domain)
define('SMART_HOST', smtp:my.relay.host.domain)
define('UUCP_RELAY', smtp:my.relay.host.domain)
define('LOCAL_MAILER_PATH', '/bin/deliver')
define('LOCAL_MAILER_ARGS', 'deliver $u')
```

Lahůdky v sendmailu v8

Existuje několik rozdílů, které předpokládám u uživatelů IDA. Zatím jsem se setkal s tímto: místo „runq“ pro spuštění fronty zadáváte „sendmail -q“.

Agenti pro lokální doručování

Na rozdíl od většiny operačních systémů neměl Linux přímo zabudovanou podporu pošty: potřebovali jste program, který by ji lokálně doručoval (například „lmail“, „procmail“ nebo „deliver“).

Nyní již každá nová distribuce obsahuje lokálního doručovatele!

Dokumentaci k jejich využití ve vaší distribuci naleznete v binární podobě sendmail5.67b+IDA1.5 (na adrese uvedené výše).

Uživatelské doručovací programy

Mutt

S překladem, instalací a spuštěním programu mutt byste neměli mít problémy. Uživatelé programu qmail mohou buď použít záplatu nebo jej mohou spustit s parametrem –f, který zajistí čtení z lokální poštovní složky.

Pokud vás bude po aktualizaci distribuce mutt obtěžovat chybovým hlášením „unknown terminal error“, znovu jej přeložte.

Elm

Elm se pod Linuxem kompiluje, instaluje a spouští bez problémů. Více informací naleznete ve zdrojových kódech elmu a v instalačních instrukcích. Elm a filter musí mít práva 2755 (skupina mail), /var/spool/mail 775 a skupina mail.

Uživatelé programu qmail mohou použít záplatu umožňující přístup k podivným funkcím programu qmail nebo mohou elm spustit s parametrem –f a získat tak přístup k lokální poštovní složce.

Je nutné si dávat pozor na to, že pokud máte Elm přeložen s podporou MIME, musíte mít nainstalován a v cestě nastaven metamail, jinak Elm nebude schopen číst poštu došlou ve formátu MIME. Metamail je k dispozici na `thumper.bellcore.com` a samozřejmě také službou Archie.

Jestliže používáte binární distribuci, budete muset vytvořit soubor „/usr/local/lib/elm/elm.rc“, čímž přepíšete vkompilovaný název serveru a informace o doméně:

- nahraďte „subdomain.domain“ názvem vaší domény
- nahraďte „myhostname“ názvem vašeho serveru (bez názvu domény)

```
#---------- /usr/local/lib/elm/elm.rc ------------------
#
# toto je nekvalifikovaný název hostitele
hostname = myhostname
#
# toto je lokální doména
hostdomain = subdomain.domain
#
# toto je plně kvalifikovaný název hostitele
hostfullname = myhostname.subdomain.domain
#
#-------------------------------------------------------
```

Do kategorie „příliš dobré, než aby to byla pravda“ spadá distribuce Elm-2.4.24, která podporuje PGP. Naleznete ji na adrese *ftp://ftp.viewlogic.com/pub/elm-2.4pl24pgp3.tar.gz,* což je elm2.4.24 s přidanou podporou PGP. Konfiguruje se a instaluje stejně jako normální Elm, což pravděpodobně znamená přidání výše zmíněných dodatků. Za zmínku stojí, že já tuto verzi používám a jsem s ní spokojen. Samozřejmě, že k dispozici je jistě i mnoho novějších verzí, včetně elm-ME+.

Následující věc sice není specifická pro Linux, ale často je považována (neprávem) za chybu v Elmu. Slyšeli jsme, že Elm někdy padá a hlásí, že nemůže alokovat v paměti nějaký vysoký počet bajtů. Náprava je v odstranění zpracovaných globálních poštovních přezdívek (aliases.dir a aliases.pag).

TOHLE ALE NENÍ CHYBA ELMU, je to chyba v konfiguraci Elmu, kterou prováděl ten, od koho máte binární distribuci.

Elm má rozšířený a nekompatibilní formát přezdívek; musíte zajistit, aby cesta, kterou Elm pro přezdívky využívá, byla odlišná od cesty, kterou využívá sendmail/smail. Vzhledem k množství zpráv o tomhle problému je zřejmé, že alespoň jedna z větších distribucí „z ulice“ byla špatně nakonfigurována (podle informací od Scota W. Stevensona, *scot@catzen.gun.de*).

Aktuální balík metamailu vyžaduje pro některé skripty csh. Nemáte-li csh (nebo tcsh), objeví se zajímavé chyby...

Mailx

Ušetřete si námahu: sežeňte si ze Slackware verze 2.1.0 nebo pozdější mailx kit, který obsahuje pěknou implementaci mailx5.5. Jestliže chcete kompilovat ze zdrojů, mailx v5.5 se v Linuxu kompiluje bez dodatků, pokud máte nainstalován „**pmake**“**.** Jestli je to ještě aktuální, doporučuji náhradu starého „edmailu“ ze SLS1.00 za mailx.

Vzdálená práce s poštou

Tato část popisuje vzdálené zpracování pošty protokoly POP a IMAP.

Další možností je pomocí nfs připojit na klientské stanice diskovou oblast spool (to ovšem není bezpečné, protože by všichni museli používat stejný mechanismus zámků) nebo použití webového rozhraní pro přístup k poště (což je dnes velmi oblíbené řešení).

Historie

Na síti s pracovními stanicemi představovala pošta vždycky problém:

- Buď jste používali adresy typu *user@computer.foo.com*, což přinášelo problémy pokud byl „*computer*“ vypnutý, navíc to všem prozrazovalo strukturu vaší sítě a nutilo vás to používat různé adresy podle toho, na kterém počítači jste se přihlásili.
- Druhá možnost byla použít „hlavní počítač“, *mailhost.foo.com* s přepisovacími pravidly, takže všichni uživatelé měli jednotné adresy bez ohledu na to, že pracovali na různých počítačích.

Jak si ale v takovém případě přečíst poštu?

Použít rsh a elm?

To by náš poštovní uzel přetěžovalo! Další možnost bylo předávat poštu pomocí UUCP, SMTP a podobně, jenže to je moc složité.

Pak se objevily protokoly POP a IMAP (oba s počátečními bezpečnostními problémy, které ale byly díky ssh odstraněny). Vyžadují sice specifické nastavení lokálních poštovních programů, celkový přístup k poště se ale výrazně usnadnil.

Čtení pošty

Hlavní nevýhody protokolu POP jsou:

- heslo se po síti posílá nešifrovaně
- musíte používat poštovní program, který protokol POP umí; ovšem takových je dnes hodně (například Pine, Emacs, Mozilla, Netscape, Mutt, Internet Explorer, Pegasus, Eudora, Claris, ...)
- pokud se uživatel stěhuje (čte si poštu na různých počítačích), je docela nepříjemné pokud pošta zůstává vždy na té stanici, odkud byla vyzvednuta
- některé servery protokolu POP (například qpopper, ipop3d) a obecně jakýkoliv hodně zatížený server může mít značné nároky na systém. Zvažte nastavení různých voleb (například nenechávat poštu na serveru), použijte jiný poštovní server (například cucipop) a nespouštějte jej prostřednictvím inetd.

Problém s hesly je možné vyřešit vytvořením šifrovaného kanálu na němž POP poběží, nebo pomocí doplňků APOP nebo RPOP. Problémy se čtením pošty je možné vyřešit změnou poštovního programu (nepodceňujte problémy spojené s přeškolením uživatelů na něco jiného) nebo pomocí automatického stahovacího programu vedle normálního poštovního programu.

Lepší alternativou k protokolu POP je protokol IMAP, který by měl být použit zejména při vzdáleném přístupu. Použití protokolu POP by mělo být omezeno na lokální síť, kde není odchycení hesla tak riskantní. (Mark Aitchison navrhl řešení pomocí souborů hosts.deny a hosts.allow, viz dokument Net-3 HOWTO, nicméně k tomu musíte démona protokolu spouštět přes inetd.)

Zda poštu ponechávat nebo neponechávat na serveru závisí na dostupné diskové kapacitě, na druhé straně se tím usnadňuje zálohování a zabezpečení pošty i práce na různých počítačích. Volba řešení závisí na konkrétních podmínkách. Nezajistíte tak, že vaši poštu nikdo nepřečte, nicméně alespoň vám ji nikdo nebude moci smazat. Takže pokud používáte poštu chráněnou pomocí PGP, je to nejlepší řešení.

Dále uvádíme některé POP programy, které stojí za vyzkoušení:

- gwpop (a Good Way to POP) je dobře zabezpečený, protože vytváří šifrovaný kanál a ukládá poštu přímo do adresáře spool, nicméně potřebuje ke své práci Perl.
- popclient, velmi jednoduché použití. Pokud se například přihlašujete jako *john* a máte heslo *PrettySecret*, spustíte:

  ```
  $ popclient -3 -v mail.acme.net -u john -p
      "PrettySecret" -k -o JOHN-INET-MAIL
  ```

  Toto řešení nedoporučujeme v případě víceuživatelského počítače, protože kdokoliv bude schopen přečíst vaše heslo, například příkazem „ps –auxv“.

- fetchmail, který je značně podporován a velmi snadno se konfiguruje: konfiguruje se v souboru `~/.fetchmailrc` a spouštíte jej tedy pouze tehdy, pokud chcete převzít poštu. Můj soubor `.fetchmailrc` vypadá takto:

  ```
  poll mail.server protocol pop3:
  forcecr
  password PrettySecret;
  ```

Nezapomeňte provést „chmod 600 .fetchmailrc“, jinak vás o to fetchmail sám požádá. Volba *forcecr* je nutná pro použití s programem qmail, který striktně dodržuje RFC.

Poslání pošty

Nejprve potřebujete nějaký poštovní program ovládající SMTP, například qmail, smail, vmail nebo mozilla (tento program umí všechno, čtení pošty, příjem přes POP a odesílání přes SMTP).

V některé z předcházejících částí si najděte jak nainstalovat ten, který se vám líbí. Až jej budete testovat, zkuste poslat poštu na nějaký lokální účet na „hlavním“ poštovním systému.

Čtení pošty

Pokud váš oblíbený program neumí všechno sám, můžete si nainstalovat elm, pgp, mush, pine ... existuje řada dobrých programů, které jsou volně dostupné.

Testování

Abyste ověřili zda hlavní poštovní systém podporuje protokol POP, zkuste

```
$ telnet mailhost 110
```

Pokud to funguje, objeví se hlášení typu „*OK Pop server (...) starting*“. Zadejte „quit“!

K instalaci bezpečného kanálu prostřednictvím ssh nejprve vyzkoušejte

```
$ ssh mailhost date
```

Pokud se objeví datum, mělo by být všechno v pořádku. ssh vás nepožádá o heslo, takže budete muset na poštovním serveru vytvořit soubor „.shosts“, obsahující jméno klienta. K ověření přesměrování na port ssh (které používá gwpop) zadejte

```
$ ssh -n -f -L 12314:localhost:110 mailhost sleep 30

a pak
$ telnet localhost 12314
```

Měli byste vidět hlášení POP serveru na hlavním poštovním počítači. Pokud nepoužíváte ssh, nezapomeňte ve skriptu gwpop zakomentovat $ssh. Zda běží procmail ověříte příkazem „procmail –v“.

Použití

Dále byste měli upravit Perl skript programu gwpop, zkontrolovat zda je všechno v pořádku a pak spustit gwpop:

```
$ gwpop -v your-username
POP password on mailhost: yoursecretpasword
```

Pokud se neobjeví žádné chybové hlášení, měla by se vaše pošta na hlavním poštovním systému stáhnout na váš lokální počítač (ověřte si to!).

Kromě toho můžete gwpop spustit jako démona:

```
$ gwpop -d $HOME/tmp your-username
```

Zprávy programu gwpop se pak zapisují do syslogu a program běží trvale. Signál HUP jej přinutí k okamžitému převzetí pošty.

Výše popsané poštovní programy můžete získat na adresách:

```
ftp://ftp.unina.it/pub/Unix/pkgs/network/mail/gwpop
ftp://ftp.informatik.rwth-aachen.de/pub/packages/procmail
http://www.cs.hut.fi/ssh/
```

Poděkování

Následující lidé přispěli svými informacemi a zkušenostmi, čímž napomohli dokončení tohoto dokumentu:

Steve Robbins, Ian Kluft, Rich Braun, Ian Jackson, Syd Weinstein, Ralf Sauther, Martin White, Matt Welsh, Ralph Sims, Phil Hughes, Scot Stevenson, Neil Parker, Stephane Bortzmayer a zvláštní díky patří Vince Skahanovi za jeho jedinečnou spolupráci.

Eric S. Raymond tento dokument pročetl, opravil nějaké chyby a doplnil sekci „Jak funguje elektronická pošta“ ze svého dokumentu Mail User's HOWTO.

Hitoshi Hayakawa kontrolovat část věnovanou qmailu, Jun Morimoto dodal různé poznámky k programům popclient a fetchmail a Takeo Nakano provedl kontrolu pravopisu :-).

Jestliže jsem na někoho zapomněl, omlouvám se: stačí mi poslat email!

KAPITOLA 4

Autor: Eric Steven Raymond, Thyrsus Enterprises (http://www.tuxedo.org/~esr/), esr@thyrsus.com,

Pošta z pohledu uživatele

Tento dokument je úvodem do problematiky elektronické pošty pod operačním systémem Linux. Je zaměřen především na běžné uživatele a typické konfigurace Linuxu určené pro domácí použití a pro použití v menších organizacích. Tento dokument byste si měli prostudovat v případě, kdy plánujete využívat lokální či vzdálenou komunikaci prostřednictvím elektronické pošty. Tento dokument naopak nemusíte číst v případě, kdy si nevyměňujete elektronickou poštu s ostatními uživateli na vašem systému ani s ostatními počítači. Informace o nastavení a administraci prostředí pro elektronickou poštu naleznete v dokumentu Pošta z pohledu administrátora.

Úvod

Tento dokument obsahuje informace o tom, jak funguje elektronická pošta a odpovědi na často kladené dotazy ohledně programového vybavení určeného pro podporu elektronické pošty pod operačním systémem Linux.

Většina současných distribucí Linuxu nabízí snadno použitelné a přednastavené konfigurace pro práci s elektronickou poštou, obvykle tyto distribuce vycházejí z nejnovější verze programu sendmail-v8. Tento dokument předpokládá, že máte k dispozici správně nastavené a plně funkční připojení k Internetu.

Informace o tom, jak správně nastavit připojení pomocí protokolů PPP či SLIP najdete v dokumentu Hookup HOWTO (http://www.linuxdoc.org/HOWTO/ISP-Hookup_HOWTO.html).

Z tohoto důvodu je tento dokument, na rozdíl od verzí 1.x od Vince Skahana, zaměřen na uživatelské problémy a architekturu. Byly tak odstraněny podrobné technické informace o UUCP, IDA sedmail a některá další dříve významná témata.

Nové verze tohoto dokumentu

Anglická podoba tohoto dokumentu je rozesílána jednou měsíčně prostřednictvím diskusní skupiny comp.os.linux.answers. Aktuální verze by také měla být k dispozici na adrese http://www.linuxdoc.org/HOWTO/ISP-Hookup_HOWTO.html.

Hardwarové nároky na programy pro elektronickou poštu

Podpora elektronické pošty nevyžaduje žádné další speciální hardwarové vybavení. Máte-li k dispozici hardware zajišťující funkční připojení k Internetu, bude tento hardware dostačují také pro práci s elektronickou poštou.

Kde najít programy pro elektronickou poštu?

Programové vybavení určené pro práci s elektronickou poštou pravděpodobně najdete ve vámi použité distribuci Linuxu. Opravy a nové verze můžete najít také v archivu *ibiblio Linux Archive* (http://www.ibiblio.org/pub/Linux).

Poštovní klienti

Tato kapitola je věnována poštovním klientům (MUA – Mail User Agent), tedy programům umožňujícím uživatelům používat elektronickou poštu (prohlížet došlé zprávy, vytvářet nové atd.). Tyto programy využívají tzv. přenosové agenty (MTA – Mail Transfer Agent) popsané v dokumentu Pošta z pohledu administrátora, který se také zabývá nastavováním poštovních klientů a obsahuje celou řadu tipů a rad pro řešení problémů, se kterými se administrátoři mohou setkat.

Nastavení editoru zpráv

Poštovní klienti používají pro vytváření zpráv některý z dostupných editorů. Jaký editor je implicitní záleží na konkrétním poštovním klientovi. Hodně z nich dodržuje konvenci zavedenou v počátcích unixů – použitý editor je určen obsahem systémové proměnné `VISUAL`. Není-li tato proměnná definovaná, bere se do úvahy proměnná `EDITOR`.

Velmi často proměnná `EDITOR` nabývá hodnot **vi** a **emacs**. Pokud patříte mezi uživatele, kteří mají neustále spuštěné prostředí Emacs, je velmi užitečné nastavit proměnnou `EDITOR` na hodnotu **emacsclient**. Toto nastavení používejte v kombinaci s následujícími řádky v konfiguračním souboru `.emacs`:

```
(server-start)
```

Poštovní klient mutt

Tento program používáme a doporučujeme jej. Vychází ze svého předchůdce, programu elm – má i velmi podobné příkazy, nabízí ovšem mnohem více funkcí a větší nastavitelnost. Program mutt může být využíván jako klient pro poštovní protokoly POP3 i IMAP, a má navíc zabudovanou podporu MIME a šifrování PGP. Více informací a vlastní program je k dispozici na adrese http://www.mutt.org/.

Program mutt dodržuje konvenci proměnných `EDITOR/VISUAL`.

Poštovní klient elm

Program elm byl první moderní unixový poštovní klient, který umožňoval stránkové zobrazení. Poslední dobou však jeho vývoj poněkud zaostává a začíná být nahrazován právě klientem mutt. Některé verze mají také zabudovanou podporu poštovního protokolu POP3, navíc mohou uživatele některé vlastnosti potrápit – například nedostatečná podpora šifrování PGP. Existují sice některá rozšíření umožňující spolupráci s PGP, v programu mutt je ale PGP podporováno mnohem lépe. Pokud tedy uvažujete o používání PGP, je pro vás poštovní klient mutt lepší volbou.

Více informací o tomto klientovi můžete najít ve zdrojových kódech a instalačních pokynech, které jsou k dispozici například na adrese http://www.instinet.org/elm.

Program elm dodržuje konvenci proměnných `EDITOR/VISUAL`.

Poštovní klient pine

Program pine je určen a mnohými uživateli doporučován především pro začátečníky. Má vestavěnou podporu pro příspěvky diskusních serverů a podporu protokolu IMAP. Tento poštovní klient nabízí pouze omezenou sadu příkazů a možnosti nastavení. Vestavěný editor je navíc poměrně složitý na běžné používání. Pokud si chcete program pine vyzkoušet, jeho distribuci naleznete na adrese http://www.washington.edu/pine.

Program pine dodržuje konvenci proměnných `EDITOR/VISUAL`.

Netscape

Internetový prohlížeč Netscape má vestavěnou podporu pro zpracování pošty pomocí protokolů POP3 a IMAP, takže může být použit také jako poštovní klient. Toto použití ovšem není ideální, protože Netscape se nespecializuje na podporu elektronické pošty, a tak nenabízí takové spektrum funkcí jako běžný poštovní klient (chybí například podpora aliasů či PGP). Samozřejmostí je podpora protokolů LDAP a SSL.

Netscape disponuje vlastním minieditorem, který je používaný také v jiných situacích (například pro textové položky ve formulářích).

Módy rmail a vm Emacsu

Prostředí Emacs nabízí mód pro odesílání zpráv. Pro čtení zpráv pak nabízí mód rmail. Mód pro odesílání zpráv je výhodný především tehdy, pokud chcete vytvářet poštovní zprávy zcela v rámci prostředí Emacs (další možností je využít nastavení emacsclient, kterému se věnuje odstavec 2.1. Nastavení editoru zpráv).

Mód rmail však nebývá příliš doporučován – pokaždé, když je spuštěný, transformuje vaši poštovní schránku do formátu BABYL. Díky tomu nemusí ostatní poštovní klienti pracovat s vaší schránkou zcela správně. Pokud však k takové situaci dojde, můžete ji napravit pomocí příkazu prostředí Emacs **M-x unrmail**.

Pro prostředí Emacs existuje speciální prohlížeč zpráv – vm. Ten dokáže spolupracovat s poštovními schránkami ve formátu podle standardu V7. Tento program ovšem není distribuován spolu se samotným prostředím Emacs, můžete jej ale získat na adrese http://www.wonderworks.com/vm/.

Módy rmail a prohlížeč vm nepodporují konvenci proměnných `EDITOR/VISUAL`.

Poštovní klient BSD mail

Pokud napíšete na příkazové řádce Linuxu nebo dalšího moderního Unixu příkaz mail, vyvoláte některou z variant poštovního klienta BSD Mail. Je to řádkově orientované rozhraní původně navržené pro terminály. Ale v dnešní době se jedná pouze o historickou zajímavost.

Program BSD Mail zavedl konvenci podpory proměnných `EDITOR/VISUAL`.

Další poštovní klienti

V případě Linuxu jsou jako poštovní klienti známy také následující programy (pro jejich vyhledání na Internetu použijte například službu archie):

`mush`

velmi mocný klient z hlediska filtrování a dávkového zpracování

`mh`

další poštovní klient

V současné době nemáme dostatek informací k podrobnému popisu obou těchto programů. Oba nabízejí komplexní rozhraní a jsou navrženy spíše pro důmyslné uživatele.

Pokročilá témata

Podpora aliasů

Aliasy umožňují používání pseudoadres, pomocí kterých můžete nahradit složitější a úplnou adresu srozumitelnějším a jednodušším tvarem. Existují dva druhy aliasů – MUA a MTA.

Aliasy typu MUA jsou nastavené v rámci vašeho poštovního klienta a jedná se tedy o jakési osobní zkratky. Pro ostatní uživatele jsou tyto aliasy nedostupné. Můžete tedy použít například následující zápis:

```
alias esr     Eric S. Raymond<esr@thyrsus.com>
```

Předávání zpráv

Nastavení aliasů typu MTA velmi často vyžaduje administrátorská práva. Pro běžné uživatele je ale doporučeno, aby si mohli nastavit předávání jim patřících zpráv bez nutnosti spolupráce s administrátorem.

Pro zajištění tohoto požadavku většina MTA uznává „většinovou“ pozici programu sendmail a používá pro předávání zpráv soubor `.forward` umístěný ve vašem domovském adresáři. Obsah tohoto souboru je interpretován podobně jako v případě aliasů. Nejčastější použití předávání zpráv je přesměrování vašich zpráv na účet na jiném stroji.

Automatické odpovídání

Další často používanou vlastností využívající soubor .forward je přesměrování zprávy do programu vacation. Tento program zpracovává všechny příchozí zprávy a automaticky generuje nastavené odpovědi. Označení odpovědi se ujalo ze způsobu nejčastějšího použití automatického odpovídání – tím je zaslání zpětné informace o tom, že adresát je na dovolené a nebude k zastižení před uvedeným datem.

Neexistuje jeden standardní odpovídací program, který by umožňoval univerzální použití. Je tomu tak ze dvou dobrých důvodů. Prvním je fakt, že tento typ programu lze poměrně snadno vytvořit ve skriptovacím jazyku daného příkazového interpretu. Druhým důvodem je skutečnost, že odpovídací programy špatně spolupracují s poštovními konferencemi.

Před tím, než si nastavíte automatické odpovídání, byste se měli z poštovních konferencí odhlásit. V opačném případě všichni členové vámi přihlášených konferencí najdou ve svých schránkách záplavu zpráv vygenerovaných vaším odpovídacím programem. Toto je považováno za velmi hrubé chování a buďte si jisti, že po návratu z dovolené budete přivítáni velmi chladně.

Poštovní konference

Poštovní konference jsou pseudoadresy, pomocí kterých můžete zasílat zprávy více než jednomu uživateli.

V základní podobě jsou poštovní konference aliasy typu MTA s více než jedním adresátem, jedná se však o řešení vhodné pouze pro menší počet adresátů. Program sendmail podporuje tento případ navíc pomocí speciální syntaxe v souboru `/etc/aliases`, díky které můžete definovat přiřazení souboru se seznamem adres ke konkrétnímu aliasu, například takto:

```
admin-list:   ":include:/usr/home/admin/admin-list"
```

Soubor se seznamem adres může být uložen v libovolném adresáři, administrátorská práva jsou nutná pouze pro úpravu souboru `/etc/aliases`. Podobné možnosti nabízejí i některé další MTA.

Tyto jednoduché poštovní konference jsou obvykle označovány jako „mail reflectors“. V praxi je s nimi svázáno několik problémů – například ten, že informace o chybném doručení některé zprávy jsou rozesílány všem členům konference či ten, že všechna přihlášení a odhlášení musí být prováděna ručně administrátorem.

S cílem vyřešit tyto problémy byl vyvinut speciální druh programů, který bývá označován jako správce poštovní konference. Velmi důležitou funkcí je především již zmiňovaná možnost přihlašování/odhlašování členů bez nutnosti zásahu administrátora.

Správce poštovní konference i nadále využívá možností aliasů, například pro správce SmartList může soubor `/etc/aliases` vypadat například takto:

```
admin-list: "|/usr/home/smartlist/bin/flist admin-list"
admin-list-request: "|/usr/home/smartlist/bin/flist admin-list-request"
```

Uvědomte si, že v tomto příkladě jde o dvojici aliasů. Toto vychází z požadavku oddělit běžnou poštovní konferenci od adresy pro přihlašování a odhlašování. Je totiž velmi hrubé zasílat tyto požadavky na hlavní adresu konference – nedělejte to.

Robot zpracovávající požadavky na přihlášení a odhlášení může nabídnout mnohem více – například poskytovat nápovědu, umožňovat přístup k archivům či zajišťovat dotazy na informace o členech. Nebo může zajistit automatické přihlášení odesilatelů, kteří ještě nejsou členy, případně nastavit bezpečnostní opatření. Jednotliví správci poštovních konferencí se liší právě těmito funkcemi.

Bohužel formát pro zasílání příkazů do poštovních konferencí není nijak standardizován. Někteří správci očekávají příkazy v poli předmět zasílané zprávy, někteří naopak očekávají příkazy v těle zprávy a pole předmět zcela ignorují. Vyplatí se, pokud si podrobně přečtete první zprávu, která vám bude doručena jako odpověď na vaše první přihlášení. Uložte si tuto zprávu pro pozdější potřebu.

Mezi nejznámější správce poštovních konferencí patří majordomo, listserv, listproc či smartlist. Nejpopulárnějším je právě majorodomo, především pak pro své nadstandardní možnosti. Poslední dobou se taky stává velmi populárním správce mailman (http://www.gnu.org/software/mailman/mailman.html), který nabízí webové rozhraní pro přihlašování, odhlašování a administraci konferencí. Podrobný seznam správců můžete najít na adrese http://www.catalog.com/vivian/mailing-list-software.html.

Více informací o těchto správcích naleznete v odpovídajících zdrojích na Internetu (například na adrese http://www.greatcircle.com/list-managers/), a to včetně souborů často kladených dotazů

a odpovědí (uvědomte si ale, že tato skupina není vhodná pro kladení otázek typu „jak udělat to a to“).

Filtrování zpráv

O filtrování zpráv se starají programy, které jsou umístěny mezi místního doručovacího agenta a vás, automaticky vyřizují či odmítají zprávy před tím, než si je prohlédnete.

Filtrování zpráv nabízí mnohé – nejčastěji se používá jako obrana proti nevyžádaným zprávám, automatické odpovídání a třídění do více poštovních schránek podle předmětu dané zprávy.

Filtrování zpráv nastavíte v mnoha případech tak, že zadáte alias na odpovídají program pro filtrování do vašeho souboru .forward a vytvoříte soubor s pravidly pro filtrování. Formát a umístění těchto souborů s pravidly je závislý na konkrétním programu.

Seznam vlastností třech hlavních programů pro filtrování, tedy procmail, mailagent a deliver, je obsažen ve třetí části dokumentu Chrise Lewise Email Software Survey, který můžete najít na adrese http://www.faqs.org/faqs/mail/setup/unix/part3/index.html. Nejpopulárnější je první z uvedených – procmail – a to i přes poměrně složitou syntaxi pro zápis pravidel. Procmail je standardní součástí linuxových systémů (obvykle je používán jako systémový lokální doručovací agent).

Obrana proti nevyžádaným zprávám

Pod spamem si většinou každý vybaví nevyžádané zprávy, mnohdy pak komerčního charakteru (UCE – Unsolicited Commercial Email). Za spamem se ale také skrývají nevyžádané zprávy značného obejmu (UCB – Unsolicited Bulk Email). Jedná se zpravidla o obtěžující formu reklamy, která zahlcuje vaše poštovní schránky. Mimochodem – termín spam pochází ze žertu použitého v Monty Pythonově létajícím cirkusu, v rámci kterého zpívají Vikingové nekonečně popěvek „Spam spam spam spam spam ...“

Mnohé nevyžádané zprávy obsahují naléhavou prosbu pyramidového charakteru, přidávají pornografii či obtěžující výzvy k prodeji. Několik individuálních nevyžádaných zpráv (například MAKE MONEY FAST či Craig Shergold) jsou již věčné a staly se legendami. Nevyžádané zprávy mnohdy obsahují celé množství gramatických chyb. Uvědomte si, že vždy jde o plýtvání vaším časem a obrovským plýtváním z hlediska síťového přenosu.

Epidemie nevyžádaných zpráv vyvrcholila v polovině roku 1997, ale i přesto, že pomalu ustupuje, může vás kdykoli nepříjemně potrápit. Více informací o nevyžádaných zprávách můžete najít na serverech Fight Spam on The Internet (http://spam.abuse.net/) či Death To Spam! (http://www.mindworkshop.com/alchemy/nospam.html).

Další zdroje informací

Diskusní skupiny USENET

Další informace jsou k dispozici v několika skupinách na Usenetu zaměřených na technické problémy o elektronické poště.

`comp.mail.elm`

Tato diskusní skupina je věnována systému ELM.

`comp.mail.mh`

Tato diskusní skupina je věnována systému Message Handling.

`comp.mail.mime`

Tato diskusní skupina je věnována používání MIME pro poštu na Internetu.

`comp.mail.misc`

Tato diskusní skupina je věnována elektronické poště a jedná se o hlavní skupinu věnovanou této problematice.

`comp.mail.multi-media`

Tato diskusní skupina je zaměřena na problematiku multimediálních zpráv.

`comp.mail.mush`

Tato diskusní skupina je zaměřena na MUSH (Mail User's Shell)

`comp.mail.sendmail`

Tato diskusní skupina je věnována poštovnímu serveru sendmail.

`comp.mail.smail.`

Tato diskusní skupina je věnována poštovnímu serveru smail.

`comp.mail.uucp`

Tato diskusní skupina je věnována zprávám v prostředí UUCP.

Knihy

V případě zájmu můžete další informace najít také v následujících publikacích:

Sendmail

Tato publikace z nakladatelství O'Reilly a Associates se věnuje programům sendmail-v8 a sendmail+IDA. Měla by být v knihovničce každého, kdo to s programy sendmail myslí vážně.

The Internet Complete Reference

Tato publikace z nakladatelství Osborne je skvělá referenční příručka pro objasnění mnoha služeb dostupných na Internetu a je vynikajícím zdrojem informací o diskusních serverech, poště a mnoha dalších internetových zdrojích.

The Linux Networking Administrators' Guide

Tato publikace Olafa Kircha je k dispozici na Internetu a byla také vydána v nakladatelstvích O'Reilly a SSC. Jedná se o vynikající knížku pro seznámení se síťovou problematikou v unixových operačních systémech.

Periodicky rozesílané informace

Za zmínku zcela jistě stojí periodické rozesílání informací od Chrise Lewise, které jsou k dispozici na adrese ftp://rtfm.mit.edu/pub/usenet/comp.mail.misc, konkrétně v souborech UNIX_Email_Software_Survey_*. Formát HTML je k dispozici na adrese http://www.faqs.org/faqs/setup/unix. Je nutné si ale uvědomit, že tento dokument nebyl od roku 1996 příliš obnovován.

Kde nápovědu nezískáte

Není vhodné obracet se se žádostí o pomoc s obecnými problémy s elektronickou poštou do specializovaných diskusních skupin věnovaných pouze linuxovým prostředím. Výjimkou jsou dotazy typu – „řekněte mi, jaké směrovače jsou již zakompilovány do SLS1.03 smail 3.1.28). V ostatních případech se obracejte s žádostí o pomoc do některé ze skupin uvedených v kapitole 4.1.

Dovolte nám to zopakovat ...

Nejsou důvody pro zasílání obecných příspěvků do diskusních skupin comp.os.linux, protože pro ně jsou určeny skupiny comp.mail.*.

Pokud zašlete dotaz do skupin comp.os.linux.* netýkající se Linuxu, jste na špatném místě pro vyřešení svého problému. Specialisté věnující se elektronické poště jsou přihlášeni do obecných diskusních skupin. Jedná se tedy o plýtvání vašeho času a také času dalších členů. S největší pravděpodobností se tak odpovědi dočkáte, pokud vůbec, s velkým zpožděním.

Administrativa

Zpětná vazba

(Tuto kapitolu napsal Vince, ale má politika je stejná.)

Zajímáme se o jakoukoli odpověď, kladnou i zápornou, pokud jde o obsah tohoto dokumentu, zaslanou elektronickou poštou. Určitě nás kontaktujte v případě, kdy objevíte nějakou chybu či zřejmé opomenutí něčeho podstatného.

Čteme všechny přijaté zprávy, na všechny ale neodpovídáme. Požadavky na rozšíření tohoto dokumentu budou zváženy a zapracovány v závislosti na volném čase a důležitosti požadavku.

Standardy pro cesty v linuxových souborových systémech jsou ve vývoji. Konkrétní případy uvedené v tomto dokumentu jsou pouze pro ilustraci v okamžiku vzniku tohoto dokumentu. Proto je nezbytně nutné si zjistit případné umístění jednotlivých souborů a adresářů daných vaší konkrétní distribucí.

Zprávy ohledně aktuálního formátu tohoto dokumentu by měly být zasílány na koordinátora HOWTO s poštovní adresou linux-howto@metalab.unc.edu.

Autorská práva

kud máte otázky, obraťte se, prosím, na koordinátora HOWTO s poštovní adresou linux-howto@metalab.unc.edu.

Zřeknutí se odpovědnosti

Autoři se zříkají odpovědnosti plynoucí z obsahu tohoto dokumentu. Použití pojmů, příkladů a/nebo dalšího obsahu tohoto dokumentu je výhradně na vaše riziko.

Poděkování

Původním autorem tohoto dokumentu je Vince Skahan. Eric Steven Raymond jej přepsal do moderní podoby odpovídající současnému stavu, který je orientován především na svět ISP, kde je UUCP již jen malou vzpomínkou.

V květnu 1999 se název tohoto dokumentu změnil z „The Linux Electronic Mail HOWTO“ z důvodů možných kolizí s dokumentem od Guylhema Aznarema, který se změnil na „Mail Administrator HOWTO“.

KAPITOLA 5

Nicolai Langfeldt, janl@math.uio.no

DNS

Jak se stát úplně malým administrátorem systému DNS.

Předmluva

Klíčová slova: DNS, bind, bind-4, bind-8, named, dialup, ppp, slip, isdn, Internet, domain, name, hosts, resolving.

Autorská práva

Poděkování a další

Chtěl bych poděkovat Arntu Gulbrandsenovi, který mnohokrát četl návrhy této práce a navrhl mnohá užitečná vylepšení. Chtěl bych také poděkovat lidem, kteří mi poslali návrhy a poznámky.

Tento dokument nebude nikdy dokončen. Pošlete mi prosím zprávy o vašich problémech a úspěších, dokument tak mohu vylepšovat. Peníze, komentáře nebo otázky posílejte na *janl@math.uio.no*. Jestliže budete odesílat elektronickou zprávu, zajistěte prosím, aby byla návratová adresa funkční. Předtím, než mi napíšete, si také prosím přečtěte část *Otázky a odpovědi*. Poslední maličkost – rozumím pouze anglicky a norsky.

Jestliže chcete tento dokument překládat, sdělte mi to, abych věděl, ve kterých jazycích jsem byl publikován, současně vás mohu upozornit na změny v dokumentu.

Věnování

Dokument bych rád věnoval Anne Line Norheim Langfeldtové. I když jej pravděpodobně nikdy nebude číst, protože není takovým typem dívky, která by jej četla.

Úvod

Co to je a co není

DNS je Domain Name System. DNS převádí názvy strojů na IP adresy, které mají všechny počítače na síti. Mapuje názvy na adresy, adresy na názvy a plní i některé další funkce. Tento dokument ukazuje, jak takové mapování provádět v Linuxu. Mapování je vlastně vztah mezi dvěma věcmi, v našem případě mezi názvem stroje (například *ftp.linux.org*) a IP adresou stroje (zde 198.182.196.59).

DNS je pro nezasvěcené jednou z méně průhledných oblastí síťové správy. Tento dokument by měl alespoň něco osvětlit. Popisuje, jak nastavit *jednoduchý* DNS server. Začneme přitom s caching-only DNS serverem a přejdeme až k nastavení primárního DNS serveru domény. Složitější nastavení naleznete v části *Otázky a odpovědi*. Jestliže zde nenaleznete to, co potřebujete, musíte si pročíst opravdovou dokumentaci. K té se dostanu v poslední kapitole.

Před dalším postupem je vhodné nakonfigurovat váš počítač, aby bylo možné se z něj a na něj přihlásit pomocí telnetu a provést všechny druhy připojení na síť. Zejména je nutné vyzkoušet `telnet 127.0.0.1` na vlastní stroj (zkuste to ihned!). Jako počáteční bod potřebujete také dobře nastavené soubory `/etc/nsswitch.conf` (nebo `/etc/host.conf`), `/etc/resolv.conf` a `/etc/hosts`. Nebudu zde totiž vysvětlovat jejich funkci. Jestliže ještě nemáte všechno nastavené a funkční, pomohou vám dokumenty *Networking-HOWTO* a *PPP-HOWTO*. Pročtěte si je.

Když říkám „váš počítač“, myslím tím počítač, na kterém má běžet DNS server. Ne tedy jakýkoliv jiný váš počítač, který používáte ve vašem síťovém snažení.

Předpokládám, že se nenacházíte za žádným druhem firewallu, který by blokoval DNS dotazy. Jestliže se ale za ním nacházíte, budete potřebovat speciální konfiguraci, viz část *Otázky a odpovědi*.

Převody názvů jsou v Unixu prováděny programem `named`. Ten je součástí balíku „bind“, který je koordinován Paulem Vixiem z Internet Software Consortium. `Named` je obsažen ve většině distribucí Linuxu a instaluje se jako `/usr/sbin/named`. Jestliže už `named` máte, pravděpodobně jej také můžete používat; jestliže jej nemáte, můžete si z linuxového FTP serveru stáhnout binární podobu, nebo si můžete z *ftp://ftp.isc.org/isc/bind/src/cur/bind-8/* stáhnout poslední zdrojovou podobu. Tento dokument se zabývá balíkem bind verze 8. Používáte-li bind verze 4, stará verze tohoto dokumentu (vztahující se k bind verze 4) je pro vás stále k dispozici na *http://www.math.uio.no/janl/DNS/*. Jestliže manuálová stránka k named hovoří o `named.conf`, máte bind 8, jestliže hovoří o `named.boot`, máte bind 4. Jestliže máte 4 a staráte se o zabezpečení, měli byste přejít na 8.

DNS je databází rozloženou na síti. Starejte se o to, co do ní vkládáte. Jestliže sem vložíte nesmysly, uvidí je všichni. Svůj systém DNS udržujte konzistentní a jasný – získáte tak kvalitnější služby. Naučte se jej používat, spravovat, odlaďovat a bude z vás další „hodný“ síťový správce, který chrání síť před přetížením daným špatnou správou.

V tomto dokumentu nastíním několik věcí, které nejsou zcela pravdivé (jsou to spíše polopravdy). Vše v zájmu zjednodušení. Když mi budete věřit, všechno bude (pravděpodobně) fungovat.

Tip: U všech souborů, které vám navrhuji změnit, si vytvořte záložní kopie. Kdyby se stalo, že nebude nic fungovat, můžete se ještě vrátit ke staré konfiguraci.

Caching-only DNS server

První zásah do konfigurace DNS, velmi užitečný pro uživatele modemů

DNS server typu caching-only nalezne odpověď na vaše dotazy a bude si pro příště pamatovat odpověď. Tím se příště zkrátí doba čekání, zejména pokud se nacházíte na pomalých připojeních.

Nejprve potřebujete soubor nazvaný `/etc/named.conf`. Ten je čten při spuštění *named*. Pro tentokrát by měl jednoduše obsahovat:

```
// Konfigurační soubor pro caching only DNS server

options {
  directory "/var/named";
  // Zrušení následujícího komentáře vám může pomoci
  // pokud při průchodu firewallem narazíte na problémy
  // query-source address * port 53;
} ;
zone "." {
  type hint;
  file "root.hints";
} ;

zone "0.0.127.in-addr.arpa" {
  type master;
  file "pz/127.0.0";
} ;
```

Řádek `directory` sděluje programu *named*, kde se mají soubory hledat. Všechny další uvedené názvy souborů se budou vztahovat k tomuto adresáři. `pz` je pak podadresář pod `/var/named`, tedy `/var/named/pz`. `/var/named` je adresář, který odpovídá Linux File System Standard.

Soubor `/var/named/root.hints` by měl obsahovat následující:

```
.       6D IN NS        G.ROOT-SERVERS.NET.
.       6D IN NS        J.ROOT-SERVERS.NET.
.       6D IN NS        K.ROOT-SERVERS.NET.
.       6D IN NS        L.ROOT-SERVERS.NET.
.       6D IN NS        M.ROOT-SERVERS.NET.
.       6D IN NS        A.ROOT-SERVERS.NET.
.       6D IN NS        H.ROOT-SERVERS.NET.
.       6D IN NS        B.ROOT-SERVERS.NET.
.       6D IN NS        C.ROOT-SERVERS.NET.
.       6D IN NS        D.ROOT-SERVERS.NET.
.       6D IN NS        E.ROOT-SERVERS.NET.
.       6D IN NS        I.ROOT-SERVERS.NET.
.       6D IN NS        F.ROOT-SERVERS.NET.

G.ROOT-SERVERS.NET.    5w6d16h IN A          192.112.36.4
J.ROOT-SERVERS.NET.    5w6d16h IN A          198.41.0.10
K.ROOT-SERVERS.NET.    5w6d16h IN A          193.0.14.129
L.ROOT-SERVERS.NET.    5w6d16h IN A          198.32.64.12
M.ROOT-SERVERS.NET.    5w6d16h IN A          202.12.27.33
```

```
A.ROOT-SERVERS.NET.    5w6d16h IN A          198.41.0.4
H.ROOT-SERVERS.NET.    5w6d16h IN A          128.63.2.53
B.ROOT-SERVERS.NET.    5w6d16h IN A          128.9.0.107
C.ROOT-SERVERS.NET.    5w6d16h IN A          192.33.4.12
D.ROOT-SERVERS.NET.    5w6d16h IN A          128.8.10.90
E.ROOT-SERVERS.NET.    5w6d16h IN A          192.203.230.10
I.ROOT-SERVERS.NET.    5w6d16h IN A          192.36.148.17
F.ROOT-SERVERS.NET.    5w6d16h IN A          192.5.5.241
```

Soubor popisuje světové kořenové DNS servery. Ty se během doby mění a musí být udržovány. Jak tento seznam aktualizovat je popsáno v kapitole *Údržba*.

Další část souboru `named.conf` definuje *zónu*. Co to je si vysvětlíme později, pro tuto chvíli potřebujeme v adresáři `pz` vytvořit soubor `127.0.0`:

```
@               IN      SOA     nslinux.bogus. hostmaster.linux.bogus. (
                                1               ; Serial
                                8H              ; Refresh
                                2H              ; Retry
                                1W              ; Expire
                                1D)             ; Minimum TTL
                        NS      ns.linux.bogus.
1                       PTR     localhost.
```

Pak potřebujete soubor `/etc/resolv.conf`, který vypadá přibližně takto:

```
search subdomain.your-domain.edu your-domain.edu
nameserver 127.0.0.1
```

Řádek `search` určuje, ve kterých doménách budou vyhledány názvy hostitelů, ke kterým se chcete připojit. Řádek `nameserver` určuje adresu vašeho DNS serveru – zde tedy vašeho stroje, protože na něm spouštíte `named` (správně je 127.0.0.1, nezávisle na případných dalších adresách vašeho stroje). Jestliže hodláte použít několik DNS serverů, pro každý z nich vložte jeden řádek `nameserver`. (Poznámka: *named* tento soubor nikdy nečte, ale resolver, který používá *named*, jej čte).

Aby bylo jasné, co tento soubor dělá: Jestliže se klient pokusí vyhledat název `foo`, pak se nejprve vyzkouší `foo.subdomain.your-domain.edu`, pak `foo.your-domain.edu` a nakonec `foo`. Jestliže se klient pokouší vyhledat `sunsite.unc.edu`, vyzkouší se nejprve `sunsite.unc.edu.subdomain.your-domain.edu` (ano, je to hloupé, ale tak to funguje), potom `sunsite.unc.edu.your-domain.edu` a nakonec `sunsite.unc.edu`. Na vyhledávací řádek tedy raději nedávejte příliš mnoho domén, vyhledávání by mohlo trvat dlouho.

Příklad předpokládá, že patříte do domény `subdomain.your-domain.edu`, váš stroj se tedy nazývá `your-machine.subdomain.your-domain.edu`. Vyhledávací řádek by neměl obsahovat vaši TLD (doménu nejvyšší úrovně, zde `edu`). Jestliže se často připojujete na hostitele v jiné doméně, můžete tuto doménu přidat do vyhledávacího řádku:

```
search subdomain.your-domain.edu your-domain.edu other-domain.com
```

A tak dále. Přirozeně, že zde použijete funkční názvy domén. Povšimněte si, že za názvy domén chybí tečky.

Poté musíte upravit buď `/etc/nsswitch.conf`, nebo `/etc/host.conf`. Jestliže máte soubor `nsswitch.conf`, budeme jej upravovat, jestliže jej nemáte, budeme upravovat `host.conf`.

/etc/nsswitch.conf

Tohle je dlouhý soubor, který určuje, odkud se vezmou různé datové typy, ze kterého souboru nebo databáze. Na začátku souboru jsou obvykle užitečné komentáře, které si raději hned přečtěte. Poté nalezněte řádek začínající hosts:, měl by vypadat takto:

```
hosts: files dns
```

Jestliže zde není žádný řádek, který by začínal `hosts:`, pak jej vložte. Sděluje, že program by se měl dívat nejprve do souboru `/etc/hosts`, potom podle `resolv.conf` použít DNS.

/etc/host.conf

Obsahuje pravděpodobně několik řádků, jeden by měl začínat `order` a vypadat takto:

```
order hosts,bind
```

Jestliže zde není žádný řádek `order`, přidejte jej. Sděluje, že rozhodovací rutiny jména hostitele budou hledat nejprve v `/etc/hosts` a poté požádají DNS server (v `resolv.conf` jste sdělili, že je to 127.0.0.1).

Spuštění named

Takže nastal čas spustit *named*. Jestliže používáte připojení přes modem, tak se nejprve připojte. Zadejte `ndc start` (žádné parametry) a stiskněte Enter. Jestli to zlobí, zkuste místo toho `/usr/sbin/ndc start`. Pokud to pořád zlobí, přejděte k části *Otázky a odpovědi*. Pokud si můžete prohlédnout zprávy démona syslog (obvykle `/var/log/messages`, ale občas jde o adresář `/var/adm` a občas jde o soubor `syslog`) při spouštění programu *named* (zadejte `tail -f /var/log/messages`), měli byste vidět něco jako:

(řádky ukončené „\“ pokračují na následujícím řádku)

```
Feb 15 01:26:17 roke named[6091]: starting. named 8.1.1 Sat Feb 14 \
00:18:20 MET 1998 ^Ijanl@roke.uio.no:/var/tmp/bind-8.1.1/src/bin/named
Feb 15 01:26:17 roke named[6091]: cache zone "" (IN) loaded (serial 0)
Feb 15 01:26:17 roke named[6091]: master zone "0.0.127.in-addr.arpa" \
(IN) loaded (serial 1)
Feb 15 01:26:17 roke named[6091]: listening [127.0.0.1].53 (lo)
Feb 15 01:26:17 roke named[6091]: listening [129.240.230.92].53 (ippp0)
Feb 15 01:26:17 roke named[6091]: Forwarding source address is [0.0.0.0].1040
Feb 15 01:26:17 roke named[6092]: Ready to answer queries.
```

Pokud se objeví nějaká chybová hlášení, něco není v pořádku. Named by měl vypsat soubor, ve kterém k chybě došlo (typicky `named.conf` nebo `root.hints`). Ukončete *named* a opravte příslušný soubor.

Nyní můžete otestovat nastavení. Spusťte *nslookup* a vyzkoušejte váš systém.

```
$ nslookup
Default Server: localhost
Address: 127.0.0.1
>
```

Jestli dostanete uvedený výsledek, tak to funguje. Doufejme. Pokud se objeví něco jiného, vraťte se a všechno zkontrolujte. Pokaždé, když změníte soubor `named.conf`, musíte pomocí příkazu `ndc restart` *named* znovu spustit.

Nyní můžete zadávat dotazy. Zkuste vyhledávat některé blízké stroje. Pro mě je to `pat.uio.no` (na univerzitě v Oslo):

```
> pat.uio.no
Server: localhost
Address: 127.0.0.1

Name: pat.uio.no
Address: 129.240.130.16
```

nslookup nyní požádal váš *named* o nalezení stroje `pat.uio.no`. Ten poté kontaktoval jeden z DNS serverů vyjmenovaných ve vašem souboru `root.hints`, a požádal o odpověď odtud. Výsledek můžete dostat až za chvíli, protože se prohledávají všechny domény vyjmenované v `/etc/resolv.conf`.

Jestliže se na totéž dotážete ještě jednou, dostanete následující:

```
> pat.uio.no
Server: localhost
Address: 127.0.0.1

Non-authoritative answer:
Name: pat.uio.no
Address: 129.240.2.50
```

Všimněte si řádku `Non-authoritative answer:`, který se nám nyní objevil. Znamená to, že *named* se pro odpověď neodebral na síť, ale jen do své vyrovnávací paměti, kde ji také našel. Ale tato informace by mohla být stará. Proto jste na tohle (velmi malé) nebezpečí upozorněni sdělením `Non-authoritative answer:` (neautorizovaná odpověď). Když vám to *nslookup* sdělí při druhém dotazu na stejnou lokaci, jedná se o znamení, že *named* funguje a ukládá si informace do vyrovnávací paměti. *nslookup* ukončíte příkazem `exit`.

Další vylepšení

Na větších, dobře organizovaných akademických sítích nebo sítích poskytovatelů můžete zjistit, že byla vytvořena hierarchie předávajících DNS serverů, které optimalizují zatížení interní sítě i externích serverů. Nelze ovšem jednoduše zjistit, že se v takové síti nacházíte. Není to ale důležité a pokud použijete DNS server vašeho poskytovatele jako předávající server, můžete zrychlit odezvy a odlehčit zátěž. Používáte-li modem, je to k nezaplacení. Pro náš příklad předpokládáme, že váš poskytovatel používá dva DNS servery s adresami `10.0.0.1` a `10.1.0.1`. Pak v souboru `named.conf` uvnitř úvodní části `options` vložíte řádky

```
forward first;
forwarders {
  10.0.0.1;
  10.1.0.1;
};
```

Restartujte server a vyzkoušejte jej programem *nslookup*. Všechno by mělo fungovat.

Blahopřejeme

Nyní víte, jak nastavit *named* s vyrovnávací pamětí. Dejte si na oslavu pivo, mléko nebo to, čemu dáváte přednost.

Jednoduchá doména

Jak nastavit vlastní doménu

Nejprve ale trochu suché teorie

Předtím, než opravdu začneme tuto část, vám předložím teorii o funkci DNS. A vy si ji pročtete, protože je pro vás poučná. Jestli se vám nechce, tak ji alespoň v rychlosti projděte. Ale nezapomeňte se zastavit u obsahu souboru `named.conf`.

DNS je hierarchický, stromově strukturovaný systém. Jeho vrchol se označuje . a vyslovuje `root`. Pod . je mnoho domén nejvyšší úrovně (TLD). Nejznámější jsou `ORG`, `COM`, `EDU` a `NET`, ale je zde mnoho dalších. Stejně jako strom má i DNS svůj kořen a větve. Pokud máte nějaké počítačové vzdělání, rozeznáte v DNS vyhledávací strom a víte co jsou uzly, listy a hrany.

Při vyhledávání stroje se dotaz zpracovává v hierarchii rekurzivně od vrcholu. Jestliže hodláte nalézt adresu pro `prep.ai.mit.edu`, musí váš systém nalézt DNS server, který obsluhuje doménu `edu`. Požádá kořenový server (kořenové servery jsou známy, od toho je zde soubor `root.hints`). Kořenový server poskytne seznam serverů domény `edu`:

```
$ nslookup
Default Server: localhost
Address: 127.0.0.1
```

Budeme se ptát kořenového serveru:

```
> server c.root-servers.net.
Default Server: c.root-servers.net
Address: 192.33.4.12
```

Nastavení typu dotazu na NS (záznamy DNS serverů):

```
> set q=ns
```

Dotaz na `edu`:

```
> edu.
```

Tečka je zde důležitá – sděluje serveru, že se ptáme na `edu` přímo pod .(tím se vyhledávání zjednoduší).

```
edu     nameserver = A.ROOT-SERVERS.NET
edu     nameserver = H.ROOT-SERVERS.NET
edu     nameserver = B.ROOT-SERVERS.NET
edu     nameserver = C.ROOT-SERVERS.NET
edu     nameserver = D.ROOT-SERVERS.NET
edu     nameserver = E.ROOT-SERVERS.NET
edu     nameserver = I.ROOT-SERVERS.NET
edu     nameserver = F.ROOT-SERVERS.NET
edu     nameserver = G.ROOT-SERVERS.NET
```

```
A.ROOT-SERVERS.NET     internet address = 198.41.0.4
H.ROOT-SERVERS.NET     internet address = 128.63.2.53
B.ROOT-SERVERS.NET     internet address = 128.9.0.107
C.ROOT-SERVERS.NET     internet address = 192.33.4.12
D.ROOT-SERVERS.NET     internet address = 128.8.10.90
E.ROOT-SERVERS.NET     internet address = 192.203.230.10
I.ROOT-SERVERS.NET     internet address = 192.36.148.17
F.ROOT-SERVERS.NET     internet address = 192.5.5.241
G.ROOT-SERVERS.NET     internet address = 192.112.36.4
```

Tohle nám říká, že všechny kořenové servery `root-servers.net` obsluhují i doménu `edu`, takže můžeme požádat kterýkoliv z nich. Nyní chceme vědět, kdo obsluhuje následující úroveň názvu domény – `mit.edu`:

```
> mit.edu.
Server: c.root-servers.net
Address: 192.33.4.12

Non-authoritative answer:
mit.edu nameserver = W20NS.mit.edu
mit.edu nameserver = BITSY.mit.edu
mit.edu nameserver = STRAWB.mit.edu

Authoritative answers can be found from:
W20NS.mit.edu internet address = 18.70.0.160
BITSY.mit.edu internet address = 18.72.0.3
STRAWB.mit.edu internet address = 18.71.0.151
```

Takže doménu `mit` obsluhují servery `steawb`, `w20ns` a `bitsy`. Vyberte si jeden a pokračujte s `ai.mit.edu`:

```
> server W20NS.mit.edu.
```

U názvů hostitelů nezáleží na velkých a malých písmenech, zde jsou přímo okopírovány (pomocí paste) z obrazovky.

```
Server: W20NS.mit.edu
Address: 18.70.0.160

> ai.mit.edu.
Server: W20NS.mit.edu
Address: 18.70.0.160

Non-authoritative answer:
ai.mit.edu      nameserver = BEET-CHEX.ai.mit.edu
ai.mit.edu      nameserver = FEDEX.ai.mit.edu
ai.mit.edu      nameserver = LIFE.ai.mit.edu
ai.mit.edu      nameserver = ALPHA-BITS.ai.mit.edu

Authoritative answers can be found from:
BEET-CHEX.ai.mit.edu    internet address = 128.52.32.22
FEDEX.ai.mit.edu        internet address = 192.148.252.43
LIFE.ai.mit.edu internet address = 128.52.32.80
ALPHA-BITS.ai.mit.edu   internet address = 128.52.32.5
```

Takže DNS server pro `ai.mit.edu` je třeba `fedex.ai.mit.edu`:

```
> server fedex.ai.mit.edu
Default Server:  fedex.ai.mit.edu
Address:  192.148.252.43
```

Nyní změním typ požadavku (query). Nalezli jsme DNS server, takže se ho zeptáme na vše, co ví o `prep.ai.mit.edu`.

```
> set q=any
> prep.ai.mit.edu.
Server:  fedex.ai.mit.edu
Address:  192.148.252.43

prep.ai.mit.edu CPU = i686      OS = unix
prep.ai.mit.edu internet address = 198.186.203.77
prep.ai.mit.edu preference = 1, mail exchanger = mail.gnu.org
ai.mit.edu      nameserver = alpha-bits.ai.mit.edu
ai.mit.edu      nameserver = beet-chex.ai.mit.edu
ai.mit.edu      nameserver = fedex.ai.mit.edu
ai.mit.edu      nameserver = life.ai.mit.edu
alpha-bits.ai.mit.edu   internet address = 128.52.32.5
beet-chex.ai.mit.edu    internet address = 128.52.32.22
fedex.ai.mit.edu        internet address = 192.148.252.43
life.ai.mit.edu internet address = 128.52.32.80
```

Takže počínaje od . jsme postupně nalezli všechny DNS servery pro následující úrovně názvu domény. Pokud byste místo použití všech ostatních serverů použili vlastní DNS server, váš *named* by si samozřejmě všechny získané informace ukládal do vyrovnávací paměti a případný další dotaz na totéž by už zodpověděl sám.

V analogii se stromem je každá tečka v názvu místo, kde se odděluje další větev a cokoliv mezi dvěma tečkami jsou jednotlivé větve stromu.

„Prolezli“ jsme stromem tak, že jsme si vzali požadovaný název (`prep.ai.mit.edu`), nejprve jsme nalezli kořen a pak jsme postupovali po první větvi, v tomto případě `edu`. Vylezli jsme na ni a přepnuli se na server, který tuto větev obsluhuje. Dále jsme na větvi `edu` hledali větev `mit` (celý název je `mit.edu`) a přepnuli se na server, který obsluhuje větev `mit.edu`. Znovu jsme hledali další větev, `ai.mit.edu` a znovu se přepnuli na příslušný server. Teď jsme na správném serveru a hledáme `prep.ai.mit.edu`, což je jednoduché. V počítačové teorii představuje `prep` *list* stromu.

Méně diskutovanou, ale stejně důležitou doménou je `in-addr.arpa`. Je to „stejná“ doména jako ostatní. Umožňuje získat názvy hostitelů podle adres. Zde je důležité si povšimnout, že IP adresy jsou v doméně `in-addr.arpa` napsány obráceně. Jestliže máte adresu stroje `192.128.52.43`, *named* pokračuje stejně jako u předchozího příkladu `prep.ai.mit.edu`. Nalezne servery `arpa.`, nalezne servery `in-addr.arpa.`, nalezne servery `192.in-addr.arpa.`, nalezne servery `128.192.in-addr.arpa.`, nalezne servery `52.128.192.in-addr.arpa.`. Nalezne požadovaný záznam pro `43.52.128.192.in-addr.arpa`. Jasné? (Odpovězte „Ano!“) Tak dva roky se vám obrácené pořadí čísel může ještě zdát matoucí.

Teď jsem lhal. DNS sice nepracuje tak, jak jsem říkal, ale zato dost podobně.

Naše vlastní doména

Nyní k definici vaší vlastní domény. Chystáme se vytvořit doménu `linux.bogus` a definovat v ní stroje. Využívám zde fiktivní (*bogus*) název domény, aby bylo jasné, že tam nikoho nevyrušíme.

Ještě něco, než začneme: Ve jménech hostitelů nejsou povoleny všechny znaky. Jsme omezeni na znaky anglické abecedy (`a-z`), číslice (`0-9`) a pomlčky „`-`“. Držte se toho. Pro DNS se nerozlišují velká a malá písmena. Takže `pat.uio.no` je totéž jako `Pat.UiO.No`.

Tuto část jsme již začali s tímto řádkem v `named.conf`:

```
zone "0.0.127.in-addr.arpa" {
 type master;
 file "pz/127.0.0";
} ;
```

Všimněte si, že zde na koncích názvů domén není tečka. To znamená, že nyní budeme definovat zónu `0.0.127.in-addr.arpa`, pro kterou jsme hlavním serverem a která je uložena v souboru `pz/127.0.0`. Tento soubor už jsme nastavili, vypadá takto:

```
@       IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
                        1       ; Serial
                        8H      ; Refresh
                        2H      ; Retry
                        1W      ; Expire
                        1D)     ; Minimum TTL
                NS      ns.linux.bogus.
1               PTR     localhost.
```

Zde už tečky na konci všech plných názvů domén jsou, což je v kontrastu se souborem `named.conf`. Někteří lidé rádi začínají každý zónový soubor direktivou `$ORIGIN`, ale je to zbytečné. Původ (kam náleží v hierarchii DNS) zónového souboru je určen na zónové řádce v souboru `named.conf`, v tomto případě to je `0.0.127.in-addr.arpa`.

Tento *zónový soubor* obsahuje tři *zdrojové záznamy* (RR): SOA, NS a PTR. SOA je zkratkou pro začátek autority. „`@`“ je speciální označení, znamenající počátek, a protože „doménový“ sloupec pro tento soubor říká `0.0.127.in-addr.arpa`, první řádek ve skutečnosti znamená

```
0.0.127.in-addr.arpa. IN SOA ...
```

NS je zdrojový záznam pro DNS server. Na začátku tohoto řádku není žádné „`@`“, je to implicitní, protože poslední řádek začínal na „`@`“. Tím se ušetří trocha psaní. Takže NS-řádek skutečně vypadá takto:

```
0.0.127.in-addr.arpa. IN NS ns.linux.bogus
```

Sděluje DNS, který stroj je DNS serverem domény `0.0.127.in-addr.arpa`. Je to `ns.linux.bogus`. „ns“ je obecný název pro DNS servery, ale název může být i jiný, stejně jako u webových serverů, které by měly být pojmenovány `www.něco`, ale často jsou pojmenovány jinak.

A konečně záznam PTR sděluje, že hostitel na adrese 1 v podsíti `0.0.127.in-addr.arpa` (`127.0.0.1`) je pojmenován `localhost`.

Záznam SOA je úvodem ke všem zónovým souborům a v každém zónovém souboru by měl být právě jeden – jako první záznam. Popisuje zónu, ze které pochází (stroj pojmenovaný `ns.linux.bogus`), kdo je zodpovědný za jeho obsah (`hostmaster@linux.bogus`), jaké verze je tento

zónový soubor (`serial: 1`) a další věci, které mají co do činění se sekundárními DNS servery a vyrovnávací pamětí. Pro zbylá pole (`refresh`, `retry`, `expire` a `minimum`) použijte čísla z našeho dokumentu a měli byste být bez problémů.

Nyní znovu spusťte *named* (příkaz je `ndc restart`) a to co jste vytvořili otestujte pomocí *nslookup*:

```
$ nslookup

Default Server: localhost
Address: 127.0.0.1

> 127.0.0.1
Server: localhost
Address: 127.0.0.1

Name: localhost
Address: 127.0.0.1
```

Takže ze `127.0.0.1` se stal `localhost`, to je dobré. Nyní pro náš hlavní úkol (doména `linux.bogus`) vložte do `named.conf` novou zónovou část:

```
zone "linux.bogus" {
  notify no;
  type master;
  file "pz/linux.bogus";
} ;
```

Všimněte si, že na konci názvu domény v souboru `named.conf` opět chybí tečka.

Do zónového souboru `linux.bogus` vložíme nějaká zcela fiktivní data:

```
;
; Zone file for linux.bogus
;
; The full zone file
;
@       IN      SOA     ns.linux.bogus.hostmaster.linux.bogus. (
                        199802151               ; serial, todays date +
                                                  todays serial #
                        8H                      ; refresh, seconds
                        2H                      ; retry, seconds
                        1W                      ; expire, seconds
                        1D )                    ; minimum, seconds
;
                NS      ns                      ; Inet Address of DNS server
                MX      10 mail.linux.bogus     ; Primary Mail Exchanger
                MX      20 mail.friend.bogus. ; Secondary Mail Exchanger
;
localhost       A       127.0.0.1
ns              A       192.168.196.2
mail            A       192.168.196.4
```

K záznamu SOA musí být poznamenány ještě dvě věci. `ns.linux.bogus` musí být skutečným strojem s A záznamem. Pro stroj zmíněný v záznamu SOA není povoleno mít záznam CNAME. Jeho

jméno nemusí být `ns` a může to být jakýkoliv povolený název hostitele. Dále `hostmaster.linux.bogus` by měl být čten jako `hostmaster@linux.bogus` a měl by to být poštovní alias nebo schránka, kde by měla osoba(y), spravující DNS, často číst poštu. Jakákoliv pošta týkající se domény, je odesílána na zde vypsanou adresu. Název nemusí být `hostmaster`, ale nějaká jiná povolená e-mailová adresa. Ale i ten `hostmaster` by měl být funkční.

V tomto souboru je jeden nový typ zdrojového záznamu, MX neboli Mail eXchanger (poštovní server). Sděluje poštovním systémům kam mají posílat poštu určenou na `někdo@linux.bogus`. V našem případě půjde pošta na servery `mail.linux.bogus` nebo `mail.friend.bogus`. Číslo před každým názvem stroje je priorita daného MX. Na MX s nejnižším číslem (10) by měla být pošta odesílána primárně. Jestliže to není možné, pošta je odeslána na MX s druhým nejvyšším číslem (sekundární), zde `mail.friend.bogus` s prioritou 20.

Spuštěním `ndc restart` znovu spusťte *named*. Výsledky zkontrolujte pomocí *nslookup*:

```
$ nslookup
> set q=any
> linux.bogus
Server: localhost
Address: 127.0.0.1

linux.bogus
        origin = ns.linux.bogus
        mail addr = hostmaster.linux.bogus
        serial = 199802151
        refresh = 28800 (8 hours)
        retry = 7200 (2 hours)
        expire = 604800 (7 days)
        minimum ttl = 86400 (1 day)
linux.bogus       nameserver = ns.linux.bogus
linux.bogus       preference = 10, mail exchanger = mail.linux.bogus.linux.bo-
gus
linux.bogus       preference = 20, mail exchanger = mail.friend.bogus
linux.bogus       nameserver = ns.linux.bogus
ns.linux.bogus    internet address = 192.168.196.2
mail.linux.bogus  internet address = 192.168.196.4
```

Po podrobném prozkoumání objevíte chybu. Řádek

```
linux.bogus preference = 10, mail exchanger = mail.linux.bogus.linux.bogus
```

je celý špatně. Měl by vypadat

```
linux.bogus preference = 10, mail exchanger = mail.linux.bogus
```

Tu chybu jsem udělal schválně, abyste se mohli poučit. Když se podíváte do zónového souboru, zjistíte, že v řádku

```
MX 10 mail.linux.bogus ; Primary Mail Exchanger
```

chybí tečka. Jestliže název stroje nekončí v zónovém souboru tečkou, na jeho konec je přidán počátek, takže máme dvojitý `linux.bogus.linux.bogus`. Proto správně je buď

```
MX 10 mail.linux.bogus. ; Primary Mail Exchanger
```

nebo

```
MX 10 mail ; Primary Mail Exchanger
```

Já dávám přednost druhé formě, je zde méně psaní. Existují úzkoprsí uživatelé, kteří teď nesouhlasí, ale také uživatelé, kteří souhlasí. V zónovém souboru by měla být doména buď vypsána a zakončena „.“, nebo by neměla být vůbec zmíněna a brala by se implicitně z počátku.

Musím zde znovu zdůraznit, že v `named.conf` se tečky za názvy neuvádějí. Nedokážete si představit, jak často tyto přebývající nebo chybějící tečky komplikují život.

Takže zde je nový zónový soubor s některými informacemi navíc:

```
;
; Zone file for linux.bogus
;
; The full zone file
;
@       IN      SOA     ns.linux.bogus.        hostmaster.linux.bogus. (
                        199802151              ; serial, todays date +
                                                 todays serial #
                        8H                     ; refresh, seconds
                        2H                     ; retry, seconds
                        1W                     ; expire, seconds
                        1D )                   ; minimum, seconds
;
                TXT     "Linux.Bogus, your DNS consultants"
                NS      ns              ; Inet Address of DNS server
                NS      ns.friend.bogus.
                MX      10 mail         ; Primary Mail Exchanger
                MX      20 mail.friend.bogus. ; Secondary Mail Exchanger

localhost       A       127.0.0.1

gw              A 1     92.168.196.1
                HINFO   "Cisco" "IOS"
                TXT     "The router"

ns               A      192.168.196.2
                MX      10 mail
                MX      20 mail.friend.bogus.
                HINFO   "Pentium" "Linux 2.0"
www             CNAME   ns

donald  A       192.168.196.3
                MX      10 mail
                MX      20 mail.friend.bogus.
                HINFO   "i486" "Linux 2.0"
                TXT     "DEK"

 mail           A       192.168.196.4
                MX      10 mail
                MX      20 mail.friend.bogus.
                HINFO   "386sx" "Linux 1.2"
```

```
ftp             A       192.168.196.5
                MX      10 mail
                MX      20 mail.friend.bogus.
                HINFO   "P6" "Linux 2.1.86"
```

Nalezneme zde množství nových zdrojových záznamů: HINFO (informace o hostiteli) má dvě části a je zvykem je oddělovat. První částí je hardware nebo procesor stroje a druhou částí je software nebo operační systém stroje. Stroj nazvaný `ns` má procesor (CPU) Pentium a používá Linux 2.0. CNAME (kanonické jméno) je způsob pojmenování stroje více názvy. Takže `www` je aliasem pro `ns`.

Používání záznamu CNAME je trochu kontroverzní. Bezpečné pravidlo ale říká, že záznamy MX, CNAME nebo SOA by se neměly odkazovat na záznam CNAME, měly by se odkazovat pouze na něco se záznamem A. Takže následující by bylo špatně

```
foobar  CNAME   www                     ; NO!
```

ale toto by bylo správně

```
foobar  CNAME   ns                      ; Yes!
```

Pro adresy elektronické pošty je také bezpečné předpokládat, že CNAME není platný název hostitele:

`webmaster@www.linux.bogus` je podle předchozího nastavení neplatnou adresou elektronické pošty. Můžete předpokládat, že pouze málo poštovních správců „zvenku“ bude toto pravidlo akceptovat (i když vám funguje dobře). Způsob, jakým se situace vyřeší, je použití A záznamů (a snad i dalších, jako jsou MX):

```
www             A       192.168.196.2
```

V určitých kruzích se použití CNAME nedoporučuje. Takže CNAME neberte příliš vážně. Ale tento dokument a množství hostitelů se tímto pravidlem neřídí.

Spuštěním `ndc reload` nahrajte novou databázi. Tím přinutíte *named*, aby svoje soubory znovu načetl.

```
$ nslookup
Default Server: localhost
Address: 127.0.0.1
> ls -d linux.bogus
```

To znamená, že všechny záznamy by měly být vypsány. Výsledek je tento:

```
[localhost]
$ORIGIN linux.bogus.
@               1D IN SOA       ns hostmaster (
                                199802151       ; serial
                                8H              ; refresh
                                2H              ; retry
                                1W              ; expiry
                                1D )            ; minimum

                1D IN NS        ns
                1D IN NS        ns.friend.bogus.
                1D IN TXT       "Linux.Bogus, your DNS consultants"
```

```
                1D IN MX        10 mail
                1D IN MX        20 mail.friend.bogus.
gw              1D IN A         192.168.196.1
                1D IN HINFO     "Cisco" "IOS"
                1D IN TXT       "The router"
mail            1D IN A         192.168.196.4
                1D IN MX        10 mail
                1D IN MX        20 mail.friend.bogus.
                1D IN HINFO     "386sx" "Linux 1.0.9"
localhost       1D IN A         127.0.0.1
www             1D IN CNAME     ns
donald          1D IN A         192.168.196.3
                1D IN MX        10 mail
                1D IN MX        20 mail.friend.bogus.
                1D IN HINFO     "i486" "Linux 1.2"
                1D IN TXT       "DEK"
ftp             1D IN A         192.168.196.5
                1D IN MX        10 mail
                1D IN MX        20 mail.friend.bogus.
                1D IN HINFO     "P6" "Linux 1.3.59"
ns              1D IN A         192.168.196.2
                1D IN MX        10 mail
                1D IN MX        20 mail.friend.bogus.
                1D IN HINFO     "Pentium" "Linux 1.2"
@               1D IN SOA       ns hostmaster (
                                199802151       ; serial
                                8H      ; refresh
                                2H      ; retry
                                1W      ; expiry
                                1D )    ; minimum
```

Toto je správné. Jak je vidět, vypadá to spíše jako samotný zónový soubor. Podívejme se, co se zde říká o samotném `www`:

```
> set q=any
> www.linux.bogus.
Server: localhost
Address: 127.0.0.1

www.linux.bogus canonical name = ns.linux.bogus
linux.bogus    nameserver = ns.linux.bogus
linux.bogus    nameserver = ns.friend.bogus
ns.linux.bogus internet address = 192.168.196.2
```

Jinými slovy `www.linux.bogus` má pravý název `ns.linux.bogus` a dává vám některé informace o `ns`, postačující k připojení.

Nyní máme polovinu za sebou.

Reverzní zóna

Nyní mohou programy převádět názvy z `linux.bogus` na adresy, ke kterým se mohou připojit. Nutná je ale také reverzní zóna, která DNS umožňuje převod adresy na název. Tento název je využíván mnoha servery různých druhů (FTP, IRC, WWW a další), aby se rozhodly, jestli s vámi bu-

dou mluvit nebo ne, a když se rozhodnou kladně, tak se ještě musí určit, jaká obdržíte práva. Pro plný přístup ke všem službám Internetu je nutná reverzní zóna.

Následující vložte do `named.conf`:

```
zone "196.168.192.in-addr.arpa" {
        notify no;
        type master;
        file "pz/192.168.196";
        } ;

Je to stejná situace jako u 0.0.127.in-addr.arpa a obsahy jsou podobné:
@       IN      SOA     ns.linux.bogus. hostmaster.linux.bogus. (
                        199802151       ; Serial, todays date + todays serial
                        8H              ; Refresh
                        2H              ; Retry
                        1W              ; Expire
                        1D)             ; Minimum TTL
                NS      ns.linux.bogus.

1               PTR     gw.linux.bogus.
2               PTR     ns.linux.bogus.
3               PTR     donald.linux.bogus.
4               PTR     mail.linux.bogus.
5               PTR     donald.linux.bogus.
```

Nyní znovu spusťte *named* (`ndc restart`) a znovu otestujte výsledky vaší práce pomocí *nsloo-kup*:

```
> 192.168.196.4
Server: localhost
Address: 127.0.0.1

Name: mail.linux.bogus
Address: 192.168.196.4
```

vypadá to v pořádku, takže vyzkoušejte úplně všechno:

```
> ls -d 196.168.192.in-addr.arpa
[localhost]
$ORIGIN 196.168.192.in-addr.arpa.
@               1D IN SOA       ns.linux.bogus. hostmaster.linux.bogus. (
                                199802151       ; serial
                                8H              ; refresh
                                2H              ; retry
                                1W              ; expiry
                                1D )            ; minimum

                1D IN NS        ns.linux.bogus.
1               1D IN PTR       gw.linux.bogus.
2               1D IN PTR       ns.linux.bogus.
3               1D IN PTR       donald.linux.bogus.
4               1D IN PTR       mail.linux.bogus.
5               1D IN PTR       donald.linux.bogus.
```

```
@               1D IN SOA       ns.linux.bogus. hostmaster.linux.bogus. (
                                199802151       ; serial
                                8H              ; refresh
                                2H              ; retry
                                1W              ; expiry
                                1D )            ; minimum
```

Opět správně! Pokud by výstup vypadal jinak, podívejte se do syslogu (jak to udělat jsem vysvětloval na začátku kapitoly).

Upozornění

Tady bych měl doplnit ještě několik věcí. IP adresy, které jsem v příkladu využil, jsou vyňaty z bloku „privátních sítí“. Tyto adresy není povoleno na Internetu veřejně používat. Proto je jejich použití v našem příkladu bezpečné. Další záležitostí je řádek `notify no;`. Named z něho zjistí, že při aktualizaci jednoho ze svých zónových souborů nemá upozorňovat svoje sekundární servery. V bind-8 může named při aktualizaci zóny upozornit další servery, vypsané v záznamech NS. To je vhodné pro pravidelné využívání, ale pro soukromé experimenty se zónami by tato funkce měla být vypnuta. Přece nechceme, aby náš experiment zamořil Internet.

A samozřejmě musím dodat, že tato doména a její adresy jsou všechny smyšlené. Příklad skutečné domény bude v následující části.

Proč nefunguje reverzní hledání

Při reverzním vyhledávání se vyskytují některé „úskoky“, které se normálního vyhledávání obvykle netýkají. Než budeme pokračovat, potřebujete, aby reverzní vyhledávání ve vaší doméně fungovalo alespoň na vašem DNS serveru. Pokud nefunguje, vraťte se zpět a napravte chybu.

Zmíníme se o dvou problémech s reverzním vyhledáváním, které se někdy projevují „vně“ Internetu.

Reverzní zóna není delegována

Když požádáte poskytovatele síťových služeb o rozsah síťových adres a o registraci názvu domény, rozumí se samo sebou, že zajistí i delegaci názvu domény. Delegace je reprezentována takzvanými „tmelicími“ záznamy v DNS, které zajišťují přechod z jednoho serveru na jiný tak, jak jsme si to demonstrovali v teoretickém popisu. Četli jste jej, ne? Pokud vám reverzní hledání nefunguje, tak si jej přečtěte teď.

I reverzní zóna musí být delegována. Pokud používáte síť `192.168.196` a doménu `linux.bogus`, musí váš poskytovatel zanést do DNS NS záznamy jak pro „normální“ zónu, tak i pro reverzní zónu. Když zkusíte sledovat řetězec `in-addr.arpa` směrem k vaší síti, narazíte někde na přerušený řetězec. Pravděpodobně někde na serveru vašeho poskytovatele. Jakmile přerušení naleznete, kontaktujte poskytovatele a požádejte jej o nápravu.

Máte beztřídovou podsíť

Teď se dostáváme k trochu složitějšímu tématu, nicméně beztřídové podsítě jsou dnes velmi běžné a pokud nejste střední či větší společnost, pravděpodobně ji máte také.

Beztřídové podsítě jsou věc, která spasila dnešní Internet. Několik let zpátky bylo hodně problémů kolem docházejících IP adres. Pár chytrých lidí od IETF (Internet Engineering Task Force, sdružení, které udržuje Internet v provozu) dalo hlavy dohromady a vymysleli řešení. Ovšem za jistou cenu. A tou cenou je, že dostanete méně než adresu třídy C a některé věci nemusí fungovat. Ze-

ptejte se „pana DNS“ na adrese *http://www.acmebw.com/askmrdns/archive.php?question=7* a dozvíte se, oč jde a jak to řešit.

Ne? Tak to udělejte, protože já to vysvětlovat nebudu.

První část problému spočívá v tom, že poskytovatel musí rozumět technice, kterou pan DNS doporučuje. Ne všichni drobnější poskytovatelé tomu rozumí. Pokud to ten váš nechápe, tak jim to pečlivě vysvětlete. Nejprve musíte mít jistotu, že vám porozuměli. Pak na svém serveru vytvoří pěknou reverzní zónu, kterou budete moci programem *nslookup* otestovat.

Druhá a poslední část problému spočívá v tom, že sami musíte rozumět použitému postupu. Pokud si nejste jisti, přečtěte si popis ještě jednou. Pak si můžete vlastní reverzní zónu klidně nastavit sami.

Skrývá se v tom ještě jedna záludnost. Starší resolvery nedokáží přijmout trik se záznamem `CNAME` v řetězci názvů a nepodaří se jim reverzní převod názvu. To může vést k situacím, že vám budou přiřazena chybná přístupová práva, bude vám odepřen přístup k určitým službám a podobně. Pokud taková situace nastane, jediná možnost je domluvit se s poskytovatelem, aby umístil záznam PTR vašeho počítače přímo do jejich reverzní zóny a vyhnete se tak použití triku se záznamem `CNAME`.

Někteří poskytovatelé s takovým řešením počítají a nabízejí například webové formuláře nebo jiná automatická řešení, jak zajistit reverzní překlad adres ve vaší doméně.

Příklad skutečné domény

Kde nalezneme některé soubory skutečných domén

Uživatelé navrhli, abych vedle ukázkového příkladu použil příklad skutečné funkční domény.

Příklad jsem použil se souhlasem Davida Bullocka z LAND-5. Tyto soubory byly aktuální 24. 9. 1996 a potom byly mnou upraveny, aby odpovídaly omezením bind-8 a využívaly nějaké mnou navržené doplňky. Takže to, co uvidíte v dalším textu, se bude lišit od současné situace na DNS serverech LAND-5.

/etc/named.conf (nebo /var/named/named.conf)

Zde nalezneme základní informace o dvou potřebných reverzních zónách: síti `127.0.0` a podsíti `206.6.177` společnosti LAND-5. Zároveň zde uvidíme hlavní záznam přímé zóny `land-5.com`. Všimněte si také, že místo hromadění souborů v adresáři `pz` (jak to dělám já v tomto dokumentu) jsou soubory v adresáři `zone`.

```
// Boot file for LAND-5 DNS server

options {
        directory "/var/named";
} ;

zone "." {
        type hint;
        file "root.hints";
} ;

zone "0.0.127.in-addr.arpa" {
        type master;
```

```
        file "zone/127.0.0";
} ;

zone "land-5.com" {
        type master;
        file "zone/land-5.com";
} ;

zone "177.6.206.in-addr.arpa" {
        type master;
        file "zone/206.6.177";
} ;
```

Pokud budete chtít tento příklad použít jako začátek pro váš soubor `named.conf`, uveďte PROSÍM v obou částech zóny `land-5` parametr „`notify no`“, aby se předešlo zmatkům v DNS systému.

/var/named/root.hints

Tento soubor má dynamickou povahu a ten, který si prohlížíte, už je zastaralý. Použijte proto novější, který můžete získat například nástrojem *dig*.

```
; <<>> DiG 8.1 <<>> @A.ROOT-SERVERS.NET. =
; (1 server found)
;; res options: init recurs defnam dnsrch
;; got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10
;; flags: qr aa rd; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 13
;; QUERY SECTION:
;; ., type = NS, class = IN

;; ANSWER SECTION:
.                       6D IN NS                G.ROOT-SERVERS.NET.
.                       6D IN NS                J.ROOT-SERVERS.NET.
.                       6D IN NS                K.ROOT-SERVERS.NET.
.                       6D IN NS                L.ROOT-SERVERS.NET.
.                       6D IN NS                M.ROOT-SERVERS.NET.
.                       6D IN NS                A.ROOT-SERVERS.NET.
.                       6D IN NS                H.ROOT-SERVERS.NET.
.                       6D IN NS                B.ROOT-SERVERS.NET.
.                       6D IN NS                C.ROOT-SERVERS.NET.
.                       6D IN NS                D.ROOT-SERVERS.NET.
.                       6D IN NS                E.ROOT-SERVERS.NET.
.                       6D IN NS                I.ROOT-SERVERS.NET.
.                       6D IN NS                F.ROOT-SERVERS.NET.

;; ADDITIONAL SECTION:
G.ROOT-SERVERS.NET.    5w6d16h IN A             192.112.36.4
J.ROOT-SERVERS.NET.    5w6d16h IN A             198.41.0.10
K.ROOT-SERVERS.NET.    5w6d16h IN A             193.0.14.129
L.ROOT-SERVERS.NET.    5w6d16h IN A             198.32.64.12
M.ROOT-SERVERS.NET.    5w6d16h IN A             202.12.27.33
A.ROOT-SERVERS.NET.    5w6d16h IN A             198.41.0.4
H.ROOT-SERVERS.NET.    5w6d16h IN A             128.63.2.53
B.ROOT-SERVERS.NET.    5w6d16h IN A             128.9.0.107
```

```
C.ROOT-SERVERS.NET.    5w6d16h IN A          192.33.4.12
D.ROOT-SERVERS.NET.    5w6d16h IN A          128.8.10.90
E.ROOT-SERVERS.NET.    5w6d16h IN A          192.203.230.10
I.ROOT-SERVERS.NET.    5w6d16h IN A          192.36.148.17
F.ROOT-SERVERS.NET.    5w6d16h IN A          192.5.5.241

;; Total query time: 215 msec
;; FROM: roke.uio.no to SERVER: A.ROOT-SERVERS.NET. 198.41.0.4
;; WHEN: Sun Feb 15 01:22:51 1998
;; MSG SIZE sent: 17 rcvd: 436
```

/var/named/zone/127.0.0

Velmi jednoduché – povinný záznam SOA a záznam, který mapuje `127.0.0.1` na `localhost`. Nutné jsou oba. V tomto souboru by nemělo být nic dalšího. Tyto soubory nemusí být nikdy aktualizovány, pokud se nezmění adresy vašeho DNS serveru.

```
@               IN      SOA     land-5.com. root.land-5.com. (
                                199609203       ; Serial
                                28800           ; Refresh
                                7200            ; Retry
                                604800  ; Expire
                                86400)  ; Minimum TTL
                        NS      land-5.com.

1                       PTR     localhost.
```

/var/named/zone/land-5.com

Zde vidíme povinný záznam SOA a potřebné záznamy NS. Je vidět, že sekundární DNS server je `ns2.psi.net`. Tak to má být – jako zálohu vždy používat sekundární server na jiné síti. Je zde také patrné, že jako hlavní hostitel je zde `land-5`, který se stará o mnoho různých internetových služeb. Využívá k tomu záznamy CNAME (jako alternativy záznamů A).

Jak je vidět ze záznamu SOA, zónové soubory pochází z `land-5.com`, kontaktní osobou je `root@land-5.com`. Další často používanou adresou kontaktní osoby je `hostmaster`. Sériové číslo je v obvyklém formátu yyyymmdd (rok, měsíc, den) s připojenými sériovými čísly za daný den – zde se jedná pravděpodobně o šestou verzi ze dne 20. 9. 1996. Uvědomte si, že sériové číslo se musí stále zvyšovat, zde máme pouze jednu číslici pro dnešní sériová čísla, takže po 9 editacích souboru se může v jeho editacích pokračovat až následující den. Zvažte využití dvou číslic.

```
@       IN      SOA l   and5.com. root.land-5.com. (
                        199609206       ; serial, todays date
                                          + todays serial #
                        8H              ; refresh, seconds
                        2H              ; retry, seconds
                        1W              ; expire, seconds
                        1D )            ; minimum, seconds
                NS      land-5.com.
                NS      ns2.psi.net.
                MX 10   land-5.com.     ; Primary Mail Exchanger

localhost       A       127.0.0.1
```

```
router  A       206.6.177.1

land-5.com.     A       206.6.177.2
ns              A       206.6.177.3
www             A       207.159.141.192

ftp             CNAME   land-5.com.
mail            CNAME   land-5.com.
news            CNAME   land-5.com.

funn            A       206.6.177.2
@               TXT     "LAND-5 Corporation"

;
;       Workstations
;
ws-177200       A       206.6.177.200
                MX      10 land-5.com. ; Primary Mail Host
ws-177201       A       206.6.177.201
                MX      10 land-5.com. ; Primary Mail Host
ws-177202       A       206.6.177.202
                MX      10 land-5.com. ; Primary Mail Host
ws-177203       A       206.6.177.203
                MX      10 land-5.com. ; Primary Mail Host
ws-177204       A       206.6.177.204
                MX      10 land-5.com. ; Primary Mail Host
ws-177205       A       206.6.177.205
                MX      10 land-5.com. ; Primary Mail Host
; { Many repetitive definitions deleted - SNIP}
ws-177250       A       206.6.177.250
                MX      10 land-5.com. ; Primary Mail Host
ws-177251       A       206.6.177.251
                MX      10 land-5.com. ; Primary Mail Host
ws-177252       A       206.6.177.252
                MX      10 land-5.com. ; Primary Mail Host
ws-177253       A       206.6.177.253
                MX      10 land-5.com. ; Primary Mail Host
ws-177254       A       206.6.177.254
                MX      10 land-5.com. ; Primary Mail Host
```

Jestliže otestujete nameserver `land-5`, zjistíte, že názvy hostitelů mají formát `ws_`*`číslo`*. Od pozdějších verzí bind 4 začal *named* omezovat znaky, které mohou být použity v názvech hostitelů. Takže v bind-8 by to nefungovalo a já jsem pomlčkami „-“ nahradil podtržítka „_“.

Dále za zmínku stojí to, že pracovní stanice nemají jednotlivé názvy, ale spíše předpony, následované posledními dvěma částmi IP adres. Použití takové konvence značně zjednoduší údržbu, ale je trochu neosobní a může být zdrojem odporu vašich uživatelů.

Zřejmě je také `funn.land-5.com` aliasem pro `land-5.com`, ale s použitím záznamu A, ne CNAME. Jak už bylo řečeno, jedná se o rozumný způsob nastavení.

/var/named/zone/206.6.177

Tento soubor ještě později okomentuji.

```
@               IN      SOA     land-5.com. root.land-5.com. (
                                199609206       ; Serial
                                28800           ; Refresh
                                7200            ; Retry
 6                              04800           ; Expire
                                86400)          ; Minimum TTL
                                NS              land-5.com.
                                NS              ns2.psi.net.
;
;       Servers
;
1               PTR     router.land-5.com.
2               PTR     land-5.com.
2               PTR funn.land-5.com.
;
;       Workstations
;
200             PTR     ws-177200.land-5.com.
201             PTR     ws-177201.land-5.com.
202             PTR     ws-177202.land-5.com.
203             PTR     ws-177203.land-5.com.
204             PTR     ws-177204.land-5.com.
205             PTR     ws-177205.land-5.com.
; { Many repetitive definitions deleted - SNIP}
250             PTR     ws-177250.land-5.com.
251             PTR     ws-177251.land-5.com.
252             PTR     ws-177252.land-5.com.
253             PTR     ws-177253.land-5.com.
254             PTR     ws-177254.land-5.com.
```

Reverzní zóna je částí konfigurace, která způsobuje nejvíce neštěstí. Používá se k nalezení názvu hostitele, když máte IP adresu stroje. Příklad: jste IRC server a přijímáte připojení IRC klientů. Jste ale norský IRC server a chcete přijímat připojení pouze z Norska a dalších skandinávských zemí. Když obdržíte od klienta připojení, můžete zjistit IP adresu připojovaného stroje, protože IP adresa klienta je obsažena ve všech paketech posílaných po síti. Nyní můžete zavolat funkci *gethostbyaddr*, která vyhledá název hostitele podle IP adresy. Tato funkce požádá DNS server o nalezení názvu Předpokládejme, že se klient připojuje z ws-177200.land-5.com. IP adresa, kterou zjistí IRC server, je 206.6.177.200. Abychom nalezli název stroje, musíme nalézt záznam `200.177.6.206.in-addr.arpa`. DNS server nejprve nalezne servery `arpa`, potom servery `in-addr.arpa`, pak bude pokračovat obrácenou cestou přes 206, poté 6, až nakonec nalezne server pro zónu `177.6.206.in-addr.arpa` na `land-5`. Odtud získá odpověď, že pro `200.177.6.206.in-addr.arpa` máme záznam `PTR ws-177200.land-5.com`, což znamená, že název, který odpovídá `206.6.177.200`, je `ws-177200.land-5.com`. Stejně jako u příkladu vyhledávání `prep.ai.mit.edu` je toto vysvětlení zjednodušené.

Vraťme se k příkladu IRC serveru. IRC server přijímá připojení pouze ze skandinávských zemí (*.no, *.se, *.dk), přičemž název `ws-177200.land-5.com` zjevně neodpovídá podmínce a server spojení nepovolí. Kdyby zde nebylo žádné zpětné mapování `206.6.177.200` přes zónu `in-addr.arpa`, server by vůbec nebyl schopen nalézt název. Musel by `206.6.177.200` srovnávat s `*.no`, `*.se` a `*.dk`, takže by nic neodpovídalo.

Někteří lidé vám řeknou, že zpětná mapování jsou důležitá pouze pro servery nebo nejsou důležitá vůbec. Ne tak zcela: Množství serverů FTP, news, IRC a dokonce HTTP (WWW) nepřijme při-

pojení ze stroje, ke kterému nejsou schopny nalézt název. Takže zpětné mapování je u strojů v podstatě povinné.

Údržba

Udržujte v chodu

U named musíte kromě samotného „udržování v chodu“ zajistit ještě jeden aspekt údržby. Jedná se o aktualizace souboru `root.hints`. Nejjednodušším způsobem je použití programu `dig`. Nejprve jej spusťte bez parametrů, tak získáte `root.hints` podle vlastního serveru. Poté požádejte jeden z vypsaných kořenových serverů pomocí `dig @rootserver`. Zjistíte, že výstup vypadá velmi podobně jako soubor `root.hints`. Uložte jej do souboru (`dig @e.root-servers.net .ns >root.hints.new`) a nahraďte jím starý `root.hints`.

Po nahrazení souboru nezapomeňte znovu spustit `named`.

Al Longyear mi poslal následující skript, kterým se aktualizuje `root.hints`, můžete jej spouštět automaticky jednou za měsíc démonem `cron` a zapomenout na něj. Skript předpokládá, že máte funkční poštu a že je definován poštovní alias *hostmaster*. Aby skript odpovídal vašemu nastavení, musíte jej ještě upravit.

```
#!/bin/sh
#
# Update the nameserver cache information file once per month.
# This is run automatically by a cron entry.
#
(
echo "To: hostmaster <hostmaster>"
echo "From: system <root>"
echo "Subject: Automatic update of the root.hints file"
echo

export PATH=/sbin:/usr/sbin:/bin:/usr/bin:
cd /var/named

dig @rs.internic.net . ns >root.hints.new

echo "The root.hints file has been updated to contain the following
information:"
echo
cat root.hints.new

chown root.root root.hints.new
chmod 444 root.hints.new
rm -f root.hints.old
mv root.hints root.hints.old
mv root.hints.new root.hints
ndc restart
echo
echo "The nameserver has been restarted to ensure that the update is
    complete."
echo "The previous root.hints file is now called
```

```
/var/named/root.hints.old."
) 2>&1 | /usr/lib/sendmail -t
exit 0
```

Někteří z vás možná zjistili, že soubor `root.hints` je přes FTP k dispozici na Internicu. Prosím, nepoužívejte FTP k aktualizaci `root.hints`. Výše popsaná metoda je pro síť daleko přijatelnější.

Přechod z verze 4 na verzi 8

Tuto část původně napsal David E. Smith (*dave@bureau42.ml.org*) a zabývala se použitím programu bind 8. Doplnil jsem ji tak, aby odpovídala svému novému názvu.

Není zde moc co říct. Kromě použití `named.conf` místo `named.boot` je všechno stejné. A bind 8 se dodává s perlovým skriptem, který převádí soubory starých verzí na nové. Ukázkový soubor `named.boot` (stará verze) pro DNS servery pouze s vyrovnávací pamětí:

```
directory /var/named
cache .  root.hints
primary 0.0.127.IN-ADDR.ARPA 127.0.0.zone
primary localhost  localhost.zone
```

Na příkazovém řádku zadejte v adresáři `bind8/src/bin/named` (zde se předpokládá, že máte zdrojovou distribuci, jestliže máte binární – skript někde je, ale já nevím kde – ed.):

```
./named-bootconf.pl < named.boot > named.conf
```

Čímž se vytvoří `named.conf`:

```
// generated by named-bootconf.pl

options {
        directory "/var/named";
} ;

zone "." {
        type hint;
        file "root.hints";
} ;

zone "0.0.127.IN-ADDR.ARPA" {
        type master;
        file "127.0.0.zone";
} ;

zone "localhost" {
        type master;
        file "localhost.zone";
} ;
```

Funguje to pro všechno, co se může v souboru `named.boot` objevit, ačkoliv se nepřidávají všechna nová rozšíření a konfigurační volby, které bind 8 umožňuje. Zde následuje úplnější `named.conf`, který provádí to stejné, ale trochu efektivněji.

```
// This is a configuration file for named (from BIND 8.1 or later).
```

```
// It would normally be installed as /etc/named.conf.
// The only change made from the `stock' named.conf (aside from this
// comment :) is that the directory line was uncommented, since I
// already had the zone files in /var/named.

options {
        directory "/var/named";
        check-names master warn;  /* default. */
        datasize 20M;
} ;

zone "localhost" IN {
        type master;
        file "localhost.zone";
        check-names fail;
        allow-update {  none; } ;
        allow-transfer {  any; } ;
} ;

zone "0.0.127.in-addr.arpa" IN {
        type master;
        file "127.0.0.zone";
        check-names fail;
        allow-update {  none; } ;
        allow-transfer {  any; } ;
} ;

zone "." IN {
        type hint;
        file "root.hints";
} ;
```

Toto všechno najdete v adresáři `bind8/src/bin/named/test`, kde jsou i kopie zónových souborů, které můžete ihned využít.

Formáty zónových souborů a souborů `root.hints` jsou shodné, stejně jako příkazy pro jejich aktualizace.

Otázky a odpovědi

Než mi budete psát, přečtěte si tuto kapitolu.

Můj named požaduje soubor `named.boot`.

Čtete špatný dokument. Přečtěte si prosím starou verzi tohoto dokumentu, která pokrývá bind 4. Naleznete ji na *http://www.math.uio.no/janl/DNS/*.

Jak mám DNS používat zpoza firewallu?

Zkuste `forward only`; Kromě toho ještě asi bude nutné v části „options“ souboru `named.conf` (jak bylo uvedeno v části věnované caching-only serverům) uvést

```
query-source port 53;
```

Jak DNS přinutit k cyklickému procházení adres, které jsou k dispozici pro nějakou službu (řekněme například *www.busy.site*), abychom dosáhli distribuce zátěže?

Vytvořte pro *www.busy.site* několik A záznamů a používejte bind 4.9.3 nebo pozdější. Bind cyklicky obslouží odpovědi. Nebude to fungovat ve starších verzích bind.

Chci DNS nastavit na (uzavřeném) intranetu. Co mám dělat?

Zbavte se souboru `root.hints` a vytvořte zónové soubory. Znamená to také, že nemusíte tento soubor periodicky aktualizovat.

Jak mám nastavit sekundární (podřízený) DNS server?

Jestliže má primární server adresu 127.0.0.1, pak na svém sekundárním vložíte do souboru `named.conf` tento řádek:

```
zone "linux.bogus" {
  type slave;
  file "sz/linux.bogus";
  masters {  127.0.0.1; } ;
} ;
```

V seznamu masters můžete vypsat několik alternativních hlavních serverů, ze kterých je možné kopírovat zónu. Oddělujte je středníkem.

Chci, aby bind běžel, i když jsem ze sítě odpojen.

Na to lze odpovědět více způsoby

- Od Iana Clarka (*ic@deakin.edu.au*) jsem obdržel dopis, ve kterém popisuje způsob, který k tomu používá:

  Named spouštím na svém maškarádovacím stroji. Mám dva soubory `root.hints` – jeden se jmenuje `root.hints.real` (obsahuje pravé názvy kořenových serverů) a druhý se jmenuje `root.hints.fake` a obsahuje

  ```
  ----
  ; root.hints.fake
  ; tento soubor neobsahuje žádné informace
  ----
  ```

  Když se odpojuji, kopíruji soubor `root.hints.fake` na `root.hints` a znovu spouštím `named`.

  Když se připojuji, kopíruji soubor `root.hints.real` na `root.hints` a znovu spouštím `named`.

  Tohle provádím přes `ip-down` (respektive `ip-up`).

  Jakmile poprvé požaduji název domény, když nejsem připojen, `named` nemá k dispozici žádné podrobnosti, takže oznámí následující zprávu..

  ```
  Jan 28 20:10:11 hazchem named[10147]: No root nameserver for class IN
  ```

  s čímž dokáži vyžít.

  U mě vše spolehlivě funguje. DNS server mohu pro lokální stroje využívat bez pauz při neúspěšném vyhledávání externích názvů domén (když nejsem připojen k Internetu). Když jsem připojen k Internetu, vyhledávání probíhá normálně.

- Kromě toho jsem od Karla-Maxe Wangera dostal informaci o tom, jak na většinu času odpojeném počítači spolupracuje bind s NFS a portmapperem.

  Používám vlastní *named* na všech počítačích, které jsou k Internetu připojeny jen občas modemem. Nameserver funguje pouze jako cache, není autoritativní pro žádnou zónu a na všechno se ptá kořenových serverů podle souboru *root.cache*. Jak je u Slackware obvyklé, spouští se před *nfsd* a *mountd*.

  Na jednom z počítačů (notebook Libretto 30) jsem měl problém, protože občas jsem si jej sice mohl připojit z jiných systémů v mé lokální síti, ale ve většině případů to nefungovalo. Efekt byl stejný při použití PLIP, PCMCIA síťové karty nebo PPP přes sériové rozhraní.

  Po nějaké době hádání a experimentování jsem zjistil, *named* zjevně způsobil zmatek v procesu registrace *nfsd* a *mountd* se pak nedohodne s *portmapperem* (všechny démony spouštím obvyklým postupem při bootování systému). Když se *named* spustí až po *nfsd* a *mountd*, problém se odstraní.

  Protože tato změna v sobě neskrývá žádné nevýhody, doporučuji ji provést všem, aby se předešlo případným problémům.

- Konečně se tomuto tématu věnuje HOWTO na stránkách *Ask Mr. DNS* na adrese *http://www.acmebw.com/askmrdns/#linux-ns*. Je zaměřen na bind 4, takže pro bind 8 jej musíte upravit.

Kam si DNS-server s vyrovnávací pamětí ukládá vyrovnávací paměť? Je zde nějaký způsob, jakým bych mohl kontrolovat její velikost?

Vyrovnávací paměť je celá uložena v paměti, nikdy se neukládá na disk. Kdykoliv ukončíte named, vyrovnávací paměť je ztracena. Vyrovnávací paměť *není* možné žádným způsobem řídit. Jestli k tomu máte přesto chuť, musíte si programově upravit celý named. To samozřejmě příliš nedoporučujeme.

Ukládá named svoji vyrovnávací paměť mezi jednotlivými restarty? Mohu ho přinutit, aby ji uložil?

Ne, named při svém zániku vyrovnávací paměť neukládá. To znamená, že vyrovnávací paměť se musí při každém restartu znovu vytvořit. Neexistuje žádný způsob, jak named přinutit uložit svoji vyrovnávací paměť do souboru. Jestli k tomu máte přesto chuť, musíte si opět upravit celý named. To ale samozřejmě příliš nedoporučujeme.

Jak můžu získat doménu? Chci si nastavit doménu pojmenovanou (řekněme) *linux-rules.net*.

Kontaktujte svého poskytovatele připojení. Měl by být schopen vám pomoci. Mějte na paměti, že ve většině zemí budete muset za vlastní doménu platit.

Jak se stát lepším správcem DNS

Dokumentace a nástroje

Vhodná dokumentace existuje. Na síti i v tištěné podobě. K tomu, aby se z obyčejného správce DNS stal správce lepší, je nutné si část této dokumentace pročíst. Tištěná dokumentace, to je zejména kniha *DNS and BIND* od C. Liu a P. Albitze, vydaná v O'Reilly & Associates, Sebastopol, CA, ISBN 0-937175-82-X. Četl jsem ji, je skvělá a i když je určena pro bind 4, nepředstavuje to problém. O DNS pojednává také část knihy *TCP/IP Network Administration* od Craiga Hunta, vydaná

v O'Reilly..., ISBN 0-937175-82-X. Další nutnou koupí pro dobrou správu DNS (nebo spíše něčeho jiného) je *Zen and the Art of Motorcycle Maintenance* od Roberta M. Prisiga :-). K dispozici pod ISBN 0688052304 a další.

Na síti naleznete další zajímavosti na *http://www.dns.net/dnsrd/* (DNS Resources Directory), *http://www.isc.org/bind.html*; jsou to FAQ, referenční příručka (BOG, Bind Operation Guide) a definice protokolů spolu s úpravami DNS (většina z toho, plus některá níže zmíněná RFC jsou obsažena v distribuci bindu). Většinu z toho jsem nečetl, ale já také nejsem žádný skvělý správce. Na druhou stranu Arnt Gulbrandsen četl BOG a líbilo se mu to :-). K DNS se vztahuje skupina *news–comp.protocols.tcp-ip.domains*. Navíc k DNS existuje množství RFC, z nichž nejdůležitější jsou pravděpodobně tato:

RFC 2052

A. Gulbrandsen, P. Vixie, *A DNS RR for specifying the location of services (DNS SRV)*, October 1996

RFC 1918

Y. Rekhter, R. Moskowitz, D. Karrenberg, G. de Groot, E. Lear,

Addre*ss Allocation for Private Internets*, 02/29/1996.

RFC 1912

D. Barr, *Common DNS Operational and Configuration Errors*, 02/28/1996.

RFC 1912 Errors

B. Barr, *Errors in RFC 1912*, k dispozici také na *http://www.cis.ohio-state.edu/~barr/rfc1912-errors.html*

RFC 1713

A. Romao, *Tools for DNS debugging*, 11/03/1994.

RFC 1712

C. Farrell, M. Schulze, S. Pleitner, D. Baldoni, *DNS Encoding of Geographical Location*, 11/01/1994.

RFC 1183

R. Ullmann, P. Mockapetris, L. Mamakos, C. Everhart, *New DNS RR Definitions*, 10/08/1990.

RFC 1035

P. Mockapetris, *Domain names – implementation and specification*, 11/01/1987.

RFC 1034

P. Mockapetris, *Domain names – concepts and facilities*, 11/01/1987.

RFC 1033

M. Lottor, *Domain administrators operations guide*, 11/01/1987.

RFC 1032

M. Stahl, *Domain administrators guide*, 11/01/1987.

RFC 974

C. Partridge, *Mail routing and the domain system*, 01/01/1986.

KAPITOLA 6

El Dev, alavoor@yahoo.com

Jádro Linuxu

Úvod

Patříte mezi ty, kteří by si měli tento dokument přečíst? Ano, pokud jste se setkali s některou z následujících situací:

- „Tento balíček 46.4.6 tvrdí, že potřebuje jádro 2.8.193 a vy máte pouze jádro 1.0.9!“
- Ovladač vašeho nového zařízení je jen v novém jádru.
- Nemáte představu o tom, jak přeložit jádro.
- „Jsou ty nesmysly v souboru README opravdu pravdivé?“
- Zkusili jste něco a přesto to nefunguje?
- Potřebujete zodpovědět dotazy někoho dalšího o instalaci jádra.

Začínáme

Některé příklady v tomto dokumentu předpokládají, že máte nainstalovány utility tar, find a xargs. Jedná se o naprostý standard – s tímto požadavkem by neměly být žádné problémy. Předpokládáme také, že se vyznáte ve struktuře souborového systému – pokud ne, je velmi důležité, abyste si uschovali výstup příkazu `mount` (či obsah souboru `/etc/fstab`, pokud k němu máte přístup). Tato informace je důležitá a nezmění se, pokud nebudete přerozdělovat pevný disk, přidávat či přeinstalovávat operační systém či něco podobného.

Poslední produkční verze jádra je v okamžiku vzniku tohoto dokumentu 2.2.9, takže veškeré odkazy a příklady vychází z tohoto jádra. I přesto jsme se v mezích možností snažili tento dokument vytvořit tak, aby nebyl na konkrétní verzi jádra závislý. Jádro je neustále ve vývoji a každá nová verze vyvolá několik změn. Tato skutečnost by neměla způsobit příliš vážné problémy, ale drobné zmatky možná ano.

Existují dvě základní verze linuxového jádra – produkční a vývojová. Produkční verze jsou označovány sudým číslem na druhém místě v čísle verze. Produkční verze jsou tak 1.2.x, 2.0.x, 2.2.x či 2.4.x. Tyto verze jsou stabilní a bezchybné. Vývojové verze (například 2.1.x, 2.3.x atd.) jsou jádra určená pro testování a odhalování chyb.

Typografické konvence

`Text v tomto formátu` reprezentuje výstupy na obrazovce, jména souborů a cokoli, co zadáváte z klávesnice, jako jsou příkazy či volby příkazů. Příkazy a jejich výstupy jsou někdy uvozeny apostrofy, což způsobuje následující klasický problém: pokud se tečka objeví na konci věty, často ji uživatelé zadají společně s příkazem. Americká typografická konvence totiž přikazuje umísťovat tečku v rámci citace. Příklady v tomto dokumentu vychází z toho, že tečka není součástí ci-

tace – pokud tedy bude nutno zadat příkaz „make config“, bude v tomto dokumentu uvedeno `'make config'` , případně (`make config`) a ne `'make config.'`

Stručný přehled překladu jádra

Nejnovější verze tohoto dokumentu je dostupná na adrese http://www.aldev.8m.com (v sekci Quick Steps to recompile kernel). Dokument je také dostupný na adrese http://aldev.webjump.com/ a serverech angelfire (http://www.angelfire.com/nv/aldev), geocities (http://www.geocities.com/alavoor/index.html), virtualave (http://aldev.virtualave.net/), bizland (http://aldev.bizland.com/), theglobe (http://members.theglobe.com/aldev/index.html), spree (http://members.spree.com/technology/aldev), infoseek (http://homepages.infoseek.com/~aldev1/index.html), bcity (http://www3.bcity.com/aldev), 50megs (http://aldev.50megs.com/), NBCi (http://members.nbci.com/alavoor), Terrashare (http://aldev.terrashare.com/), Fortunecity (http://members.fortunecity.com/aldev), Freewebsites (http://aldev.freewebsites.com/) a Tripod (http://members.tripod.lycos.com/aldev). Na těchto serverech je mnoho zajímavých informací a tipů pro Linux.

Překlad jádra zajišťuje, že jádro není rozsáhlé a ve výsledku je tak Linux rychlejším operačním systémem. Pomocí překladu jádra je také umožněna podpora nových zařízení.

> **POZNÁMKA:** Následující zápis `'bash#'` označuje příkazový řádek příkazového interpretu bash, měli byste zadávat až příkazy uvedené za tímto zápisem. Následující příkazy byly testovány na distribuci RedHat Linux, ale měly by být bez větších úprav funkční také na jiných distribucích.

1. Uvědomte si, že můžete mít ve vašem systému více než jedno jádro. Následujícím postupem si nepřepíšete ani si nezničíte vaše současné jádro. Tento postup je **zcela bezpečný** a současné jádro nebude nijak dotčeno.
2. Přihlaste se jako uživatel root. Připojte jednotku CD-ROM a nainstalujte zdrojové balíčky s jádrem:

   ```
   bash$ su - root
   bash# cd /mnt/cdrom/RedHat/RPMS
   bash# rpm -i kernel-headers*.rpm
   bash# rpm -i kernel-source*.rpm
   bash# rpm -i dev86*.rpm
   bash# rpm -i bin86*.rpm
   ```

   Balíček bin86*.rpm a assembler as86 je nutný pouze pro starší verze Linuxu, jako je například RedHat 5.x. Intel assembler as86 můžete získat z balíčku dev86*.rpm nebo z adres http://rpmfind.net/linux/RPM/mandrake/7.1/Mandrake/RPMS/bin86-0.4-12mdk.i586.html a http://rpmfind.net/linux/RPM/kondara/jirai/i586/bin86-0.4-8k.i586.html.

3. Spusťte X-window:

   ```
   bash# man startx
   bash# startx
   bash# cd /usr/src/linux
   bash# make xconfig
   ```

Poslední příkaz `make xconfig` spustí konfigurační program s uživatelsky přívětivým rozhraním. Vyjma případů, kdy nemůžete spustit X-window, nepoužívejte příkaz `make config`, který se bude ptát na mnoho otázek.

4. Povolte podporu nahrávání modulů. Pak můžete nahrávat/odstraňovat ovladače zařízení dynamicky na spuštěném systému. Prostudujte následující manuálové stránky s nápovědou:

```
bash# man lsmod
bash# man insmod
bash# man rmmod
bash# man depmod
```

5. Uložte změny a ukončete práci s konfiguračním programem. Všechny provedené změny jsou nyní uloženy do konfiguračního souboru `/usr/src/linux/.config` (konfigurační soubor s tečkou) a proveďte:

```
bash# make dep
bash# make clean
```

6. Prostudujte následující soubor (obsahuje mnoho informací o sestavování jádra).

   TIP: Pro lepší čitelnost můžete použít editor *gvim* (viz dokument http://linuxdoc.org/HOWTO/Vim-HOWTO.html), který podporuje barevné zobrazení.

```
bash# gvim -R /usr/src/linux/arch/i386/config.in
bash# man less
bash# less /usr/src/linux/arch/i386/config.in
```

   Pro nápovědu stiskněte `h` a pro navigaci používejte písmena `i`, `j`, `k`, `l`, `h` nebo šipky a klávesy PAGE UP/PAGE DOWN.

7. Nyní proveďte vlastní sestavení jádra pomocí příkazu `make`:

```
bash# cd /usr/src/linux
bash# man nohup
bash# nohup make bzImage &
bash# tail -f nohup.out (.... pro sledování průběhu)
```

   Tento příkaz vytvoří jádro do souboru `/usr/src/linux/arch/i386/boot/bzImage`

```
bash# man tail
```

8. Poté, co je obraz jádra bzImage úspěšně vytvořen, zkopírujte jej do adresáře `/boot`. Toto je podmínka nutná, protože v opačném případě by nebylo možné toto jádro pro spuštění systému použít. Poté si prostudujte nápovědu k příkazu `lilo` (viz dokument http://www.linuxdoc.org/HOWTO/LILO-crash-rescue-HOWTO.htm) a ukázkový konfigurační soubor `lilo.conf`. Vždy do názvu souboru s jádrem přidávejte informace o tom, kdy bylo jádro sestaveno.

```
bash# cp /usr/src/linux/arch/i386/boot/bzImage /boot/bzImage.my-
ker.26mar2001
bash# man lilo
bash# man lilo.conf
```

Do souboru lilo.conf přidejte následující řádky:

```
image=/boot/bzImage.myker.26mar2001
label=myker
root=/dev/hda1
read-only
```

Název startovacího zařízení pro položku root můžete zjistit takto:

```
bash# df /boot
```

9. Nyní zadejte:

```
bash# lilo
bash# lilo -q
```

Program lilo musíte spustit vždy, když je opětovně přeloženo jádro bzImage.

10. Restartujte počítač a po stisku tabulátoru v úvodní nabídce programu lilo napište `myker`. Pokud se operační systém spustí, bylo vše provedeno správně. V opačném případě vyberte původní jádro a vše proveďte znovu. Vaše původní jádro zůstalo nedotčené v souboru `/boot/vmlinuz-2.0.34-0.6`.
11. Moduly – zkontrolujte existenci příkazu `insmod`, který se pro nahrávání modulů používá nejčastěji:

```
bash# man insmod
bash# insmod
bash# rpm -i /mnt/cdrom/Redhat/RPMS/modutils*.rpm
```

Následující kroky nejsou nezbytně nutné, ale jsou doporučeny pro případ, kdy by došlo k poškození souborů s moduly v adresáři `/lib/modules`. Pokud adresář `/lib/modules` již existuje a chcete jej nahradit, použijte při instalaci balíčku volbu `--force`. Nezapomeňte vybrat balíček pro správnou architekturu. Pro nové verze RedHat Linux (6.0 a novější), jsou moduly jádra obsaženy v balíčcích kernel-2.2*.rpm. Moduly a jádro nainstalujete takto (první příkaz zobrazí seznam nainstalovaných balíčků):

```
bash# rpm -qa | grep -i kernel
bash# rpm -U --force /mnt/cdrom/Redhat/RPMS/kernel-2.2.14-5.0.i686.rpm
```

(nebo)

```
bash# rpm -U --force /mnt/cdrom/Redhat/RPMS/kernel-2.2.14-5.0.i586.rpm
```

(nebo)

```
bash# rpm -U --force /mnt/cdrom/Redhat/RPMS/kernel-2.2.14-5.0.i386.rpm
```

Následující příkaz je určen pro starší verze RedHat Linux (5.2 a starší), které nemají předinstalovanou podporu modulů (chybí příkaz `insmod`). Spusťte nové jádro a nainstalujte moduly z CD-ROM označeného „RedHat Linux contrib“:

```
bash# rpm -i /mnt/cdrom/contrib/kernel-modules*.rpm
```

12. Tento krok je nutný pouze v případě, kdy máte staženou novou verzi zdrojového kódu jádra. Moduly jsou umístěny v adresáři `/lib/modules`.

```
bash# cd /usr/src/linux
bash# make modules
bash# make modules_install
```

13. Pokud bylo jádro *myker* spuštěno a pracuje správně, vytvořte si startovací disketu. Zasuňte do mechaniky čistou disketu a zadejte:

```
bash# cd /usr/src/linux
bash# make bzdisk
```

Podívejte se také na příkaz mkbootdisk

```
bash# rpm -i mkbootdisk*.rpm
bash# man mkbootdisk
```

Řešení běžných chyb

Následující chyba se přihodí mnoha novým uživatelům. Pokud se přeložené jádro nespustí, získáte varovné hlášení:

```
Warning: unable to open an initial console
Kernel panic: no init found. Try passing init= option to kernel
```

Tento problém je nejčastěji způsoben špatně nastavenou hodnotou parametru `root =` v souboru `/etc/lilo.conf`. Ve výše uvedeném příkladě jsme použili nastavení `root=/dev/hda1`, ale ve vašem případě tento parametr může nabývat jiných hodnot, jako je `/dev/hdb2` či `/dev/hda7`.

Jádro se pokouší využít příkaz `init` v adresáři `/sbin`, který je umístěn právě na kořenovém oddílu. Další informace získáte z příslušné manové stránky:

```
bash# man init
```

Ukázkový soubor lilo.conf

Pro soubor `/etc/lilo.conf` byste měli dodržovat konvenci názvosloví – pro jádro 2.2.17 ker2217, pro jádro 2.2.14 ker2214 atd. Na jednom systému můžete mít více obrazů jádra:

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
default=firewall

image=/boot/vmlinuz-2.2.14-5.0
        label=ker2214
        read-only
        root=/dev/hda9

image=/boot/vmlinuz-2.2.17-14
```

```
        label=ker2217
        read-only
        root=/dev/hda9

#image=/usr/src/linux/arch/i386/boot/bzImage
#       label=myker
#       root=/dev/hda7
#       read-only

image=/boot/bzImage.myker.11feb2001
        label=myker11feb
        root=/dev/hda9
        read-only

image=/boot/bzImage.myker.01jan2001
        label=myker01jan
        root=/dev/hda9
        read-only

image=/boot/bzImage.myker-firewall.16mar2001
        label=firewall
        root=/dev/hda9
        read-only
```

Důležité otázky a odpovědi

O co se jádro stará

Unixové jádro je prostředníkem mezi vašimi programy a vaším hardwarem. Obhospodařuje správu paměti pro všechny spuštěné programy (procesy) a zajišťuje, že všechny programy slušně (či neslušně, pokud to tak chcete) sdílejí strojový čas procesoru. Navíc poskytuje přenositelné rozhraní pro přístup vašich programů k hardwaru.

Pochopitelně se jádro stará o mnohem více věci, ale tyto základní funkce jsou nejdůležitější.

Kdy je nutné provést inovaci jádra

Nové verze jádra nabízejí podporu většího množství hardwaru (v Unixu se k hardwaru přistupuje přes tzv. ovladače zařízení), mají vylepšenou správu procesů, jsou rychlejší oproti předchozím verzím, měly by být stabilnější než předchozí verze a odstraňují chyby předchozích verzí. Mnoho uživatelů inovuje jádro právě z důvodů podpory nových zařízení či odstranění chyb.

Jaký druh hardwaru podporuje nové jádro

Podívejte se do souboru `Hardware-HOWTO`. Nebo se také můžete podívat do souboru `config.in`, případně na výstup příkazu `make config`. Zde zjistíte veškerý standardní distribucí podporovaný hardware, ale ne veškerý hardware, který Linux podporuje. Mnoho běžných ovladačů zařízení (jako jsou ovladače pro PCMCIA a páskové jednotky) je řešeno pomocí modulů a jsou distribuovány odděleně.

Jakou musíme mít verzi překladače gcc a knihovny libc

Linus Torvalds informuje o požadavcích na překladač a knihovny v souboru `README`. Pokud nemáte tuto verzi, dokumentace k požadované verzi gcc by vám měla poskytnout informace o tom, zda je nutné inovovat také knihovnu libc. Toto není obtížná procedura, ale je velmi důležité dodržet přesný postup.

Co jsou to moduly

Existuje kód jádra, který ale není přilinkován (vložen) přímo do vlastního jádra. Tyto moduly jsou překládány samostatně a mohou být připojovány a odpojovány za běhu operačního systému. Vzhledem k této pružnosti je dnes tento postup pro rozšiřování funkčnosti jádra doporučován. Mnoho populárních ovladačů zařízení, jako jsou ovladače karet PCMIA či ovladač páskové jednotky QIC-80/40 je realizováno právě pomocí modulů.

Kolik diskového prostoru jádro vyžaduje

Požadavek na diskový prostor velmi závisí na konkrétní konfiguraci Linuxu. Komprimovaný zdrojový kód pro verzi 2.2.9 je veliký 14 MB, ovšem na mnoha serverech je uložena nekomprimovaná podoba. Nekomprimovaný a přeložený tvar pro běžné nastavení zabírá okolo 67 MB.

Kolik času překlad jádra vyžaduje

Na novějších počítačích je oproti minulosti překlad jádra mnohem rychlejší. Na konfiguraci s procesorem AMD K6-2/300 s rychlým diskem trvá překlad jádra 2.2.x přibližně čtyři minuty. Na počítačích s procesory Pentium, 486 či 386 se připravte na čekání, možná v hodinách, možná ve dnech...

Pokud je časová prodleva pro vás problémem a máte přístup k rychlejšímu počítači, můžete jádro přeložit na něm a následně je přenést na pomalejší počítač (pochopitelně musíte při překladu použít správné parametry a ověřit si, že jednotlivé utility nejsou příliš zastaralé).

Jak nastavit jádro

Jak získat zdrojový kód

Zdrojový kód jádra můžete získat z anonymního serveru `ftp.kernel.org`. Jednotlivé verze jádra jsou k dispozici v adresáři `/pub/linux/kernel/vx.y`, kde x.y je verze (například 2.2), a jak bylo řečeno v kapitole 1, lichou poslední číslicí jsou označovány vývojové verze a ty mohou být nestabilní. Typický název souboru se zdrojovým kódem jádra je `linux-x.y.z.tar.gz`, kde x.y.z je číslo verze. Na serverech s distribucemi velmi často bývají také k dispozici soubory s příponou `.bz2`, které jsou komprimovány pomocí programu `bzip2` (tyto soubory bývají menší, a tak vyžadují méně času na stažení).

Nejlepší je použít adresu serveru ftp.xx.kernel.org, kde xx je kód vašeho státu, například ftp.cz.kernel.org pro Českou republiku či ftp.us.kernel.org pro Spojené státy americké.

Dekomprese

Přihlaste se jako uživatel root a přejděte do adresáře `/usr/src` (pomocí příkazu `cd /usr/src`). Pokud jste spolu s první instalací Linuxu instalovali také zdrojový kód jádra (tak tomu je ve většině běžných případů), měl by v tomto adresáři existovat podadresář `linux`, který obsahuje zdrojový kód nainstalovaného jádra. Pokud máte k dispozici dostatek místa na disku a chce-

te mít jistotu možnosti návratu a překladu této starší verze, uchovejte si obsah tohoto podadresáře. Rozumný přístup je přejmenovat tento podadresář podle verze jádra, které je aktuálně nainstalováno a spuštěno. Verzi aktuálně spuštěného jádra zjistíte pomocí příkazu `uname -r`. Tedy, pokud tento příkaz zobrazí `1.0.9`, můžete původní podadresář `linux` přejmenovat na `linux-1.0.9` (pomocí příkazu mv). Pokud si myslíte, že zálohu nebudete potřebovat, můžete celý podadresář `linux` odstranit. Před dekompresí nového jádra si ověřte, že v adresáři `/usr/src` žádný podadresář `linux` neexistuje.

Nyní můžete v adresáři `/usr/src` dekomprimovat zdrojový kód jádra pomocí příkazu `tar zxpvf linux-x.y.z.tar.gz` (pokud máte soubor s příponou `.tar` a ne `.gz`, použijte příkaz `tar xpvf linux-x.y.z.tar`). Obsah archivu bude rozbalen do správné adresářové struktury. Po dokončení se zobrazí opět příkazová řádka – přejděte do adresáře `linux` a prostudujte si soubor `README`. V tomto souboru je sekce nazvaná INSTALLING the kernel, ve které jsou uvedeny důležité informace – jaké mají být vytvořeny symbolické odkazy, informace o odstranění zastaralých souborů `*.o` atd.

Pokud máte k dispozici soubor s příponou `.bz2` a program `bzip2` (http://www.muraroa.deamon.co.uk), použijte příkaz:

```
bzcat linux-x.y.z.tar.bz2 | tar xvf -
```

Nastavení jádra

> **POZNÁMKA:** Některé informace uvedené v této kapitole doplňují či opakují informace uvedené v příslušném souboru `README` od Linuse Torvaldse.

Příkaz `make config` zadaný v adresáři `/usr/src/linux` spustí konfigurační skript, který vám položí několik otázek. Pokud tento skript vyžaduje bash, tak zkontrolujte, že současný příkazový interpret je uložen v `/bin/bash`, `/bin/sh` či `$BASH`.

Samozřejmě existuje několik příjemnějších alternativ k příkazu `make config` a vy je můžete oprávněně považovat za komfortnější a jednodušší. Pravděpodobně světově nejpoužívanější možností je příkaz `make menuconfig`. Bez ohledu na to, jakou možnost si zvolíte, je velmi vhodné seznámit se s daným rozhraním, protože se k němu můžete vrátit dříve, než jste předpokládali. Pokud máte spuštěn systém X-window, můžete zkusit příkaz `make xconfig`. Příkaz `make menuconfig` je pro ty, co preferují textově orientované nabídky. Tato rozhraní mají jednu základní výhodu – pokud zvolíte špatnou volbu během konfigurace, není problém svůj omyl jednoduše napravit. Jednotlivé volby jsou v obou těchto případech zobrazovány v podobě hierarchicky orientované nabídky.

Na většinu otázek budete odpovídat ano (y-es) či ne (n-o), ovladače zařízení zpravidla navíc nabízí možnost volby `m`, tedy možnost použití modulu místo přímého zařazení k vlastnímu jádru. Takový ovladač je tedy sice přeložen, ale jako samostatný modul. Některé zlé jazyky tuto volbu překládají jako možná (maybe). Některé zřejmé a nepříliš důležité volby zde popsány nejsou – podívejte se do části věnované dalším volbám, ve které jsou popsány některé další volby. Pokud jste zvolili cestu s `make menuconfig`, pro označení voleb můžete používat mezerník.

Ve verzích 2.0.x a novějších se objevila volba `?`, která poskytuje krátký popis daného konfiguračního parametru. Tyto popisy bývají velmi aktuální.

Emulace matematických operací v jádru (Typ procesoru a některé vlastnosti) {Kernel math emulation (Processor type and features)}

Pokud nemáte k dispozici matematický koprocesor (máte tedy samotný procesor 386 či 486SX), musíte na tuto otázku odpovědět kladně (y). Pokud k dispozici koprocesor máte a přesto odpovíte kladně, nic se neděje – je používán koprocesor a emulace je ignorována. Pro většinu současných počítačů by měla být odpověď záporná (n), ale jak již bylo řečeno, pokud odpovíte kladně, nic se nestane.

Rozšířená podpora disků MFM/RLL a IDE disků/CD-ROM (bloková zařízení) {Enhanced (MFM/RLL) disk and IDE disk/cdrom support (Block Devices)}

Pravděpodobně budete tuto podporu potřebovat. Tato volba říká, že jádro bude podporovat standardní pevné disky pro osobní počítače (a ty má většina uživatelů). Tato volba nezahrnuje podporu zařízení SCSI (o nich více později).

Budete dotázáni na podporu pouze starých disků (`old disk only`) a nových IDE (`new IDE`) zařízení. Vyberte jednu volbu – rozdíl je v tom, že starší ovladač podporuje pouze dva disky na jednom rozhraní, kdežto nový ovladač podporuje druhé rozhraní a zařízení IDE/ATAPI CD-ROM. Nový ovladač je o 4 KB větší a obsahuje jiné množství chyb, může zvýšit výkon vašeho disku, především tehdy, pokud máte k dispozici nejnovější hardware (EIDE).

Podpora sítí {Networking support (General Setup)}

V zásadě byste měli odpovídat kladně v případě, že váš počítač je připojen k Internetu či se k Internetu připojujete pomocí vytáčené linky a protokolů SLIP, PPP, TERM apod. Mnoho balíčků (jako je X-window) vyžaduje podporu sítí i když nejste připojení ke skutečné síti. V tomto případě byste také měli odpovídat kladně. Pokud si nejste absolutně jisti svým rozhodnutím, odpovězte kladně i na otázku (bude položena později) týkající se podpory sítí založených na sadě protokolů TCP/IP.

Systém V IPC {System V IPC (General Setup)}

Jedna z nejlepších definic IPC (Interprocess Communication) je uvedena ve slovníčku knih o Perlu. To není nijak překvapující, někteří programátoři v Perlu využívají IPC k vzájemné komunikaci mezi procesy, stejně jako mnoho dalších balíčků (nejvíce proslulý je například DOOM), není tedy dobrý nápad odpovědět záporně, pokud zcela přesně nevíte, co děláte.

Typ procesoru {Processor family (Processor type and features)}

Použití volby –m486 pro optimalizace pro 486.

Tradičně je překlad optimalizován pro konkrétní procesor. Jádra mohou správně fungovat také na dalších procesorech, ale pravděpodobně pak bude jádro o něco větší. V novějších jádrech toto již není pravdivé tvrzení, měly byste tedy určit, pro jaký procesor bude jádro překládáno. Jádro pro 386 bude správně fungovat na všech počítačích.

Podpora zařízení SCSI {SCSI support}

Pokud máte v počítači nějaká zařízení SCSI, odpovězte na tento dotaz kladně. Budete dotázáni na další doplňující informace, jako je podpora jednotek CD-ROM, pevných disků a jaký typ adaptéru SCSI máte k dispozici. Více informací najdete v dokumentu SCSI-HOWTO.

Podpora síťových zařízení {Network device support}

Pokud máte ve svém počítači síťovou kartu či budete chtít používat protokoly SLIP, PPP či paralelní adaptéry pro připojení k Internetu, odpovězte kladně. Následně budete dotázáni na konkrétní typ vaší síťové karty a jaký protokol budete používat.

Souborové systémy {Filesystems}

Konfiguračním skriptem budete dotázáni na typy podporovaných souborových systémů:

Standard (minix) – novější distribuce nevytvářejí tento souborový systém a mnoho uživatelů jej nepoužívá, ale může být pořád důvod na výběr této volby. Používají jej totiž některé záchranné programy a pořád ještě mnoho disket může být naformátováno pod tímto systémem, protože je jeho použití na disketách relativně bezproblémové.

Second extended – toto je standardní souborový systém pro Linux. Téměř vždy jej používáte a proto odpovězte kladně.

msdos – pokud budete chtít používat diskové oddíly či diskety vytvořené pod MS-DOS, odpovězte kladně.

K dispozici máte také několik dalších podporovaných souborových systémů.

/proc – Tento souborový systém (původem z Bell Labs) nevytváří klasický souborový systém na disku. Jedná se o rozhraní mezi jádrem a procesy. Mnoho programů (jako je například `ps`) tento systém používá. Zkuste někdy zadat příkaz `cat /prcoc/meminfo` či `cat /proc/devices`. Některé příkazové interprety (například `rc`) využívají pro vstupy a výstupy `/proc/self/fd` (na jiných systémech známé jako `/dev/fd`). Na tuto otázku byste měli téměř vždy odpovědět kladně, mnoho důležitých nástrojů je na tomto systému závislých.

NFS – pokud je váš počítač připojený k síti a chcete využívat souborové systémy umístěné na jiných počítačích, odpovězte na tuto otázku kladně.

ISO9660 – na tomto souborovém systému je založeno mnoho disků CD-ROM. Pokud chcete využívat pod Linuxem jednotku CD-ROM, odpovězte na tuto otázku kladně.

Co když ale nevíte, jaký souborový systém potřebujete? V takovém případě zadejte příkaz `mount`. Výstup by měl vypadat nějak takto:

```
/dev/hda1 on / type ext2 (defaults)
/dev/hda3 on /usr type ext2 (defaults)
none on /proc type proc (defaults)
/dev/fd0 on /mnt type msdos (defaults)
```

Podívejte se na každý jednotlivý řádek. Slovo následující za `type` určuje typ souborového systému. V tomto příkladě souborové systémy `/` a `/usr` jsou typu second extended, používá se `/proc` a je připojena disketová mechanika pomocí souborového systému typu msdos.

Pokud máte povolenou podporu /proc, můžete zkusit zadat příkaz `cat /proc/filesystems` a získáte tak seznam aktuálně podporovaných souborových systémů.

Konfigurace využívající zřídkakdy používané souborové systémy mohou způsobit nárůst velikosti jádra. Podívejte se do kapitoly 10 věnované modulům, jak tomuto nepříjemnému důsledku zabránit a do kapitoly 8 věnované některým léčkám, proč takováto jádra nejsou žádoucí.

Znaková zařízení {Character devices}

Zde můžete povolit podporu zařízení pro tisk (paralelní tiskárny), myší (i typu PS/2 – mnoho notebooků využívá pro vestavěná polohovací zařízení právě protokol PS/2), některých páskových jednotek a dalších znakových zařízení. Pokud je používáte, odpovězte kladně.

> **POZNÁMKA:** Program `gpm`, který umožňuje používat myši i mimo X-window (například pro kopírování mezi virtuálními konzolemi). Nenastávají žádné problémy, pokud máte myš pro sériové rozhraní, protože ty naprosto správně spolupracují s X-window. V opačném případě ale budete potřebovat některé speciální triky.

Zvuk {Sound}

Pokud máte neodolatelnou touhu pracovat se zvukem, odpovězte na tuto otázku kladně. Následně budete dotázání na doplňující informace o vaší zvukové kartě (na dotaz, zda instalovat plnou verzi ovladače, můžete odpovědět záporně – snížíte tak paměťové nároky pro jádro a budete využívat pouze vlastnosti, které opravdu potřebujete).

Pokud se o podporu zvukových karet vážně zajímáte, podívejte se jak na ovladače dostupné zdarma na adrese http://www.linux.org.uk/OSS/, tak na komerční ovladače Open Sound System (http://www.opensound.com) a také na projekt ALSA (http://www.alsa-project.org).

Další volby {Other configuration options}

Zde nejsou uvedeny všechny dostupné volby – jednak se jejich hodnoty nedají zobecnit, jednak je jejich hodnota snadno odvoditelná (například volba 3Com 3C509 zajistí podporu tohoto typu síťové karty). Úplný seznam všech voleb (spolu s popisem, jak je umístit do skriptu `Configure`) je spravován Axelem Boldtem (boltd@math.ucsb.edu). Je dostupný jako nápověda i jako jeden velký soubor `Documentation/Configure.help` ve zdrojovém kódu jádra.

Kernel hacking

Citace ze souboru `README` od Linuse Torvaldse:

Tyto volby obvykle vedou k nárůstu nebo zpomalení jádra (nebo obojímu) a mohou snížit stabilitu jádra. Je to způsobeno tím, že některé rutiny se pokouší přerušit špatný kód k nalezení problémů jádra (kmalloc()). Takže byste měli na tuto otázku pro produkční verzi jádra odpovědět záporně.

Co dál? (Makefile)

Poté, co dokončíte nastavení budete vyzváni ke kontrole nadřazených souborů `Makefile` pro případné dodatečné nastavení. Podívejte se tedy na soubor `Makefile` – s největší pravděpodobností jej nebudete muset měnit, ale neuškodí si jej prohlédnout. Můžete také změnit volby příkazem `rdev` dříve, než umístíte nové jádro. Pokud se v souboru `Makefile` ztratíte, nic si z toho nedělejte.

Překlad jádra

Úklid a závislosti

Po ukončení konfiguračního skriptu, jste také vyzvání k příkazu `make dep` a (možná) `make clean`. Zadejte tedy příkaz `make dep`, který zajistí, že všechny závislosti, jako jsou hlavičkové soubory, jsou správně nastaveny a vyžadované soubory jsou na svých místech. Tato kontrola netrvá nijak dlouho, pokud tedy váš počítač není příliš pomalý. Pro starší verze jádra byste měli po ukončení zadat příkaz `make clean`. Tento příkaz odstraní všechny soubory `*.o` a některé další věci, které v systému zůstaly po předchozích verzích. V žádném případě nezapomeňte provést tento krok před samotným překladem jádra.

Vlastní překlad

Po provedení kontroly závislostí a úklidu můžete zadat příkaz `make bzImage` či `make bzdisk` (tato část vyžaduje nejvíce času). První příkaz `make bzImage` přeloží jádro a umístí jej pod název `bzImage` do adresáře `arch/i386/boot` (včetně dalších věcí). Toto je nové komprimované jádro. Druhý příkaz `make bzdisk` vykonává totéž, ale současně umístí nové jádro `bzImage` na disketu, kterou jste zasunuli do jednotky A: Tuto disketu můžete použít pro testování nového jádra – pokud se toto nové jádro nechová správně či se nespustí vůbec, můžete jednoduše vyndat disketu z mechaniky a nastartovat počítač s původním jádrem. Tuto disketu můžete také použít v případě, kdy jste omylem z pevného disku toto nové jádro odstranili (či provedli něco podobně chybného). Můžete ji také použít pro instalaci nového systému ve chvíli, kdy přenášíte obsah jednoho disku na druhý.

Takřka všechna dnešní jádra jsou komprimována, odtud umístění písmenek bz před vlastní název. Komprimované jádro je automaticky dekomprimováno před vlastním spuštěním.

Ve starších jádrech nemáte možnost volby pro vytvoření komprimovaného jádra. Tato volba je dnes dostupná i z toho důvodu, že velikost nových verzí jádra se neustále zvyšuje a starší metody neumožňují obsluhovat tato velká jádra.

Další možnosti příkazu make

Příkaz `make mrproper` je rozsáhlejší obdobou příkazu `make clean`. Někdy je provedení tohoto příkazu potřebné, můžete jej chtít provést například při každé nové záplatě. Tento příkaz také vymaže váš konfigurační soubor s nastavením jádra. Proto si soubor `.config` můžete chtít zálohovat, pokud tedy vlastní obsah uznáte za důležitý.

Příkaz `make oldconfig` se pokusí nastavit nové jádro podle původního konfiguračního souboru. Tento příkaz je spouštěný v rámci příkazu `make config`. Pokud nemáte žádné dříve překládané jádro či žádný starší konfigurační soubor, tak byste neměli tento příkaz spouštět.

Podrobnosti o příkazu `make modules` najdete v kapitole 10 věnované modulům.

Instalace jádra

Poté, co máte nové jádro odpovídající vašim požadavkům, nastává vhodný okamžik pro jeho instalaci. Mnoho uživatelů pro tuto činnost preferuje LILO (Linux Loader). Příkaz `make bzlilo` nainstaluje jádro, spustí LILO a připraví vše pro spuštění systému. To vše ale pouze za předpokladu, že LILO je na vašem systému nastaveno takto: jádro je `/vmlinuz`, program `lilo` je umístěn v adresáři `/sbin` a konfigurační soubor pro LILO je `/etc/lilo.conf`.

V opačném případě musíte použít LILO přímo – je jednoduché jej nainstalovat a používat, ale vykazuje občasné tendence zmást uživatele právě svým konfiguračním souborem. Pro starší verze je konfigurační soubor `/etc/lilo/config` a pro nové verze `/etc/lilo.conf`. Podívejte se na jeho obsah platný pro současné nastavení, který bude vypadat přibližně takto:

```
image = /vmlinuz
        label = Linux
        root = /dev/hda1
        ...
```

Parametr `image =` je nastaven na aktuálně nainstalované jádro. Mnoho uživatelů používá právě `/vmlinuz`. Parametr `label =` je používán pro určení, jaké jádro či operační systém bude spuštěn a parametr `root =` je kořenový adresář konkrétního operačního systému. Udělejte si záložní kopii původního souboru s jádrem a nahraďte jej novým jádrem `bzImage` (pokud používáte `/vmlinuz`, tak pak například pomocí `příkazu cp bzImage /vmlinuz`). Následně spusťte program `lilo` (na novějších systémech stačí napsat příkaz `lilo`, na starších možná bude muset zadat příkaz `/etc/lilo/install` či `/etc/lilo/lilo -C /etc/lilo/config`).

Pokud se chcete o nastavení pro LILO dozvědět více či LILO vůbec nemáte nainstalováno, stáhněte si nejnovější verzi z vašeho oblíbeného serveru ftp a řiďte se instalačními pokyny.

Pro spuštění některé ze starších verzí jádra (další z možností jak se můžete ochránit před důsledky plynoucími z problémů s novým jádrem), nakopírujte řádky následující za `image = xxx` (pochopitelně včetně tohoto řádku) na konec konfiguračního souboru a změňte parametr `image = xxx` na `image = yyy`, kde yyy je název původního jádra. Tento název je včetně cesty a tuto zálohu máte vytvořenou z předchozího kroku. Poté změňte parametr `label = zzz` na `label = linux-backup` a spusťte `lilo`. Můžete také chtít vložit do konfiguračního souboru řádek s parametrem `delay = x`, kde x je počet desetin sekundy, během kterých LILO čeká před automatickým spuštěním aktuálního jádra. Tento proces lze před uplynutím definované doby zastavit stiskem klávesy (například shift) a například v případě problémů s novým jádrem zadat název původního jádra (určený hodnotou parametru `label`).

Opravy jádra

Uplatnění záplat

Přírůstkové inovace jádra jsou distribuovány jako záplaty. Například, pokud máte verzi 1.1.45 a máte k dispozici záplatu `patch46.gz`, znamená to, že můžete provést upgrade na verzi 1.1.46 pouhou aplikací této záplaty. Můžete také nejprve chtít vytvořit si záložní kopii zdrojových kódů jádra verze 1.1.45 (nejprve zadejte příkaz `make clean` a pak příkazy `cd /usr/src` a `tar zcvf old-tree.tar.gz linux`, čímž získáte komprimovanou zálohu zdrojového kódu).

Pokračujme ve výše uvedeném příkladě – budeme předpokládat, že máte soubor `patch46.gz` nakopírován do adresáře `/usr/src`. Přejděte do tohoto adresáře a zadejte příkaz `zcat patch46.gz | patch -p0` (případně `patch -p0 < patch46` není li soubor se záplatou komprimován).Výstup vás bude informovat o pokusech aplikovat změny a zda tyto pokusy dopadly úspěšně či nikoli. Ve většině případů tato akce proběhne příliš rychle na to, abyste si mohli vše pečlivě pročíst, proto můžete pro program patch použít volbu `-s`. Ta určuje, že ve výstupu budou obsažena pouze chybová hlášení (neuvidíte tedy hlášení typu „Ahoj, můj počítač je připraven na aplikování záplaty“, ale proti gustu ...). Pro seznámení se s částmi, které neproběhly hladce, přejděte do adresáře `/usr/src/linux` a najděte soubory s příponou `.rej`. Některé verze záplat (starší verze, které mo-

hou být přeloženy na horším souborovém systému) ponechávají tyto soubory s příponou #. Pro nalezení těchto souborů můžete použít příkaz `find`:

```
find . -name '*.rej' -print
```

Tento příkaz vypíše na standardní výstup všechny soubory v aktuálním adresáři a všech podadresářích, které mají příponu `.rej`.

Pokud vše proběhlo v pořádku, zadejte příkaz `make clean`, `make config` a `make dep`, jak bylo popsáno v kapitolách 3 a 4.

Příkaz `patch` má také několik voleb. Jak bylo zmíněno výše, příkaz `patch -s` potlačí zobrazení všech hlášení kromě chybových. Pokud jste ponechali zdrojový kód vašeho jádra v původním adresáři (`/usr/src/linux`), příkaz `patch -p1` (v tomto adresáři) provede opravu jádra korektně. Další volby příkazu `patch` jsou popsány v nápovědě.

Pokud něco nefunguje správně?

POZNÁMKA: Tato kapitola se většinou odkazuje na starší verze jádra.

Častým problémem je, že při aplikování záplaty dochází ke změně souboru `config.in` a ty se neprovedenou vždy správně, protože jste změnili volby na vašem počítači. Toto je problém, se kterým se můžete setkat zejména ve starších verzích. Pro nápravu se podívejte na obsah souboru `config.in.rej` a zjistíte, co zůstalo z původní záplaty. Změny bývají nejčastěji označeny + a – na začátku každého řádku. Podívejte se také na okolní řádky a zapamatujte si, zda jsou volby nastaveny na y či n. Nyní změňte v souboru `config.in` odpovídající parametry na stejné hodnoty. Pak zadejte příkaz:

```
patch -p0 < config.in.rej
```

a pokud se nezobrazí žádná chyba, můžete pokračovat s dalším nastavováním a překladem. Soubor `config.in.rej` sice zůstane, ale můžete jej smazat.

Pokud se setkáte s dalšími problémy, mohli jste instalovat záplaty ve špatném pořadí. Pokud se zobrazí chybové hlášení „`previously applied patch detected: Assume -R?`“, možná se pokoušíte aplikovat záplatu, která je nižní verze, než verze aktuálního jádra. Pokud odpovíte kladně, dojde k pokusu o „snížení“ zdrojového kódu jádra, ale takřka vždy dojde k chybě. Následně budete muset získat kompletní zdrojový kód jádra (což by nebyl špatný nápad hned jako první krok).

Pro zrušení záplaty můžete použít příkaz `patch -R`.

Pokud se objeví nějaké problémy se záplatami, je nejlepší začít vše odzačátku nad nově rozbaleným zdrojovým kódem jádra.

Odstranění souborů s příponou .orig

Po aplikování několika záplat se na vašem disku začnou hromadit soubory s příponou `.orig`, po třech záplatách (u nás například u verze 1.1.51 z 1.1.48) zabíraly tyto soubory okolo půl megabajtu.

Následující příkaz vás zbaví starostí s těmito soubory:

```
find . -name '*.orig' -exec rm -f {} ';'
```

Verze programu patch, které používají pro odmítnuté změny značku #, používají místo přípony `.orig` znak tilda (~).

V závislosti na použití utility *xargs* můžete tyto soubory odstranit také takto:

```
find . -name '*.orig' | xargs rm
```

Další z možností je:

```
find . -name '*.orig' -print0 | xargs --null rm -
```

Další záplaty

Mimo standardních originálních záplat existují také další nestandardní záplaty. Pokud tyto záplaty použijete, nebudete možná schopni originální záplaty správně aplikovat a budete muset instalovat celý původní zdrojový kód a aplikovat jednotlivé originální záplaty. To může být velmi vyčerpávající, zejména tehdy, pokud nechcete měnit zdrojové kódy (s pravděpodobně špatným výsledkem). Po instalaci původního zdrojového kódu jádra máte na výběr spokojit se s tímto starším jádrem, pokusit se přinutit ke spolupráci originální a nestandardní záplaty či počkat na novou verzi záplaty.

Jaké jsou známé nestandardní záplaty? Pravděpodobně o nich uslyšíte. My používáme „neblikající" záplatu pro virtuální konzole, protože nemáme rádi blikající kurzory (tato záplata je, či alespoň dříve byla, pravidelně obnovovaná v souvislosti s novými verzemi jádra). S příchodem ovladačů zařízení ve tvaru modulů dochází k poklesu významu těchto nestandardních ovladačů.

Další balíčky

Linuxové jádro nabízí mnoho vlastností, které nejsou obsaženy přímo ve zdrojovém kódu jádra. Tyto vlastnosti jsou nejčastěji zajištěny pomocí externích balíčků. V této kapitole jsou uvedeny některé z těch nejznámějších.

Balíček kbd

Linuxová konzole nabízí pravděpodobně více vlastností, než si zaslouží. Mezi tyto vlastnosti patří přepínání písem, definice rozložení kláves, přepínání grafických režimů (v novějších verzích jádra) atd. Balíček kbd obsahuje programy, které umožňují uživatelům využívat všechny tyto vlastnosti a navíc nabízí mnoho dalších písem, přednastavené definice mnoha typů klávesnic a je dostupný na stejných místech jako vlastní zdrojové kódy jádra.

Balíček util-linux

Tento balíček, původně od Ricka Faitha (faith@cs.unc.edu) obsahuje velkou kolekci utilit. Nyní jej spravuje Andries Brouwer (util-linux@math.uio.no) a je dostupný v rámci anonymního serveru ftp www.ibiblio.org v adresáři `/pub/Linux/system/misc`. Součástí balíčku jsou například programy `setterm`, `rdev` či `ctrlaltdel`. Dle slov autora byste ale tento balíček neměli instalovat bezmyšlenkovitě – nebudete potřebovat vše, co je v balíčku obsaženo. V opačném případě se můžete setkat s vážnými problémy.

Balíček hdparm

Stejně jako mnoho balíčků, byl i tento záplatou jádra a podpůrnými programy. Záplaty se staly součástí oficiálního jádra a programy vylepšující a rozšiřující možnosti využití pevných disků jsou distribuovány odděleně.

Balíček gpm

Tento balíček je určen pro podporu práce s myší a umožňuje používat kopírovací funkce mezi dvěma konzolemi a některé další vlastnosti s mnoha typy myší.

Některé léčky

make clean

Pokud se vaše nové jádro chová opravdu podivně, je možné, že jste zapomněli před překladem nového jádra provést příkaz `make clean`. Příznaky mohou být libovolné – od úplného pádu operačního systému přes vážné V/V problémy až po snížení výkonu. Ujistěte se také, že nezapomenete na příkaz `make dep`.

Velké nebo pomalé jádro

Pokud nové jádro potřebuje mnoho operační paměti, je příliš velké a/nebo jeho překlad trvá neúměrně dlouho, pravděpodobně jste nekonfigurovali příliš mnoho vlastností jádra (ovladačů zařízení, souborových systémů apod.). Pokud je nepoužíváte, nenastavujte je, protože zabírají příliš paměti. Nejčastějším příznakem je extrémní práce se souborem virtuální paměti – pokud tento případ nastane, podívejte se ještě jednou na nastavení jádra.

Celkovou velikost paměti obsazenou jádrem můžete zjistit odečtem dostupné paměti (zjistíte ji pomocí příkazu `free` či informací v `/proc/meminfo`) od velikosti operační paměti daného počítače.

Paralelní port nepracuje, tiskárna netiskne

V kategorii General setup zkontrolujte nastavení voleb Parallel port support a PC-style hardware. Poté v kategorii Character devices zkontrolujte nastavení volby Parallel printer support.

Dalším možným problémem může být změna v názvosloví pro paralelní zařízení, dle kterého je od verze jádra 2.2 první paralelní port označován jako `lp0` a ne `lp1`, jak tomu bylo v dřívějších verzích jádra. V odhalení tohoto problému vám může pomoci program `dmesg` či protokolové soubory v adresáři `/var/log`.

Jádro nelze přeložit

Pokud nelze jádro přeložit, mohou být špatně nastavené cesty či zdrojové kódy mohou být poškozeny. Také nemusíte mít správnou verzi překladače `gcc` nebo může být poškozený také (chyba může být například v připojovaných souborech). Ujistěte se, že symbolické odkazy popsané v souboru `README` jsou správně nastavené. Obecně platí, že pokud nelze standardní jádro přeložit, jsou nějaké problémy v celém systému – dost možná bude nutné přeinstalovat některé nástroje.

V některých případech může být pád překladače `gcc` způsoben hardwarovými problémy. Chybové hlášení v takových případech vypadá přibližně takto: !`xxx exited with signal 15`! a vlastní překlad se chová velmi podivně. My jsme se s uvedeným problémem setkali pouze jednou,

když jsme měli problémy s vyrovnávací pamětí a překladač padal zcela náhodně. Nejprve zkuste reinstalaci překladače `gcc`. Podezření mějte pouze v případě, kdy překlad jádra proběhne správně při vypnuté externí vyrovnávací paměti či snížení velikosti operační paměti apod.

Většinu uživatelů vyděsí, pokud zjistí, že mohou mít problémy s hardware – pomoc hledejte v odpovídajícím souboru často kladených dotazů a odpovědí, který najdete na adrese http://www.bitwizard.nl/sig11/.

Nelze spustit novou verzi jádra

Pokud se zdá, že nemůžete spustit novou verzi jádra, je možné, že jste zapomněli spustit LILO nebo nebylo správně nastaveno. Problém může být například v konfiguračním souboru, kdy parametr `boot` máte nastavený na hodnotu `/dev/hda` místo `/dev/hda1`.

Zapomněli jste spustit LILO nebo systém není možné spustit

Tak to je opravdu problém. Nejlepší řešení tohoto problému je spuštění systému z diskety nebo CD-ROM a připravení jiné startovací diskety (například pomocí příkazu `make zdisk`). Musíte vědět, kde je umístěn hlavní souborový systém (kořenový, /) a jakého typu tento souborový systém je (například second extended, minix). V následujícím příkladě musíte také znát na jakém souborovém systému máte uložený zdrojový kód jádra a jak je tento souborový systém za normálních okolností připojen.

V následujícím příkladě je kořenovým adresářem zařízení `/dev/hda1` a souborový systém se zdrojovými kódy jádra je na zařízení `/dev/hda3`, které bývá za normálních okolností připojeno jako `/usr`. Oba souborové systémy jsou typu second extended. Funkční jádro je uloženo v adresáři `/usr/src/linux/arch/i386/boot` a soubor se jmenuje `bzImage`.

Základní myšlenka záchrany je, že pokud máte k dispozici funkční jádro, bude možné toto jádro použít pro novou startovací disketu. Další možnost, která se ovšem nemusí vždy podařit (úspěšnost závisí na konkrétním způsobu poškození systému), je popsána za tímto příkladem.

Nejprve nastartujte systém ze záchranné diskety či startovacích disket pro instalaci a připojte souborový systém s funkčním jádrem:

```
mkdir /mnt
mount -t ext2 /dev/hda3 /mnt
```

Pokud se vytvoření adresáře `/mnt` nepovede z toho důvodu, že již existuje, netrapte se tím a hlášení o chybě ignorujte. Nyní přejděte do adresáře s funkčním jádrem. Uvědomte si, že platí:

```
/mnt + /usr/src/linux/arch/i386/boot - /usr = /mnt/src/linux/arch/i386/boot
```

Vložte do disketové mechaniky (A:) prázdnou naformátovanou disketu, přeneste na ni obraz jádra a nastavte ji pro kořenový souborový systém.

```
cd /mnt/src/linux/arch/i386/boot
dd if=bzImage of=/dev/fd0
rdev /dev/fd0 /dev/hda1
```

Přejděte do kořenového adresáře a odpojte souborový systém `/usr`.

```
cd /
umount /mnt
```

Nyní byste měli být bez problémů schopni nastartovat systém přímo z výše vytvořené diskety. Nezapomeňte po startu systému spustit LILO (či cokoli jiného, co jste předtím udělali chybně).

Jak již bylo uvedeno výše, máte k dispozici ještě jednu možnost. Máte-li k dispozici funkční jádro v kořenovém adresáři (například `/vmlinuz`), můžete je použít pro startovací disk. Předpokládáme-li všechny dříve vyřčené požadavky, můžete tuto možnost realizovat například takto – změňte `/dev/hda3` na `/dev/hda1` (kořenový adresář), `/mnt/src/linux` na `/mnt` a `if=bzImage` na `if=vmlinuz`.

Problémy také může způsobit použití LILO s velkými disky (více než 1024 válců), více informací najdete v nápovědě a dokumentaci.

Objeví se hlášení: „warning: bdflush not running“

Toto může být opravdu vážný problém. S jádrem novějším než 1.0 (okolo 20. dubna 1994) je dodávaná nová opravená verze programu `update`, který opakovaně vyprazdňuje vyrovnávací paměti souborového systému. Získejte program `bdflush` (měl by být na stejném zdroji, ze kterého jste získali zdrojový kód jádra) a nainstalujte jej. Program `bdflush` se sám nainstaluje jako `update` a po novém startu systému by vše již mělo být bez problému.

Jednotka IDE/ATAPI CD-ROM nepracuje

Mnoha uživatelům se nedaří zprovoznit jejich mechaniky CD-ROM, pravděpodobně z toho důvodu, že problém může být způsoben mnoha důvody.

Pokud je jednotka CD-ROM jediným zařízením na konkrétním rozhraní IDE, musí být nastavena jako master či single. Toto je jedna z nejčastějších příčin.

Někteří výrobci, například Creative Labs nyní umísťují rozhraní IDE na zvukové karty. Tato skutečnost může způsobit problémy – dříve mnoho uživatelů mělo pouze jeden kanál IDE, poté dva integrované na základní desce (nejčastěji přerušení IRQ 15) a nyní třetí kanál IDE (pravděpodobně přerušení IRQ 11). Třetí kanál IDE způsobuje problémy s jádrem 1.2.x (podpora třetího kanálu IDE začíná až u verze 1.3.x a nepodporuje autodetekci). Pokud se setkáte s tímto problémem, máte na výběr z několika možností.

Máte-li druhý kanál IDE, je velmi pravděpodobné, že jej nepoužíváte či jej nepoužíváte pro obě možná zařízení. Vypněte tedy zařízení na zvukové kartě a přesměrujte je na druhý kanál IDE. Následně můžete zakázat rozhraní zvukové karty, čímž ušetříte jedno přerušení.

Pokud druhé rozhraní IDE nemáte, nastavte kanál IDE na zvukové kartě tak, aby využíval přerušení IRQ 15. Vše by poté mělo fungovat korektně.

Objevují se podivné problémy při požadavcích na směrování

Stáhněte si novou verzi programu `route` a dalších programů, které se při směrování využívají, protože soubor `/usr/include/linux/route.h` (fyzicky je soubor uložen v adresáři `/usr/src/linux`) byl změněn.

Ve verzi 1.2.0 nefunguje správně firewall

Proveďte inovaci jádra alespoň na verzi 1.2.1.

Objeví se hlášení „Not a compressed kernel Image file“

Nepoužívejte jako startovací obraz jádra soubor `vmlinuz` vytvořený v adresáři `/usr/src/linux`, správný soubor je `/arch/i386/boot/bzImage`.

Objevují se problémy s konzolou po inovaci jádra na verzi 1.3.x

V souboru `/etc/termcap` změňte hodnotu parametru `termcap` z `dumb` na `linux`. Možná také budete muset založit parametr `terminfo`.

Po inovaci jádra není možné překládat další programy

Součástí zdrojového kódu jádra je mnoho hlavičkových souborů (těch, které končí příponou .h), které jsou standardně odkazovány z adresáře `/usr/include`. Nejčastěji se na tyto hlavičkové soubory odkazují programy takto (`xyzzy.h` je libovolný hlavičkový soubor v adresáři `/usr/include/linux`):

```
#include <linux/xyzzy.h>
```

Za normálních okolností existuje odkaz `/usr/include/linux` zastupující adresář `include/linux` ve zdrojovém kódu vašeho jádra (tedy na typickém systému adresář `/usr/src/linux/include/linux`). Pokud tento odkaz neexistuje či odkazuje na špatný adresář, mnoho dalších programů nepůjde přeložit. Velmi často tento problém vzniká tak, že se rozhodnete pročistit disk a vymažete zdrojové kódy jádra. Dalším častým problémem bývají špatně nastavená práva – pokud jste přihlášen jako uživatel `root`, máte předefinované takové masky pro nově vytvářené soubory, které neumožní ostatním uživatelům k těmto souborům přistupovat (pokud jste rozbalili zdrojový kód jádra bez volby `-p` (preserve modification)) a ostatní uživatelé tak nebudou moci používat překladač jazyka C. Přestože lze pro vyřešení tohoto problému použít příkaz `chmod`, jednodušší cesta je rozbalit zdrojový kód jádra znovu. Toto rozbalení proběhne stejně jako předtím, ovšem s přidáním dalšího parametru:

```
blah# tar zxvpf linux.x.y.z.tar.gz linux/include
```

POZNÁMKA: Pokud odkaz `/usr/src/linux` neexistuje, příkaz `make config` jej vytvoří.

Zvýšení limitů

Následující ukázkové příkazy mohou být užitečné pro zvýšení limitů určovaných jádrem:

```
echo 4096 > /proc/sys/kernel/file-max
echo 12288 > /proc/sys/kernel/inode-max
echo 300 400 500 > /proc/sys/vm/freepages
```

Poznámky k inovaci jádra na verze 2.0.x a 2.2.x

Verze jádra 2.0.x a 2.2.x zavedly některé změny v instalaci jádra – podrobnosti o těchto změnách a nové postupy jsou uvedeny v souboru `Documentation/Changes`. Budete muset inovovat několik klíčových balíčků, jako je `gcc`, `libc` či `SysVInit` a možná také několik systémových souborů, takže s tím počítejte a nepanikařte.

Moduly

Moduly především šetří paměť a usnadňují nastavení jádra a používají se zejména v oblasti souborových systémů, ovladačů síťových karet a páskových jednotek, tiskáren apod.

Instalace utilit pro podporu modulů

Utility pro podporu práce s moduly jsou pod názvem `modutils-x.y.z.tar.gz` dostupné na stejných zdrojích, kde jste získali zdrojový kód jádra. Zvolte nejvyšší možnou verzi, která je shodná či nižší s verzí vámi používaného jádra. Rozbalte tento soubor pomocí příkazu `tar zxvf modutils-x.y.z.tar.gz`, přejděte do vytvořeného adresáře (`modutils-x.y.z`), přečtěte si soubor `README` a dodržte instalační instrukce (ty bývají velmi snadné, zpravidla stačí zadat příkaz `make install`). Nyní byste měli mít v adresáři `/sbin` k dispozici programy `insmod`, `rmmod`, `ksyms`, `lsmod`, `genksyms`, `modprobe` a `depmod`. Správnost instalace si můžete ověřit pomocí ukázkového příkladu hw – podrobnosti najdete v souboru `INSTALL` v příslušném podadresáři.

Program `insmod` jednotlivé moduly vkládá do běžícího jádra. Moduly mají nejčastěji příponu `.o`, ukázkový ovladač zmíněný výše má název `drv_hello.o`, do běžícího jádra tedy můžete vložit pomocí příkazu `insmod drv_hello.o`. Přehled modulů, které jsou aktuálně aktivní získáte pomocí příkazu `lsmod`. Výstup bude vypadat například takto:

```
blah# lsmod
Module:       #pages:  Used by:
drv_hello          1
```

Údaj `drv_hello` je název modulu, který používá jednu stránku paměti (4 KB) a na tomto modulu v tomto okamžiku žádné další moduly nezávisí. Pro odstranění tohoto modulu z jádra stačí zadat příkaz `rmmod drv_hello`. Uvědomte si, že příkaz `rmmod` vyžaduje jako parametr název modulu (tak jak jej vidíte ve výstupu příkazu `lsmod`), nikoli název souboru s modulem. Ostatní utility pro práci s moduly jsou popsány v příslušných manuálových stránkách.

Moduly distribuované spolu s jádrem

Od verze jádra 2.0.30 jsou mnohé vlastnosti dostupné prostřednictvím modulů. Předtím, než začnete moduly používat si rozmyslete, zda pro vás nebude lepší použít klasické vlastnosti jádra. Pokud ne, neodpovídejte kladně během průběhu příkazu `make config`. Přeložte nové jádro a nastartujte systém s tímto jádrem. Poté přejděte do adresáře `/usr/src/linux` a zadejte příkaz `make modules`, který přeloží všechny požadované moduly a vytvoří symbolické odkazy do adresáře `/usr/src/linux/modules`. Moduly pak můžete používat přímo z tohoto adresáře nebo zadejte příkaz `make modules_install`, který nainstaluje jednotlivé moduly do adresáře `/lib/modules/x.y.z`, kde x.y.z je číslo verze jádra.

Tento přístup může být výhodný pro práci se souborovými systémy. Nebudete často používat souborové systémy typu minix a msdos? Setkáte se s disketou ve formátu msdos? Zadáte příkaz `insmod /usr/src/linux/modules/msdos.o` a po ukončení práce s disketou modul odstraníte pomocí příkazu `rmmod msdos`. Tento přístup vám během normální práce ušetří okolo 50 KB operační paměti. Drobná poznámka na závěr – souborový systém minix byste z důvodu případné obnovy systému měli přeložit přímo do vlastního jádra.

Tipy a triky

Přesměrování výstupu příkazů make a patch

Pokud chcete získat výstupy programů `make` a `patch`, můžete úspěšně přesměrovat jejich výstupy do souboru. Nejprve zjistěte, jaký příkazový interpret používáte (například pomocí příkazu `grep root /etc/passwd`).

Pro příkazové interprety bash a sh provedete přesměrování výstupu takto:

```
(command) 2>&1 | tee (výstupní soubor)
```

Pro příkazové interprety csh a tcsh provedete přesměrování výstupu takto:

```
(command) |& tee (výstupní soubor)
```

Pro příkazový interpret rc (pravděpodobně jej nepoužíváte) provedete přesměrování výstupu takto:

```
(command) >[2=1] | tee (výstupní soubor)
```

Instalace více jader

Pro testování nového jádra existují také další metody, nejen instalace jádra na disketu. Na rozdíl od mnoha dalších unixů umožňuje LILO spustit jádro umístěné kdekoli na pevném disku (pokud máte velký disk – 500 MB a více – podívejte se do dokumentace k LILO a seznamte se s možnými problémy). Pokud přidáte na konec konfiguračního souboru pro LILO kód:

```
image = /usr/src/linux/arch/i386/boot/bzImage
        label = new_kernel
```

budete moci spouštět nově přeložená jádra bez nutnosti přepisu původního jádra `/vmlinuz` (pochopitelně musíte spustit `lilo`). Nejjednodušší způsob, jak vybrat jádro, které se má použít, je stisknout klávesu shift v okamžiku startu systému (při zobrazení hlášky `LILO`) a následně zadat `new_kernel`.

Pokud chcete na svém disku uchovávat zdrojové kódy pro více verzí jádra (ale pozor na místo na disku), můžete používat oblíbený způsob pojmenování adresářů s jednotlivými verzemi jádra podle vzoru /usr/src/linux-x.y.z, kde x.y.z je číslo verze jádra. Poté můžete nastavit platný zdrojový kód pomocí symbolického odkazu (například příkazem `ln -sf linux-1.2.2 /usr/src/linux` nastavíte jako aktuální zdrojový kód jádra verze 1.2.2). Předtím, než vytvoříte tento symbolický odkaz si ověřte, že poslední argument neodpovídá skutečnému adresáři – v opačném případě nebude výsledek takový, jaký předpokládáte.

Inovace jádra

Změny mezi jednotlivými verzemi jádra sleduje a zaznamenává Russell Nelson (nelson@crynwr.com). Pokud se chcete na tyto změny před inovací jádra podívat, je tento dokument k dispozici na adrese http://www.crynwr.com/kchanges nebo na anonymním serveru ftp ftp.emlist.com v adresáři `pub/kchanges`.

Další užitečné dokumenty

- Sound-HOWTO: zvukové karty a utility
- SCSI-HOWTO: vše o SCSI řadičích a zařízeních
- NET-2-HOWTO: sítě
- PPP-HOWTO: sítě pomocí protokolu PPP
- PCMCIA-HOWTO: o ovladačích pro notebook
- ELF-HOWTO:
- Hardware-HOWTO: přehled podporovaného hardwaru
- Module mini-HOWTO: podrobnosti o modulech
- Kerneld mini-HOWTO: informace o kerneld
- BogoMips mini-HOWTO: pro případ, že budete udiveni

Různé

O autorovi

Autorem a správcem tohoto dokumentu je Brian Ward (bri@cs.uchicago.edu) a očekává libovolné komentáře, připomínky a opravy (ty nejvíce). Domovské stránky autora jsou dostupné na následujících adresách:

http://www.math.psu.edu/bri/

http://blah.math.tu-graz.ac.at/~bri/

Přestože se autor pokouší být pozorný a vstřícný jak to je jen možné, uvědomte si, že není v jeho silách odpovědět na všechny zprávy. Zejména pokud se na něj obracíte s otázkami, snažte se být co nejvíce jasní a přesní. Pokud se zaslané zprávy týkají nefunkčního hardwaru, nezapomeňte ke zprávě přiložit přesný popis hardwarové konfigurace. Pokud zasíláte hlášení o chybě, nepište zprávy stylu: „Zkusil jsem toto a objevila se chyba“, podstatné jsou všechny dostupné informace o vlastní chybě. Autor by také uvítal informace o verzi použitého jádra, překladače gcc a knihovny libc. Pokud autorovi sdělíte tyto informace či jakou distribuci používáte, bude vědět vše, co potřebuje. Autor nebere ohledy na jednoduché otázky – uvědomte si, že ne vždy musíte získat jeho odpověď. Stejně tak nemusí zodpovědět otázky netýkající se jádra či ty, které budou položeny v neznámém jazyce. I přesto bude autor vděčný za jakoukoli zpětnou vazbu.

Pokud autorovi zašlete závažnou zprávu a nedostane se vám odpovědi během tří týdnů či měsíce, je velká pravděpodobnost, že byla zpráva odstraněna, za což se autor omlouvá. Zopakujte, prosím, případně svůj dotaz.

Autorovi je zasíláno mnoho zpráv týkajících se hardwarové problematiky – to je v pořádku, ale pamatujte, že nemůže být specialistou na všechen existující hardware. Používá procesory AMD, řadiče SCSI firmy Adaptec a Symbios a disky SCSI od IBM.

Verze 0.1 byla napsána 3. října 1994 a tento dokument je dostupný ve formátech SGML, PostScript, TeX, roff a v ascii textu.

Plány do budoucna

V budoucnu by měla být rozšířena především kapitola věnovaná tipům a trikům, stejně jako kapitola věnovaná rozšiřujícím balíčkům. Zapracovány by měly být také další informace o debuggingu a zotavení z pádu systému.

Poděkování

Poděkování patří Linusu Torvaldsovi za zahrnutou část jeho souboru `README`. Na tomto dokumentu dále spolupracovali:

uc@brian.lunetix.de (Ulrich Callmeier): programy patch -s a xargs.

quinlan@yggdrasil.com (Daniel Quinlan): opravy a připomínky

nat@nat@nataa.fr.eu.org (Nat Makarevitch): utility mrproper, tar –p a některé další

boldt@math.ucsb.edu (Axel Boldt): ucelený popis voleb pro konfiguraci jádra dostupný na Internetu

lembark@wrkhors.psyber.com (Steve Lembark): problematika podpory koexistence více verzí jádra

kbriggs@earwax.pd.uwa.edu.au (Keith Briggs): některé opravy a připomínky

rmcguire@freenet.columbus.oh.us (Ryan McGuire): připomínky k programu make

dumas@excalibur.ibp.fr (Eric Dumas): překlad do francouzštiny

simazaki@ab11.yamanashi.ac.jp (Yasutada Shimazaki): překlad do japonštiny

jjamor@lml.ls.fi.upm.es (Juan Jose Amor Iglesias): překlad do španělštiny

mva@sbbs.se (Martin Wahlen): překlad do švédštiny

jzp1218@stud.u-szeged.hu (Zoltan Vamosi): překlad do maďarštiny

bart@mat.uni.torun.pl (Bartosz Maruszewski): překlad do polštiny

donahue@tiber.nist.gov (Michael J Donahue): sazba

rms@gnu.ai.mit.edu (Richard Stallman): koncept poznámek o distribuci a autorských právech

dak@Pool.Informatik.RWTH-Aachen.DE (David Kastrup): problematika NFS

esr@snark.thyrsus.com (Eric Raymond): některé zajímavosti

Poděkování patří také všem, kteří autorovi zaslali zprávu s otázkami a problémy.

Licenční ujednání a autorská práva

Další formáty tohoto dokumentu

V současné době je tento dokument k dispozici ve 12 různých formátech – DVI, PostScript, Latex, Adobe Acrobat PDF, LyX, GNU-info, HTML, RTF (Rich Text Format), ASCII text, manové stránky, jeden soubor HTML a SGML.

- Tento dokument jako jeden sbalený soubor tar ve formátu HTML, DVI, PostScript a SGML můžete získat z adresy http://www.linuxdoc.org/docs.html#howto.
- Tento dokument ve formátu čistého textu můžete získat z adresy http://www.linuxdoc.org/docs.html#howto
- Tento dokument jako jeden soubor ve formátu HTML můžete získat z adresy http://www.linuxdoc.org/docs.html#howto
- Překlady do francouzštiny, němčiny, španělštiny, čínštiny a japonštiny můžete získat z adresy http://www.linuxdoc.org/docs.html#howto. Jakékoli další překlady jsou vítány.

Tento dokument byl vytvořen pomocí nástrojů ze sady SGML, kterou můžete získat z adresy http://www.sgmltools.org/. Překlad zdrojového kódu tohoto dokumentu zajistíte například takto:

```
sgml2html Kernel-HOWTO.sgml (překlad do formátu HTML)
sgml2rtf Kernel-HOWTO.sgml (překlad do formátu RTF)
sgml2latex Kernel-HOWTO.sgml (překlad do formátu LaTeX)
```

Dokumenty formátu LaTeX mohou být jednoduše převedeny do formátu PDF pomocí utility `distill` (http://www.adobe.com), například takto:

```
bash$ man sgml2latex
bash$ sgml2latex jmenosouboru.sgml
bash$ latex jmenosouboru.tex
bash$ man dvips
bash$ dvips -o jmenosouboru.ps jmenosouboru.dvi
bash$ distill jmenosouboru.ps
bash$ man ghostscript
bash$ man ps2pdf
bash$ ps2pdf vstup.ps vystup.pdf
bash$ acroread vystup.pdf &
```

Můžete také použít příkaz `ps2pdf` z balíku Ghostscript. Tento příkaz pracuje téměř shodně s výše uvedeným příkazem `distill` – tedy převádí soubory ve formátu PostScript do formátu PDF (Portable Document Format). Tento příkaz je implementován jako malý skript volající Ghostscript a využívá speciální výstupní zařízení nazvané `pdfwrite`. Toto zařízení musí být ovšem při překladu balíku Ghostscript vybráno, více informací najdete v dokumentaci.

Anglickou podobu tohoto dokumentu můžete získat primárně na adrese

http://www.linuxdoc.org/

případně na těchto dalších:

http://www.caldera.com/LDP/HOWTO/Kernel-HOWTO.html

http://www.WGS.com/LDP/HOWTO/Kernel-HOWTO.html

http://www.cc.gatech.edu/linux/LDP/HOWTO/Kernel-HOWTO.html

http://www.redhat.com/linux-info/ldp/HOWTO/Kernel-HOWTO.html

Další umístění, která pro vás mohou být dostupnější, najdete na adrese http://sunsite.unc.edu/LDP/hmirrors.html (vlastní dokument pak naleznete v adresáři `/LDP/HOWTO/Kernel-HOWTO.html`).

Pro prohlížení dokumentu ve formátu dvi můžete použít program `xdvi`, který je součástí balíčku `tetex-xdvi*.rpm` (pro RedHat Linux):

```
xdvi -geometry 80x90 howto.dvi
man xdvi
```

Velikost okna můžete změnit pomocí myši, pro pohyb můžete používat šipky, klávesy PageUp, PageDown či klávesy f, d, u, c, l, r, p a n. Do rozšířené nabídky se dostanete stiskem klávesy x.

Pro prohlížení dokumentu ve formátu PostScript můžete použít program `gv` (ghostview) či `ghostscript`. První program je součástí balíčku `gv*.rpm`, druhý pak `ghostsript*.rpm`. Program `gv` je oproti programu `ghostscript` mnohem více uživatelsky přívětivý. Oba jsou dostupné také na dalších platformách, jako je OS/2, Windows 9x/NT/2000, takže tyto dokumenty můžete prohlížet i na těchto platformách. Ghostscript pro tyto platformy můžete získat na adrese http://www.cs.wisc.edu/~ghost. Vlastní dokument v tomto formátu pak můžete zobrazit například takto:

```
gv howto.ps
ghostscript howto.ps
```

Pro prohlížení dokumentu ve formátu HTML můžete používat webové prohlížeče Netscape Navigator, Microsoft Internet Explorer, RedHat Baron Web Browser a asi deset dalších.

Pro prohlížení dokumentu ve formátu latex můžete použít program `LyX` – X-window klienta pro latex.

KAPITOLA 7

Tavis Barr, tavis@mahler.econ.columbia.edu; Nicolai Langfeldt, janl@linpro.no
Seth Vidal, skvidal@phy.duke.edu

NFS

Preambule

Práva

Odvolání

Tento dokument je poskytován bez záruk, včetně obchodovatelnosti nebo vhodnosti pro dílčí použití. Autoři nejsou zodpovědní za to, zda následující instrukce v tomto dokumentu povedou ke zničení zařízení nebo dat, naštvou sousedy, poruší cizí zvyky, povedou k rozvodu nebo nějaké jiné kalamitě.

Připomínky

Toto nikdy nebude dokončený dokument; uvítáme připomínky vedoucí k jeho zlepšení. Od října 2000 je domovská stránka Linux NFS umístěna na adrese http://nfs.sourceforge.net. Podívejte se tam na diskusní skupiny, opravy chyb a aktualizace, a také se tam dozvíte, kdo aktuálně spravuje tento dokument.

Překlad

Pokud jste schopni přeložit tento dokument do jiného jazyka, budeme vám vděčni a budeme se snažit udělat vše pro to, abychom vám pomohli. Upozorněte prosím autory.

Věnování

NFS pro Linux bylo vytvořeno společným úsilím mnoha lidí, ale několik z nich zaslouží speciální uznání. Původní verze byla vyvinuta Olafem Kirchem a Alanem Coxem. Kód serveru verze 3 sestavil Neil Brown na základě práce Saadia Khan, James Yarbrough, Allen Morris, H.J. Lu a dalších (včetně jeho samotného). Kód klienta napsal Olaf Kirch a aktualizoval ho Trond Myklebust. Lock manager verze 4 vyvinul Saadia Khan. Dave Higgen a H. J. Lu společně provedli nevděčnou práci rozsáhlé správy a oprav chyb, aby kód skutečně pracoval předpokládaným způsobem. H. J. také hodně pracoval na vývoji balíčku s nástroji nfs-utils. Toto věnování samozřejmě neobsahuje všechny lidi.

Původní verzi tohoto dokumentu vytvořil Nicolai Langfeldt. V roce 2000 jej hodně přepracovali Tavis Barr a Seth Vidal, aby odpovídal podstatným změnám ve fungování NFS v Linuxu, které se objevily mezi jádry 2.0 a 2.4. Thomas Emmel, Neil Brown, Trond Myklebust, Erez Zadok a Ion Badulescu také poskytli cenné komentáře a příspěvky.

Úvod

Co je NFS?

Network File System (NFS) byl vyvinut pro umožnění připojování diskových oddílů na vzdáleném počítači jako kdyby se jednalo o místní pevný disk. Umožňuje rychlé a celistvé sdílení souborů v síti.

Také to dává možnost přístupu k vašemu pevnému disku po síti nevítaným lidem (a tudíž umožňuje čtení vašich e-mailů a odstraňování všech vašich souborů stejně jako průlom do vašeho systému) pokud jej nastavíte nesprávně. Takže si prosím pečlivě přečtěte část Zabezpečení tohoto dokumentu pokud hodláte implementovat NFS.

Existují další systémy, které poskytují podobné funkce jako NFS. Samba poskytuje souborové služby pro klienty Windows. Souborový systém Andrew od IBM (http://www.transarc.com/Product/EFS/AFS/index.html), nedávno uvolněn pod open-source, poskytuje mechanismus sdílení souborů s určitým dalším zabezpečením a funkcemi pro výkon. Souborový systém Coda (http://www.coda.cs.cmu.edu/) je stále ve vývoji, ale je navržen tak, aby pracoval dobře s odpojenými klienty. Mnoho funkcí souborových systémů Andrew a Coda je připraveno pro zahrnutí do další verze NFS (verze 4) (http://www.nfsv4.org). Dnešní výhodou NFS je to, že je vyzrálý, standardní, dobře pochopitelný a hodně podporovaný pro různé platformy.

Co je toto HOWTO a co není

Toto HOWTO je myšleno jako kompletní návod krok za krokem pro správné a efektivní nastavení NFS. Nastavování NFS zahrnuje dva kroky, jmenovitě konfiguraci serveru a konfiguraci klienta. Oba tyto kroky spolu souvisí. Dokument také nabízí některé tipy pro lidi s konkrétními potřebami a nastavení hardwaru, stejně jako rady pro zabezpečení a odstraňování problémů.

Toto HOWTO není popisem struktury NFS. Kvůli tomu si možná budete chtít přečíst knihu *Managing NFS and NIS*, kterou napsal Hal Stern a vydalo ji nakladatelství O'Reilly & Associates, Inc. I když je tato kniha hodně zastaralá, většina struktury NFS se nezměnila a kniha jej popisuje velmi dobře. Mnohem dokonalejší a novější technický popis NFS je k dispozici v *NFS Illustrated* od Brenta Callaghana.

Tento dokument také není myšlen jako kompletní referenční příručka a neobsahuje kompletní seznam funkcí linuxové implementace NFS. Pro to se můžete podívat na manuálové stránky pro *nfs(5)*, *exports(5)*, *mount(8)*, *fstab(5)*, *nfsd(8)*, *lockd(8)*, *statd(8)*, *rquotad(8)* a *mountd(8)*.

Také nebude zahrnovat PC-NFS, který je považován za zastaralý (uživatelům se doporučuje používat pro sdílení souborů s PC systém Samba) nebo NFS verze 4, která je stále ve vývoji.

Předpokládané znalosti

Před čtením tohoto HOWTO byste měli znát určité základy sítí TCP/IP. Pokud jste na pochybách, přečtěte si Networking-Overview-HOWTO.

Předpoklady pro software: verze jádra a nástroje NFS

Rozdíl mezi NFS verze 2 a 3 bude vysvětlen později; nyní byste mohli jednoduše přijmout doporučení použít NFS verze 3, pokud instalujete dedikovaný nebo rozsáhlý souborový server. NFS verze 2 by měla být dostačující pro nevýznamné použití.

NFS verze 2 byla k dispozici už před nějakou dobou (alespoň od série jádra 1.2). Jestliže budete chtít udělat něco z následujícího, budete potřebovat jádro verze alespoň 2.2.18:

- Míchat Linux NFS s NFS jiných operačních systémů
- Použít spolehlivé zamykání souborů v NFS
- Použít NFS verze 3.

Existují také záplaty pro jádra verzí nad 2.2.14, které umožňují výše uvedené funkce. Některé z nich je možno stáhnout z domovské stránky Linux NFS. Pokud máte jádro verze 2.2.14 - 2.2.17 a máte zdrojové kódy, můžete říci, zda byly tyto záplaty přidány, protože podpora serveru NFS verze 3 bude konfigurační volbou. Samozřejmě, pokud nemáte nějaký konkrétní důvod pro použití staršího jádra, měli byste provést inovaci, protože mezitím bylo opraveno mnoho chyb.

Funkce verze 3 budou také vyžadovat balíček nástrojů nfs-utils verze alespoň 0.1.6, a mount verze 2.10m nebo novější. Protože jsou samozřejmě nfs-utils a mount plně zpětně kompatibilní, a protože mají novější verze mnoho oprav chyb a zabezpečení, neexistuje dobrý důvod pro nenainstalování nejnovějších balíčků nfs-utils a mount, když začínáte nastavovat NFS.

Všechna jádra 2.4 a vyšší mají všechny funkce NFS verze 3.

Všechna jádra po 2.2.18 podporují NFS přes TCP na straně klienta. V době vytváření tohoto dokumentu je NFS na straně serveru přes TCP k dispozici pouze v pozdějších verzích 2.2 (ale ne ještě v jádrech 2.4), je považováno za experimentální a obsahující chyby.

Protože mnoho z výše uvedených funkcí bylo představeno v jádře verze 2.2.18, byl tento dokument napsán tak, aby byl konzistentní s jádry od této verze (včetně 2.4.x). Pokud máte starší jádro, nemusí tento dokument popisovat váš systém NFS správně.

Když jsme psali tento dokument, byl NFS verze 4 stále ve vývoji jako protokol a nebudeme se jím zde zabývat.

Kde získat nápovědu a další informace

Od listopadu 2000 je domovská stránka Linux NFS na adrese http://nfs.sourceforge.net. Najdete tam odkazy na diskusní skupiny týkající se NFS, stejně jako poslední verzi nfs-utils, záplaty jádra NFS a další balíčky související s NFS.

Také se možná budete chtít podívat na manuálové stránky pro *nfs(5)*, *exports(5)*, *mount(8)*, *fstab(5)*, *nfsd(8)*, *lockd(8)*, *statd(8)*, *rquotad(8)* a *mountd(8)*.

Nastavení serveru NFS

Úvod do nastavení serveru

Předpokládejme, že budete instalovat server i klienta. Pokud pouze instalujete klienta pro práci se serverem někoho jiného (řekněme ve vašem oddělení), můžete přejít rovnou na sekci 4. Samozřejmě, každý klient, který je nastaven, vyžaduje úpravy na serveru, aby ověřil, že klient k serveru může přistupovat (pokud není server nastaven jako velmi nezabezpečený), takže i když nenastavujete server, možná si budete chtít tuto část přečíst, abyste věděli, jaké druhy problémů s ověřováním mohou nastat.

Nastavování serveru bude prováděno ve dvou krocích: nastavování konfiguračních souborů pro NFS a pak spouštění služeb NFS.

Nastavování konfiguračních souborů

Jsou zde tři hlavní konfigurační soubory, které budete pro nastavování serveru NFS potřebovat upravit: /etc/exports, /etc/hosts.allow, a /etc/hosts.deny. Přesněji řečeno, pro spuštění NFS budete potřebovat upravit pouze /etc/exports, ale bylo by to extrémně nezabezpečené nastavení. Možná budete také potřebovat upravit vaše spouštěcí skripty;

/etc/exports

Tento soubor obsahuje seznam položek; každá položka indikuje svazek, který je sdílen, a to, jak je sdílen. Podívejte se do manuálových stránek (**man exports**) na kompletní popis všech možností nastavení tohoto souboru, ačkoliv následující popis bude pravděpodobně pro potřeby většiny lidí stačit.

Položka v /etc/exports bude typicky vypadat jako tato:

```
directory machine1(option11,option12) machine2(option21,option22)
```

kde

directory

adresář, který chcete sdílet. Může i nemusí to být celý svazek. Pokud sdílíte adresář, pak budou sdíleny všechny adresáře, které se v něm nachází a jsou na stejném souborovém systému.

machine1 a machine2

počítače klientů, které budou mít přístup k adresáři. Počítače mohou být uvedeny jako jejich adresa IP nebo jejich jméno DNS (např. *machine.company.com* nebo *192.168.0.8*). Použití adres IP je spolehlivější a bezpečnější.

optionxx

seznam možností pro každý počítač, který popisuje, jež druh přístupu tento počítač bude mít. Důležité možnosti jsou:

- **ro**: Adresář je sdílen pouze pro čtení; počítač klienta do něj nebude moci zapisovat. Toto je implicitní nastavení.
- **rw**: Počítač klienta bude mít přístup k adresáři pro čtení a zápis.
- **no_root_squash**: Implicitně jsou všechny požadavky na soubory vytvořené uživatelem root na počítači klienta zpracovávány jako by je prováděl na serveru uživatel nobody (přesněji, ke kterému UID je požadavek mapován, závisí na UID uživatele „nobody“ na serveru, nikoliv u klienta). Pokud je zvoleno no_root_squash, pak root na počítači klienta bude mít stejnou úroveň přístupu k souborům systému jako root na serveru. To může mít vážný vliv na zabezpečení, ačkoliv to může být nezbytné pokud chcete provádět nějakou správu na počítači klienta, kterého se týkají exportované adresáře. Pokud k tomu nemáte dobrý důvod, neměli byste tuto volbu zadávat.
- **no_subtree_check**: Pokud je exportována pouze část svazku, rutina nazývaná ověřování podstromu (subtree checking) ověřuje, že soubor, který je požadován klientem je v příslušné části svazku. Pokud je exportován celý svazek, vypnutí této kontroly urychlí přenosy.
- **sync**: Implicitně server NFS verze 2 sdělí počítači klienta, že je zápis souboru dokončen, když NFS dokončí práci zápisu na systém souborů; samozřejmě, systém souborů jej nemů-

že synchronizovat s diskem, i když klient provede volání sync() na systému souborů. Implicitní chování proto může způsobit poškození souboru v případě restartu serveru. Tato volba si vynutí v systému souborů synchronizaci s diskem pokaždé, když NFS dokončí operaci zápisu. To podstatně zpomaluje zápisy, ale může to být nezbytné při provozování NFS verze 2 v produkčním prostředí. NFS verze 3 má operaci, kterou může klient zavolat a to se projeví synchonizací disku na straně serveru.

Předpokládejme, že máme dva počítače klientů, *slave1* a *slave2*, které mají adresy IP *192.168.0.1* a *192.168.0.2*. Chceme sdílet náš software a domovské adresáře s těmito počítači. Typické nastavení /etc/exports by mohlo vypadat takto:

```
/usr/local   192.168.0.1(ro) 192.168.0.2(ro)
/home        192.168.0.1(rw) 192.168.0.2(rw)
```

Zde sdílíme /usr/local pouze pro čtení s počítači slave1 a slave2, protože pravděpodobně obsahuje náš software a nebudou zde žádné výhody, když umožníme počítačům slave1 a slave2 zápis do něj, a to zvýší zabezpečení. Na druhou stranu je potřeba domovské adresáře exportovat pro čtení i zápis, pokud do nich uživatelé mají ukládat svou práci.

Pokud máte rozsáhlou instalaci, můžete zjistit, že máte nějaké počítače, které jsou všechny na stejné místní síti a vyžadují přístup k vašemu serveru. Existuje několik způsobů zjednodušování odkazů na velký počet počítačů. Můžete najednou zadat přístup pro rozsah počítačů uvedením sítě a masky sítě. Například, pokud chcete umožnit přístup všem počítačům s adresami IP mezi *192.168.0.0* a *192.168.0.255* pak byste měli mít tyto položky:

```
/usr/local 192.168.0.0/255.255.255.0(ro)
/home      192.168.0.0/255.255.255.0(rw)
```

Podívejte se do Networking-Overview HOWTO, kde najdete více informací o tom, jak pracují masky sítě a také byste se mohli podívat na manuálové stránky pro init a hosts.allow.

Další možností je použití síťových skupin NIS ve vaší položce. Pro zadání síťové skupiny ve vašem souboru /etc/exports jednoduše předřaďte název síťové skupiny s „@“. Podívejte se do NIS HOWTO na detaily o tom, jak síťové skupiny pracují.

Další možností je použití zástupných znaků jako **.foo.com* nebo *192.168.* namísto názvů hostitelů.

Samozřejmě byste měli myslet na to, že některé z těchto zjednodušení může způsobit bezpečnostní riziko, pokud jsou ve vaší síťové skupině nebo lokální síti počítače, kterým zcela nedůvěřujete.

Několik varování se váže k tomu, co nelze (nebo by nemělo být) exportovat. Nejprve, pokud je adresář exportován, jeho nadřazené nebo vnořené adresáře nemohou být exportovány, pokud se nachází na stejném souborovém systému. Samozřejmě, exportování obojího by nemělo být nutné, protože uvedení nadřazeného adresáře v souboru /etc/exports způsobí to, že všechny vnořené adresáře v tomto souborovém systému budou exportovány.

Dále, není dobrý nápad exportovat systém souborů FAT nebo VFAT (tj. MS-DOS nebo Windows 95/98) přes NFS. FAT není navržen pro použití na víceuživatelském stroji, a díky tomu operace, které závisí na oprávněních, nebudou pracovat správně. Kromě toho je hlášeno, že některé ze systémů souborů v nižší vrstvě pracují špatně v NFS.

Dále se soubory zařízení nebo jiné speciální soubory nemusí exportovat správně pro klienty, kteří nepoužívají Linux. Podívejte se do sekce 8 na detaily ke konkrétním operačním systémům.

/etc/hosts.allow a /etc/hosts.deny

Tyto dva soubory určují, které počítače v síti mohou používat služby na vašem počítači. Každý řádek souboru je položkou seznamu služeb a nastavení počítačů. Když server obdrží požadavek od počítače, provede následující:

- Nejprve zkontroluje hosts.allow, aby zjistil, zda počítač odpovídá popisu uvedenému zde. Pokud ano, pak je počítači povolen přístup.
- Pokud počítač neodpovídá položce v hosts.allow, pak server zkontroluje hosts.deny, aby zjistil, zda klient odpovídá seznamu v něm. Pokud ano, je počítači odepřen přístup.
- Pokud klient neodpovídá seznamům ani v jednom z těchto souborů, je mu povolen přístup.

Kromě řízení přístupu ke službám obsluhovaným inetd (jako např. telnet a FTP), tento soubor může také řídit přístup k NFS omezením připojení k démonům, které poskytují služby NFS. Omezování se provádí podle služeb.

První démon, ke kterému je omezen přístup je portmapper. Tento démon v podstatě říká klientům jak nalézt všechny služby NFS v systému. Omezení přístupu k démonu portmapper je nejlepší obranou proti komukoliv, kdo se pokouší prolomit do vašeho systému skrze NFS, protože neautorizovaní klienti nebudou vědět, jak nalézt démony NFS. Samozřejmě, existují dvě věci, na které byste si měli dát pozor. Zaprvé omezení portmapperu nestačí, pokud vetřelec již ví, z nějakého důvodu, jak tyto démony nalézt. Zadruhé, pokud provozujete NIS, omezení portmapperu také omezí požadavky pro NIS. To by mělo být obvykle neškodné, jelikož obvykle chcete omezit NFS i NIS podobným způsobem, ale mějte se na pozoru (Provozování NIS je obecně dobré, pokud provozujete NFS, protože počítače klientů potřebují způsob zjišťování toho, kdo vlastní které soubory na exportovaných svazcích. Samozřejmě pro to existují jiné způsoby, jako např. synchronizace souborů hesel. Podívejte se do NIS HOWTO na informace o nastavování NIS).

Obecně je dobré u NFS (jako u většiny internetových služeb) explicitně zakázat přístup k hostitelům, kterým nepotřebujete povolit přístup.

Prvním krokem je přidání následující položky do /etc/hosts.deny:

```
portmap:ALL
```

Pomocí nfs-utils 0.2.0 můžete být trochu opatrnější při řízení přístupu k jednotlivým démonům. Je to dobré opatření, jelikož vetřelec bude často schopen obejít portmapper. Pokud máte novější verzi NFS-utils, přidejte položky pro všechny démony NFS (viz. následující sekce, kde najdete, o které démony se jedná; nyní jen vložte jejich položky do hosts.deny):

```
lockd:ALL
mountd:ALL
rquotad:ALL
statd:ALL
```

I když máte starší verzi *nfs-utils*, je přidání těchto položek v nejhorším případě neškodné (jelikož prostě budou ignorovány) a v nejlepším případě vám ušetří některé problémy při inovaci. Někteří správci systému vkládají **ALL:ALL** do souboru /etc/hosts.deny, což způsobí, že služby, které se dívají do těchto souborů pro odepření přístupu všem hostitelům, pokud není explicitně povolen. I když je to bezpečnější chování, může to také způsobit problémy při instalaci nových služeb, jelikož zapomenete, že jste to vložili a nemůžete přijít na to, proč nechtěji fungovat.

Dále, je potřeba přidat položku do hosts.allow, abyste umožnili přístup některým hostitelům, kterým ho chcete umožnit (pokud jen ponecháme výše uvedené řádky v souboru hosts.deny, pak nikdo nebude mít přístup do NFS). Položky v hosts.allow mají formát

```
service: host [nebo network/netmask] , host [nebo network/netmask]
```

Zde je host adresa IP potenciálního klienta; v některých verzích může být umožněno používat název DNS hostitele, ale nedoporučuje se to.

Předpokládejme, že musíme nastavit výše uvedené a chceme povolit přístup pouze pro *slave1.foo.com* a *slave2.foo.com*, a předpokládejme, že adresy IP těchto počítačů jsou *192.168.0.1* a *192.168.0.2*. Měli bychom přidat následující položku do /etc/hosts.allow:

```
portmap: 192.168.0.1 , 192.168.0.2
```

Pro poslední verze nfs-utils bychom měli také přidat následující (znovu, tyto položky jsou neškodné, i když nejsou podporovány):

```
lockd:   192.168.0.1 , 192.168.0.2
rquotad: 192.168.0.1 , 192.168.0.2
mountd:  192.168.0.1 , 192.168.0.2
statd:   192.168.0.1 , 192.168.0.2
```

Pokud hodláte spustit NFS pro velký počet počítačů v lokální síti, /etc/hosts.allow také umožňuje položky se sítí nebo maskou sítě stejným způsobem jako /etc/exports výše.

Spouštění služeb

Předpoklady

Server NFS by měl být nyní nastaven a můžeme jej spustit. Nejprve budete potřebovat mít nainstalované příslušné balíčky. Zejména dostatečně nové jádro a dostatečně nový balíček nfs-utils. Pokud máte problémy, podívejte se do sekce 2.4.

Dále, předtím, než můžete spustit NFS, budete potřebovat na vašem počítači správně fungující síť TCP/IP. Pokud můžete používat telnet, FTP apod., pak pravděpodobně vaše síť TCP funguje správně.

To znamená, že ve většině nových distribucí Linuxu můžete NFS nastavit a spustit jednoduše restartováním vašeho počítače, a spouštěcí skripty by měly detekovat, že máte nastavený váš soubor /etc/exports a spustí NFS správně. Pokud to zkoušíte, podívejte se na sekci 3.4. Pokud nepracuje nebo pokud nemůžete počítač restartovat, pak vám následující sekce řekne, které démony je potřeba spustit, aby se spustily služby NFS. Pokud z nějakého důvodu nfsd již běžel, když jste upravovali výše uvedené konfigurační soubory, budete muset zrušit vaši konfiguraci; na detaily se podívejte do sekce 3.5.

Spouštění Portmapperu

NFS závisí na démonu portmapper, nazývaném **portmap** nebo **rpc.portmap**. Bude jej potřeba nejprve spustit. Měl by se nacházet v /sbin, ale někdy je v /usr/sbin. Většina posledních distribucí systému Linux spouští tohoto démona ve spouštěcích skriptech, ale je dobré se ujistit, že běží, než začnete pracovat s NFS (jednoduše napište **ps aux | grep portmap**).

Démoni

Práce NFS je zajišťována pěti démony: rpc.nfsd, který provádí většinu práce; rpc.lockd a rpc.statd, kteří se starají o zamykání souborů; rpc.mountd, který zpracovává počáteční požadavky na připojení, a rpc.rquotad, který se stará o kvóty souborů uživatelů na exportovaných svazcích. Od 2.2.18 je lockd volán na požadavek nfsd, takže se nemusíte starat o jeho spouštění. statd bude potřeba spustit odděleně. Většina posledních distribucí systému Linux bude mít pro tyto démony spouštěcí skripty.

Démoni jsou všechny součástí balíčku nfs-utils a mohou být v adresáři /sbin nebo /usr/sbin.

Pokud je vaše distribuce neobsahuje ve spouštěcích skriptech, pak byste je měli přidat, nastavené pro spuštění v následujícím pořadí:

rpc.portmap

rpc.mountd, rpc.nfsd

rpc.statd, rpc.lockd (pokud je to nutné),

rpc.rquotad

Balíček nfs-utils obsahuje ukázkové spouštěcí skripty pro RedHat a Debian. Pokud používáte jinou distribuci, obecně můžete prostě zkopírovat skript pro RedHat, ale pravděpodobně budete muset odstranit řádek, který obsahuje:

```
. ../init.d/functions
```

abyste se vyhnuli chybovým zprávám.

Ověřování, že NFS funguje

Za tímto účelem se dotážete portmapperu, příkazem **rpcinfo -p**, jaké služby poskytuje. Měli byste obdržet něco jako:

```
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100011    1   udp    749  rquotad
100011    2   udp    749  rquotad
100005    1   udp    759  mountd
100005    1   tcp    761  mountd
100005    2   udp    764  mountd
100005    2   tcp    766  mountd
100005    3   udp    769  mountd
100005    3   tcp    771  mountd
100003    2   udp   2049  nfs
100003    3   udp   2049  nfs
300019    1   tcp    830  amd
300019    1   udp    831  amd
100024    1   udp    944  status
100024    1   tcp    946  status
100021    1   udp   1042  nlockmgr
100021    3   udp   1042  nlockmgr
100021    4   udp   1042  nlockmgr
100021    1   tcp   1629  nlockmgr
100021    3   tcp   1629  nlockmgr
100021    4   tcp   1629  nlockmgr
```

Toto říká, že máme NFS verze 2 a 3, rpc.statd verze 1, network lock manager (název služby pro rpc.lockd) verze 1, 3 a 4. Je zde seznam jiných služeb v závislosti na tom, zda NFS jde přes TCP nebo UDP. Systémy Linux používají implicitně UDP, pokud není explicitně požadován TCP; samozřejmě, jiné operační systémy, jako např. Solaris, implicitně používají TCP.

Pokud neuvidíte alespoň řádek, který obsahuje „portmapper", řádek, který obsahuje „nfs", a řádek, který obsahuje „mountd", budete se muset vrátit zpět a zkusit znovu spustit démony.

Pokud jsou tyto služby uvedeny, pak byste měli být připraveni pro nastavování klientů NFS pro přístup k souborům na vašem serveru.

Pozdější změny v /etc/exports

Pokud se vrátíte a změníte váš soubor /etc/exports, neprojeví se změny, které jste provedli, ihned. Měli byste spustit příkaz **exportfs -ra** pro znovunačtení souboru /etc/exports nfsd. Pokud nemůžete najít příkaz **exportfs**, pak můžete poslat nfsd signál **HUP** (podívejte se do manuálových stránek na detaily o kill).

Pokud to stále nepracuje, nezapomeňte se podívat do hosts.allow, abyste se ujistili, že jste zde nezapomněli uvést nové klientské počítače. Také zkontrolujte seznam hostitelů na firewallech, které byste mohli mít nastavené.

Nastavování klienta NFS

Připojování vzdálených adresářů

Předtím, než začnete, byste měli dvakrát zkontrolovat, že je váš program mount dostatečně nový (verze 2.10m, pokud chcete používat NFS verze 3), a že počítač klienta podporuje připojování NFS, ačkoliv to většina standardních distribucí dělá. Pokud používáte jádro 2.2 nebo novější se systémem souborů /proc, můžete druhou možnost zkontrolovat přečtením souboru /proc/filesystems a ujištěním se, že je zde řádek obsahující nfs. Pokud ne, budete muset sestavit (nebo stáhnout) jádro, které má vestavěnou podporu NFS.

Abyste mohli začít používat počítač jako klienta NFS, budete na něm potřebovat spuštěný portmapper a používat zamykání souborů NFS a budete také potřebovat rpc.statd a rpc.lockd spuštěné na klientovi i na serveru. Většina posledních distribucí spouští tyto služby implicitně při spuštění; pokud ta vaše ne, další na informace o tom, jak je spustit najdete dále v této kapitole.

Pokud běží portmapper, lockd a statd, měli byste být nyní schopni se připojit ke vzdálenému adresáři na vašem serveru jako kdybyste připojovali místní pevný disk, pomocí příkazu mount. Při pokračování v našem příkladu z předchozí sekce předpokládáme, že se náš server nazývá *master.foo.com*, a chceme připojit adresář /home na *slave1.foo.com*. Pak všechno, co musíme udělat na konzole na *slave1.foo.com*, je napsat:

```
# mount master.foo.com:/home /mnt/home
```

a adresář /home na master se objeví jako adresář /mnt/home na *slave1*.

Pokud to nefunguje, podívejte se do sekce Odstraňování problémů (Sekce 7).

Tohoto souborového systému se můžete zbavit napsáním

```
# umount /mnt/home
```

stejně jako byste to dělali pro místní souborový systém.

Připojování souborových systémů NFS při spuštění

Souborové systémy NFS lze přidat do vašeho souboru /etc/fstab stejným způsobem, jako kdyby se jednalo o místní souborové systémy, aby se připojily při spuštění vašeho systému. Jediný rozdíl je v tom, že druh souborového systému bude nastaven na **nfs** a dump a pořadí fsck (poslední dvě položky) budou muset být nastaveny na nuly. Takže pro náš příklad by položka v /etc/fstab vypadala takto:

```
# device        mountpoint      fs-type      options       dump fsckorder
...
master.foo.com:/home  /mnt     nfs          rw            0    0
...
```

Podívejte se do manuálové stránky na fstab, pokud neznáte syntaxi tohoto souboru. Pokud používáte automonter, jako např. amd nebo autofs, měly by možnosti v odpovídajících polích vašeho seznamu pro připojování vypadat velmi podobně, pokud ne stejně.

V tuto chvíli byste měli mít fungující NFS, ačkoliv může být potřeba udělat několik malých oprav, aby pracoval správně.

Možnosti připojování

Připojování Soft vs. Hard

Existují některé volby, jejichž přidání byste měli zvážit. Řídí způsob, kterým klient NFS zpracovává havárii serveru nebo výpadek sítě. Jednou ze skvělých věcí na NFS je to, že to umí elegantně zpracovat. Pokud nastavíte práva klientů; existují dva odlišné režimy selhání:

soft

Pokud požadavek na soubor selže, klient NFS nahlásí chybu procesu na klientském počítači, který požaduje přístup k souboru. Některé programy to umí zpracovat s klidem, většina ne. Nedoporučujeme používat toto nastavení; je to recept na poškození souborů a ztrátu dat. Speciálně byste to neměli používat pro disky s poštou - pokud si vážíte vaší pošty.

hard

Program přistupující k souboru na připojeném souborovém systému NFS bude zastaven při selhání serveru. Proces nelze přerušit pokud nenastavíte také intr. Když je server NFS opět online, program bude pokračovat nerušeně tam, kde skončil. Doporučujeme používat hard, intr u všech připojených souborových systémů NFS.

Pro předchozí příklad by položka fstab vypadala takto:

```
# device              mountpoint  fs-type   options       dump  fsckorder
...
master.foo.com:/home  /mnt/home   nfs       rw,hard,intr  0     0
...
```

Nastavení velikosti bloku pro optimalizaci rychlosti přenosu

Volby připojení **rsize** a **wsize** určují velikost částí dat, které cestují mezi klientem a serverem.

Implicitní hodnoty mohou být příliš velké nebo příliš malé; neexistuje žádná velikost, která pracuje dobře na všech nebo většině nastavení. Na jednu stranu, některé kombinace jádra systému Linux a síťové karty (převážně na starších počítačích) nemohou pracovat s bloky této velikosti. Na druhou stranu, pokud umí pracovat s většími bloky, větší velikost může být rychlejší.

Správné nastavení velikosti bloku je důležitým faktorem pro výkon a je nutné, pokud plánujete používat server NFS v produkčním prostředí. Detaily naleznete v sekci 5.

Optimalizace výkonu NFS

Správné nastavení sítě může mnohonásobně zvýšit výkon NFS – desateronásobné zvýšení přenosové rychlosti není nic neslýchaného. Nejdůležitější věci, které je potřeba nastavit na správné hodnoty, jsou volby **rsize** a **wsize** příkazu **mount**. Jiné níže uvedené faktory mohou ovlivnit jednotlivce s konkrétními nastaveními hardwaru.

Nastavení velikosti bloků pro optimalizaci přenosové rychlosti

Volby **rsize** a **wsize** pro **mount** určují velikost částí dat, které si předávají klient a server. Pokud volby **rsize** a **wsize** nejsou uvedeny, liší se implicitní hodnoty podle verze NFS, kterou používáme. 4096 bajtů je nejčastější implicitní hodnota, ačkoliv pro připojení přes TCP v jádrech 2.2, a pro všechna připojení počínaje jádry 2.4, server určuje implicitní velikost bloku.

Implicitní hodnoty mohou být příliš velké nebo příliš malé. Na jednu stranu, některé kombinace jader systému Linux a síťových karet (zejména na starších počítačích) neumí zpracovávat bloky této velikosti. Na druhou stranu, pokud umí zpracovávat bloky větší velikosti, může být větší velikost rychlejší.

Takže budeme experimentovat a najdeme rsize a wsize, které pracují a jsou tak rychlé, jak je to jen možné. Rychlost vašich voleb můžete otestovat pomocí některých jednoduchých příkazů.

První z těchto příkazů přenáší 16384 bloků po 16k ze speciálního souboru /dev/zero (který, pokud jej čtete prostě velmi rychle produkuje nuly) na připojovaný oddíl. Budeme zjišťovat, kolik času to zabere. Takže na počítači klienta napište:

```
# time dd if=/dev/zero of=/mnt/home/testfile bs=16k count=16384
```

Toto vytvoří soubor o velikosti 256 MB nulových bajtů. Obecně byste měli vytvořit soubor, který je alespoň dvakrát větší než paměť RAM na serveru, ale ujistěte se, že máte dostatek místa na disku! Pak zpětně přečtěte soubor do skvělé černé díry na počítači klienta (/dev/null) pomocí:

```
# time dd if=/mnt/home/testfile of=/dev/null bs=16k
```

Opakujte to několikrát a vypočtěte, jak dlouho to v průměru trvá. Nezapomeňte pokaždé odpojit a znovu připojit souborový systém (na straně klienta a, pokud jste horliví, také místně na serveru), což by mělo vyprázdnit vyrovnávací paměť.

Pak proveďte znovu odpojení a opět připojení s větší a menší velikostí bloku. Měly by to být násobky 1024, a ne větší než 8192 bajtů, jelikož to je maximum v NFS verze 2 (ačkoliv, pokud používáte verzi 3, mohli byste zkusit až 32768). Moudré je používat velikost bloku o velikosti mocniny dvou, jelikož většina parametrů, které by to omezily (jako např. velikosti bloků souborového systému a velikost síťových paketů), jsou také mocniny dvou. Samozřejmě, někteří uživatelé hlásili větší úspěch s velikostmi bloku, které nejsou mocninami dvou, ale jsou stále násobky velikosti bloků souborového systému a velikosti síťových paketů.

Přímo po připojení s větší velikostí přejděte do připojeného souborového systému a provádějte věci jako ls, podívejte se na souborový systém, abyste se ujistili, že je vše tak, jak by mělo být. Pokud je velikost rsize/wsize příliš velká, jsou symptomy velmi náhodné a ne 100% zřejmé. Ty-

pickým symptomem je nekompletní seznam souborů při provádění 'ls' a žádné chybové zprávy. Nebo čtení souborů selhává záhadně bez chybových zpráv. Po stanovení daných rsize/wsize, které fungují, můžete provést testy rychlosti znovu. Různé platformy serveru budou mít pravděpodobně různé optimální velikosti. SunOS a Solaris jsou podle obecného mínění mnohem rychlejší s bloky o velikosti 4096 bajtů než s jakoukoliv jinou.

Nezapomeňte upravit /etc/fstab tak, aby odpovídal velikostem rsize/wsize, které jste zjistili.

Velikost paketů a ovladače sítě

Existuje mnoho špatných ovladačů pro Linux, včetně těch pro některé docela standardní karty.

Zkuste ping mezi dvěma počítači s velkými pakety pomocí voleb -f a -s u **ping** (viz. **man ping**) pro více informací a zjištění toho, zda byly pakety přijaty nebo zda trvala odpověď moc dlouho. Pokud ano, možná máte problém s výkonem vaší síťové karty.

Pro opravu takového problému budete možná muset přenastavit velikost paketu, kterou používá vaše síťová karta. Velmi často je omezení někde jinde v síti (jako např. směrovač), které snižuje maximální velikost paketů mezi dvěma počítači, zatímco síťové karty v počítačích zvládnou více. TCP by mělo automaticky zjistit příslušnou velikost paketu pro síť, ale UDP jednoduše zůstane na výchozí hodnotě. Takže zjišťování příslušné velikosti paketu je důležité zejména pokud používáte NFS přes UDP.

Velikost síťového paketu můžete testovat pomocí příkazu tracepath: na klientském počítači jednoduše napište **tracepath [server] 2049** a měla by být vrácena hodnota MTU. Pak můžete nastavit MTU pro vaši síťovou kartu stejnou jako MTU pomocí volby MTU příkazu **ifconfig** a sledovat, zda bylo ztraceno menší množství paketů. Podívejte se do manuálových stránek pro **ifconfig** na detaily ohledně nastavení MTU.

Počet instancí NFSD

Většina spouštěcích skriptů v Linuxu spouští 8 instancí nfsd. V dávné minulosti NFS Sun stanovil toto číslo jako pravidlo a všichni ostatní je kopírovali. Neexistuje žádné dobré měřítko toho, kolik instancí je optimální, ale zatíženější server může vyžadovat více. Pokud používáte jádro 2.4 nebo vyšší a chcete zjistit, jak silně je každé vlákno nfsd využíváno, můžete se podívat do souboru /proc/net/rpc/nfsd. Posledních deset čísel na řádce *th* v tomto souboru indikuje počet sekund, kdy bylo vlákno maximálně využíváno. Pokud jsou první tři čísla velká, možná budete chtít zvýšit počet instancí **nfsd**. To je prováděno při spouštění **nfsd** pomocí počtu instancí zadaného jako volba na příkazovém řádku. Podívejte se do manuálových stránek **nfsd** na další informace.

Omezení paměti ve vstupní frontě

Na jádrech 2.2 a 2.4 vstupní fronta socketů, kde se řadí požadavky zatímco jsou aktuálně zpracovávány, má nízké omezení výchozí velikosti na 64k. To znamená, že pokud používáte 8 instancí **nfsd**, každá bude mít pouze 8k pro uložení požadavků, které jsou zpracovávány.

Měli byste zvážit zvýšení této velikosti na alespoň 256k pro **nfsd**. Toto omezení je nastaveno v souborovém systému proc pomocí souborů /proc/sys/net/core/rmem_default a /proc/sys/net/core/rmem_max. Může být zvýšeno ve třech krocích; následující metoda by měla fungovat a neměla by způsobit nějaké problémy:

a. Zvýšení velikosti uvedené v souboru:

```
echo 262144 > /proc/sys/net/core/rmem_default
echo 262144 > /proc/sys/net/core/rmem_max
```

b. Restartování **nfsd**, např. **/etc/rc.d/init.d/nfsd restart** pro Red Hat

c. Vrácení omezení velikosti na jejich normální velikost pro případ, že na tom závisí jiné systémy jádra:

```
echo 65536 > /proc/sys/net/core/rmem_default
echo 65536 > /proc/sys/net/core/rmem_max
```

d. Ujistěte se, že jste provedli tento poslední krok, protože byly hlášeny havárie počítačů, když tyto hodnoty byly ponechány změněné po delší dobu.

Přetečení fragmentovaných paketů

Protokol NFS používá fragmentované pakety UDP. Jádro má omezení toho, kolik fragmentů neúplných paketů může udržet ve vyrovnávací paměti předtím, než se začnou zahazovat pakety. U jader 2.2, která podporují souborový systém /proc, můžete zadat počet úpravou souborů /proc/sys/net/ipv4/ipfrag_high_thresh a /proc/sys/net/ipv4/ipfrag_low_thresh.

Jakmile počet nezpracovaných, fragmentovaných paketů dosáhne počtu zadaného pomocí **ipfrag_high_thresh** (v bajtech), jádro jednoduše začne zahazovat fragmentované pakety dokud počet neúplných paketů nedosáhne počtu zadaného pomocí **ipfrag_low_thresh**. (U jader 2.2 je výchozí hodnota obvykle 256K). To bude vypadat jako ztráta paketů a pokud je dosažena horní prahová hodnota, sníží se hodně výkon vašeho serveru.

Jedním ze způsobů sledování tohoto sledování pole IP: ReasmFails v souboru /proc/net/snmp; pokud se zvyšuje příliš rychle v průběhu silné souborové aktivity, mohl by nastat problém. Nebyly hlášeny žádné dobré alternativní hodnoty **ipfrag_high_thresh** a **ipfrag_low_thresh**; pokud máte dobrou zkušenost s konkrétní hodnotou, dejte prosím vědět autorům a vývojovému týmu.

Vypínání automatického nastavení NIC a rozbočovačů

Někdy se síťové karty špatně automaticky domlouvají s rozbočovači a přepínači a to může mít podivné následky. Kromě toho rozbočovače mohou ztrácet pakety, pokud mají různé porty pracující na různých rychlostech. Zkoušejte nastavení rychlosti sítě a duplexu.

Prostředky nevztahující se k NFS pro zvyšování výkonu serveru

Nabídka obecných vodítek pro nastavování dobře fungujícího souborového serveru je mimo rámec tohoto dokumentu, ale několik rad stojí za zmínku: Nejprve, RAID 5 vám poskytuje dobrou rychlost při čtení, ale malou rychlost zápisu; pokud je důležitá rychlost zápisu i nadbytečnost, zvažte RAID 1/0. Dále, použití žurnálového souborového systému drasticky sníží čas pro znovuspuštění v případě havárie systému; v době vytváření tohoto dokumentu byl ext3 (ftp://ftp.uk.linux.org/pub/linux/sct/fs/jfs/) jediným žurnálovacím souborovým systémem, který pracuje správně s NFS verze 3, ale nebojte se, brzo se to změní. Zejména to vypadá, že Reiserfs by měl pracovat s NFS verze 3 na jádrech 2.4, ačkoliv teď nepracuje na jádrech 2.2. Konečně, použití automatického připojování (jako např. autofs nebo amd) může zabránit zpomalení, když připojíte soubory na vašich počítačích křížem (ať už záměrně nebo omylem) a jeden z těchto počítačů spadne. Detaily najdete v Automount Mini-HOWTO.

Zabezpečení a NFS

Tento seznam tipů pro zabezpečení a vysvětlivky nezabezpečí vaše servery kompletně. *NIC* nezabezpečí vaše servery kompletně. Může vám to pomoci získat představu o problémech zabezpeče-

ní s NFS. Toto není obsažná příručka a nebude vždy odpovídat změnám. Pokud máte nějaké tipy nebo rady, které byste nám mohli předat, pošlete je prosím správci HOWTO.

Pokud jste v síti bez přístupu do okolního světa (ani přes modem) a důvěřujete všem interním počítačům a všem vašim uživatelům, pak tato sekce nebude pro vás. Domníváme se samozřejmě, že existuje relativně málo sítí v této situaci, takže bychom doporučovali důkladné pročtení této sekce každému, kdo nastavuje NFS.

Připojování nebo přístup k souborům NFS se děje ve dvou krocích. Prvním krokem je připojení. Přístup pro připojení je dosažen počítačem klienta, který se pokouší připojit k serveru. Zabezpečení pro něj je nastaveno pomocí souboru /etc/exports. Tento soubor obsahuje názvy nebo adresy IP počítačů, kterým je povolen přístup ke sdílené položce. Pokud adresa IP klienta odpovídá jedné položce v seznamu přístupů, pak mu bude povoleno připojení. Toto není velké zabezpečení. Pokud je někdo schopen převzít důvěryhodnou adresu, pak má přístup k vašim připojovacím bodům. Příklad z reálného světa tohoto druhu „ověření": Je to to samé, jako když se vám někdo představuje a vy věříte jeho tvrzení, že je ta osoba, protože nosí jmenovku, na které je „Ahoj, jmenuji se ...."

Druhým krokem je přístup k souborům. Toto je funkce řízení přístupu k normálnímu souborovému systému a nikoliv speciální funkce NFS. Jakmile je jednotka připojena, převezmou řízení přístupu oprávnění uživatelů a skupin.

Příklad: Bob je na serveru mapován na UserID 9999. Bob vytvoří soubor na serveru, ke kterému má přístup pouze uživatel (0600 oktalově). Klientovi je povoleno připojení jednotky, kde je soubor uložen. Klient Mary je mapován na UserID 9999. To znamená, že klient mary má přístup k Bobovu souboru, který je označen jako přístupný pouze jemu. Je horší, když někdo má root na počítači klienta a může **su - [username]** a stát se NĚJAKÝM uživatelem. NFS z toho nebude moudré.

To všechno není tak strašné. Existuje několik způsobů zjišťování, které můžete provést na serveru pro kompenzaci nebezpečí klientů. Krátce si je popíšeme.

Pokud si myslíte, že se vás míra zabezpečení netýká, pravděpodobně nemáte pravdu. V následujících řádcích si popíšeme zabezpečení portmapperu, zabezpečení serveru a klienta. Nakonec se budeme krátce zabývat správnými firewally pro váš server NFS.

Nakonec je důležité, aby všichni vaši démoni NFS a klientské programy byly aktuální. Pokud si myslíte, že je chyba zjištěna teprve nedávno, aby to byl pro vás problém, pak jste již pravděpodobně udělali kompromis.

Dobrým způsobem pro to, abyste šli s dobou, co se týká varování o zabezpečení, je zapsání se do e-mailových diskusních skupin bugtraq. Přečíst si o tom, jak se přihlásit a různé další informace, můžete na: http://www.securityfocus.com/forums/bugtraq/faq.html

Další hledání ohledně *NFS* na securityfocus.com vám ukáže všechny zprávy ohledně zabezpečení týkající se NFS.

Také byste měli pravidelně kontrolovat poradnu CERT. Podívejte se na webovou stránku CERT na adrese www.cert.org.

Portmapper

Portmapper udržuje seznam toho, které služby běží na kterých portech. Tento seznam je používán připojujícím se počítačem, aby mohl zjistit, na kterých portech má hledat konkrétní služby.

Portmapper už není tak špatný, jako býval před několika lety, ale stále znepokojuje mnoho správců systému. K portmapperu, jako NFS a NIS, by se opravdu nemělo dát připojit z venku (mimo

důvěryhodné místní sítě LAN). Pokud jej musíte zpřístupnit okolnímu světu, buďte opatrní a tyto systémy pečlivě sledujte.

Všechny distribuce systému Linux nebyly vytvořeny stejně. Některé distribuce, které vypadají jako aktuální, neobsahují bezpečný portmapper. Zda je váš portmapper dobrý nebo ne, zjistíte jednoduše spuštěním *strings(1)* a sledováním, zda čte relevantní soubory, /etc/hosts.deny a /etc/hosts.allow. Za předpokladu, že je váš portmapper /sbin/portmap, jej můžete zkontrolovat tímto příkazem:

```
strings /sbin/portmap | grep hosts.
```

Na zabezpečeném počítači se objeví něco jako toto:

```
/etc/hosts.allow
/etc/hosts.deny
@(#) hosts_ctl.c 1.4 94/12/28 17:42:27
@(#) hosts_access.c 1.21 97/02/12 02:13:22
```

Nejprve upravíme /etc/hosts.deny. Měl by obsahovat řádek

```
portmap: ALL
```

který zamezí přístupu komukoliv. Až jej uzavřete, spusťte:

```
rpcinfo -p
```

jen pro kontrolu, že váš portmapper opravdu čte tento soubor a řídí se jím. Rpcinfo by nemělo mít žádný výstup nebo chybovou zprávu. Soubory /etc/hosts.allow a /etc/hosts.deny účinkují okamžitě po jejich uložení. Není potřeba restartovat žádný démon.

Uzavření portmapperu komukoliv je trochu drastické, takže jej znovu otevřeme úpravou /etc/hosts.allow. Nejprve ale potřebujeme zjistit, co do něj vložit. Měl by zde být seznam všech počítačů, které by měly mít přístup k vašemu portmapperu. Na obyčejném systému Linux existuje jen několik málo počítačů, které potřebují z nějakého důvodu přístup. Portmapper spravuje **nfsd**, **mountd**, **ypbind**/**ypserv**, **pcnfsd** a služby 'r' jako např. **ruptime** a **rusers**. Z těchto pouze **nfsd**, **mountd**, **ypbind**/**ypserv** a možná **pcnfsd** jsou nějak důležité. Všem počítačům, které potřebují přístup ke službám na vašem počítači, by to mělo být povoleno. Řekněme, že adresa IP vašeho počítače je *192.168.0.254* a že je v podsíti *192.168.0.0*, a že všechny počítače v podsíti by k němu měly mít přístup (to jsou termíny z Networking-Overview-HOWTO, podívejte se do něj a osvěžte si znalosti, pokud to potřebujete). Pak napíšeme:

```
portmap: 192.168.0.0/255.255.255.0
```

do /etc/hosts.allow. To je toté,ž jako byste dali síťovou adresu route a masku podsítě **ifconfig**. Pro zařízení eth0 na tomto počítači by **ifconfig** měl ukázat:

```
...
eth0   Link encap:Ethernet  HWaddr 00:60:8C:96:D5:56
    inet addr:192.168.0.254  Bcast:192.168.0.255 Mask:255.255.255.0
    UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
    RX packets:360315 errors:0 dropped:0 overruns:0
    TX packets:179274 errors:0 dropped:0 overruns:0
    Interrupt:10 Base address:0x320
...
```

a netstat -rn by měl ukázat:

```
Kernel routing table
Destination     Gateway         Genmask         Flags Metric Ref Use    Iface
...
192.168.0.0     0.0.0.0         255.255.255.0   U     0      0   174412 eth0
...
```

(Síťová adresa v prvním sloupci).

Soubory /etc/hosts.deny a /etc/hosts.allow jsou popsány v manuálových stránkách s příslušnými názvy.

> **DŮLEŽITÉ:** Nevkládejte nic jiného než číselné adresy IP na řádky portmap v těchto souborech. Vyhledávání názvů hostitelů může nepřímo způsobit činnost portmap, která spustí vyhledávání názvů hostitelů, což může nepřímo způsobit činnost portmap, která spustí...

Verze balíčku nfs-utils 0.2.0 a vyšší také používá soubory hosts.allow a hosts.deny, takže byste měli vložit položky pro **lockd**, **statd**, **mountd**, a **rquotad** také do těchto souborů.

Výše uvedené věci by měly učinit váš server uzavřenějším. Jediný zbývající problém (ano, opravdu!) je někdo narušující root (nebo spouštějící MS-DOS) na důvěryhodném počítači a používající tato oprávnění pro odesílání požadavků ze zabezpečeného portu jako jiný uživatel.

Zabezpečení serveru: nfsd a mountd

Na serveru můžeme určit, že nebudeme důvěřovat účtu root klienta. To můžeme provést pomocí volby **root_squash** v /etc/exports:

```
/home slave1(rw,root_squash)
```

To je ve skutečnosti implicitní nastavení. Mělo by být vždy zapnuto, pokud nemáte VELMI dobrý důvod pro jeho vypnutí. Pro jeho vypnutí použijte možnost **no_root_squash**.

Nyní, pokud se uživatel s *UID* 0 (tj. číslo ID uživatele root) na počítači klienta pokouší o přístup (čtení, zápis, odstranění) v souborovém systému, server dosadí *UID* účtu 'nobody' na serveru. To znamená, že uživatel root na počítači klienta nemá přístup nebo nemůže měnit soubory, k nimž má přístup pouze root na serveru. To je dobré a pravděpodobně byste měli použít **root_squash** na všech souborových systémech, které exportujete. „Ale uživatel root na počítači klienta může stále použít **su** k tomu, aby se stal nějakým jiným uživatelem a zpřístupnit a měnit tak soubory tohoto uživatele!" říkáte. Odpověď zní: Ano, takhle to jde, a musí to být v Unixu s NFS. Má to jeden důležitý důsledek: všechny důležité kódy a soubory by měly být vlastněny uživatelem root a ne bin nebo jiným účtem odlišným od root, jelikož jediný účet, ke kterému uživatel root klienta nemá přístup je účet root serveru. V manuálové stránce *exports(5)* je uvedeno více jiných možností squash, takže se můžete rozhodnout, že nebudete komukoliv z klientů důvěřovat.

Porty TCP 1-1024 jsou rezervovány pro použití uživatelem root (a proto jsou někdy nazývány „zabezpečenými porty"). Uživatel z jiného účtu než root se nemůže připojit na tyto porty. Přidání volby secure do položky /etc/exports si vynutí jeho spuštění na portu nižším než 1024, takže poťouchlý uživatel odlišný od uživatele root nemůže přijít a otevřít z legrace dialog NFS na nerezervovaném portu. Tato možnost je implicitně nastavena.

Zabezpečení klienta

Volba nosuid příkazu mount

U klienta se můžeme rozhodnout, že nechceme tolik důvěřovat serveru několika způsoby pomocí voleb příkazu mount. Například můžeme zakázat, aby suid programy pracovaly na souborovém systému NFS pomocí volby nosuid. Některé unixové programy, jako např. passwd, jsou nazývány „suid“ programy: Nastavují id osoby, která je spouští na id vlastníka souboru. Pokud je vlastníkem souboru root a je suid, pak se program provede jako root, aby mohly být provedeny operace (jako např. zápis do souboru hesel), které jsou povoleny pouze uživateli root. Použití volby nosuid je dobré a měli byste zvážit její použití u všech připojovaných disků NFS. To znamená, že uživatel root serveru nemůže provádět program suid-root na tomto souborovém systému, přihlásit se ke klientovi jako normální uživatel a pak použít program suid-root k tomu, aby se stal uživatelem root také u klienta. Dalo by se také zakázat spouštění souborů na připojeném souborovém systému společně s volbou **noexec**. Ale toto bude pravděpodobně méně praktické než nosuid jelikož souborový systém pravděpodobně obsahuje alespoň nějaké skripty nebo programy, které je potřeba spouštět.

Volba broken_suid příkazu mount

Některé starší programy (**xterm** je jedním z nich) se spoléhají na to, že uživatel root může zapisovat kamkoliv. To už neplatí na nových jádrech a připojených NFS. V důsledku toho programy, které provádí tento druh akce suid, mohou být použity pro změnu jejich zdánlivého uid na serverech NFS, které provádí mapování uid. Takže je **broken_suid** implicitně v jádře systému Linux vypnuto.

Celkově z toho plyne: Pokud používáte starou distribuci systému Linux, některé druhy programů suid nebo starší druhy Unixu, budete se *možná* muset připojovat z vašich klientů s volbou **broken_suid** u příkazu **mount**. Samozřejmě, většina posledních distribucí Unixů a Linuxů obsahuje **xterm** a podobné programy jako normální spustitelné programy bez nastavení bitu suid, které volají programy podle jejich setuid.

Výše uvedené možnosti zadáváte do sloupce options s **rsize** a **wsize** oddělenými čárkami.

Zabezpečení portmapperu, rpc.statd a rpc.lockd u klienta

V aktuální implementaci NFS (2.2.18+) je podporováno zamykání celého souboru. To znamená, že **rpc.statd** a **rpc.lockd** musí běžet u klienta, aby zamykání fungovalo správně. Tyto služby vyžadují, aby běžel portmapper. Takže většina problémů, se kterými se setkáte u NFS na serveru, může být také způsobena klientem. Podívejte se do sekce o portmapperu, kde najdete informace o jeho zabezpečení.

NFS a firewally (ipchains a netfilter)

IPchains (pod jádry 2.2.X) a netfilter (pod jádry 2.4.x) umožňují dobrou úroveň zabezpečení - místo spoléhání se na démona (nebo v tomto případě tcp-wrapper) pro zjištění toho, kdo se smí připojit a zda je pokus o připojení povolen nebo zamítnut na nižší úrovni. V tomto případě můžete zastavit spojení mnohem dříve a globálněji, což vás může chránit před všemi druhy útoků.

Popis toho, jak nastavit firewall v systému Linux přesahuje rámec tohoto dokumentu. Zájemci se mohou podívat do Firewall-HOWTO nebo IPCHAINS-HOWTO. Uživatelé jádra 2.4 a novějšího se mohou podívat na webovou stránku netfilter na adrese: http://netfilter.filewatcher.org. Pokud již znáte práci programů ipchains nebo netfilter, podá vám tato sekce několik tipů o tom, jak nastavit váš firewall, aby lépe pracoval s NFS.

Dobrým pravidlem, kterým byste se měli řídit při konfiguraci vašeho firewallu je odepření přístupu všem a povolení pouze někomu – toto vás pomůže ochránit před náhodným povolením něčeho více, než jste plánovali.

Porty, kterými byste se měli zabývat:

a. Portmapper je na 111 (tcp a udp)

b. nfsd je na 2049 a může být pod TCP i UDP. Ačkoliv NFS přes TCP je aktuálně na straně serveru experimentálně a obvykle se setkáte na straně serveru s UDP, použití TCP je na straně klienta trochu stabilnější.

c. mountd, **lockd**, a **statd** se mohou lišit (to je to, proč potřebujeme portmapper) – to způsobuje problémy. To můžete řešit dvěma základními způsoby:

   i. Můžete více méně zakázat všechny porty, ale explicitně povolit většinu portů pro konkrétní adresy IP.

   ii. Novější verze těchto nástrojů mají volbu „-p“, která vám umožňuje přiřadit je ke konkrétnímu portu. Podívejte se do manuálových stránek, abyste se ujistili, zda to podporuje vaše verze. Pak můžete povolit přístup k portům, které jste uvedli pro vaše klienty NFS a oddělit všechny ostatní porty, dokonce i pro vaši místní síť.

Pomocí IPCHAINS by se jednoduchý firewall pro první způsob podobal tomuto:

```
ipchains -A input -f -j ACCEPT
ipchains -A input -s trusted.net.here/trusted.netmask
    -d host.ip/255.255.255.255 -j ACCEPT
ipchains -A input -s 0/0 -d 0/0 -p 6 -j DENY -y -l
ipchains -A input -s 0/0 -d 0/0 -p 17 -j DENY -l
```

Ekvivalentní sada příkazů pro netfilter (nástroj pro firewally ve verzi 2.4) je:

```
iptables -A INPUT -f -j ACCEPT
iptables -A INPUT -s trusted.net.here/trusted.netmask -d \
    host.ip/255.255.255.255 -j ACCEPT
iptables -A INPUT -s 0/0 -d 0/0 -p 6 -j DENY --syn --log-level 5
iptables -A INPUT -s 0/0 -d 0/0 -p 17 -j DENY --log-level 5
```

První řádek říká, aby byly akceptovány všechny fragmenty paketů (s výjimkou prvního paketu souboru, který bude považován za normální paket). Teoreticky nebude žádný paket předán, dokud není složen a nebude složen, dokud nebude předán první fragment paketu. Samozřejmě, existují útoky, které mohou být generovány pomocí přetížení počítače fragmenty paketů. Ale NFS nebude pracovat správně dokud nepovolíte procházení paketů. Podívejte se na detaily v sekci 7.

Další tři řádky říkají, že důvěřujete vaší místní síti a zakazujete a logujete všechno ostatní. Není to moc dobré a vyplatí se více specifických pravidel, ale více specifických pravidel je mimo rámec této diskuse.

Několik tipů, pokud byste rádi byli trochu paranoidní nebo přísní při stanovování vašich pravidel. Pokud budete chtít nulovat pravidla vašeho firewallu při každém přesunu **statd**, **rquotad**, **mountd** nebo **lockd** (což je možné), budete se chtít ujistit, že jste povolili fragmenty pro váš server NFS z vašeho klienta (nebo klientů) NFS. Pokud to neuděláte, obdržíte některé velmi zajímavé zprávy od jádra o odmítnutí paketů. Zprávy budou říkat, že paket z portu 65535 u klienta na 65535 na serveru byl odmítnut. Toto vyřeší povolení fragmentů.

Shrnutí

Pokud používáte hosts.allow, hosts.deny, root_squash, **nosuid** a funkce pro povolování portů v softwaru portmapper/NFS, vyhnete se mnoha současným chybám v NFS a můžete se cítit alespoň téměř bezpečně. Ale přece jen: pokud má vetřelec přístup do vaší sítě, mohou se objevit neznámé příkazy ve vašem .forward při čtení vaší pošty, když je exportován /home nebo /var/mail přes NFS. Ze stejného důvodu byste nikdy neměli zpřístupnit váš privátní klíč PGP přes NFS. Nebo byste alespoň měli znát riziko. A nyní o tom něco víte.

NFS a portmapper tvoří komplexní podsystém a proto není zcela nepravděpodobné, že se objeví nové chyby, v základní konstrukci nebo použité implementaci. Mohou zde být známé díry, kterých může někdo zneužít. Ale to je život.

Odstraňování problémů

Toto je zamýšleno jako příručka „krok za krokem“, ve které byste nalezli to co byste měli dělat když něco v NFS nefunguje. Obvykle se problémy objevují na straně klienta, takže začneme s diagnostikou odsud.

Nejsou vidět soubory na připojeném souborovém systému

Nejprve zkontrolujte, zda je souborový systém momentálně připojen. To můžete provést několika způsoby. Nejspolehlivější způsob je podívat se na soubor /proc/mounts, který obsahuje všechny připojené souborové systémy a obsahuje detaily o nich. Pokud to nepracuje (například pokud nemáte zakompilovaný souborový systém /proc do vašeho jádra), můžete napsat ‘mount -f’, ačkoliv obdržíte méně informací.

Pokud to vypadá, že je souborový systém připojen, pak můžete mít na něm připojen jiný souborový systém (v tomto případě byste jej měli odpojit a oba svazky znovu připojit), nebo můžete mít na serveru exportovaný souborový systém před tím, než jste jej připojili sem, v tom případě NFS exportuje nižší přípojný bod (pokud ano, pak je potřeba restartovat NFS na serveru).

Pokud není souborový systém připojen, pak se jej pokuste připojit. Pokud to nefunguje, podívejte se na symptom 3.

Požadavky na soubory se zastaví nebo vyprší čas pro čekání na přístup k souboru

Toto obvykle znamená, že klient není schopen komunikovat se serverem. Podívejte se na *symptom 3* písmeno b.

Není možno připojit souborový systém

Existují dva společné problémy, které mount produkuje, když nemůže připojit svazek. Jsou to:

a. failed, reason given by server: Permission denied

   To znamená, že server nepozná, že máte přístup ke svazku.

   i. Zkontrolujte váš soubor /etc/exports a ujistěte se, že je svazek exportován a že váš klient má správný druh přístupu k němu. Například, pokud má klient přístup pouze pro čtení, pak musíte připojit svazek s volbou ro místo s volbou rw.

ii. Ujistěte se, že jste NFS oznámili změny, které jste provedli v /etc/exports od spuštění nfsd pomocí spuštění příkazu exportfs. Nezapomeňte napsat **exportfs -ra**, abyste si byli opravdu jisti, že je možno exporty znovu přečíst.

iii. Zkontrolujte soubor /proc/fs/nfs/exports a ujistěte se, že jsou svazek a klient uvedeni správně (také se můžete podívat do souboru /var/lib/nfs/xtab na nezkrácený seznam nastavení aktivních voleb exportu). Pokud ne, pak jste nepřeexportovali správně. Pokud jsou uvedeni, ujistěte se, že server rozpoznává vašeho klienta na počítači tak, jak si myslíte, že by měl. Například můžete mít starý záznam pro klienta v /etc/hosts, který shazuje server, nebo nemusíte mít uvedenu celou adresu klienta a může se překládat na počítač v jiné doméně. Zkuste ping na klienta ze serveru, a zkuste ping na server z klienta. Pokud to nepracuje, nebo pokud dojde ke ztrátě paketů, může jít o síťové problémy na nižší úrovni.

b. RPC: Program Not Registered (nebo jiná chyba „RPC“):

To znamená, že klient nedetekuje NFS běžící na serveru. To by mohlo nastat z několika důvodů.

i. Nejprve zkontrolujte, že NFS na serveru aktuálně běží napsáním **rpcinfo -p** na serveru. Měli byste obdržet něco jako toto:

```
program vers proto   port
100000    2   tcp    111  portmapper
100000    2   udp    111  portmapper
100011    1   udp    749  rquotad
100011    2   udp    749  rquotad
100005    1   udp    759  mountd
100005    1   tcp    761  mountd
100005    2   udp    764  mountd
100005    2   tcp    766  mountd
100005    3   udp    769  mountd
100005    3   tcp    771  mountd
100003    2   udp   2049  nfs
100003    3   udp   2049  nfs
300019    1   tcp    830  amd
300019    1   udp    831  amd
100024    1   udp    944  status
100024    1   tcp    946  status
100021    1   udp   1042  nlockmgr
100021    3   udp   1042  nlockmgr
100021    4   udp   1042  nlockmgr
100021    1   tcp   1629  nlockmgr
100021    3   tcp   1629  nlockmgr
100021    4   tcp   1629  nlockmgr
```

ii. Toto říká, že máme NFS verzí 2 a 3, rpc.statd verze 1, network lock manager (název služby pro rpc.lockd) verzí 1, 3 a 4. Existují také různé seznamy služeb v závislosti na tom, zda NFS pracuje přes TCP nebo UDP. UDP je obvykle (ale ne vždy) implicitní, pokud není explicitně požadováno TCP.

iii. Pokud nevidíte alespoň portmapper, nfs a mountd, pak bude potřeba restartovat NFS. Pokud nemůžete úspěšně restartovat, pokračujte symptomem 9.

iv. Nyní zkontrolujte, že jej vidíte od klienta. Na straně klienta napište **rpcinfo -p [server]** kde **[server]** je název DNS nebo adresa IP vašeho serveru.

Pokud obdržíte seznam, pak se ujistěte, že je podporován druh připojení, které se pokoušíte provést. Například, pokud se pokoušíte připojit NFS verze 3, ujistěte se, že je verze 3 uvedena v seznamu; pokud se pokoušíte připojit k NFS přes TCP, ujistěte se, že je registrováno (někteří klienti z jiných operačních systémů než je Linux implicitně používají TCP). Podívejte se na man rpcinfo na detailnější informace o tom, jak číst výstup. Pokud druh připojení, o které se pokoušíte není v seznamu uveden, zkuste jiný druh připojení.

Pokud obdržíte chybu No Remote Programs Registered, pak budete muset zkontrolovat vaše soubory /etc/hosts.allow a /etc/hosts.deny na serveru a ujistit se, že má váš klient aktuálně povolen přístup. Znovu, pokud se položky objevují správně, zkontrolujte /etc/hosts (nebo váš server DNS) a ujistěte se, že je počítač v seznamu uveden správně a ujistěte se, že funguje ping mezi serverem a klientem. Zkontrolujte také chybové protokoly v systému, zda v nich nejsou užitečné zprávy: Chyby ověřování ze špatných položek /etc/hosts.allow se budou obvykle objevovat v souboru /var/log/messages, ale mohou se objevit kdekoliv jinde v závislosti na tom, jak jsou protokoly vašeho systému nastaveny. Manuálové stránky pro syslog vám mohou pomoci vyřešit nastavení vašich protokolů. Konečně, některé starší operační systémy se mohou chovat špatně, když jsou směrování mezi dvěma počítači asymetrická. Zkuste napsat **tracepath [server]** z klienta a podívejte se, zda se ve výstupu neobjevuje slovo „asymmetric“. Pokud ano, může to způsobovat ztrátu paketů. Samozřejmě, asymetrická směrování nejsou obvykle problémem na posledních distribucích systému Linux.

Pokud obdržíte chybu Remote system error - No route to host, ale ping mezi serverem a klientem pracuje správně, pak jste obětí přehnaného firewallu. Zkontrolujte firewally, které mohou být nastaveny, buď na serveru nebo na směrovačích mezi klientem a serverem. Podívejte se na manuálové stránky **ipchains**, **netfilter** a **ipfwadm**, stejně jako do IPChains-HOWTO a Firewall-HOWTO.

Nemám oprávnění pro přístup k souborům na připojeném svazku

To by mohl být jeden ze dvou problémů.

Pokud je to problém s oprávněním pro zápis, zkontrolujte volby exportu na serveru kontrolou /proc/fs/nfs/exports a ujistěte se, že souborový systém není exportován pouze pro čtení. Pokud ano, budete muset provést znovu export pro čtení/zápis (nezapomeňte spustit **exportfs -ra** po úpravě /etc/exports). Také zkontrolujte /proc/mounts a ujistěte se, že je svazek připojen pro čtení i zápis (ačkoliv pokud je připojen pouze pro čtení, měli byste obdržet konkrétnější chybovou zprávu). Pokud ne, pak budete potřebovat provést připojení znovu s volbou rw.

Druhý problém má co do činění s mapováním uživatelských jmen a je různý v závislosti na tom, zda se pokoušíte to provést jako root nebo jako jiný uživatel.

Pokud nejste root, pak názvy hostitelů nemusí být synchronizovány u klienta a na serveru. Napište **id [user]** u klienta i na serveru a ujistěte se, že udávají stejné číslo *UID*. Pokud ne, pak máte problémy s NIS, NIS+, rsync nebo jakýmkoliv systémem, který používáte pro synchronizaci jmen uživatelů. Zkontrolujte názvy skupin, abyste se ujistili, že odpovídají. Také se ujistěte, že neexportujete s volbou **all_squash**. Pokud jména uživatelů odpovídají, pak má uživatel obecnější problém s oprávněními, který se nevztahuje k NFS.

Pokud jste root, pak jste pravděpodobně neexportovali s volbou **no_root_squash**; zkontrolujte /proc/fs/nfs/exports nebo /var/lib/nfs/xtab na serveru a ujistěte se, že je zde uvedena tato volba. Obvykle je možnost zápisu na server NFS jako root špatně, pokud pro to nemáte nějakou urgentní potřebu – což je to proč se tomu Linux NFS implicitně brání.

Pokud máte root squashing, chcete jej zachovat a pouze se pokoušíte nastavit uživatele root tak, aby měl stejná oprávnění pro soubor jaká by měl mít uživatel *nobody*, pak si pamatujte, že to je tím, že server určuje, na které UID je root mapován. Server implicitně používá *UID* a *GID* uživatele *nobody* v souboru /etc/passwd, ale to může být také potlačeno volbami **anonuid** a **anongid** v souboru /etc/exports. Ujistěte se, že souhlasí *UID* na které je mapován uživatel *nobody,* na klientu i serveru.

Když přenáším velké soubory, NFS zabere všechny cykly CPU na serveru a moc nepracuje

Toto je problém funkce fsync() v jádrech 2.2, který způsobuje kumulování všech požadavků na synchronizaci disku, což se projeví časem zápisu, který roste kvadraticky podle velikosti souboru. Pokud můžete, inovujte na jádro 2.4, což by mělo vyřešit tento problém. Také export s volbou **no_wdelay** způsobí, že program bude používat o_sync(), což se může projevit zvýšením rychlosti.

Neznámé chyby nebo zprávy v protokolu

a. Zprávy následujícího formátu:

```
Jan 7 09:15:29 server kernel: fh_verify:
    mail/guest permission failure, acc=4, error=13
Jan 7 09:23:51 server kernel: fh_verify:
    ekonomi/test permission failure, acc=4, error=13
```

Nastanou, když je proveden pokus o operaci NFS setattr na souboru, ke kterému nemáte právo zápisu. Zprávy jsou neškodné.

Následující zprávy se často objevují v protokolech:

```
kernel: nfs: server server.domain.name not responding, still trying
kernel: nfs: task 10754 can't get a request slot
kernel: nfs: server server.domain.name OK
```

b. Zpráva „can't get a request slot" znamená, že kód RPC na straně klienta detekoval hodně prodlev (pravděpodobně z důvodu přetížení sítě, pravděpodobně z důvodu přetíženého serveru), a snižuje počet souběžných nevyřízených požadavků v pokusu pro snížení zatížení. Příčinou těchto zpráv je malý výkon.

c. Po připojení se objevuje u klienta následující zpráva:

```
nfs warning: mount version older than kernel
```

Znamená to to, co říká: měli byste inovovat váš balíček s mount a/nebo am-utils (pokud je z nějakého důvodu inovace problém, můžete to vyřešit jen jejich překompilováním, aby byly novější funkce jádra rozpoznány při kompilaci).

d. Chyby v protokolu startup/shutdown pro lockd

Ve vašem protokolu spouštění můžete vidět zprávy následujícího druhu:

```
nfslock: rpc.lockd startup failed
```

Jsou neškodné. Starší verze rpc.lockd bylo potřeba spouštět ručně, ale novější verze jsou spouštěny automaticky pomocí knfsd. Mnoho z výchozích spouštěcích skriptů se stále pokouší spouštět lockd ručně, v případě, že je to nutné. Jestliže se chcete těchto zpráv zbavit, můžete upravit vaše spouštěcí skripty.

e. Následující zpráva se objevuje v protokolech:

```
kmem_create: forcing size word alignment - nfs_fh
```

Toto vyplývá z toho, že identifikátor souboru je 16bitový, místo násobku 32 bitů, což se jádru nelíbí. Je neškodná.

Skutečná oprávnění neodpovídají těm, která jsou v /etc/exports

/etc/exports je VELMI citlivý na bílé znaky, takže následující příkazy nejsou stejné:

```
/export/dir hostname(rw,no_root_squash)
/export/dir hostname (rw,no_root_squash)
```

První přidělí hostiteli přístup rw pro /export/dir bez squashingu oprávnění uživatele root. Druhý přidělí hostiteli oprávnění rw s root squashem a přidělí KOMUKOLIV jinému přístup pro čtení a zápis bez squashingu oprávnění uživatele root. Pěkné, že?

Nestabilní a nespolehlivé chování

Jednoduché příkazy jako **ls** pracují, ale něco, co přenáší velké množství informací, způsobuje zamknutí bodu připojení.

To by mohl být jeden ze dvou problémů:

i. Nastává to, pokud máte ipchains na serveru nebo u klienta a neumožnili jste průchod fragmentovaných paketů skrze řetěz. Povolte fragmenty od vzdáleného hostitele a bude to zase fungovat.

ii. Můžete používat větší rsize a wsize ve vašich volbách pro mount, než podporuje server. Zkuste snížit rsize a wsize na 1024 a podívejte se, zda problém zmizel. Pokud ano, pak je pomalu zvyšujte na přijatelnější hodnotu.

nfsd se nespustí

Zkontrolujte soubor /etc/exports a ujistěte se, že má uživatel root oprávnění pro čtení. Zkontrolujte binární soubory a ujistěte se, že jsou spustitelné. Ujistěte se, že vaše jádro bylo zkompilováno s podporou serveru NFS. Možná budete potřebovat přeinstalovat vaše binární soubory pokud žádný z těchto nápadů nepomůže.

Používání Linux NFS s jinými operačními systémy

Každý operační systém, včetně Linuxu, má triky a odchylky v chování jeho implementace NFS – někdy z toho důvodu, že protokoly jsou nejasné, někdy z důvodu bezpečnostních děr. Linux bude pracovat správně se všemi implementacemi NFS hlavních dodavatelů, alespoň pokud víme. Samozřejmě, mohou zde být dodatečné kroky týkající se zajištění toho, aby dva operační systémy spolu komunikovaly čistě. Tato sekce tyto kroky popisuje.

Obvykle je velmi nešťastné se pokoušet používat počítač se systémem Linux s jádrem před 2.2.18 jako server NFS pro klienty z jiných operačních systémů než Linuxu. Implementace se staršími jádry mohou pracovat dobře jako klienti; samozřejmě, pokud používáte jedno z těchto jader a objevují se chyby, první radou, kterou vám dáme, je inovovat vaše jádro a zkontrolovat, zda problémy zmizely. Implementace NFS uživatelského prostoru také nepracují správně s klienty z jiných operačních systémů, než je Linux.

Zde je seznam známých problémů při použití systému Linux společně s hlavními operačními systémy.

AIX

Klienti se systémem Linux a servery se systémem AIX

Formát pro soubor /etc/exports pro náš příklad v sekci 3 je:

```
/usr   slave1.foo.com:slave2.foo.com,access=slave1.foo.com:slave2.foo.com
/home  slave1.foo.com:slave2.foo.com,rw=slave1.foo.com:slave2.foo.com
```

Klienti se systémem AIX a servery se systémem Linux

AIX používá soubor /etc/filesystems místo /etc/fstab. Vzorová položka, vypadá podobně jako:

```
/mnt/home:
    dev             = "/home"
    vfs             = nfs
    nodename        = master.foo.com
    mount           = true
    options         = bg,hard,intr,rsize=1024,wsize=1024,vers=2,proto=udp
    account         = false
```

i. Verze 4.3.2 AIX vyžaduje, aby byly souborové systémy exportovány s volbou insecure, což způsobí, že NFS naslouchá požadavkům přicházejícím z nezabezpečených portů (tj., portů nad 1024, ze kterých se mohou připojit i jiní uživatelé než root). Starší verze AIX to nevyžadovaly.

ii. Klienti AIX se budou implicitně připojovat k NFS verze 3 přes TCP. Pokud to váš server Linux nepodporuje, pak možná budete potřebovat zadat vers=2 nebo proto=udp ve volbách vašeho mount.

iii. Použití síťových masek v /etc/exports, se zdá, někdy způsobuje ztrátu připojení klientů, když je jiný klient resetován. To by se mělo dát opravit explicitním uvedením hostitelů.

iv. Podle všeho automount je v AIX 4.3.2 docela poškozený.

BSD

Servery se systémy BSD a klienti se systémy Linux

Jádra BSD mají sklony pracovat lépe s většími velikostmi bloků.

Servery se systémy Linux a klienti se systémy BSD

Některé verze BSD mohou vytvářet požadavky na server z nezabezpečených portů, v tom případě budete potřebovat exportovat vaše svazky s volbou insecure. Podívejte se na detaily na manuálové stránce pro *exports(5)*.

Compaq Tru64 Unix

Servery se systémy Tru64 Unix a linuxové klienti

Obvykle servery Tru64 Unix pracují s klienty se systémy Linux bez problémů. Formát souboru /etc/exports pro náš příklad je:

```
/usr          slave1.foo.com:slave2.foo.com \
```

```
    -access=slave1.foo.com:slave2.foo.com \
/home        slave1.foo.com:slave2.foo.com \
    -rw=slave1.foo.com:slave2.foo.com \
    -root=slave1.foo.com:slave2.foo.com
```

Tru64 kontroluje soubor /etc/exports při každém požadavku na připojení, takže není potřeba spouštět příkaz **exportfs**; ve skutečnosti na mnoha verzích systému Tru64 Unix tento příkaz neexistuje.

Servery se systémy Linux a klienti se systémy Tru64 Unix

Existují dva problémy, na které je zde potřeba dát pozor. Zaprvé, Tru64 Unix se připojuje implicitně přes NFS verze 3. Setkáte se s chybami připojení, pokud váš server Linux nepodporuje NFS verze 3. Za druhé, v Tru64 Unix 4.x jsou požadavky NFS na uzamčení vytvářeny démonem. Proto budete potřebovat zadat volbu **insecure_locks** pro všechny svazky, které exportujete pro klienta Tru64 Unix 4.x; na detaily ohledně **exports** se podívejte do manuálových stránek.

HP-UX

Servery se systémem HP-UX a klienti se systémem Linux

Příklad položky /etc/exports na HP-UX vypadá podobně jako toto:

```
/usr -ro,access=slave1.foo.com:slave2.foo.com
/home -rw=slave1.foo.com:slave2.fo.com:root=slave1.foo.com:slave2.foo.com
```

(volba **root** je uvedena v poslední položce pouze pro informační účely; její použití se nedoporučuje dokud to není nezbytné).

Servery se systémem Linux a klienti se systémy HP-UX

Klienti HP-UX bez disků budou vyžadovat alespoň jádro verze 2.2.19 (nebo záplatované 2.2.18) pro správný export souborů zařízení.

IRIX

Servery se systémy IRIX a klienti se systémy Linux

Příklad /etc/exports položky v systému IRIX vypadá podobně jako toto:

```
/usr -ro,access=slave1.foo.com:slave2.foo.com
/home -rw=slave1.foo.com:slave2.fo.com:root=slave1.foo.com:slave2.foo.com
```

(Volba **root** je uvedena v poslední položce pouze pro informační účely; její použití se nedoporučuje dokud to není nezbytné).

Byly zde hlášeny údajné problémy při použití volby nohide pro exporty do systémů Linux 2.2. Tento problém je opraven v jádře 2.4. Abyste jej obešli, můžete exportovat a připojovat nižší systémy odděleně.

Klienti se systémy IRIX a servery se systémy Linux

Nejsou zde žádné známé problémy se spoluprácí.

Solaris

Servery se systémy Solaris

Solaris má na straně serveru trochu odlišný formát od jiných operačních systémů. Namísto /etc/exports je konfiguračním souborem /etc/dfs/dfstab. Položky jsou ve tvaru příkazu „share“, kde syntaxe pro příklad ze sekce 3 by vypadala takto:

```
share -o rw=slave1,slave2 -d "Master Usr" /usr
```

a namísto spouštění **exportfs** po úpravách musíte spustit **shareall**.

Servery Solaris jsou citlivé zejména na velikost paketů. Pokud používáte klienta se systémem Linux a server se systémem Solaris, zajistěte nastavení **rsize** a **wsize** na 32768 v době připojení.

Konečně, existuje zde problém s root squashingem na systému Solaris: root je mapován na uživatele *noone*, což není totéž jako uživatel *nobody*. Pokud máte problémy s oprávněními k souborům jako root na počítači klienta, ujistěte se, že jste zkontrolovali, že mapování funguje jak jste očekávali.

Klienti se systémy Solaris

Klienti se systémy Solaris budou pravidelně produkovat následující zprávu:

```
svc: unknown program 100227 (me 100003)
```

To nastane z důvodu, že klienti Solaris, když se připojují, se pokouší získat informaci ACL - tu Linux obvykle nemá. Zprávy mohou být bezpečně ignorovány.

Existují dva známé problémy s bezdiskovými klienty Solaris: Zaprvé, je potřeba jádro verze alespoň 2.2.19, aby se /dev/null exportovalo správně. Zadruhé, velikost paketu může být potřeba nastavit extrémně malou (tj. 1024) na bezdiskových klientech sparc, protože klienti neví, jak sestavovat pakety v opačném pořadí. To může být provedeno v /etc/bootparams na straně klientů.

SunOS

SunOS má pouze NFS verze 2 přes UDP.

Servery se systémy SunOS

Na straně serveru SunOS používá tradiční formát pro soubor /etc/exports. Příklad by vypadal takto:

```
/usr    -access=slave1.foo.com,slave2.foo.com
/home   -rw=slave1.foo.com,slave2.foo.com, root=slave1.foo.com,slave2.foo.com
```

8.7.2. Klienti se systémy SunOS

Pamatujte si, že SunOS provádí všechny požadavky NFS na uzamknutí jako démon, a proto budete potřebovat přidat volbu **insecure_locks** k svazkům, které exportujete pro počítače se systémem SunOS. Podívejte se na detaily ohledně **exports** do manuálových stránek.

KAPITOLA 8

Mike G., mikkey@dynamo.com.ar

Úvod do programování v BASH

Tento text vám má pomoci začít programovat jednoduché a složitější skripty příkazového interpretu. Není určen jako dokument pro pokročilé. Nejsem ani expert ani zkušený programátor v příkazovém interpretu. Rozhodl jsem se tento text napsat, protože mi to pomůže naučit se spoustu věcí a může to být užitečné i ostatním. Je vítána jakákoliv odezva, zejména ve formě doplňků.

Úvod

Kde získat nejnovější verzi

http://www.linuxdoc.org/HOWTO/Bash-Prog-Intro-HOWTO.html

Předpoklady

Vhodná je obeznámenost s příkazovou řádkou Linuxu a se základními programátorskými dovednostmi. I když tento text nepředstavuje úvod do programování, objasňuje (nebo se alespoň snaží) řadu základních postupů.

Použití dokumentu

Tento dokument vám bude užitečný v následujících případech:

- Umíte programovat a chcete začít psát nějaké skripty příkazového interpretu.
- Máte jakési ponětí o programování v příkazovém interpretu a potřebujete nějakou referenční příručku.
- Chcete se podívat na nějaké skripty a vysvětlení, abyste mohli začít psát vlastní.
- Přecházíte nebo jste přešli z prostředí DOS/Windows a chcete psát „dávkové soubory".
- Jste úplný blázen a čtete všechna HOWTO.

Velmi jednoduché skripty

Tento dokument by vám měl poskytnout základní představu o programování příkazového interpretu na základě různých příkladů.

V této části najdete několik krátkých skriptů, které by vám měly pomoci pochopit některé techniky.

Klasický skript „Hello World“

```
#!/bin/bash
echo Hello World
```

Tento skript má pouhé dva řádky. První řádek říká systému, jaký program má ke spuštění souboru použít.

Druhý řádek definuje jedinou akci prováděnou skriptem, totiž vypsání textu „Hello World“ na terminál.

Pokud uvidíte hlášení typu *./hello.sh: Command not found*, pak je zřejmě chyba v prvním řádku `#!/bin/bash`. Zkuste příkaz `whereis bash` nebo se podívejte do části *Jak najít bash* abyste zjistili, jak má řádek správně vypadat.

Velmi jednoduchý zálohovací skript

```
#!/bin/bash
tar -cZf /var/my-backup.tgz /home/me/
```

V tomto skriptu nevypisujeme nic na terminál, místo toho vytváříme tarový archiv obsahující domovský adresář uživatele. Tento skript NENÍ určen k běžnému použití, později si ukážeme vhodnější zálohovací skript.

Vše o přesměrování

Teorie a stručná reference

Existují tři souborové deskriptory, *stdin*, *stdout* a *stderr* (kde *std* znamená *standard*).

V zásadě můžete:

- přesměrovat *stdout* do souboru
- přesměrovat *stderr* do souboru
- přesměrovat *stdout* na *stderr*
- přesměrovat *stderr* na *stdout*
- přesměrovat *stderr* a *stdout* do souboru
- přesměrovat *stderr* a *stdout* na *stdout*
- přesměrovat *stderr* a *stdout* na *stderr*

Symbol *1* „reprezentuje“ *stdout*, *2* pak *stderr*.

Malá poznámka k této problematice: příkazem `less` můžete prohlížet jak *stdout* (který zůstává v bufferu), tak i *stderr*, který se tiskne na obrazovce, ale vymaže se, když „procházíte“ bufferem.

Příklad: stdout do souboru

Tímto způsobem se výstup programu zapíše do souboru.

```
ls -l > ls-l.txt
```

V tomto příkladu se vytvoří souboru *ls-l.txt*, který bude obsahovat to, co byste viděli na obrazovce po zadání a spuštění příkazu `ls -l`.

Příklad: stderr do souboru

Tímto způsobem se chybový výstup programu zapíše do souboru.

```
grep da * 2> grep-errors.txt
```

Vytvoří se soubor *grep-errors.txt*, který bude obsahovat chybový výstup příkazu `grep da *`.

Příklad: stdout na stderr

Tímto způsobem se bude výstup programu vypisovat do stejného souborového deskriptoru jako standardní chybový výstup.

```
grep da * 1>&2
```

Standardní výstup příkazu se posílá na standardní chybový výstup, což můžete ověřit různými způsoby.

Příklad: stderr na stdout

Tímto způsobem se bude chybový výstup programu vypisovat do stejného souborového deskriptoru jako standardní výstup.

```
grep * 2>&1
```

Standardní chybový výstup programu se nyní posílá na *stdout*, takže když si výstup prohlédnete programem `less`, zjistíte, že řádky, které normálně „mizí“ (protože jsou zapisovány na *stderr*) tentokrát zůstávají (protože se vypisují na *stdout*).

Příklad: stderr a stdout do souboru

Tímto způsobem se celý výstup programu přesměruje do souboru. Je to užitečné například u programů spouštěných démonem *cron*, pokud chcete, aby program proběhl v úplné tichosti.

```
rm -f $(find / -name core) &> /dev/null
```

Tento příkaz (zamýšlený ke spouštění démonem *cron*) vymaže všechny soubory *core* ve všech adresářích. Pokud úplně potlačíte výstup nějakého programu, měli byste si být velmi přesně jisti tím, co program dělá.

Roury

V této části se jednoduše a prakticky vysvětluje, jak používat roury a k čemu se dají použít.

Co to je a k čemu je to dobré

Roury umožňují velice jednoduše předat výstup jednoho programu jako vstup dalšímu programu.

Příklad: jednoduchá roura s programem sed

Takto vypadá jednoduchý příklad použití roury:

```
ls -l | sed -e "s/[aeio]/u/g"
```

Stane se toto: spustí se první příkaz (`ls -l`) a jeho výstup se místo vypsání předá programu `sed`, který pak vypíše to, co s výpisem provede.

Příklad: alternativa k ls-l *.txt

Následující příklad je složitější alternativou k příkazu `ls -l *.txt`, ovšem my jej tady uvádíme jako příklad použití roury, ne jako návod pro vypisování souborů:

```
ls -l | grep "\.txt$"
```

Výpis příkazu `ls -l` se předává programu `grep`, který pak vypíše všechny řádky odpovídající regulárnímu výrazu `"\.txt$"`.

Proměnné

Stejně jako v jiných programovacích jazycích můžete i ve skriptech používat proměnné. Proměnná může obsahovat číslo, znak nebo řetězec znaků.

Proměnné nemusíte deklarovat, automaticky se vytvoří, když jim přiřadíte hodnotu.

Příklad: Hello World s proměnnými

```
#!/bin/bash
STR="Hello World!"
echo $STR
```

Ve druhém řádku se vytvoří proměnná pojmenovaná STR a bude mít hodnotu „Hello World!“. Hodnota proměnné se dá přečíst tak, že se před název proměnné uvede symbol $. Všimněte si (a vyzkoušejte si), že pokud nepoužijete symbol $, bude výstup programu jiný a pravděpodobně ne takový, jaký jste chtěli.

Příklad: Velmi jednoduchý zálohovací skript *(vylepšený)*

```
#!/bin/bash
OF=/var/my-backup-$(date +%Y%m%d).tgz
tar -cZf $OF /home/me/
```

Tento skript obsahuje několik novinek. V první řadě si musíme vyjasnit vytvoření a inicializaci proměnné na druhém řádku. Všimněte si výrazu *$(date +%Y%m%d)*. Pokud jej spustíte ve skriptu, zjistíte, že provede příkaz v závorkách a převezme jeho výstup.

Všimněte si také, že v našem skriptu bude název výstupních souborů každý den jiný díky přepínači *+%Y%m%d*. Můžete jej změnit a nastavit si jiný formát.

Některé další příklady:

```
echo ls
echo $(ls)
```

Lokální proměnné

Lokální proměnné se vytvářejí pomocí klíčového slova *local*.

```
#!/bin/bash
HELLO=Hello
function hello {
local HELLO=World
echo $HELLO
}
echo $HELLO
hello
echo $HELLO
```

Tento příklad by měl stačit k pochopení, jak pracovat s lokálními proměnnými.

Podmínky

Podmínky umožňují rozhodnout, zda provést nebo neprovést nějakou operaci. Rozhodnutí se provádí na základě vyhodnocení nějakého výrazu.

Suchá teorie

Podmínky mají mnoho formátů. Nejjednodušší typ je **if** *výraz* **then** *příkaz*, kdy se *příkaz* provede jenom tehdy, pokud bude hodnota *výrazu* pravdivá. Například *2<1* je výraz, který je nepravdivý, zatímco *2>1* je pravdivý výraz.

Jiná forma podmínky může být **if** *výraz* **then** *příkaz1* **else** *příkaz2*. V tomto případě se provede *příkaz1* tehdy, je-li *výraz* pravdivý, v opačném případě se provede *příkaz2*.

Další forma podmínky může být **if** *výraz1* **then** *příkaz1* **else if** *výraz2* **then** *příkaz2* **else** *příkaz3*. V tomto příkladu jsme doplnili část **else if ...**, která vede k tomu, že *příkaz* se provede pouze pokud je pravdivý *výraz2*. Zbylé chování je takové, jak asi očekáváte (viz předchozí příklady).

Poznámka k syntaxi:

Základní syntaxe konstrukce *if* vypadá v BASH takto:

```
if [výraz];
then
kód je-li výraz pravdivý
fi
```

Příklad: jednoduchá podmínka if ... then

```
#!/bin/bash
if [ "foo" = "foo" ]; then
    echo expression evaluated as true
fi
```

Kód, který se provede v případě, že je výraz v lomených závorkách pravdivý, se nachází mezi slovy `then` a `fi`, která označují konec podmíněně prováděného kódu.

Příklad: jednoduchá podmínka if ... then ... else

```
#!/bin/bash
if [ "foo" = "foo" ]; then
    echo expression evaluated as true
else
    echo expression evaluated as false
fi
```

Příklad: podmínka s proměnnými

```
#!/bin/bash
T1="foo"
T2="bar"
if [ "$T1" = "$T2" ]; then
    echo expression evaluated as true
else
    echo expression evaluated as false
fi
```

Smyčky for, while a until

V této části se seznámíme se smyčkami **for**, **while** a **until**.

Smyčka **for** se poněkud liší od jiných programovacích jazyků. V zásadě se používá k iteraci přes posloupnost „slov“ v řetězci.

Smyčka **while** opakovaně provádí část kódu, pokud je řídící výraz pravdivý, a zastaví se teprve až výraz bude nepravdivý (nebo pokud se v těle smyčky explicitně provede příkaz *break*).

Smyčka **until** je prakticky totožná se smyčkou **while** s tím rozdílem, že tělo se provádí tak dlouho, dokud je řídící výraz nepravdivý.

Pokud máte pocit, že smyčky **while** a **until** jsou velice podobné, máte pravdu.

Příklad smyčky for

```
#!/bin/bash
for i in $( ls ); do
    echo item: $i
done
```

Ve druhém řádku deklarujeme *i* jako proměnnou, která bude nabývat jednotlivé hodnoty obsažené v *$(ls)*.

Třetí řádek může být klidně delší, případně mohou před příkazem *done* následovat i další řádky.

Slovíčko *done* na čtvrtém řádku označuje konec těla smyčky, proměnná *i* nabude další hodnotu ze vstupní množiny a tělo se provede znovu.

Tento příklad nemá valného praktického použití, rozumnější příklad by mohl používat smyčku **for** k nalezení pouze souborů vyhovujících nějaké požadované vlastnosti.

Smyčka for jako v C

Fiesh navrhl doplnit následující typ smyčky. Jedná se o smyčku **for** podobnou té, jak ji známe z jazyka C, Perlu a podobně.

```
#!/bin/bash
for i in `seq 1 10`;
do
    echo $i
done
```

Příklad smyčky while

```
#!/bin/bash
COUNTER=0
while [ $COUNTER -lt 10 ]; do
    echo The counter is $COUNTER
    let COUNTER=COUNTER+1
done
```

Tento skript „emuluje“ chování struktury *for* známé z Pascalu, C a podobně.

Příklad smyčky until

```
#!/bin/bash
COUNTER=20
until [ $COUNTER -lt 10 ]; do
    echo COUNTER $COUNTER
    let COUNTER-=1
done
```

Funkce

Ve většině programovacích jazyků můžete používat funkce, které dovolují členit kód na logické celky nebo provozovat tajemné umění rekurze.

Deklarace funkce obnáší pouhé zapsání `function moje_funkce { můj_kód }`.

Volání funkce je stejné jako volání jakéhokoliv programu – stačí zapsat její název.

Příklad funkce

```
#!/bin/bash
function quit {
    exit
}
function hello {
    echo Hello!
}
hello
quit
echo foo
```

Řádky 2 až 4 obsahují funkci `quit`. Řádky 5 až 7 obsahují funkci `hello`. Pokud si nejste úplně jisti, co tento skript udělá, vyzkoušejte si jej.

Funkce nemusejí být deklarovány v žádném specifickém pořadí.

Když skript spustíte, zjistíte že nejprve zavolá funkci `hello`, pak funkci `quit` a na řádek 10 se už vůbec nedostane.

Příklad funkce s parametry

```
#!/bin/bash
function quit {
    exit
}
function e {
    echo $1
}
e Hello
e World
quit
echo foo
```

Skript je prakticky shodný s předchozím. Hlavní rozdíl je ve funkci `e`. Tato funkce vytiskne první předaný parametr. Parametry se uvnitř funkcí obsluhují stejně jako parametry předávané celému skriptu.

Uživatelské rozhraní

Jednoduchá nabídka příkazem select

```
#!/bin/bash
OPTIONS="Hello Quit"
select opt in $OPTIONS; do
    if [ "$opt" = "Quit" ]; then
      echo done
      exit
    elif [ "$opt" = "Hello" ]; then
      echo Hello World
    else
```

```
      clear
      echo bad option
    fi
done
```

Když si tento skript spustíte, zjistíte, že jde o sen každého programátora, co se týče textových nabídek. Jistě si také všimnete, že celá konstrukce je velmi podobná konstrukci **for** s tím rozdílem, že smyčka neprobíhá přes všechna „slova“ v $OPTIONS, ale dotáže se uživatele, kterou větev provést.

Použití příkazového řádku

```
#!/bin/bash
if [ -z "$1" ]; then
    echo usage: $0 directory
    exit
fi
SRCD=$1
TGTD="/var/backups/"
OF=home-$(date +%Y%m%d).tgz
tar -cZf $TGTD$OF $SRCD
```

Mělo by být jasné, co tento skript dělá. Výraz v podmínce testuje, zda program obdržel parametr ($1) a pokud ne, ukončí jej a zobrazí malou nápovědu. Zbytek skriptu už by měl být v této chvíli jasný.

Různé

Při řadě různých příležitostí potřebujete uživatele požádat o zadání nějakého vstupu a existuje několik způsobů, jak to udělat. Toto je jeden z nich:

```
#!/bin/bash
echo Please, enter your name
read NAME
echo "Hi $NAME!"
```

Další možnost je získat příkazem `READ` více vstupních hodnot. Demonstruje to následující příklad:

```
#!/bin/bash
echo Please, enter your firstname and lastname
read FN LN
echo "Hi! $LN, $FN !"
```

Aritmetické operace

V příkazovém řádku nebo v shellu zkuste toto:

```
echo 1 + 1
```

Pokud jste očekávali, že uvidíte `2`, budete zklamáni. Co když ale budete potřebovat, aby BASH vyhodnotil nějaký aritmetický výraz? Řešení vypadá takto:

```
echo $((1+1))
```

Získáme „logičtější“ výsledek. Tímto způsobem se vyhodnocují aritmetické výrazy. Alternativou je zápis

```
echo $[1+1]
```

Pokud chcete pracovat se zlomky nebo potřebujete složitější matematické operace, můžete počítat pomocí programu `bc`.

Pokud byste v příkazovém řádku zadali `echo $[3/4]`, výsledek by byl `0`, protože BASH pracuje pouze s celočíselnou aritmetikou. Pokud zadáte `echo 3/4|bc -l`, pak pravděpodobně dostanete `0.75`.

Jak najít bash

Ze zprávy od Mikeho (viz *Poděkování*).

Ve všech skriptech se uvádí `#!/bin/bash`... možná by byl užitečný příklad jak zjistit, kde je *bash* uložen.

Nejlepší metoda je `locate bash`, program *locate* však není na všech systémech.

Obvykle bude z kořenového adresáře fungovat `find ./ -name bash`.

Doporučená místa k prohlédnutí:

ls -l /bin/bash

ls -l /sbin/bash

ls -l /usr/local/bin/bash

ls -l /usr/bin/bash

ls -l /usr/sbin/bash

ls -l /usr/local/sbin/bash

(Nedokážu si představit žádný jiný rozumný adresář, na různých systémech byl *bash* vždy v jednom z nich.)

Můžete také zkusit `which bash`.

Získání návratové hodnoty programu

Návratová hodnota spuštěného programu se ukládá ve speciální proměnné `$?`.

Následující skript demonstruje, jak získat návratovou hodnotu. Předpokládáme, že adresář *dada* neexistuje. (Příklad rovněž navrhl Mike.)

```
#!/bin/bash
cd /dada &> /dev/null
echo rv: $?
cd $(pwd) &> /dev/null
echo rv: $?
```

Zachycení výstupu příkazu

Následující skript vypíše všechny tabulky ve všech databázích (za předpokladu, že máte nainstalováno MySQL). Zkuste změnit příkaz *mysql* tak, aby používal platné uživatelské jméno a heslo.

```
#!/bin/bash
DBS=`mysql -uroot -e"show databases"`
for b in $DBS ;
do
    mysql -uroot -e"show tables from $b"
done
```

Více zdrojových souborů

Skript může být rozdělen do více souborů...

Tabulky

Operátory porovnávání řetězců

s1=s2	s1 se shoduje s s2
s1!=s2	s1 se neshoduje s s2
s1<s2	...
s1>s2	...
-n s1	s1 není prázdný (obsahuje jeden nebo více znaků)
-z s1	s1 je prázdný

Příklady porovnávání řetězců

Porovnání dvou řetězců:

```
#!/bin/bash
S1='string'
S2='String'
if [ $S1=$S2 ];
then
    echo "S1('$S1') is not equal to S2('$S2')"
fi
if [ $S1=$S1 ];
then
    echo "S1('$S1') is equal to S1('$S1')"
fi
```

Doplňuji zde poznámku, kterou poslal Andreas Beck a která se vztahuje k zápisu *if [$S1 = $S2]*. Není to příliš vhodné, protože pokud je *$S1* nebo *$S2* prázdný, dojde k chybě. Rozumnější je zápis *x$1=x$2* nebo *"$1"="$2"*.

Aritmetické operátory

+
-
*
/
% (zbytek po dělení)

Aritmetické relační operátory

-lt	<
-gt	>
-le	<=
-ge	>=
-eq	==
-ne	!=

Užitečné příkazy

Tuto část přepsal Kees (viz *Poděkování*).

Některé z následujících příkazů obsahují prakticky úplné vlastní programovací jazyky. Uvedeme si zde pouze základní popis vybraných příkazů. Podrobnější informace získáte na příslušných manuálových stránkách.

sed (dávkový editor)

Sed je neinteraktivní editor. Namísto modifikace souboru prostřednictvím pohybu kurzoru po obrazovce zadáte programu **sed** název modifikovaného souboru a skript s editačními příkazy. Sed můžete chápat také jako filtr. Podívejme se na některé příklady:

```
$sed 's/to_be_replaced/replaced/g' /tmp/dummy
```

Tento příkaz nahradí řetězec *to_be_replaced* řetězcem *replaced*, přičemž vstup čte ze souboru */tmp/dummy*. Výsledek bude poslán na standardní výstup (typicky tedy na konzolu), pomocí přesměrování symbolem > však můžete výstup zapsat do dalšího souboru.

```
$sed 12,18d /tmp/dummy
```

Dojde k vypsání všech řádků kromě řádků 12 až 18. Původní soubor přitom nebude změněn.

awk
(manipulace s datovými soubory, čtení a zpracování textu)

Existuje řada implementací jazyka AWK (nejznámější jsou GNU *gawk* a „new awk" *mawk*). Princip je jednoduchý: AWK hledá zadaný vzor a pro každou shodu se vzorem provede určitou akci.

Vytvořili jsme si soubor */tmp/dummy*, který obsahuje následující řádky:

test123

test

tteesstt

```
$awk '/test/ {print}' /tmp/dummy
vypíše
test123
test
```

Vyhledávaným vzorem je *test* a operace, která se má provést s daným řetězcem je vytisknout jej.

```
$awk '/test/ {i=i+1} END {print i}' /tmp/dummy
2
```

Pokud chcete najít více vzorů, je rozumné nahradit text v apostrofech výrazem *–f file.awk* a pak všechny vzory a operace definovat v souboru *file.awk*.

grep (tisk řádků odpovídajících hledanému vzoru)

Několik příkladů příkazu *grep* jsme už viděli, jedná se o příkaz, který vypíše řádky odpovídající zadanému vzoru. Dokáže však ještě víc.

```
$grep "look for this" /var/log/messages -c
12
```

V souboru /var/log/messages byl řetězec look for this nalezen dvanáctkrát. (Dobře, tenhle příklad je podvrh, museli jsme soubor /var/log/messages modifikovat.)

wc (počítání řádků, slov a bajtů)

V následujícím příkladu uvidíme, že výstup nebude odpovídat našim očekáváním. Soubor *dummy*, s nímž pracujeme, obsahuje text *bash introduction howto test file*.

```
$wc --words --lines --bytes /tmp/dummy
2 5 34 /tmp/dummy
```

Příkaz wc se nestará o pořadí parametrů. Spočítané hodnoty vypisuje vždy v neměnném pořadí, tedy řádky, slova a bajty.

sort (seřazení řádků textu)

Tentokrát obsahuje náš soubor text

b
c
a

```
$sort /tmp/dummy
```

Výstup bude

a
b
c

Žádný příkaz by neměl být tak jednoduchý :-).

bc (kalkulačka)

Příkaz *bc* přijímá vstup z příkazového řádku (nebo ze souboru, ne z přesměrování nebo roury), případně i z uživatelského rozhraní. Následující demonstrace předvede některé příklady. Všimněte si, že *bc* spustíme s parametrem `-q`, aby nevypisoval uvítací hlášení. (Kurzívou jsou uváděny texty vytisknuté programem.)

```
$bc -q

1 == 5
0
0.05 == 0.05
1
5 != 5
```

```
0
2 ^ 8
256
sqrt(9)
3
while (i != 9) {
i = i + 1;
print i
}
123456789
quit
```

tput (inicializace terminálu nebo dotaz na databázi terminfo)

Malá demonstrace možností programu *tput*:

```
$tput cup 10 4
```

Prompt se objeví na souřadnicích (y10, x4).

```
$tput reset
```

Vymaže obrazovku a zobrazí prompt na (y1, x1). Levý horní roh je (y0, x0).

```
$tput cols
80
```

Vypíše počet znaků na řádku.

Vřele doporučujeme se (přinejmenším) s tímto programem seznámit. Existuje spousta malých programků, které vám dovolí dělat na příkazové řádce hotové divy.

Některé příklady jsme převzali z manuálu nebo z dokumentů FAQ.

Další skripty

Použití příkazu na všechny soubory v adresáři

Příklad: Velmi jednoduchý zálohovací skript (*vylepšený*)

```
#!/bin/bash
SRCD="/home/"
TGTD="/var/backups/"
OF=home-$(date +%Y%m%d).tgz
tar -cZf $TGTD$OF $SRCD
```

Přejmenování souborů

```
#!/bin/sh
# renna: přejmenuje více souborů podle zadaných pravidel
# napsal felix hudson Jan - 2000

# nejprve ověříme různé "režimy" tohoto programu
# pokud první parametr ($1) vyhovuje podmínce, provedeme příslušnou
```

```
# část programu a skončíme

# budeme přejmenovávat prefix?
if [ $1 = p ]; then

# nyní máme v ($1) režim a ve ($2) prefix
  prefix=$2 ; shift ; shift

# rychlá kontrola, jestli byly zadány názvy souborů
# pokud ne, raději neuděláme nic, než abychom přejmenovali neexistující soubo-
ry!!
  if [$1 = ]; then
    echo "no files given"
    exit 0
  fi

# následující smyčka prochází přes všechny zadané soubory
# pro každý soubor se provede přejmenování
  for file in $*
  do
    mv ${file} $prefix$file
  done

# a ukončíme program
  exit 0
fi

# budeme přejmenovávat suffix?
# zbytek je prakticky shodný s předchozím kódem, komentáře viz výše
if [ $1 = s ]; then
  suffix=$2 ; shift ; shift

  if [$1 = ]; then
    echo "no files given"
    exit 0
  fi

  for file in $*
  do
    mv ${file} $file$suffix
  done

  exit 0
fi

# bude přejmenovávat celé soubory?
if [ $1 = r ]; then
  shift

# následující kousek slouží čistě jako ochrana, abychom nepoškodili
# žádné soubory pokud uživatel nespecifikuje, co se má udělat
  if [ $# -lt 3 ] ; then
    echo "usage: renna r [expression] [replacement] files... "
```

```
    exit 0
  fi

# smazání ostatních informací
  OLD=$1 ; NEW=$2 ; shift ; shift

# následující smyčka prochází přes všechny zadané soubory a přejmenovává
# je pomocí programu sed
# jde o jednoduchý příkaz, který čte standardní vstup a nahrazuje
# jeden řetězec jiným
# my mu předáváme název souboru (na standardní vstup) a nahrazujeme
# požadovaný text
  for file in $*
  do
    new=`echo ${file} | sed s/${OLD}/${NEW}/g`
    mv ${file} $new
  done
  exit 0
fi

# pokud jsme se dostali až sem, nebyly programu předány žádné
# rozumné parametry, takže řekneme uživateli, co má dělat
echo "usage;"
echo " renna p [prefix] files.."
echo " renna s [suffix] files.."
echo " renna r [expression] [replacement] files.."
exit 0

# konec
```

Přejmenování souborů (jednoduché)

```
#!/bin/bash
# renames.sh
# jednoduchý přejmenovávací filtr

criteria=$1
re_match=$2
replace=$3

for i in $( ls *$criteria* );
do
  src=$i
  tgt=$(echo $i | sed -e "s/$re_match/$replace/")
  mv $src $tgt
done
```

Když něco nefunguje (ladění)

Jak spustit BASH

Rozumné je zadat na první řádek

```
#!/bin/bash -x
```

Díky tomu uvidíme spoustu zajímavých informací navíc.

O tomto dokumentu

Uvítáme všechny opravy a doplňky, případně další náměty na to, co by se mělo v dokumentu objevit. Budu jej aktualizovat, jak budu moci.

Bez záruky

Dokument je poskytován bez jakýchkoliv záruk.

Překlady

italština: William Ghelfi (*wizzy@tiscalinet.it*), *http://web.tiscalinet.it/penguin_rules/*

francouzština: Laurent Martelli

korejština: Minseok Park, *http://kldp.org/*

korejština: Chun Hye Jin

španělština: anonym, *http://www.insflug.org/*

Určitě existují i další překlady, nemám však o nich informace. Pokud o nějakém víte, laskavě mě informujte.

Poděkování

- Překladatelům do jiných jazyků (viz předchozí odstavec).
- Nathanu Hurstovi za řadu oprav.
- Jonu Abbottovi za komentáře k vyhodnocování aritmetických výrazů.
- Felixi Hudsonovi za napsání skriptu *renna*.
- Keesi van der Broekovi za řadu oprav a přepsání části věnované užitečným programům.
- Mikeovi za části věnované nalezení *bash*e a za testovací soubory.
- Fieshovi za pěkný nápad do části o smyčkách.
- Lionovi za zmínku a časté chybě (`./hello.sh: Command not found`).
- Andreasi Beckovi za opravy a komentáře.

Historie

Doplnění nových překladů a drobné opravy.

Doplněna Keesova část věnovaná užitečným příkazům.

Další opravy a doplnění.

Dodány příklady porovnávání řetězců.

v0.8 zrušeno číslování verzí, datum by snad mělo stačit

v0.7 Další opravy a dopsání některých TO-DO částí

v0.6 Drobné opravy

v0.5 Přidána část o přesměrování

v0.4 Díky bývalému šéfovi byl dokument odstraněn z původního umístění a byl přesunut na správné místo na *www.linuxdoc.org*.

předchozí: nepamatuji si a nepoužívám rcs ani cvs :-(

Další informace

Introduction to bash (under BE), *http://org.laol.net/lamug/beforever/bashtut.htm*

Bourne Shell Programming, *http://207.213.123.70/book/*

KAPITOLA 9

Daniel Lopez Ridruejo, ridruejo@apache.org

Apache

Tento dokument obsahuje informace o webovém serveru Apache a souvisejících projektech. Obsahuje odkazy na další zdroje informací a detaily o implementaci.

Úvod

Tento dokument obsahuje přehled webového serveru Apache a souvisejících projektů. **Apache je nejpopulárnějším serverem na Internetu**. Noví uživatelé Apache, zejména ti, kteří přecházejí z platformyWindows, často neznají možnosti serveru Apache, užitečné přídavné moduly a, obecněji, jak to všechno spolu funguje. Tento dokument se zaměřuje na obecný obrázek takových možností společně s jejich krátkým popisem a odkazy na další zdroje informací. Informace byly získány z mnoha zdrojů, včetně webových stránek projektů, konferencí, poštovních konferencí, webových sídel Apache a mých vlastních zkušeností. Hlavní zásluhy patří autorům. Bez nich a jejich práce by tento dokument nebyl možný nebo by nebyl zapotřebí.

Odvolání: Pracuji pro Covalent. Poskytujeme produkty a podpůrné služby pro webový server Apache, a některé z nich zde uvádím, stejně jako konkurenční projekty nebo podobné open source projekty.

Pokud najdete nějaké překlepy, chyby nebo máte nějaké připomínky k možnému vylepšení nebo komentáře, dejte mi prosím vědět, abych mohl dokument opravit.

Apache

Apache je přední webový server, s více než 60 % podílu na trhu podle přehledů společnosti Netcraft. Některé klíčové faktory, které se podílely na úspěchu Apache:

- Licence Apache. Je to open source projekt, s licencí podobnou BSD, která umožňuje komerční i nekomerční využití Apache.
- Talentovaná komunita vývojářů s různým zázemím a otevřený proces vývoje.
- Modulární architektura. Uživatelé Apache mohou jednoduše přidávat funkce nebo rozvíjet Apache pro své specifické prostředí.
- Přenositelnost: Apache běží na skoro všech unixových (a linuxových) systémech, Windows, BeOS, sálových počítačích...
- Robustnost a bezpečnost

Mnozí komerční dodavatelé vytvořili řešení založená na Apache, včetně Oracle, Red Hat a IBM. Kromě toho, Covalent poskytuje přídavné moduly a podporu 24x7 pro Apache.

Následující webové servery používají server Apache nebo servery z něj odvozené. Když je Apache dost dobrý pro ně, pak je také dost dobrý pro vás :)

- Amazon.com
- Yahoo!
- W3 Consortium
- Financial Times
- Network solutions
- MP3.com
- Stanford

Z webového serveru Apache:

> *Projekt Apache je výsledkem spolupráce při vývoji software zaměřeného na vytváření robustní implementace serveru HTTP (webového serveru) na komerční úrovni s mnoha funkcemi s volně přístupným zdrojovým kódem.*

Projekt Apache se rozšířil z vytváření pouze webového serveru na další důležité technologie na straně serveru, jakými jsou např. Java nebo XML. Organizace Apache Software Foundation, popisovaná v další části, tyto projekty zastřešuje.

Apache Software Foundation

Organizace Apache Software Foundation existuje, aby poskytovala organizační, právní a finanční podporu pro softwarové open source projekty Apache. Dříve byla známa jako Apache Group, nadace byla vytvořena jako členská, nevýdělečná společnost za účelem toho, aby bylo zajištěno, že budou projekty Apache existovat i nadále bez ohledu na účast jednotlivých dobrovolníků, a aby umožnila přispívání duševního vlastnictví a zajistila finanční prostředky, a aby sloužila jako pojidlo pro zajištění práv při podílení se na open-source softwarových projektech.

Nebo jak ji popsal Roy T. Fielding, předseda ASF: *Cílem Apache Software Foundation je usnadnění a podpora spolupráce na vývoji softwarových projektů, které využívají metod spolupráce Apache přes Internet pro vytváření, správu a rozšiřování infrastruktury webu a dodržování standardů, které ji definují.*

Vývoj webových aplikací pro Apache

Existuje více způsobů poskytování obsahu pomocí Apache.

Statický obsah

Apache může poskytovat statický obsah, jakým jsou např. soubory HTML, obrázky, atd. Pokud je to vše, co potřebujete, je Apache pravděpodobně pro vás tím pravým. Podřadné Pentium s operačním systémem Linux a serverem Apache může jednoduše pokrýt statickým obsahem 10Mbps-ovou linku. Pokud je toto vaším hlavním využitím Apache, podívejte se také na část o výkonu.

Dynamický obsah

Na mnoha webových serverech se informace neustále mění a stránky je potřeba pravidelně aktualizovat nebo dynamicky generovat. To je to, proč se všude používá programování na straně

serveru: programovací jazyky, nástroje a systémy, které pomáhají vývojářům se na informace z různých zdrojů dotazovat nebo je upravovat (databáze, adresářové služby, uživatelské záznamy, další webové servery) a poskytovat obsah uživateli.

Skripty CGI

CGI znamená Common Gateway Interface. Skripty CGI jsou externí programy, které jsou volány, když uživatel požaduje konkrétní stránku. CGI obdrží informace od webového serveru (hodnoty proměnných z formulářů, druh prohlížeče, adresa IP klienta atd.) a použije tyto informace pro vytvoření webové stránky pro klienta.

Klady: Jelikož se jedná o externí program, může být vytvořen v libovolném jazyce a stejný skript bude také přenositelný mezi různými webovými servery. Protokol CGI je jednoduchý a vrácený výsledek se vytváří zápisem odpovědi na standardní výstup. Je to vyspělá technologie a existuje mnoho odkazů a příkladů, které můžete najít na Internetu nebo v knihách.

Zápory: Spouštění a inicializace procesu zabírá nějaký čas. Jelikož je CGI mimo server a instance musí být spuštěna a ukončena, dochází k ovlivnění výkonu. Pokud proces musí načíst externí knihovny nebo provést připojení k externí databázi, může být zpoždění důležité. Totéž platí, když je počet návštěv za sekundu moc velký. CGI jsou nestavová a správa relací musí být prováděna externě.

Jelikož CGI obvykle vyvolává silnou manipulaci s textem, jsou přirozenou volbou skriptovací jazyky. Část popularity jazyka Perl pramení z toho, že je programovacím jazykem CGI. A to z důvodu jeho rozšířené podpory práce s řetězci a zpracování textu. Existuje mnoho volně dostupných skriptů a knihoven CGI. Dobrým počátečním bodem je část Open Directory CGI.

Generátory serverů

Pokud je váš webový server rozsáhlý, můžete se při dynamickém generování obsahu dostat do problémů s výkonem. Alternativou jsou generátory obsahu offline. Tato řešení oddělují obsah od jeho prezentace. Generátor HTML čte zdroje a jeho výstupem jsou statické stránky, které tvoří webový server. Generátor může být spouštěn periodicky nebo při změnách obsahu.

Příští verze Cocoon budou pracovat v dávkovém režimu, aby tohoto dosáhly. Další možností je Web site meta language.

Zpracování mimo server

Webový server může předávat dynamické požadavky jinému programu. Tento program zůstává nečinný, dokud nepřijde požadavek. Požadavek je zpracován a vrácen webovému serveru, který jej předá klientovi. Toto eliminuje režii spojenou se skripty CGI. Příkladem tohoto přístupu jsou Fast CGI, Java servlets atd.

Fast CGI

Tento standard byl vyvinut, aby odstranil některé nedostatky protokolu CGI. Hlavním vylepšením je to, že jediný spuštěný proces může zpracovávat více než jeden požadavek. Existuje modul pro Apache, který implementuje protokol Fast CGI a knihovny pro Tcl, Perl atd. Více informací najdete na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-4.html#ss.4.6.

Java servlety

Externí Java virtual machine zpracovává požadavky. JVM může sídlit na stejném nebo jiném počítači. Tak pracuje mnoho aplikačních serverů. Pro zpracování na straně serveru jsou obvykle použity standardní knihovny. Můžete se podívat na JServ (http://www.linuxdoc.org/ HOWTO/ Apache-Overview-HOWTO-15.html#ss15.9) nebo Tomcat (http://www.linuxdoc.org/ HOWTO/ Apache-Overview-HOWTO-15.html#ss15.6).

Vložené interprety

Alternativou ke zpracování mimo webový server je vložení interpretu do samotného serveru. Existují zhruba dvě kategorie tohoto druhu modulů: moduly, které na požadavky odpovídají nebo je přímo upravují, a moduly určené ke zpracování příkazů vložených do stránek HTML před jejich předáním klientovi. Nejlepším příkladem jsou mod_perl (http://www.linuxdoc.org/HOWTO/ Apache-Overview-HOWTO-17.html#ss17.3) a PHP (http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-18.html).

Správa výkonu a šířky pásma

Samotný výkon je pouze jedním z faktorů, které je potřeba u webového serveru brát v úvahu (prvními jsou obvykle flexibilita a stabilita).

Jak už bylo řečeno, existují řešení pro zvýšení výkonu na velmi zatížených webových serverech, které poskytují statický obsah. Pokud provádíte hosting, server Apache také poskytuje způsoby, kterými můžete zjišťovat a řídit využití šířky pásma. Omezení v této souvislosti obvykle znamená zpomalení poskytnutí obsahu založeného na požadovaném souboru, konkrétní adrese IP klienta atd. To se dělá kvůli znemožnění zneužití.

- **mod_mmap**: Zahrnutý v aktuální verzi Apache, mapuje do paměti staticky konfigurovaný seznam často požadovaných, nezměněných souborů.
- **Mod_bandwidth**: *Umožňuje nastavení omezení šířky pásma pro servery nebo připojení, založené na konkrétním adresáři, velikosti souborů a vzdálené adrese IP nebo doméně.*
- **Bandwidth share module**: umožňuje omezování a vyrovnávání šířky pásma podle adresy IP klienta. Je aktivně spravovaný.
- **Mod_throttle**: Omezení šířky pásma pro virtuálního hostitele nebo uživatele.
- **Mod_throttle_access**: užitečný, pokud jste příliš zatíženi. Umožňuje omezování podle prostředků (soubory, adresáře atd.)

Virtuální hostitelé

Apache poskytuje rozsáhlou podporu virtuálních hostitelů, toto jsou další moduly, které poskytují specifické funkce:

- mod_dynvhost
- mod_pweb
- mod_v2h

Kromě toho, Apache 2.0 umožňuje podřízené zpracování různých domén, aby měly různé identifikátory uživatelů a zvýšení zabezpečení.

Vyrovnávání zátěže

Apache obsahuje více modulů, které umožňují distribuci požadavků mezi servery, pro redundance, zvýšení dostupnosti atd.

- **Reverse proxying + mod_rewrite**: V Apache neexistuje nic, co byste nemohli dělat s mod_rewrite ... :) Tato technika umožňuje, aby předřazený server Apache pracoval jako server proxy pro záložní servery.
- **Mod_redundancy**: Převzetí webu a adresy IP v případě selhání.
- **Mod_backhand**: *Umožňuje přesměrování požadavků HTTP z jednoho webového serveru na jiný. Toto přesměrování může být prováděno na cílové stroje s méně využitými prostředky, čímž se provádí jemné vyrovnávání zátěže pro webové požadavky.* Více informací najdete na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-7.html.

Zabezpečené transakce

Existuje více řešení, která umožňují zabezpečení transakcí pro servery Apache. Toto umožňuje použití serverů Apache v e-komerci nebo jiných variant využití tam, kde se přenášejí citlivé informace (jako např. čísla kreditních karet).

- Mod_ssl a Apache-SSL jsou open source implementace. Řídí se evropskými normami, takže se na ně nevztahují patenty RSA.
- Red Hat nabízí zabezpečený server odvozený od Apache. Red Hat získal C2Net, autory StrongHold, jiného zabezpečeného serveru odvozeného od Apache.
- Covalent prodává zabezpečené verze Apache a modul CovalentSSL, který se přidává do stávajících instalací Apache.

Transakce s kreditními kartami

Specifická řešení pro Apache pro transakce s kreditními kartami:

- Cypay modul pro kreditní karty pro Apache. Založený na šablonách, výpočty poplatků.
- Covalent credator, podpora více středisek, překonávání selhání, podpora jazyků PHP, Perl a Java.

SNMP

SNMP znamená Simple Network Management Protocol. Umožňuje sledování a správu síťových serverů, vybavení, atd. Moduly SNMP pro Apache pomáhají spravovat rozsáhlé nasazení webových serverů, zjišťovat kvalitu nabízených služeb a integraci Apache ve stávajících strukturách správy.

- Open source Mod SNMP pro Apache 1.3.
- Covalent SNMP poskytuje komerční modul SNMP, podporu pro nejnovější standard SNMPv3, integraci s HP-Openview, Tivoli atd.

Moduly pro ověřování

V mnoha situacích (předplacené služby, citlivé informace, privátní oblasti) je vyžadováno ověření uživatele. Apache obsahuje základní podporu ověřování. Existují další ověřovací moduly, které umožňují připojit Apache do stávajících zabezpečených struktur nebo databází, zahrnujících: řadič domény NT, Oracle, mySQL, PostgresSQL atd.

Speciálně zajímavé jsou moduly LDAP, jelikož umožňují integraci se stávajícími firemními adresářovými službami.

Tyto moduly můžete nalézt na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-10.html.

Grafická rozhraní pro Apache

Server Apache je konfigurován pomocí textových konfiguračních souborů. To je výhoda i nevýhoda. Správa může být prováděna z libovolného počítače, který má přístup na Internet a SSH (www.openssh.com). Ruční úpravy konfiguračního souboru vyžadují studium. Existují open source grafické nástroje, které tuto úlohu zjednodušují:

- Comanche: existují pro více platforem – Unix/Linux, Windows a Mac. Podívejte se na snímky obrazovky a podrobnější informace na webu. Odvolání: Jsem hlavním autorem Comanche, takže si pamatujte, že zde nejsou žádné chyby, ale jen nezdokumentované funkce :)
- gui.apache.org: Grafická uživatelská rozhraní pro projekt Apache. Programy s různým stupněm vývoje.
- Webmin: Hezké webové rozhraní (http://www.webmin.com/webmin/default.htm).

Vytváření modulů pro Apache

Apache, jako mnoho jiných úspěšných open source projektů, má modulární architekturu. To znamená, že pro přidání nebo úpravu funkcí nepotřebujete znát celý kód. Přístup ke zdrojovému kódu Apache znamená, že si můžete sestavit server pouze s těmi moduly, které potřebujete, a můžete vložit své vlastní.

Rozšiřování Apache může být prováděno v jazyce C nebo v řadě jiných jazyků, které používají příslušné moduly. Tyto moduly zpřístupňují interní funkce Apache pro různé programovací jazyky, jako např. Perl nebo Tcl.

Vytváření modulů v C: Apache je napsán v C a zrovna tak moduly distribuované s Apache. Nejlepším způsobem započetí vytváření modulů pro Apache je přečtení knihy od Douga MacEacherna a Lincolna Steina „Writing Apache modules with Perl and C“. Je to dobře napsaná kniha, snadno čitelná pro guru serveru Apache a jazyka Perl. Pokud nemáte peníze na zakoupení této knihy nebo si ji nemůžete zapůjčit od kamaráda, existují jiné cesty. Můžete se přečíst některé z online kurzů o vytváření modulů pro Apache: Ken Coar, člen Apache Group, má hezký kurs online na adrese http://web.golux.com/coar/slides/default.htm. Přehled architektury Apache najdete na adrese http://www.grad.math.uwaterloo.ca/~oadragoi/CS746G/a1/apache_conceptual_arch.html. Webové sídlo Apache obsahuje některé poznámky k API, které vám pomohou začít, na adrese http://www.apache.org/docs/misc/API.html. Doporučuje se také prohlédnout si zdrojový kód modulů obsažených v Apache. Apache obsahuje pro tento účel jeden jednoduchý modul (mod_example.c).

Vytváření modulů pro Apache v jiných jazycích: Existuje řada modulů pro Apache, které umožňují jazykům třetích stran přístup k internímu API serveru Apache. Nejpopulárnější je mod_perl.

Pokud máte nějaké otázky k vývoji modulu pro Apache, měli byste se zapsat do e-mailové konference o modulech pro Apache na adrese http://modules.apache.org. Nejprve si udělejte domácí práci, prozkoumejte staré zprávy a vyzkoušejte všechnu výše popsanou dokumentaci. Je šance, že někdo jiný už měl stejný problém, se kterým jste se setkali a že dostal užitečnou odpověď.

Pokud se zajímáte o vývoj jádra samotného serveru Apache, měli byste se podívat na server vývoje Apache na adrese http://dev.apache.org.

Knihy o serveru Apache

Kompletní seznam knih o serveru Apache najdete na adrese http://www.apache.org/info/apache_books.html.

Několik knih, které mohu osobně doporučit:

- Writing Apache Modules with Perl and C pokud se zajímáte o vnitřní strukturu Apache.
- Apache server for dummies pokud chcete začít se serverem Apache. Nedejte se zmást názvem. Je to obsažná kniha naplněná užitečnými informacemi.

WebDAV

Z webového sídla WebDAV (www.webdav.org): *WebDAV znamená „Web-based Distributed Authoring and Versioning". Je to sada rozšíření protokolu HTTP, která uživatelům umožňují spolupracovat na úpravách a spravovat soubory na vzdálených webových serverech.*

Je to ekvivalent standardů protokolu MS FrontPage, ale posouvá tento nápad o několik kroků dále. Na jeho základě je možno sestavovat jiné protokoly (například se podívejte na adresu Subversion http://subversion.tigris.org/default.htm).

Projekty jazyka Java

Z historických důvodů je možno projekty jazyka Java najít na adrese java.apache.org i jakarta.apache.org. Konečným cílem je to, aby se všechny projekty jazyka Java přesunuly pod server Jakarta.

Cílem projektu Jakarta je umožnění poskytnutí řešení serveru komerční kvality založeného na platformě Java, který je vyvíjen v otevřeném modelu spolupráce.

Java v komunitě Apache je velmi dynamická a aktivní, což ukazuje kvantita a kvalita jeho podprojektů, které si popíšeme.

Ant

Mohli byste si myslet, že Ant je ekvivalentem make, napsaným v jazyce Java. Je to velký úspěch v projektech souvisejících s jazykem Java. Vývojáři mohou používat jazyk Java namísto příkazů prostředí. To znamená zvýšenou přenositelnost a rozšiřitelnost. Namísto souborů Makefile má Ant soubory XML. Více se o Ant dozvíte na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.1.

ORO a Regexp

ORO je kompletní balíček, který umožňuje podporu regulárních výrazů pro jazyk Java. Zahrnuje podporu regulárních výrazů jazyka Perl5 apod. Všechno pod licencí Apache. Více informací o ORO najdete na adrese jakarta.apache.org/oro/index.html. Možná zde najdete i další balíček pro podporu regulárních výrazů, Regexp.

Slide

Slide je systém správy obsahu vysoké úrovně. Koncepčně poskytuje hierarchickou organizaci binárního obsahu, který může být uložen do libovolných, heterogenních, distribuovaných datových úložišť. Kromě toho, Slide integruje služby pro zabezpečení, uzamknutí a informace o verzích.

Pokud znáte WedDAV, Slide jej hodně používá. Jednoduše řečeno, to, co Slide poskytuje, je unifikovaný, jednoduchý přístup k prostředkům a informacím. Tyto prostředky mohou být uloženy v databázi, systému souborů, atd. a zpřístupněny skrze rozhraní WebDAV nebo vlastní API Slide.

Více se můžete ozvědět na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.3.

Struts

Struts je projekt Apache, který se pokouší přenést paradigma návrhu Model-View-Controller (MVC) do vývoje pro web. Staví na technologiích Servlet a JavaServer Pages. Modelová část jsou objekty serveru Java, které reprezentují interní stav aplikací. Enterprise Java Beans jsou zde všeobecně používané. Zobrazovací část je konstruována skrze JavaServer Pages (JSP), které jsou kombinací statického HTML/XML a jazyka Java. JSP také vývojáři umožňují definovat vlastní značky. Řídící část jsou servlety, které dostávají požadavky (GET/POST) od klienta, provádí akce na modelu a aktualizují zobrazení poskytnutím příslušného JSP. Více se můžete dozvědět na adrese http://www.linuxdoc.org/ HOWTO/ Apache-Overview-HOWTO-15.html#ss15.4l.

Taglibs

Technologie JavaServer pages umožňuje vývojářům poskytovat funkce přidáním vlastních značek. Projekt Taglibs má být společným úložištěm pro tato rozšíření. Zahrnuje značky pro společné utility (for, date), přístup k databázi SQL atd.

Více se o TagLibs dozvíte na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.5. Více dokumentace je zahnuto v balíčku.

Tomcat

Tomcat je stěžejním produktem projektu Jakarta. Je to oficiální referenční implementace pro technologie Java Servlet 2.2 a JavaServer Pages 1.1.

Více se dozvíte na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.6. Projekt Tomcat byl započat věnováním kódu od Sun Microsystems.

Velocity

Velocity je systém šablon založených na jazyce Java. Může být použit jako samostatná utilita pro generování zdrojového kódu, HTML, zpráv, nebo může být kombinován s jinými systémy pro poskytování služeb šablon. Velocity má paradigma Model View Controller, které vyžaduje oddělování kódu Java od šablony HTML.

Více se o Velocity dozvíte na adrese http://www.linuxdoc.org/ HOWTO/ Apache-Overview-HOWTO-15.html#ss15.7. Velocity je součástí jiných projektů, jako např. Turbine.

Watchdog

Projekt Watchdog poskytuje ověřovací testy pro specifikace Servlet a JavaServer Pages. Více se dozvíte na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.8.

JServ

Apache JServ je 100% čisté jádro Java servlet, které je plně kompatibilní se specifikací JavaSoft Java Servlet APIs 2.0.(...) Výsledkem je čisté jádro servlet, které pracuje na kterékoliv „verzi 1.1" Java Virtual Machine.

JServ je jedním z původních projektů Java Apache. Tomcat, až bude dokončen, bude následovníkem JServ. Víc informací získáte na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.9.

JSSI

JSSI je implementací SSI (server side includes) v jazyce Java. Server side includes jsou značky vložené do souborů, které jsou zpracovávány předtím, než je stránka předána klientovi (např. pro vložení aktuálního data). Více informací se dozvíte na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.10.

Apache JMeter

Apache JMeter je 100% čistá aplikace napsaná v jazyce Java navržená pro testování chování a zjišťování výkonu. Původně byl navržen pro testování webových aplikací, ale byl dále rozšířen o další testovací funkce.

Může být použit pro testování statických a dynamických prostředků a získání okamžité vizuální zpětné vazby.

Více se můžete dozvědět a prohlédnout si snímky obrazovky na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.11.

Server Pages Foundation Classes

Je to sada knihoven, které pomáhají řešit společné problémy ve vývoji aplikací na straně serveru. Zaměřují se na dva z nich:

- **Kombinování HTML a Java**: Obsahuje knihovnu tříd, které se starají o generování HTML a mohou být integrovány se zbytkem kódu Java.
- **HTTP je nestavový protokol**: SPFC obsahuje podporu relací, takže aplikace mohou sledovat uživatele při procházení webovým serverem. Vývojář aplikace se nemusí starat o specifické detaily generování stránky. Může přemýšlet v obecnějších termínech tradičních aplikací. Více se o SPFC dozvíte na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.12.

Element Construction Set

Element Construction Set (ECS) je API jazyka JAVA pro generování prvků pro různé jazyky používající značky a přímo podporuje HTML 4.0 a XML, ale může být jednoduše rozšířeno pro vytváření značek pro jakýkoliv jazyk používající značky.

Umožňuje generování značek pomocí volání funkcí Java, což vede k čistšímu řešení než kombinování kódu HTML a Java. Více se dozvíte na adrese http://www.linuxdoc.org/ HOWTO/ Apache-Overview-HOWTO-15.html#ss15.13.

Avalon

Pokud znáte Perl nebo systémy BSD, Avalon je zhruba ekvivalentem technologií CPAN nebo Portsů pro Apache. Neposkytuje pouze pomůcku pro společné uložení kódu, je o krok napřed: *jde o program pro vytváření, návrh, vývoj a správu společného systému serverových aplikací vytvořených v jazyce Java*. Poskytuje prostředky k tomu, aby projekty v jazyce Java na straně serveru mohly být jednoduše integrovány a stavěny na sobě navzájem.

JAMES (Java Apache Mail Enterprise Server)

Jako doplněk k jiným technologiím Apache na straně serveru JAMES poskytuje *server 100% v jazyce Java navržený tak, aby byl kompletním a přenositelným podnikovým řešením poštovního systému založeného na aktuálně dostupných otevřených protokolech (SMTP, POP3, IMAP, HTTP).*

Více informací najdete na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.15.

PicoServer

Malý server HTTP/1.0 napsaný čistě v jazyce Java. Vypadá to, že se projekt zastavil a není k dispozici žádný kód. Webový server naleznete na adrese http://www.linuxdoc.org/HOWTO/ Apache-Overview-HOWTO-15.html#ss15.16.

Jetspeed

Jetspeed je webový portál vytvořený v jazyce Java. Má modulární API, které umožňuje seskupení různých zdrojů dat (XML, SMTP, iCalendar).

Turbine

Turbine je systém založený na servletech, který umožňuje zkušeným vývojářům v jazyce Java rychle sestavovat zabezpečené webové aplikace. Turbine spojuje platformu pro spouštění kódu Java *a* znovupoužitelné komponenty, všechno pod licencí Apache. Některé z jeho funkcí

- Integrace se systémy šablon
- Vývoj typu MVC
- Seznamy řízení přístupu
- Podpora lokalizace
- atd.

Pokud se o tento systém zajímáte, můžete se podívat na webový server Turbine na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.18.

Jyve

Projekt Jyve je postaven nad systémem Turbine. Je to aplikace, která poskytuje webový systém FAQ (často kladených dotazů).

Alexandria

Alexandria je integrovaný systém správy dokumentace. Spojuje technologie společné pro mnoho open source projektů jako jsou CVS a JavaDoc. Cílem je integrace zdrojového kódu a dokumentace pro udržování a sdílení dokumentace ke kódu. Více informací najdete na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.22.

Log4j

Tento balíček poskytuje systém přihlašování, který mohou aplikace v jazyce Java používat. Může být zapnut za běhu bez úprav binárního kódu a byl navržen s ohledem na výkon. Můžete jej nalézt na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss15.21.

Projekty XML

Přímo podle webového serveru Apache XML je jeho cílem:

- *Poskytovat na standardech založená řešení XML v komerční kvalitě, která jsou vyvinuta v otevřeném modelu spolupráce.*
- *Poskytovat zpětnou vazbu standardizujícím společnostem (jako např. IETF nebo W3C) z pohledu implementace.*
- *Zaměřit se na aktivity spojené s XML uvnitř projektů Apache.*

Domovská stránka projektu se nachází na adrese http://xml.apache.org. Zastřešuje různé podprojekty.

Úvod do XML

Toto je krátký úvod do XML. Pokud se chcete dozvědět o XML něco více, bude pro vás pro začátek nejlepší http://www.xml.com. XML je jazyk používající značky (jako HTML) pro popis strukturovaného obsahu pomocí značek a atributů. Jakmile je obsah oddělen od jeho prezentace, můžete si vybrat, jak jej chcete zobrazit (mobilní telefon, HTML, text) nebo změnit. Standard XML pouze popisuje, jak mohou být značky a atributy uspořádány, nikoliv názvy toho, co znamenají. Apache poskytuje nástroje popsané v následujících bodech.

Xerces

Projekt Xerces poskytuje parsery XML pro různé jazyky, včetně Java, C++ a Perl. Vazby Perl jsou založeny na zdrojích C++. Existují vazby Tcl pro Xerces ve verzi 2.0 TclXML od Steva Balla. Tato verze 2.0 je v tomto okamžiku dostupná pouze na Ajuba CVS na adrese http://dev.ajubasolutions.com/software/tcltk/netcvs.html. Parser XML je nástroj používaný pro programový přístup k dokumentům XML. Toto je popis standardů podporovaných projektem Xerces:

- DOM: DOM znamená Document Object Model. Dokumenty XML jsou přirozeně hierarchické (vnořené značky). Dokumenty XML jsou přístupné skrze tři podobná rozhraní. Proces je následující:
  - Analýza dokumentu
  - Vytvoření stromu

- ❑ Přidání/odstranění/úpravy uzlů
- ❑ Serializace stromu

- SAX: Jednoduché API pro XML. Je to API založené na proudech. To znamená, že obdržíme zpětná volání tak, jak se elementy vyskytly. Tato zpětná volání mohou být použita např. pro konstrukci stromu DOM.
- XML Namespaces
- XML Schema: Standard XML poskytuje syntaxi pro vytváření dokumentů. XML Schema poskytuje nástroj pro definování *obsahu* dokumentu XML (sémantiku). Umožňuje definovat, že určitý element v dokumentu musí být celé číslo mezi 10 a 20 atd.

Původní základna kódu projektu Xerces XML byla věnována společností IBM. Více informací najdete na domovských stránkách Xerces Java (http://xml.apache.org/xerces-j/index.html), Xerces C (http://xml.apache.org/xerces-c/index.html) a Xerces Perl (http://xml.apache.org/xerces-p/index.html) a také na stránce http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss16.2.

Xalan

Xalan je procesor XSLT dostupný pro jazyky Java a C++. XSL je jazyk šablon pro XML. T znamená Transformace. XML je dobrý pro ukládání strukturovaných dat (informací). Někdy potřebujeme tato data zobrazit uživateli nebo aplikovat nějakou transformaci. Xalan bere původní dokument XML, načítá konfiguraci transformace a jeho výstupem je HTML, holý text nebo jiný dokument XML. Více se o projektu Xalan můžete dozvědět na domovských stránkách Xalan Java (http://xml.apache.org/xalan/index.html) a Xalan C (http://xml.apache.org/xalan-c/index.html) a také na stránce http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss16.3.

FOP

Podle webového sídla je *FOP aplikace v jazyce Java, která načítá strom objektu formátu a pak jej uloží do dokumentu PDF.* Takže FOP bere dokument XML a provádí výstup PDF podobným způsobem, jakým to Xalan dělá s HTML nebo textem. Více se o FOP dozvíte na adrese http://xml.apache.org/fop.

Cocoon

Cocoon pomocí jiných technologií Apache XML jako např. Xerces, Xalan a FOP poskytuje všeobecný publikační systém. Cocoon je založen na XML a XSL a je určen pro servery se střední až vysokou složitostí. Podle popisu na webovém serveru odděluje obsah, logiku a prezentaci:

- **Vytváření XML**: *soubor XML je vytvořen vlastníky obsahu. Nepotřebují žádnou specifickou znalost toho, jak je obsah XML dále zpracovávám, ale spíše podrobně zvolený DTD/obor názvů. Tato vrstva je vždy prováděna lidmi přímo v normálních textových editorech nebo nástrojích či editorech pro XML.*
- **Generátory zpracování XML**: *logika je oddělena od souboru s obsahem.*
- **Provádění XSL**: *Vytvářený dokument je pak překládán pomocí XSL transformace a zformátován na konkrétní druh prostředku (HTML, PDF, XML, WML, XHTML).*

Více se o projektu Cocoon dozvíte na adrese http://xml.apache.org/cocoon/index.html a také na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss16.5.

Xang

Cílem projektu Xang je *usnadnění sestavování aplikací pro XML pro web v komerční kvalitě.* Logika aplikace je definována v hierarchickém souboru XML, který může být skriptován pomocí jazyka JavaScriptu. Tento soubor definuje přístup k datům (což mohou být další soubory XML, zásuvné moduly jazyka Java, atd.). Jádro Xang se stará o mapování požadavků HTTP na příslušné handlery. Více se o projektu Xang dozvíte na adrese http://xml.apache.org/xang/samples.html a adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss16.6.

SOAP

Apache SOAP („Simple Object Access Protocol") je implementace SOAP podle W3C. Je založena na a nahrazuje implementaci IBM SOAP4J a nahrazuje ji.

Z konceptu specifikace W3C: SOAP je protokol pro výměnu informací v decentralizovaném, distribuovaném prostředí. Je to protokol založený na XML, který se skládá ze tří částí:

- *Obálka, která definuje systém popisu toho, co je ve zprávě a jak se má zpracovat,*
- *sada pravidel kódování pro vyjadřování instancí aplikací definovaných datových typů,*
- *a konvence pro reprezentaci vzdálených volání procedur a odpovědí.*

Považujte SOAP za systém vzdálených volání procedur založený na XML nebo systému CORBA. Je založen na HTTP a XML. Na jednu stranu to znamená, že je ve srovnání s jinými systémy užvaněný a pomalý. Na druhou stranu zjednodušuje univerzálnost, ladění a vývoj klientů a serverů pro řadu jazyků (C, Java, Perl, Python, Tcl atd.), jelikož má většina moderních jazyků moduly HTTP a XML. Více se dozvíte na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss16.7.

Batik

Batik je sada nástrojů založená na jazyce Java pro aplikace, které chtějí obrázky ve formátu SVG (Scalable Vector Graphics) pro řadu účelů, jako např. zobrazování, generování nebo manipulaci.

Je založený na XML a kompatibilní se specifikací W3C. Je trochu netypický pro projekty Apache, v tom, že obsahuje grafický komponent. Batik obsahuje propojení pro rozšíření systému pomocí vlastních značek a to umožňuje konverzi z SVG do jiných formátů, jako např. JPEG nebo PNG.

Domovská stránka projektu Batik je na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss16.8.

Crimson

Toto je alternativní, na jazyce Java založený, parser XML s podporou XML 1.0 s různými rozhraními. Je to parser aktuálně dodávaný v produktech Sun, a je to prostřední krok před vydáním Xerces verze 2.

Domovská stránka projektu Crimson se nachází na adrese http://www.linuxdoc.org/ HOWTO/ Apache-Overview-HOWTO-15.html#ss16.9.

Další projekty XML

Existují další projekty založené na Apache a XML, které nezastřešuje Apache XML

- mod_xslt. Je to modul vytvořený v jazyce C pro předávání obsahu založeného na XML/XSL. Je pod licencí GPL.

- AxKit je aplikační server založený na XML pro mod_perl a Apache. Umožňuje oddělení obsahu a jeho prezentace.

Perl

Perl a Apache tvoří výkonnou a populární kombinaci. Existuje více projektů, které používají tyto dvě technologie.

Embperl

Umožňuje vkládání jazyka Perl do stránek HTML. Tyto stránky jsou zpracovány na serveru předtím, než jsou předány klientovi. Je podobný PHP. Více informací najdete na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-15.html#ss17.1.

Mason

Projekt Mason vkládá Perl do HTML pomocí modelu znovupoužitelných komponentů. Umožňuje ukládání do mezipaměti, použití šablon atd.

Mod_Perl

Mod_perl je jedním z nejstarších a úspěšných projektů Apache. Vkládá interpret jazyka Perl do Apache a umožňuje přístup z jazyka Perl k interním prvkům webového serveru. Umožňuje vytváření celých modulů v jazyce Perl nebo kombinaci kódu jazyků Perl a C. V serverech Apache verze 1.3 musí být každý interpreter vložen do jednoho podřízeného procesu, jelikož je server víceprocesový. V hodně zatížených dynamických serverech se může lišit velikostí. Apache 2.0 je vícevláknový, stejně jako poslední verze jazyka Perl. Další generace modulu mod_perl toho využívá a umožňuje sdílení kódu, dat a stavů relací mezi interprety. To jej dělá rychlejším a spolehlivým řešením.

Podívejte se také na Axkit.

PHP

Z webového serveru PHP (http://www.php.net): *PHP je skriptovací jazyk na straně serveru, vkládaný do HTML a nezávislý na platformě.* PHP je skriptovací jazyk jako např. Perl, Python nebo Tcl. Je to nejpopulárnější modul pro Apache, a to z různých důvodů:

- Jeho studium netrvá dlouho
- Je k dispozici skvělá dokumentace
- Rozsáhlá podpora databází
- Modularita

PHP je modulární. Existují moduly pro podporu:

- Konektivita k databázím Oracle, ODBC, mySQL, mSQL, PostgreSQL, MS-SQL... a mnoha dalším, podívejte se na webové stránky PHP http://www.php.net.
- Podpora XML
- Přenos souborů: FTP
- HTTP
- Podpora adresářů: LDAP

- Podpora pošty a news: IMAP, POP3, NNTP
- Generování dokumentů PDF
- CORBA

a mnoho dalších. Potřebujete pouze zkompilovat nebo použít moduly, které potřebujete.

PHP může být použito přímo v Apache, jako externí CGI nebo v jiných webových serverech. Je nezávislé na platformě a běží na většině unixových systémů i systémů Windows.

Pokud pracujete na platformě Windows, pravděpodobně používáte Internet Information Server s ASP (Active Server Pages) a serverem MS-SQL. Podobnou náhradou ve světě Unixu pro tuto trojici je Apache s PHP a mySQL. Jelikož PHP pracuje:

- s Apache i Microsoft IIS
- s mySQL i MS-SQL
- na Unixu i ve Windows

je zde krásná možnost k přechodu z řešení od Microsoftu k bezpečnějším, stabilnějším a výkonným řešením pro Unix (jako např. FreeBSD, Solaris, Linux nebo OpenBSD).

Python

Python je skriptovací jazyk podobný jazykům Perl nebo Tcl. Existuje několik modulů, které vkládají Python do serveru Apache:

- Mod Python
- Mod Snake: běží na serveru Apache 1.3.x i nastupujím 2.0

Oba moduly budou užitečné, pokud plánujete vytvářet moduly pro Apache v jazyce Python nebo spouštět stávající moduly CGI v jazyce Python rychleji. Mod_Snake umožňuje vkládat Python do HTML, podobně jako to dělá PHP.

Tcl

Projekt Tcl Apache integruje Tcl do webového serveru Apache. Tcl je malý, rozšiřitelný skriptovací jazyk. Více se o Tcl můžete dozvědět na adrese http://dev.ajubasolutions.com/default.htm. Pod záštitou Apache Tcl existuje aktuálně více modulů:

- Mod_dtcl umožňuje vkládání Tcl do stránek HTML, podobně jako to dělá PHP.
- Neowebscript má podobný přístup
- Mod_tcl má podobný přístup jako mod_perl a běží na verzích 1.3.x i 2.x serveru Apache.

Další projekt Tcl Apache naleznete na adrese WebSH http://websh.com/default.htm a na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-18.htm.

Moduly pro další jazyky

Tento dokument popisuje moduly pro populární jazyky pro stranu serveru, jakými jsou Perl, Python a PHP. Moduly pro další jazyky (JavaScript, Haskell atd.) naleznete v adresáři modulů pro Apache na adrese http://www.linuxdoc.org/HOWTO/Apache-Overview-HOWTO-21.html.

Apache 2.0

Aktuální verze Apache (řada 1.3) je založena na procesech. To znamená, že server se dělí na více souběžných požadavků. Podřízené procesy jsou od sebe izolované. To je spolehlivé: pokud modul funguje špatně, nadřazený proces tento podřízený proces ukončí a projeví se to pouze na obsluhovaném požadavku, nikoliv na serveru jako celku. Vlákna jsou podobná malým procesům. Vlákna mohou sdílet společná data. Pokud vlákno funguje špatně, může narušit ostatní vlákna a server jako celek může zhavarovat. Na druhou stranu model s vlákny umožňuje vytváření rychlejších a spolehlivějších webových serverů. Apache 2.0 přebírá to nejlepší z obou světů, čímž umožňuje uživateli definovat počet procesů a počet vláken na jeden proces. Apache 2.0 zavádí APR (Apache Portable Runtime), který zvyšuje ještě více přenositelnost Apache. Konečně, vrstvené vstupy a výstupy přináší novou úroveň modularity do vývoje Apache.

Přechod z webových serverů Netscape (iPlanet)

Celková práce může spočívat v konverzi vlastních modulů z NSAPI do API Apache. Skoro všechny další technologie na straně serveru (Java, Perl, CGI) by měly být přenositelné jen s malými nebo žádnými změnami. Netscape je pevně integrován se servery LDAP. Na moduly LDAP se můžete podívat na adrese http://modules.apache.org. Netscape poskytuje podporu jazyka JavaScript na straně serveru. Na ekvivalent od Apache, mod_javascript, se můžete podívat na adrese http://www.geocities.com/TimesSquare/Fortress/9743/binjs.html.

Přechod z Microsoft IIS

Častými důvody, proč lidé přecházejí od IIS k Apache, jsou stabilita, výkon a zabezpečení. To je zčásti kvůli tomu, že většina lidí, kteří provozují Apache, to dělají na systémech Unix (jako např. Solaris, FreeBSD nebo Linux). Naštěstí je Apache víceplatformový a běží na systémech Unix i Windows, čímž nabízí rozumnou cestu přechodu.

Obvyklá vývojová prostředí pro web ve Windows, jako např. Coldfusion nebo Active Server Pages, mají verze i pro Unix nebo kompatibilní prostředí (některé jsou komerční, některé volně dostupné):

- Coldfusion for Linux
- Perl ASP module
- Halcyon ASP
- OpenASP

Apache pro Windows podporuje také rozhraní ISAPI.

Pokud chcete kompletní řešení open source a přecházíte z platformy Windows (IIS + ASP + server MS-SQL), je přibližným ekvivalentem (a hodně populárním) kombinace Apache + PHP + MySQL nebo PostgresSQL. O PHP se dozvíte více na adrese v části PHP.

V nové verzi 2.0 Apache, která je v tuto chvíli ještě v betaverzi, je hodně zlepšena podpora pro Windows.

Odkazy

Další prostředky související s Apache.

Webová sídla

- Apache (http://www.apache.org)
- Adresář modulů pro Apache (http://modules.apache.org)
- Apache today (http://www.apachetoday.com)
- Sekce Apache na serveru Slashdot (http://slashdot.org/index.pl@section=apache)

Aplikační servery Java

Toto jsou aplikační open source servery, které staví na serveru Apache nebo se o nich ví, že dobře spolupracují s Apache.

- Resin: Servlety, JSP, XSL (http://www.caucho.com/default.htm)
- Enhydra: Aplikační server Java/XML (http://www.enhydra.com/default.htm).
- Locomotive: Servlety, vyrovnávání zátěže, překonání selhání (http://www.locomotive.org/default.htm).
- JBoss: Enterprise Java Beans, J2EE (http://www.jboss.org/default.htm)

Kontakt na autora

Můžete mě kontaktovat na adrese ridruejo@apache.org. Uvítám připomínky a opravy, ale prosím, prosím, neposílejte mi zprávy, ve kterých byste mě žádali o pomoc při odstraňování potíží s vaší instalací Apache. Nemám tolik času a váš e-mail bude pravděpodobně ignorován. Pokud potřebujete pomoc:

- Podívejte se do chybových protokolů, přečtěte si dokumentaci a zejména FAQ (často kladené dotazy) na adrese http://www.apache.org/docs/misc/FAQ.html.
- Vyzkoušejte comp.infosystems.www.servers.unix na http://www.google.com. Hledejte podobný problém.
- Pokud máte stále problémy, shromážděte tolik informací, kolik budete moci, příslušné položky chybového protokolu a kroky, které jste doposud provedli a pošlete vše do diskusní skupiny. To zvýší šanci, že vám někdo na vaši otázku odpoví.

Pokud chcete komerční podporu, zvažte kontaktování společnosti Covalent (http://www.covalent.net), která poskytuje odbornou podporu pro Apache (samozřejmě za poplatek). Pokud používáte Apache pod operačním systémem Linux, možná má váš dodavatel systému Linux podporu, která zahrnuje i Apache.

Překlady

Pokud chcete přispět překladem tohoto dokumentu, měli byste použít SGML. Podívejte se na další informace na http://www.linuxdoc.org. Dejte mi prosím vědět, abych vám mohl zajistit nejnovější verzi.

KAPITOLA 10

Guido Gonzato, ggonza@tin.it

Konfigurace systému

Tento dokument se zaměřuje na snadnější a jednodušší vyladění vašeho nově nainstalovaného systému Linux. Najdete zde řadu tipů pro konfiguraci nejběžnějších aplikací a služeb.

Úvod

Proč toto HOWTO

Aktuální distribuce se blíží dokonalosti, ale stále je zapotřebí nějaké jemné doladění. Mnoho nových uživatelů má strach ze zdánlivé složitosti systému Linux, a díky tomu jsem si všiml, že se některé otázky objevují na c.o.l.setup stále znovu. Z důvodu pokusu o vyřešení této situace a vlastní pohodlnosti, jsem napsal seznam co – dělat (to –do), který se nakonec stal tímto HOWTO. Najdete zde tipy pro konfiguraci a příklady nejčastěji používaných aplikací, programů a služeb, které by vám měly ušetřit značné množství času a práce.

Toto HOWTO dělám hlavně pro RedHat. V současné době mám k dispozici počítače se systémy Red Hat a Mandrake a jádry v rozsahu od 2.0.36 do 2.2.15, takže nepovažujte některé z mých tipů za dogma, pokud máte jiné distribuce. Předchozí verze tohoto HOWTO obsahovaly některé informace o systémech SuSE, Debian a Caldera, ale počítače s těmito systémy už nemám k dispozici, takže nemohu detaily aktualizovat. Žádná informace je lepší než nepřesná informace, takže adaptace mých tipů na vaše distribuce je pouze na vás.

Toto HOWTO nemůže, a není to ani záměrem, nahradit jiná. Čtení dokumentace a HOWTO se vždy vyplatí, takže vám to důrazně doporučuji, pokud se chcete dozvědět více. Také to není pro začátečníky: pokud zjistíte, že něco nechápete, podívejte se prosím do příslušného HOWTO. Dovolte mi připomenout, že správným místem pro hledání pomoci s konfigurací systému Linux je Usenet, tj. news:comp.os.linux.setup. *Prosím*, nehledejte pomoc u mě, protože jsem zcela zaneprázdněn.

Oficiální místo pro tento dokument, na kterém se nachází také ostatní HOWTO, na která se odkazuji, a některé překlady, je http://www.linuxdoc.org.

Co budeme konfigurovat

Mohlo by zde být nekonečně mnoho konfigurací hardwaru PC, ale podle mých zkušeností je toto společné: PC s velkým diskem rozděleným do tří oddílů (jeden pro DOS/Windows, jeden pro Linux, jeden odkládací), zvuková karta, modem, jednotka CD-ROM, tiskárna, myš. Velmi častá je také externí jednotka Zip připojená k paralelnímu portu. Tento počítač je možná připojen do kombinované sítě Windows-Linux, kde pracuje jako server.

To je hardware, který si myslím, že chcete nastavit, a následující tipy je snadné adaptovat na různé konfigurace. Implicitně předpokládám, že se budete při úpravách a opravách hlásit jako uživatel root.

A nyní, vážení, pojďme na to.

Obecné nastavení systému

Několik slov o zabezpečení

Ještě před spuštěním vašeho systému byste se měli rozhodnout, jakou úroveň zabezpečení chcete implementovat. Než se rozhodnete, co chcete dělat, nepřipojujte váš počítač do sítě.

Zabezpečení je obrovským tématem, které přesahuje toto HOWTO. Dvěma dobrými výchozími body jsou příručka správce zabezpečení systému Linux (Linux Security Administrator's Guide) na adrese http://www.securityportal.com/lasg a příručka zabezpečení systému Linux (Linux Security Guide) na adrese http://nic.com/~dave/SecurityAdminGuide/index.html. Měli byste zvážit alespoň následující věci: používání stínových hesel (Shadow Password HOWTO), omezení přístupu k tomuto počítači ze sítě (část Omezování přístupu ze sítě), používání Secure Shell (http://www.openssh.org) nebo zabezpečené vzdálené heslo (Secure Remote Password; http://srp.stanford.edu/srp/). Hodně štěstí.

Začněte poznámkami!

Pro udržování vaší instalace je *nezbytné*, abyste přesně věděli, co se stalo ve vašem počítači, které balíčky jste kdy nainstalovali, co jste odstranili nebo modifikovali atd. Takže první věc, co uděláte, než si začnete hrát s vaším počítačem, bude vytvoření „deníku". Do něj si budete poznamenávat *všechny* změny, které provedete jako root; já mám také svůj deník, do kterého si poznamenávám všechny úpravy systémových souborů, doinstalované balíčky .rpm a soubory .tar.gz, které jsem nainstaloval. Optimálně, díky možnosti zpětného procházení vašich změn, byste měli být schopni obnovit původní instalaci.

Dělejte si záložní kopie systémových souborů, se kterými jste něco dělali. Přes to všechno používejte RCS. Pak budete schopni vrátit všechny změny. Nikdy nepracujte jako root bez toho, abyste si poznamenávali změny!

Klávesnice

Pokud jste tento krok vynechali při instalaci nebo jste vyměnili vaši klávesnici, budete muset:

- najít příslušnou tabulku kláves nacházející se v /usr/lib/kbd/keymaps/i386; například qwuerty/it-latin1.kmap.gz podporuje italskou klávesnici;
- upravit soubor /etc/sysconfig/keyboard tak, aby obsahoval: `KEYTABLE="it-latin1"`;
- nastavit rychlost opakování a prodlevy klávesnice přidáním tohoto řádku do /etc/rc.d/rc.sysinit:

```
/sbin/kbdrate -s -r 16 -d 500  # nebo jak budete chtít
```

Pro načtení tabulky kláves spusťte

```
/etc/rc.d/init.d/keytable start
```

Dalšími speciálními klávesami se budeme zabývat v následujících částech. Pro implicitní zapnutí NumLock přidejte tyto řádky do /etc/rc.d/rc.sysinit:

```
for tty in /dev/tty[1-9]*; do
  setleds -D +num < $tty
done
```

Normálně konzola systému Linux nerozlišuje mezi šipkami a šipkami s klávesou Shift, ale některé aplikace (jmenovitě editor Jed) ano. Normálně jsou tyto vazby kláves k dispozici pouze v programu xterm. Následující mapa kláves, kterou můžete načíst při spuštění, je velmi užitečná:

```
# načtěte tuto mapu kláves pomocí: loadkey shift.map
# Shift + Up
shift keycode 103 = F100
string F100 = "\033[a"
# Shift + Left
shift keycode 106 = F101
string F101 = "\033[c"
# Shift + Right
shift keycode 105 = F102
string F102 = "\033[d"
# Shift + Down
shift keycode 108 = F103
string F103 = "\033[b"
# Ctrl + Ins
control keycode 110 = F104
string F104="\033[2^"
# Shift + Ins
shift keycode 110 = F105
string F105="\033[2$"
# Shift + PgUp
shift keycode 104 = F106
string F106 = "\033[5$"
# Shift + PgDn
shift keycode 109 = F107
string F107 = "\033[6$"
# Shift + Home
shift keycode 102 = F108
string F108 = "\033[1$"
# Shift + End
shift keycode 107 = F109
string F109 = "\033[4$"
# Shift + Del
shift keycode 111 = F110
string F110 = "\033[3$"
# Ctrl + Del
control keycode 111 = F111
string F111 = "\033[3^"
```

Spouštěcí a záchranná disketa

Vytvořte si spouštěcí diskety pro váš nově nainstalovaný systém. Vaše distribuce může obsahovat příkaz pro vytváření těchto disket (řekněme, `mkbootdisk` nebo něco podobného); pokud ne, tyto příkazy to udělají:

```
#~ dd if=/boot/vmlinuz-2.0.36-0.7 of=/dev/fd0  # použít obraz jádra
#~ rdev /dev/fd0 /dev/hda2                     # váš kořenový oddíl
```

Také mějte připravené záchranné diskety. Na ftp://ibiblio.org/pub/Linux/system/recovery jsou široké možnosti záchranných disket; pokud nevíte, kterou zvolit, doporučuji vyzkoušet projekt Tomsrtbt, jehož domovská stránka je na adrese http://www.toms.net/rb. Je velmi povedený, ale na první pohled to vypadá, že některé utility chybí; například zde není `ftp`, ale místo něj je tu `nc` (netcat). Podívejte se do jeho dokumentace.

Jádro

Podle mého názoru první věcí, kterou je dále potřeba udělat, je sestavit jádro, které se nejlépe hodí k vašemu systému. Je to velmi jednoduché, ale v každém případě se podívejte do souboru `README` v /usr/src/linux/ nebo do Kernel HOWTO. Poznámky:

- zvažte pečlivě vaše potřeby. Volba konfigurace jádra, použití záplat a jeho zkompilování jednou pro vždy je produktivnější, než překonfigurovávání a kompilace každý měsíc; zejména, když váš Linux pracuje jako server. Nezapomeňte vložit podporu pro všechen hardware, který byste v budoucnu pravděpodobně přidávali (např. SCSI, Zip, síťové karty atd.); použití modulů je obvykle nejlepší volbou;
- uživatelé notebooků: pokud plánujete používat PCMCIA modem/fax, nezapomeňte zkompilovat podporu sériových portů *do jádra*. Nekompilujte jej jako modul nebo váš modem PCMCIA nebude pracovat;
- zakompilujte všechno, co budete potřebovat; tj. nezapomeňte na moduly pcmcia nebo ovladače zvuku ALSA;
- pro ušetření času potřebného pro další překonfigurování a nové zkompilování jádra je dobré ukládat vaši konfiguraci do souboru a uchovávat ji na bezpečném místě. Pokud inovujete jádro a použijete „make oldconfig“, bude vzat váš starý konfigurační soubor a na nezahrnuté funkce budete dotázáni, zda mají být zahrnuty, což se projeví v novém, aktualizovaném konfiguračním souboru.

Výkon pevného disku

Výkon vašeho disku (E)IDE může být hodně vylepšen *opatrným* použitím `hdparm`(8). Pokud jej vaše distribuce Linuxu neobsahuje, najdete jej na adrese ftp://ibiblio.org/pub/Linux/system/hardware; hledejte soubor s názvem hdparm-X.Y.tar.gz.

Jelikož mnoho detailů závisí na vašem pevném disku a jeho řadiči, nemohu vám dát obecný recept. Riskujete, že zničíte váš systém souborů, takže si *přečtěte pozorně manuálové stránky*, než některé z těchto voleb použijete. V tom nejjednodušším případě můžete přidat následující řádek do /etc/rc.d/rc.sysinit:

```
/sbin/hdparm -c1 /dev/hda  # je předpokládána první jednotka IDE
```

což zapíná podporu 32bitového vstupu a výstupu (E)IDE. O volbě „-m“ mi psal Mark Lord, autor `hdparm`:

*(...) pokud váš systém používá starší komponenty [< 1997], bude to v pořádku. Se staršími *by mohl* být problém (je to nepravděpodobné). Skutečně špatné čipy byly CMD0646 a RZ1000, používané *hodně* na 486 a (starších) základních deskách 586 před 2-3 lety.*

V novějších počítačích by toto nastavení mělo fungovat správně:

```
/sbin/hdparm -c1 -A1 -m16 -d1 /dev/hda
```

Jednotka Zip pro paralelní port

Jádra obsahují ovladač pro starší (ppa) i novější (imm) jednoty Zip. Pokud překompilujete jádro, ujistěte se, že je zapnuta podpora pro SCSI a disky SCSI. Pamatujte si, že se mohou vyskytnout konflikty mezi tiskárnou a jednotkou Zip na stejném paralelním portu, takže bude lepší použít moduly jádra.

Disky Zip jsou prodávány předformátované na oddíl /dev/sda4. Pro zapnutí jednotky Zip přidejte toto do /etc/rc.d/rc.sysinit:

```
# Zapnutí jednotky Zip
/sbin/modprobe ppa  # imm pro novější modely
```

Disky Zip mohou být připojeny pomocí /etc/fstab jak je uvedeno níže, nebo pomocí Mtools přidáním tohoto řádku do vašeho /etc/mtools.conf:

```
drive z: file="/dev/sda4" exclusive
```

kromě toho, vám příkaz `mzip` umožňuje vysunutí, dotaz na stav, zápis a ochranu disků Zip heslem; podrobnosti najdete v manuálových stránkách pomocí `man mzip`. Domovská stránka Mtools je na adrese http://linux.wauug.org/pub/knaff/mtools.

Ovladače zařízení

Zařízení v `/dev` (nebo spíše odkazy na aktuální ovladače zařízení) mohou chybět. Zkontrolujte, čemu odpovídají vaše zařízení pro myš, modem a jednotku CD-ROM, pak udělejte následující:

```
~# cd /dev
/dev# ln -s ttyS0 mouse; ln -s ttyS1 modem; ln -s hdb cdrom; ln -s sda4 zip
```

Ve většině notebooků je ovladač myši /dev/psaux: vezměte to v úvahu při konfigurování X11. Pokud budete chtít, proveďte `chmod 666` pro tato zařízení, abyste je zpřístupnili všem uživatelům.

Zvuková karta

Mé stolní PC je osazeno starým Sound Blaster 16; dokonce i když máte něco jiného, můžete použít toto jako vodítko.

Zkompiloval jsem podporu pro zvukovou kartu jako modul (`sb.o`). Pak to dám do /etc/modules.conf:

```
options sb io=0x220 irq=5 dma=1 dma16=5 mpu_io=0x330
alias sound sb
```

Pro zapnutí zvuku se ujistěte, že je spouštěno `modprobe sound` v /etc/rc.d/rc.sysinit. Eventuálně si můžete stáhnout nástroj `sndconfig` ze serveru RedHat.

Kromě standardních ovladačů jádra pro zvuk jsou svělou volbou ovladače Alsa (http://www.alsa-project.org). Překvapivě, jsou zvukové kanály implicitně vypnuté. Budete muset použít `aumix` a tento /etc/aumixrc pro nastavení hlasitosti na 100 %:

```
vol:100:100:P
synth:100:100:P
pcm:100:100:P
line:100:100:P
mic:100:100:R
cd:100:100:P
```

Zprávy při přihlášení

Pokud chcete upravit zprávy při přihlášení, zkontrolujte, zda váš /etc/rc.d/rc.local přepisuje /etc/issue a /etc/motd (RedHat to dělá). Pokud ano, chopte se vašeho editoru.

Pokud byste chtěli barevnou zprávu při přihlášení, můžete upravit váš `rc.local` vložením řádků jako jsou tyto:

```
# místo ^[ vložte skutečný znak escape. To uděláte:
# emacs: ^Q ESC   vi: ^V ESC   joe:  ' 0 2 7   jed: ' ESC
ESC="^["  # skutečný znak escape
BLUE="$ESC[44;37m"
NORMAL="$ESC[40;37m"
CLEAR="$ESC[H$ESC[J"

> /etc/issue
echo "$CLEAR" >> /etc/issue
echo "$BLUE   Welcome to MyServer (192.168.1.1)   " >> /etc/issue
echo "$NORMAL " >> /etc/issue
echo "" >> /etc/issue
```

Název hostitele

Spuštění příkazu `hostname new_host_name` nemusí stačit. Pro obejití hrozného uzamčení `sendmail` proveďte následující kroky (platné pouze pro samostatné počítače):

- upravte /etc/sysconfig/network a zde změňte název hostitele (např. `new_host_name.your_domain`);
- stejně upravte /etc/HOSTNAME;
- přidejte nový název hostitele do řádku /etc/hosts:

```
127.0.0.1       localhost  new_host_name.your_domain
```

Myš

Služby pro myš `gpm` jsou užitečné pro provádění vyjmutí a vložení v režimu tty a pro použití myši v některých aplikacích. Ujistěte se, že máte soubor /etc/sysconfig/mouse a že obsahuje:

```
MOUSETYPE="Microsoft"
XEMU3=yes
```

Kromě toho musíte mít soubor /etc/rc.d/init.d/gpm, do kterého přidáte parametry pro příkazový řádek. Můj obsahuje:

```
...
       daemon gpm -t $MOUSETYPE -d 2 -a 5 -B 132 # dvoutlačítková myš
...
```

Samozřejmě zajistěte, aby šlo o správnou konfiguraci pro váš druh myši. Ve většině notebooků je `MOUSETYPE „PS/2“`.

Pokud byste chtěli používat nabídky v konzole pomocí klávesy Ctrl, nastavte `gpm-root`. Upravte výchozí konfiguraci v /etc/gpm-root.conf, pak spusťte `gpm-root` v /etc/rc.d/rc.local.

Připojovací body

Je šikovné mít připojovací body pro disketovou jednotku, další zařízení a adresáře NFS. Například můžete provést toto:

```
~# cd /mnt; mkdir floppy cdrom win zip server
```

Toto vytváří připojovací body pro disketovou jednotku DOS/Win, jednotku CD-ROM, oddíl pro Windows, jednotku Zip pro paralelní port a adresář NFS.

Nyní upravte soubor /etc/fstab a přidejte následující položky:

```
/dev/fd0        /mnt/floppy     auto            user,noauto 0 1
/dev/cdrom      /mnt/cdrom      iso9660         ro,user,noauto 0 1
/dev/zip        /mnt/zip        vfat            user,noauto,exec 0 1
/dev/hda1       /mnt/win        vfat            user,noauto 0 1
server:/export  /mnt/server     nfs             defaults
```

Samozřejmě musíte použít správná zařízení v prvním poli.

Všimněte si druhu souborového systému „auto“ v prvním řádku; to vám umožňuje připojovat diskety ext2 i vfat (DOS/Windows), ale potřebujete poslední verzi `mount`. Možná vám bude více vyhovovat `mtools`.

Automatické připojování

Pokud nemáte rádi připojování a odpojování věcí, zvažte použití `autofs`(5). Řeknete démonu autofs, co chcete automaticky připojovat a kde začít, souborem /etc/auto.master. Jeho struktura je jednoduchá:

```
/misc   /etc/auto.misc
/mnt    /etc/auto.mnt
```

V tomto příkladu řeknete autofs, že chcete automaticky připojovat média v /misc a /mnt, přičemž jsou připojovací body udány v /etc/auto.misc a /etc/auto.mnt. Příklad /etc/auto.misc:

```
# export NFS
server          -ro                     my.buddy.net:/pub/export
# vyměnitelná média
cdrom           -fstype=iso9660,ro      :/dev/hdb
floppy          -fstype=auto            :/dev/fd0
```

Spusťte automounter. Od této chvíle, kdykoliv se pokusíte o přístup k neexistujícímu připojovacímu bodu /misc/cdrom, bude vytvořen a jednotka CD-ROM bude připojena.

lilo(8) a LOADLIN.EXE

Mnoho uživatelů používá na svém PC systémy Linux i DOS/Windows a chtějí mít možnost volby operačního systému při spouštění počítače; to by mělo být provedeno při instalaci, ale pokud ne, proveďte následující. Předpokládejme, že /dev/hda1 obsahuje DOS/Windows a že /dev/hda2 obsahuje Linux.

```
~# fdisk
Using /dev/hda as default device!

Command (m for help):a
Partition number (1-4): 2

Command (m for help):w
~#
```

Toto udělá oddíl Linux spustitelným. Pak zapište do souboru /etc/lilo.conf toto:

```
boot = /dev/hda2
compact                 # může být v konfliktu s "linear"
delay = 100             # 10 sekund
linear                  # zbaví nás problému "1024. cylindru"
message = /boot/bootmesg.txt  # můžete zadat vlastní
root = current
image = /boot/vmlinuz  # spouštět linux implicitně tím, že je tato položka prv-
ní
  label = linux
  read-only
#  append="mem=128M"   # aby bylo možno použít více než 64M paměti
other = /dev/hda1
  table = /dev/hda
  label = win
```

Nyní spusťte /sbin/lilo a je hotovo. Protože je `lilo` rozhodující částí vaší instalace, velmi doporučuji, abyste si k němu přečetli dokumentaci.

Pro spuštění systému Linux z DOS/Windows bez resetování umístěte `LOADLIN.EXE` do adresáře (v oddílu DOS!) umístěného v cestě DOS; pak zkopírujte vaše jádro do, řekněme, `C:\TEMP\VMLINUZ`. Následující jednoduchý soubor `.BAT` spustí Linux:

```
rem   linux.bat
smartdrv /C
loadlin c:\temp\vmlinuz root=/dev/hda2 ro
```

Pokud používáte Windows 9x, nastavte vlastnosti tohoto `.BAT` tak, aby se spouštěl v režimu MS-DOS.

> **Tip pro zabezpečení:** Před nainstalováním systému Linux je dobré vytvořit záložní kopii vašeho MBR. Připravte si záchrannou disketu Windows a ujistěte se, že obsahuje `FDISK.EXE`. Pro obnovu MBR musíte provést jen
>
> ```
> A:\> fdisk /mbr
> ```

Konfigurace tiskárny

Všechny distribuce, které znám, mají konfigurační nástroj pro nastavování tiskáren (`printtool`, `yast` nebo `magicfilter`); pokud jej nemáte, zde je základní ruční konfigurace.

Předpokládejme, že máte ne-postscriptovou (také „ne pouze pro Windows"!) tiskárnu, kterou chcete používat pro tisk holého textu (např. zdrojových souborů C) a postscriptových souborů pomocí Ghostscript, o kterém předpokládejme, že je již nainstalován.

Nastavování tiskárny se provádí v několika krocích:

- zjistěte které paralelní tiskové zařízení to je: zkuste

  ```
  ~# echo "hello, world" > /dev/lp0
  ~# echo "hello, world" > /dev/lp1
  ```

  a podívejte se, které pracuje.

- Vytvořte dva adresáře pro řazení:

  ```
  ~# cd /var/spool/lpd
  /var/spool/lpd/# mkdir raw; mkdir postscript
  ```

- pokud vaše tiskárna předvádí „efekt schodiště" (většina inkoustových tiskáren to dělá), budete potřebovat filtr. Zkuste vytisknout dva řádky pomocí

  ```
  ~# echo "první řádek" > /dev/lp1 ; echo "druhý řádek" > /dev/lp1
  ```

  pokud je výstup podobný tomuto:

  ```
  první řádek
             druhý řádek
  ```

  pak uložte tento skript jako /var/spool/lpd/raw/filter:

  ```
  #!/bin/sh
  # Tento filtr eliminuje "efekt schodiště"
  awk '{print $0, "\r"}'
  ```

  a změňte jej na spustitelný pomocí `chmod 755 /var/spool/lpd/raw/filter`.

- vytvořte filtr pro emulaci PostScriptu. Vytvořte následující filtr jako /var/spool/lpd/postscript/filter:

  ```
  #!/bin/sh

  DEVICE=djet500
  RESOLUTION=300x300
  PAPERSIZE=a4
  SENDEOF=

  nenscript -TUS -ZB -p- |
  if [ "$DEVICE" = "PostScript" ]; then
    cat -
  else
    gs -q -sDEVICE=$DEVICE \
      -r$RESOLUTION \
      -sPAPERSIZE=$PAPERSIZE \
      -dNOPAUSE \
      -dSAFER \
  ```

```
    -sOutputFile=- -
fi

if [ "$SENDEOF" != "" ]; then
  printf "\004"
fi
```

(v tomto příkladu je předpokládána tiskárna HP DeskJet. Upravte si jej podle vaší tiskárny).

- nakonec přidejte následující položky do /etc/printcap:

```
# /etc/printcap
lp|ps|PS|PostScript|djps:\
       :sd=/var/spool/lpd/postscript:\
       :mx#0:\
       :lp=/dev/lp1:\
       :if=/var/spool/lpd/postscript/filter:\
          :sh:
raw:\
       :sd=/var/spool/lpd/raw:\
       :mx#0:\
       :lp=/dev/lp1:\
       :if=/var/spool/lpd/raw/filter:\
       :sh:
```

Pro složitější nebo exotičtější konfigurace tisku si přečtěte Printing-HOWTO.

Pokud používáte `printtool`, uvědomte si, že GSDEVICE zvolený nástrojem Printtool bude pracovat, ale nemusí být tím nejlepším pro vaši tiskárnu. Můžete zkusit malý podvod se souborem `postscript.cfg`; například jsem změnil GSDEVICE z `cdj500` na `djet500` a nyní je můj tisk rychlejší.

SVGATextMode

Tento nástroj, který je k dispozici na ftp://tsx-11.mit.edu/pub/linux/sources/sbin, je užitečný pro změny rozlišení obrazovky konzoly, písma a tvaru kurzoru. Uživatelé, jejichž jazyk obsahuje znaky s diakritikou, je budou moci používat v aplikacích konzoly, zatímco uživatelé notebooků mohou změnit tvar kurzoru, aby byl viditelnější.

Upravte /etc/TextConfig nebo /etc/TextMode, kde začneme výchozí definicí VGA. Evropané by měli být šťastni za sekci „LoadFont“:

```
Option "LoadFont"
FontProg "/usr/bin/setfont"
FontPath "/usr/lib/kbd/consolefonts"
FontSelect "lat1u-16.psf"   8x16 9x16 8x15 9x15
FontSelect "lat1u-14.psf"   8x14 9x14 8x13 9x13
FontSelect "lat1u-12.psf"   8x12 9x12 8x11 9x11
FontSelect "lat1u-08.psf"   8x8  9x8  8x7  9x7
```

Jakmile je to hotovo, zkuste vaši konfiguraci s příkazem jako např. `SVGATextMode "80x34x9"`, a pokud to vypadá, že všechno funguje správně, odstraňte varování z /etc/TextMode a vložte tento řádek do etc/rc.d/rc.sysinit:

```
# SVGATextMode
/usr/sbin/SVGATextMode "80x34x9"
```

Pamatujte si, že obdélníkový kurzor pracuje pouze v některých režimech; na mém notebooku „80x30x9“.

Obvyklé úlohy správy

Tady začíná legrace. Tato část je zaměřená na síť, i když na vás čeká mnoho jiných úloh.

Sítě jsou široké téma, které zde nelze zcela pokrýt. Podívejte se na NET-HOWTO. Většina distribucí obsahuje dokumentaci k nastavování síťových služeb. Zde se budeme zabývat pouze několika body.

Krátký seznam co-dělat (to-do) pro služby, které byste mohli chtít nainstalovat: cron a načasované úlohy jako kalendář nebo připomínač, Http, Samba, telnet/ssh přístup, anonymní ftp, POP/IMAP server, NFS...

Konfigurace sítě

Pokud vaše síťová karta nebyla rozpoznána při instalaci, nebojte se: ve většině případů je kompatibilní s NE2000 nebo 3c59x. Spusťte příkaz `modprobe ne` nebo `modprobe 3c59x` a podívejte se, zda je načten relevantní modul, a pak přidejte tento řádek do /etc/modules.conf:

```
alias eth0 ne  # or 3c59x
```

Nyní jste připraveni použít `netcfg` nebo podobný nástroj pro konfiguraci sítě. Relevantní soubory jsou /etc/HOSTNAME, /etc/hosts, /etc/resolv.conf, /etc/sysconfig/network a /etc/sysconfig/network-scripts/ifcfg-eth0; služby by měly být spuštěny skriptem v /etc/rc.d/init.d.

Toto je příklad /etc/hosts:

```
127.0.0.1               localhost
192.168.1.1             paleo.eocene.net        paleo
192.168.1.2             nautilus.eocene.net     nautilus
```

Toto je /etc/resolv.conf:

```
search df.unibo.it,eocene.net
nameserver 195.210.91.100
```

Toto je /etc/sysconfig/network (Red Hat):

```
NETWORKING=false
FORWARD_IPV4=true
HOSTNAME=nautilus.eocene.net
DOMAINNAME=eocene.net
```

A konečně, /etc/sysconfig/network-scripts/ifcfg-eth0. Tento je také pouze v distribuci Red Hat; musí být spustitelný.

```
DEVICE=eth0
IPADDR=192.168.1.2
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
ONBOOT=no
```

Ačkoliv může být skutečná metoda spouštění síťových služeb vaší distribuce složitější, následující skript by měl být z počátku dostačující:

```
#!/bin/sh

# net-up.sh: set up network access

DEVICE=eth0
IPADDR=192.168.1.100
NETMASK=255.255.255.0
NETWORK=192.168.1.0
GATEWAY=192.168.1.1

ifconfig $DEVICE $IPADDR netmask $NETMASK up
route add -net $NETWORK netmask $NETMASK $DEVICE
route add default gw $GATEWAY
```

Tento skript je užitečný pro zapínání přístupu k síti, když používáte záchrannou disketu. Samozřejmě vám umožňuje pouze ping, ftp a telnet; nebude spouštět žádného daemona.

Síť pro notebooky

Když zasunete vaši síťovou PCMCIA kartu, spustí se skript /etc/pcmcia/network. Všechno, co potřebujete, je správně nastavený /etc/sysconfig/network-scripts/ifcfg-eth0.

Nastavování sítě ale může být trochu záludnější. Ve skutečnosti musíte zadat správná nastavení pro každou síť, ke které se připojujete, stejně jako nastavení notebooku, když není připojený.

Vytvořil jsem hrubé, ale funkční řešení. Používám můj notebook jako samostatný počítač a k síti se připojuji přes PPP; doma mám adresu IP 192.168.1.2 a na univerzitě adresu IP 137.204.x.y. Takže jsem vytvořil sadu konfiguračních souborů pro každou síť; všechny jsou uloženy v /etc/mobnet. Pro volbu pracovního prostředí používám skript. Toto je například /etc/mobnet/home.cfg:

```
# /etc/mobnet/home.conf

HOSTNAME=nautilus.eocene.net    # kompletní název hostitele
DOMAINNAME=eocene.net           # vaše doména
IPADDR=192.168.1.2
NETMASK=255.255.255.0
NETWORK=192.168.1.0
BROADCAST=192.168.1.255
GATEWAY=192.168.1.1
FORWARD_IPV4=true
NAMESERVER=195.210.91.100       # vyžadováno
SEARCH=df.unibo.it,eocene.net   # volitelně
SERVICES="inet httpd smb sshd"
```

Toto je `mnet` – skript, který používám pro zvolení síťového profilu:

```
#!/bin/sh
# mnet: skript pro nastavení konfigurace "mobilní síť".
# Poslední úprava: 15. července 2000

# spuštění a zastavení služeb
activate_services()
```

```
{
  for service in $(echo $SERVICES) ; do
    [ -x /etc/rc.d/init.d/$service ] && /etc/rc.d/init.d/$service $1
  done
}

# využití
if [ $# = 0 ] ; then
  echo "Použití: mnet <config name>"
  echo "Příklad: mnet office"
  exit 1
fi

# kontrola zda konfigurace existuje
if [ ! -e /etc/mobnet/$1.conf ]; then
  echo "Tato konfigurace neexistuje."
  exit 1
fi

# načtení konfigurace
. /etc/mobnet/$1.conf

# nastavení názvu hostitele
echo $HOSTNAME > /etc/HOSTNAME
/bin/hostname $HOSTNAME

# nastavení názvového serveru(ů)
cat <<EOF > /etc/resolv.conf
# /etc/resolv.conf
search $SEARCH
nameserver $NAMESERVER
EOF

# zastavení předchozích služeb, pokud byly spuštěné
if [ -f /etc/mobnet/services.prev ]; then
  NEWSERVICES=$SERVICES
  . /etc/mobnet/services.prev
  activate_services stop
  SERVICES=$NEWSERVICES
fi

if [ $1 != "none" ]; then
# nastavení parametrů sítě
  cat <<EOF > /etc/sysconfig/network
  NETWORKING=yes
  FORWARD_IPV4=true
  HOSTNAME=$HOSTNAME
  DOMAINNAME=$DOMAINNAME
  GATEWAY=$GATEWAY
  GATEWAYDEV=eth0
EOF

  cat <<EOF > /etc/sysconfig/network-scripts/ifcfg-eth0
```

```
  DEVICE=eth0
  IPADDR=$IPADDR
  NETMASK=$NETMASK
  NETWORK=$NETWORK
  BROADCAST=$BROADCAST
  ONBOOT=no
EOF
  /bin/chmod +x /etc/sysconfig/network-scripts/ifcfg-eth0

  # zkopírování dalších konfiguračních souborů
  /bin/cp -f /etc/mobnet/hosts.$1       /etc/hosts
  /bin/cp -f /etc/mobnet/smb.conf.$1    /etc/smb.conf

  echo -n "Vložte síťovou PCMCIA kartu a stiskněte <enter>: "
  read

  # OK, nyní spustit služby
  activate_services start
  echo "SERVICES=\"$SERVICES\"" > /etc/mobnet/services.prev

else # it's not "none"

  cat <<EOF > /etc/sysconfig/network
  NETWORKING=false
  FORWARD_IPV4=false
  HOSTNAME=$HOSTNAME
  DOMAINNAME=$DOMAINNAME
EOF
  /bin/rm -f /etc/sysconfig/network-scripts/ifcfg-eth0*
  /sbin/ifconfig eth0 down
  echo "SERVICES=$SERVICES" > /etc/mobnet/services.prev
  echo "Nyní můžete vyjmout kartu PC."
  exit 0

fi

# konec mnet.
```

Jak jsem řekl, je neučesaný a nedodělaný: na síti mohou záviset další soubory, jako např. /etc/fstab, /etc/exports a /etc/printcap. Přemýšlejte o síťových tiskárnách a sdílených adresářích NFS. Volně si toto řešení přizpůsobte vašim potřebám.

Sdílení Internetu

Je to jedna z nejužitečnějších úloh serveru se systémem Linux. V tuto chvíli má většina jader firewall, maškarádování a předávání implicitně vypnuté; pokud jste na pochybách, podívejte se IP-Masquerade mini-HOWTO, kde se dozvíte, jak je zapnout. Pak nainstalujte ipfwadm (jádra 2.0.x; http://www.xos.nl/linux/ipfwadm/) nebo ipchains (jádra 2.2.x; http://www.adelaide.net.au/~rustcorp/ipfwchains/ipfwchains.html). Nezapomeňte zapnout moduly jádra pro služby, které potřebujete, např. pro ftp přidáte tento řádek do /etc/rc.d/rc.sysconfig:

```
/sbin/modprobe ip_masq_ftp
```

Další moduly jsou obvykle v /lib/modules/KERNEL-VERSION/ipv4.

Zapnutí maskování IP pro další počítače ve vaší místní síti je velmi jednoduché. Nejprve zkontrolujte skripty pro inicializaci sítě (měly by být v /etc/sysconfig/network) a podívejte se, zda obsahují řádek `FORWARD_IPV4=true`. Používá se pro nastavení /proc/sys/net/ipv4/ip_forward na 1 při startu síťového podsystému.

Přidejte tyto řádky do /etc/rc.d/rc.sysinit:

```
# implicitně: pakety se nemohou přeposílat ven
/sbin/ipfwadm -F -p deny
# umožní všem počítačům v místní síti přístup k internetu
/sbin/ipfwadm -F -a m -S 192.168.1.0/24 -D 0.0.0.0/0
# eventuálně mohou být povoleny pouze tyto dva počítače
# /sbin/ipfwadm -F -a m -S 192.168.1.100/24 -D 0.0.0.0/0
# /sbin/ipfwadm -F -a m -S 192.168.1.101/24 -D 0.0.0.0/0
```

Pokud používáte jádro série 2.2.x, použijte `ipfwadm-wrapper` namísto `ipfwadm` pro rychlé spuštění. Více informací najdete na adrese http://ipmasq.cjb.net.

(Pozn. odb. korektora: u novějších jader použijte spíše ipehains.)

Nyní budete chtít udělat něco, abyste počítačům klientů povolili vytáčení ISP; já používám Mserver (http://cpwright.villagenet.com/mserver/). V etc/mserver.conf upravujte pouze položky „checkhost“, „shadow“ a „cname“. Pak nadefinujte vaše připojení. Samozřejmě nainstalujte na klientské počítače odpovídajícího klienta.

Omezování přístupu ze sítě

Předpokládejme, že se připojujete k Internetu přes PPP. Jakmile jste připojeni, může být váš počítač zranitelný útoky. Vložte do /etc/hosts.allow:

```
# povolit přístup pouze pro localhost
ALL: 127.
```

a do /etc/hosts.deny:

```
# odepření přístupu všem
ALL: ALL
```

Pokud jste v síti s přímým přístupem k Internetu, měli byste z bezpečnostních důvodů vypnout finger, telnet a možná další služby; místo služby telnet použijte `ssh`. Je potřeba upravit /etc/inetd.conf. Eventuálně můžete omezit přístup ze sítě přidáním do /etc/hosts.allow:

```
in.telnetd: 192.168.1., .another.trusted.network
in.ftpd: 192.168.1., .another.trusted.network
```

a tohoto do /etc/hosts.deny:

```
in.telnetd: ALL
in.ftpd: ALL
```

Exporty NFS

Časté je exportování domovských adresářů na server; problém nastává, pokud UID a GID uživatele nejsou pro různé počítače konzistentní. Pokud uživatel `guido' má UID/GID = 500 na počítači `server` a UID/GID = 512 na počítači `client`, je vhodnou konfigurací toto:

```
# /etc/exports
/tmp            my.client.machine(rw)
/home/guido     my.client.machine(rw,all_squash,anonuid=512,anongid=512)
```

Samba

Je to skoro triviální, ale vždycky je tu potřeba něco malého udělat. Pokud se chcete připojovat ke klientům Windows 98/NT, přečetli jste si dokumentaci a, případně, zapnuli čistě textová hesla? Distribuce obsahuje soubory .reg pro Win9x/NT/2000; pokud se vaši klienti nemohou připojit k serveru se systémem Linux, načtěte je na každém klientovi.

Samba obsahuje docela kompletní příklad /etc/smb.conf, ale kupodivu postrádá část, která by vám ukázala, jak připojit nebo odpojit vyměnitelná média. Musíte použít klauzule `preexec` a `postexec`:

```
[cdrom]
  comment = CD-ROM
  path = /mnt/cdrom
  public = yes
  read only = yes
; možná budete muset použít "root preexec/postexec"
  preexec = mount /mnt/cdrom
  postexec = umount /mnt/cdrom
```

Takže: víte, co je to Swat, že ano? Zapněte jej přidáním tohoto řádku do vašeho /etc/inetd.conf:

```
swat      stream  tcp     nowait.400      root /usr/sbin/swat swat
```

a tohoto do `/etc/services`:

```
swat            901/tcp
```

Restartujte `inetd` s SIGHUP a nasměrujte váš prohlížeč na `http://localhost:901`.

Konfigurace softwaru

Zde jsou konfigurační soubory, které budeme upravovat: /etc/profile /etc/bashrc `.bashrc .bash_profile .bash_logout .inputrc .less .lessrc .xinitrc .fvwmrc .fvwm2rc95 .Xmodmap .Xmodmap.num .Xdefaults .jedrc .abbrevs.sl .joerc .emacs`. Nepřidávejte uživatele, dokud nedokončíte tuto konfiguraci vašeho systému; soubory umístíte do /etc/skel.

bash(1)

Je to nejdůležitější část softwaru po jádře. Pro přizpůsobení chování `bash` jsou zde tyto hlavní soubory:

- /etc/bashrc obsahuje systémové aliasy a funkce;
- /etc/profile obsahuje prostředí systému a programy po spuštění;
- `$HOME/.bashrc` obsahuje aliasy uživatelů a funkce;
- `$HOME/.bash_profile` obsahuje uživatelské prostředí a programy po spuštění;
- `$HOME/.inputrc` obsahuje vazby kláves a další věci.

Níže jsou uvedeny příklady těchto souborů. První je nejdůležitější: /etc/profile. Používá se pro konfiguraci mnoha funkcí systému Linux, jak uvidíte v následujících částech. Hledejte prosím obrácené uvozovky!

```
# /etc/profile

# System wide environment and startup programs
# Functions and aliases go in /etc/bashrc

# This file sets up the following features and programs:
# path, prompts, a few environment variables, colour ls, less,
# rxvt, Backspace key behaviour, xterm title.
#
# Users can override these settings and/or add others in their
# $HOME/.bash_profile

# first: root or normal user? Set PATH and umask accordingly. Note that the
# PATH is normally set by login(1), but what if you access the machine
# via ssh?

if [ $(id -gn) = $(id -un) -a $(id -u) -gt 14 ]; then
  umask 002  # normal user
  PATH="/usr/local/bin:/bin:/usr/bin:."
else
  umask 022  # root
  PATH="/sbin:/bin:/usr/sbin:/usr/bin"
fi

# Now extend the PATH.
PATH="$PATH:/usr/X11R6/bin:$HOME/bin:." # !!! Beware of ./ !!!

# notify the user: login or non-login shell. If login, the prompt is
# blue; otherwise, magenta. Root's prompt is red.
# See the Colour-ls mini HOWTO for an explanation of the escape codes.
USER=$(whoami)
if [ $LOGNAME = $USER ] ; then
  COLOUR=44  # blue
else
  COLOUR=45  # magenta
fi

if [ $USER = 'root' ] ; then
  COLOUR=41  # red
  PATH="$PATH:/usr/local/bin" # my choice
fi
```

```
ESC="\033"
PROMPT='\h'    # hostname
STYLE='m'      # plain
# PROMPT='\u'  # username
# STYLE=';1m'  # bold
PS1="\[$ESC[$COLOUR;37$STYLE\]$PROMPT:\[$ESC[37;40$STYLE\]\w\\$ "
PS2="> "

# Ulimits: no core dumps, max file size 200 Mb.
ulimit -c 0 -f 200000

# a few variables
USER=$(id -un)
LOGNAME=$USER
MAIL="/var/spool/mail/$USER"  # sendmail, postfix, smail
# MAIL="$HOME/Mailbox"        # qmail
NNTPSERVER=news.myisp.it      # put your own here
VISUAL=jed
EDITOR=jed
HOSTNAME=$(/bin/hostname)
HISTSIZE=1000
HISTFILESIZE=1000
export PATH PS1 PS2 USER LOGNAME MAIL NNTPSERVER
export VISUAL EDITOR HOSTNAME HISTSIZE HISTFILESIZE

# enable colour ls
eval $(dircolors /etc/DIR_COLORS -b)
export LS_OPTIONS='-s -F -T 0 --color=yes'

# customize less
LESS='-M-Q'
LESSEDIT="%E ?lt+%lt. %f"
LESSOPEN="| lesspipe.sh %s"
LESSCHARDEF=8bcccbcc13b.4b95.33b. # show colours in ls -l | less
# LESSCHARSET=latin1
PAGER=less
export LESS LESSEDIT LESSOPEN VISUAL LESSCHARDEF PAGER

# you might need this to fix the backspace key in rxvt/xterm
stty erase ^H  # alternative: ^?

# set xterm title: full path
case $TERM in
  xterm*|rxvt)
    PROMPT_COMMAND='echo -ne "\033]0;${USER}@${HOSTNAME}: ${PWD}\007"'
    ;;
esac

for i in /etc/profile.d/*.sh ; do
  if [ -x $i ]; then
    . $i # beware - variables and aliases might get overridden!
  fi
done
```

```
# call fortune, if available
if [ -x /usr/games/fortune ] ; then
echo ; /usr/games/fortune ; echo
fi
```

Toto je příklad /etc/bashrc:

```
# /etc/bashrc

# System wide functions and aliases
# Environment stuff goes in /etc/profile
# Insert PS1 definitions here if you experience problems.

export CDPATH="$CDPATH:~"

# common aliases
alias cp='cp -i'
alias l=less
alias ls="ls $LS_OPTIONS"
alias mv='mv -i'
alias rm='rm -i'
alias rmbk='/bin/rm -f .*~ *~ *aux *bak *log *tmp 2> /dev/null'
alias u='cd ..'
alias which="type -path"
alias x=startx

# A few useful functions
c ()    # cd to the new directory and list its contents
{
  cd $1 ; ls
}

inst()  # Install a .tar.gz archive in current directory
{
  if [ $# != 0 ]; then tar zxvf $1; fi
}

cz()    # List the contents of a .zip archive
{
  if [ $# != 0 ]; then unzip -l $*; fi
}

ctgz()  # List the contents of a .tar.gz archive
{
  for file in $* ; do
    tar ztf ${file}
  done
}

tgz()   # Create a .tgz archive a la zip.
{
  if [ $# != 0 ]; then
    name=$1.tar; shift; tar -rvf ${name} $* ; gzip -9 ${name}
```

```
  fi
}

crpm()  # list information on an .rpm file
{
  if [ $# != 0 ]; then rpm -qil $1 | less; fi
}
```

Toto je příklad `.bashrc`:

```
# $HOME/.bashrc
# Source global definitions

if [ -f /etc/bashrc ]; then
  . /etc/bashrc
fi

# this is needed to notify the user that they are in non-login shell
if [ "$GET_PS1" = "" ] ; then
  COLOUR=45; ESC="\033"; STYLE=';1m';  # STYLE='m'
  USER=$(whoami)
  export PS1="\[$ESC[$COLOUR;37$STYLE\]$USER:\[$ESC[37;40$STYLE\]\w\\$ "
fi

# personal aliases
alias backup='tar -Mcvf /dev/fd0'
alias dial='eznet up myisp'
alias f='cd ~/fortran'
alias hangup='eznet down'
alias lyx='lyx -width 580 -height 450'
alias restore='tar -M -xpvf /dev/fd0'

# personal functions
xj()    # Launch xjed and a file in background
{
  xjed $1 &
}
```

Toto je příklad `.bash_profile`:

```
# $HOME/.bash_profile

# User specific environment and startup programs
# This file contains user-defined settings that override
# those in /etc/profile

# Get user aliases and functions
if [ -f ~/.bashrc ]; then
  GET_PS1="NO"  # don't change the prompt colour
  . ~/.bashrc
fi

# set a few 'default' directories
export CDPATH="$CDPATH:$HOME:$HOME/text:$HOME/text/geology"
```

Toto je příklad `.inputrc`:

```
# $HOME/.inputrc

# key bindings
"\e[1~": beginning-of-line
"\e[3~": delete-char
"\e[4~": end-of-line
# (F1 .. F5) are "\e[[A" ... "\e[[E"
"\e[[A": "info \C-m"

set bell-style visible          # please don't beep
set meta-flag On                # allow 8-bit input (i.e, accented letters)
set convert-meta Off            # don't strip 8-bit characters
set output-meta On              # display 8-bit characters correctly
set horizontal-scroll-mode On   # scroll long command lines
set show-all-if-ambiguous On    # after TAB is pressed
```

Aby klávesy backspace a delete pracovaly správně v `xterm` a dalších aplikacích X11, je potřeba také následující:

- vložte toto do vašeho `.xinitrc`:

  ```
  usermodmap=$HOME/.Xmodmap
  xmodmap $usermodmap
  ```

- pak bude váš `.Xmodmap` obsahovat:

  ```
  keycode 22 = BackSpace
  keycode 107 = Delete
  ```

  to opraví konzolu. Pro opravu `xterm`:

- vložte toto do vašeho `.Xdefaults`:

  ```
  xterm*VT100.Translations: #override <Key>BackSpace: string(0x7F)\n\
          <Key>Delete:        string(0x1b) string("[3~")\n\
          <Key>Home:          string(0x1b) string("[1~")\n\
          <Key>End:           string(0x1b) string("[4~")\n\
          Ctrl<Key>Prior:     string(0x1b) string("[40~")\n\
          Ctrl<Key>Next:      string(0x1b) string("[41~")

  nxterm*VT100.Translations: #override <Key>BackSpace: string(0x7F)\n\
          <Key>Delete:        string(0x1b) string("[3~")\n\
          <Key>Home:          string(0x1b) string("[1~")\n\
       <Key>End:           string(0x1b) string("[4~")\n\
          Ctrl<Key>Prior:     string(0x1b) string("[40~")\n\
          Ctrl<Key>Next:      string(0x1b) string("[41~")
  ```

`rxvt` je trochu komplikovanější, jelikož některé volby zadané při kompilaci ovlivňují jeho chování. Podívejte se na /etc/profile nahoře.

Více informací o `bash`(1) a `readline`(3) najdete v manuálových stránkách.

Nečekejte, že bude každá aplikace pracovat správně! Pokud spustíte `joe` v `xterm`, některé klávesy například nebudou pracovat; to samé platí pro některé verze `rxvt`.

I18n

(Tato část se netýká anglicky mluvících uživatelů)

Je to známo také jako „lokalizace“. Uf! Toto slovo znamená „přizpůsobení systému Linux vašim místním konvencím: jazyku, formátu data, měně atd“'.

Ačkoliv má Red Hat svou vlastní metodu nastavování i18n (/etc/sysconfig/i18n), možná budete chtít zapnout váš jazyk pouze v některých případech. Třeba budu chtít zapnout i18n v `kdm` (pomocí `kdmconfig`) a `xfce`, ale když pracuji v konzole nebo xterm, chci číst zprávy v angličtině.

Podívejte se na tyto řádky:

```
LANG=it # zvolte svůj jazyk: fr, de, es, ...
LANGUAGE=it
LC_ALL=it
export LANG LANGUAGE LC_ALL
```

Pokud je přidáte do vašeho `.xinitrc` nebo `.xsession` před řádek, kde se spouští správce oken, obdržíte lokalizované zprávy – včetně těch v xterm spuštěném ze správce oken. Ale pokud byste raději dostávali zprávy v angličtině, nastavte jazyk na „en“ a vložte ty samé řádky do `.bash_profile`.

ls(1)

`ls` může zobrazovat výpisy adresářů pomocí barev pro odlišení různých druhů souborů. Pro zapnutí této funkce potřebujete právě řádky v /etc/profile uvedené výše. Toto samozřejmě nebude pracovat se staršími verzemi `rxvt`; místo toho použijte některou verzi `xterm`. Vypadá to, jako by některé staré `rxvt` měly chybu, která jim brání v některých případech v odvozování prostředí.

less(1)

S tímto excelentním pagerem můžete procházet nejen čistě textové soubory, ale také komprimované archívy gzip, tar a zip, manuálové stránky a další. Jeho konfigurace zahrnuje několik kroků:

- pro jeho použití s klávesami pro pohyb mějte tento čistě v ASCII soubor `.lesskey` ve vašem domovském adresáři:

  ```
  ^[[A   back-line
  ^[[B   forw-line
  ^[[C   right-scroll
  ^[[D   left-scroll
  ^[OA   back-line
  ^[OB   forw-line
  ^[OC   right-scroll
  ^[OD   left-scroll
  ^[[6~  forw-scroll
  ^[[5~  back-scroll
  ^[[1~  goto-line
  ^[[4~  goto-end
  ^[[7~  goto-line
  ^[[8~  goto-end
  ```

  pak spusťte příkaz `lesskey` (Jsou to escape sekvence pro terminály vt100). Toto vytvoří binární soubor `.less` obsahující vazby kláves.

- uložte následující soubor jako /usr/bin/lesspipe.sh:

```
#!/bin/sh
# This is a preprocessor for 'less'.  It is used when this environment
# variable is set:   LESSOPEN="|lesspipe.sh %s"

lesspipe() {
  case "$1" in
  *.tar) tar tf $1 2>/dev/null ;; # View contents of .tar and .tgz files
  *.tgz|*.tar.gz|*.tar.Z|*.tar.z) tar ztf $1 2>/dev/null ;;
  *.Z|*.z|*.gz) gzip -dc $1  2>/dev/null ;; # View compressed files co-
rrectly
  *.bz2) bzip2 -dc $1  2>/dev/null ;;
  *.zip) unzip -l $1 2>/dev/null ;; # View archives
  *.arj) unarj -l $1 2>/dev/null ;;
  *.rpm) rpm -qpil $1 2>/dev/null ;;
  *.cpio) cpio --list -F $1 2>/dev/null ;;
  *.1|*.2|*.3|*.4|*.5|*.6|*.7|*.8|*.9|*.n|*.l|*.man) FILE='file -L $1'
    FILE='echo $FILE | cut -d ' ' -f 2'
    if [ "$FILE" = "troff" ]; then
      groff -s -p -t -e -Tascii -mandoc $1
    fi ;;
  *) file $1 | grep text > /dev/null ;
    if [ $? = 1 ] ; then # it's not some kind of text
      strings $1
    fi ;;
  esac
}

lesspipe $1
```

pak jej učiňte spustitelným nastavením `chmod 755 lesspipe.sh`.

- umístěte proměnné, které ovlivňují `less` do /etc/profile, jak jste viděli výše.

Editor

Budou zde zmíněny pouze nejpopulárnější.

emacs(1)

Málokdy používám `emacs`, takže mám pro vás jen pár tipů. Některé distribuce `emacs` nemají předkonfigurované barvy a zvýrazňování syntaxe. Umístěte toto do vašeho `.emacs`:

```
(global-font-lock-mode t)
(setq font-lock-maximum-decoration t)
```

Toto pracuje pouze v X11. Kromě toho pro zapnutí znaků s diakritikou přidáte tento řádek:

```
(standard-display-european 1)
```

Nechám na vás pročtení veškeré dokumentace k `emacs` a nalezení toho, jak si jej přizpůsobit vlastním potřebám---také to může zabrat měsíce hackování. Dobrým pomocníkem je generátor Dotfile (viz. Konfigurační souftware).

joe(1)

Některé verze `joe` nepracují v konzole barevně a nefungují v nich některé speciální klávesy. Rychlým a nečistým (a neelegantním) řešením předchozího problému je toto:

```
~$ export TERM=vt100
~$ joe myfile
   (edituje váš soubor)
~$ export TERM=linux
```

Aby fungovaly speciální klávesy, všechno, co potřebujete udělat, je upravit `.joerc`, `.jstarrc` nebo vaši oblíbenou emulaci; můžete začít systémovými konfiguračními soubory v /usr/lib/joe. Hledejte čtvrtou sekci (vazby kláves). Následující umožní Home a End:

```
bol ^[ [ 1 ~    Go to beginning of line
eol ^[ [ 4 ~    Go to end of line
```

Najděte požadované escape sekvence pomocí `cat` následovaným speciálními znaky.

jed(1)

To je můj oblíbený editor: dělá to, co potřebuju, je malý a jednodušeji nastavitelný než `emacs` a celkem dobře emuluje jiné editory. Mnoho uživatelů na mé univerzitě používá `jed` pro emulaci `EDT`, systémového editoru VMS.

Konfigurační soubory editoru `jed` jsou `.jedrc` a /usr/lib/jed/lib/*; dříve uvedený může být přizpůsoben z `jed.rc` v uvedeném adresáři.

- pokud `xjed` zřejmě nerozpoznává klávesu DEL, přidejte nebo zakomentujte tyto řádky ve vašem `.jedrc`:

  ```
  #ifdef XWINDOWS
    x_set_keysym (0xFFFF, 0, "\e[3~");
    setkey (''delete_char_cmd'', "\e[3~");
  #endif
  ```

- aby `jed` emuloval `EDT` (nebo jiné editory), všechno, co musíte udělat, je upravit pár řádek v `.jedrc`. Pokud chcete, aby „+“ na numerické klávesnici odstraňovalo slova místo jednotlivých znaků, přidejte do `.jedrc`:

  ```
  unsetkey("\eOl");
  unsetkey("\eOP\eOl");
  setkey("edt_wdel", "\eOl");
  setkey("edt_uwdel", "\eOP\eOl");
  ```

  za řádek, který obsahuje `() = evalfile("edt")` (nebo něco podobného);

- aby `xjed` používal numerickou klávesnici pro emulaci `EDT`, vložte do `.Xmodmap`:

  ```
  keycode 77  = KP_F1
  keycode 112 = KP_F2
  keycode 63  = KP_F3
  keycode 82  = KP_F4
  keycode 86  = KP_Separator
  ```

- přizpůsobení barev `xjed` se dělá přidáním takovýchto řádků do `.Xdefaults`:

```
xjed*Geometry: 80x32+150+50
xjed*font: 10x20
xjed*background: midnight blue
# and so on...
```

- zkratky funkcí neocenitelně šetří čas. Vytvořte soubor podobný tomu následujícímu jako `$HOME/.abbrevs.sl` (tento název můžete změnit přidáním `variable Abbrev_File = "/usr/lib/jed/abbrev.sl";` do `.jedrc`):

```
create_abbrev_table ("Global", "0-9A-Za-z");
define_abbrev ("Global", "GG", "Guido onzato");
create_abbrev_table ("TeX", "\\A-Za-z0-9");
define_abbrev ("TeX", "\\beq", "\\begin{equation}");
define_abbrev ("TeX", "\\eeq", "\\end{equation}");
% a tak dále...
```

  a napište `ESC x abbrev_mode` pro jeho zapnutí. Pro implicitní zapnutí zkratek přidejte položky jako jsou tyto do vašeho `.jedrc`:

```
define text_mode_hook ()
{
  set_abbrev_mode (1);
}
%
define fortran_hook ()
{
  set_abbrev_mode (1);
  use_abbrev_table ("Fortran");
}
% a tak dále...
```

pine(1)

Upravte globální konfiguraci v /usr/lib/pine.conf a dávejte pozor alespoň na následující pole: `user-domain`, `smtp-server` a `nntp-server`. Všimněte si, že `inbox-path` závisí na vašem MTA: pokud používáte `sendmail` nebo `postfix`, bude zde `/var/spool/mail/$USER`; v Qmail /home/$USER/Mailbox (ale root bude používat /var/qmail/alias/Mailbox.

minicom(1)

Uživatelé nemohou `minicom` používat, dokud root nevytvoří globální konfiguraci. Nezapomeňte ji vytvořit.

efax(1)

Tento balíček je pravděpodobně nejvhodnější pro jednoduché odesílání a přijímání faxů. Budete muset upravit skript /usr/bin/fax nebo (v mandrake) /etc/fax.config; je to jednoduché, ale z několika věcí mě docela rozbolela hlava:

- pro zjištění toho, zda je váš modem třídy 1, 2, nebo 2.0 použijte minicom nebo podobný program pro spuštění příkazu `at+fclass=?`. Odpověď může být `0,1` nebo `2`; 1 a 2 jsou třídy podporované vaším modemem;

- DIALPREFIX: je možné, že jednoduché zadání `T' nebo `P' nebude fungovat v některých zemích – přinejmenším v Itálii. Místo toho zadejte `ATDT' nebo `ATDP';
- INIT a RESET: tyto řetězce obsahují inicializátory `-i' a `-k', které potřebuje `efax`. Pokud chcete přidat příkaz AT, přidejte jej do příslušného řetězce vynecháním `AT' a přeřazením `-i' nebo `-k' před zbytek. Příklad: pro přidání příkazu `ATX3' do INIT připojíte `-iX3'.

Když je to hotovo, je potřeba opravit několik oprávnění, aby mohli i jiní uživatelé než root odesílat a přijímat faxy. Do adresářů /var/lock a /var/spool/fax musí být možno zapisovat. K tomu vytvořte skupinu `faxusers`, přidejte do ní uživatele a pak napište:

```
~# chown root.faxusers /var/lock
~# mkdir /var/spool/fax # if it doesn't exist yet
~# chown root.faxusers /var/spool/fax; chmod g+w /var/spool/fax
```

Jako normální uživatel spustíte před odesíláním faxu `newgrp faxusers`.

Ghostscript

Tento dokonalý nástroj trpí malým problémem. Vzhledem k dobře známé regulaci exportu z USA utilita `pdf2ps` nepracuje se zašifrovanými soubory .pdf. Nevadí: nasměrujte váš prohlížeč na http://www.ozemail.com.au/~geoffk/pdfencrypt, stáhněte soubor `pdf_sec.ps` a nahraďte soubor se stejným názvem, který pochází z distribuce Ghostscript.

TeX a spol.

„Kořenem“ systému TeX je adresář $TEXMF, který je /usr/share/texmf v teTeX; ostatní distribuce se mohou lišit (hledejte ve vašem systému „texmf“).

Rozšiřování $TEXINPUTS

Pro vložení postscriptových prvků nebo souborů TeX, které jsou uloženy v podadresářích, je vhodné rozšířit cestu hledání TeXu tak, aby obsahovala podadresáře. Vložte tento příkaz do vašeho .bash_profile:

```
export TEXINPUTS="$HOME/mylib::./figures"
```

což umožní programu TeX hledání v `$HOME/mylib` *před* implicitními adresáři a v adresáři `./figures` *později*.

Vzorky dělení

Pro nastavení vzorků dělení pro váš jazyk upravte soubor $TEXMF/tex/generic/config/language.dat a pak proveďte:

```
~# texconfig init ; texconfig hyphen
```

I když nepíšete v angličtině, neodstraňujte položku „english“; TeX se bez ní neobejde.

Dvips

Pro úpravu `dvips` je potřeba upravit soubor $TEXMF/dvips/config/config.ps. Buďte si vědomi toho, že pole výchozího rozlišení také ovlivňují chování `xdvi`; pokud se setkáte s otravnými pokusy o vytváření písem pokaždé, když jej spustíte, vložte řádek

```
XDvi*mfmode:
```

Do vašeho `.Xdefaults`. Mělo by to pomoci.

Přidávání balíčků LaTeX

Další balíčky LaTeX jsou k dispozici na nejbližším zrcadle CTAN (Comprehensive TeX Archive Network), např. ftp://ftp.dante.de/pub/tex. Rozbalte balíček do $TEXMF/tex/latex.

Pokud neexistuje soubor .sty, spusťte příkaz `latex newstyle.ins` nebo `latex newstyle.dtx`, abyste jej vytvořili, pak spusťte příkaz `texhash`, aby teTeX rozpoznal nový balíček.

Vyhněte se PPProblémům!

Budu považovat za samozřejmé, že má vaše jádro zakompilovanou podporu PPP a TCP/IP, že loopback je zapnutý a že již máte správně nainstalovaný balíček `pppd` a, pokud chcete, nastavené uid root. Váš ISP samozřejmě musí podporovat PPP.

Existují dva způsoby uvedení PPP do chodu: a) ruční konfigurace, a b) konfigurační program, který to dělá automaticky. Ať už zvolíte cokoliv, připravte si následující informace:

- telefonní číslo vašeho ISP;
- název vašeho ISP, adresu poštovního serveru a adresu serveru s diskusními skupinami;
- doménu vašeho ISP;
- vaše uživatelské jméno a heslo.

Ruční konfigurace je dřina. Jsou to úpravy souborů a vytváření skriptů; není to moc práce, ale je snadné se splést a začátečníci se často bojí. Existuje PPP HOWTO. Eventuálně existují nástroje, které se vás zeptají na výše uvedené údaje a udělají všechnu práci.

Gnome a KDE obsahují, podle pořadí, `gnome-ppp` a `kppp`, pomocí kterých je nastavování jednoduché. Eventuálně vám doporučuji se podívat na pár nástrojů založených na tty, `wvdial` a `eznet`. Zadáte jim telefonní číslo vašeho ISP, vaše jméno a vaše heslo a je to hotovo. Jejich domovské stránky jsou na adresách http://www.worldvisions.ca/wvdial a http://www.hwaci.com/sw/eznet. Oba jsou skvělé, ale doporučuji ten druhý.

Rychlý start pomocí eznet

Nejdříve vytvořte /etc/resolv.conf s tímto:

```
nameserver w.x.y.z
```

kam zadáte adresu jmenného serveru ISP. Pro vytvoření účtu pomocí `eznet` spusťte následující příkaz:

```
#~ eznet add service=YOUR_ISP user=NAME password=PASSWORD phone=PHONE
```

který vytvoří soubor /var/eznet/eznet.conf, jehož vlastníkem je root.root s oprávněním 600; změňte je pomocí chmod na 666 pokud chcete, aby k nim byl přístup. Nyní vytočte číslo vašeho ISP pomocí `eznet up YOUR_ISP`. Pokud modem stále čeká na oznamovací tón a nepřipojí se, pak zkuste tento příkaz:

```
#~ eznet change YOUR_ISP init0=atx3
```

Příkaz pro zavěšení je `eznet down`. To je vše!

Rychlý start pomocí wvdial

Nastavení `wvdial`'s je ještě kratší. Napište `wvdialconf /etc/wvdial.conf`, pak upravte výsledný soubor, aby obsahoval vaše uživatelské jméno, heslo a telefonní číslo. Zkuste `wvdial` a mějte překřížené prsty. Pro zavěšení jej zastavte pomocí Ctrl-C.

Klient POP

Pro načtení vaší pošty ze serveru POP3 potřebujete klienta POP. Většina těchto klientů vyžaduje, abyste měli spuštěný MTA jako `sendmail`, `qmail` nebo `postfix`; to je trochu náročné na počítačích nižší úrovně. Samozřejmě existují klienti, kteří pracují bez MTA. První dobře reprezentuje `fetchmail`; druhé `fetchpop` nebo `frenchie`. Servery: ftp://www.ibiblio.org/pub/Linux/system/mail/pop, http://www.lowcountry.com/~jscottb/tcltk.shtml.

Pro nastavení těchto klientů:

- `fetchpop`: když jej spustíte poprvé, požádá vás o nějaké informace. Odpovězte na otázky a bude nastavený. `fetchpop` musí být použit s přepínačem `-r`, pokud server POP3 vašeho ISP nepodporuje příkaz LAST správně.
- `frenchie`: stejný, upravte /.frenchie/frenchierc;
- `fetchmail`: upravte tento vzorový `.fetchmailrc`:

  ```
  # $HOME/.fetchmailrc
  poll mbox.myisp.com with protocol pop3;
    user john there with password _Loo%ny is john here
  ```

  Jeden uživatel říkal, že přidání „smtphost localhost" na druhý řádek dramaticky zvýšilo výkon. Musíte nastavit oprávnění pro tento soubor pomocí příkazu `chmod 600 .fetchmailrc`, jinak se `fetchmail` odmítne spustit. Tento příklad je jen základní; existují nekonečné možnosti konfigurace. Podívejte se na http://www.ccil.org/~esr/fetchmail.

Základní filtrování pošty

Budete se chtít chránit před nevyžádanou poštou nebo velkými zprávami. Existují dva případy: 1) trvalé připojení k síti a 2) připojení POP. V prvním případě můžete vytvořit soubor `.procmailrc`, zatímco v druhém případě existují nástroje pro kontrolu pošty ještě před jejím stažením.

Velmi jednoduchý `.procmailrc`, který definuje nová pravidla:

```
# $HOME/.procmailrc

MAILDIR=$HOME/mail # make sure it exists

# Store messages directed to the "foo" mailing list to $HOME/mail/foo
:0
* ^To:.*foo
foo

# Discard messages that are not explicitly sent to me or to one of the
# maillling lists I subscribed to.
:0
* !^TO(guido|jed|lugvr|ldp|nobody)
/dev/null

# ditto, for messages larger than 50k.
```

```
:0
* > 50000
/dev/null
```

Pro více příkladů spusťte `man procmailex`.

Uživatelé POP budou chtít použít `poppy`, užitečný skript v jazyce Perl pro kontrolu pošty před jejím stažením. Najdete si jej na http://www.freshmeat. .net/.

Window System (XFree86)

Nastavování X Server

Pojďme na to, už to není tak obtížné, jako to bývalo... Všechny velké distribuce obsahují nástroj pro nastavování X11 (např. `XConfigurator`, `sax`, `XF86Setup` nebo alespoň `xf86config`). Konfigurace X je dnes vlastně automatická, ale z některých videokaret vás může rozbolet hlava.

Nejprve zkontrolujte na serveru XFree86 (http://www.xfree86.org), zda je vaše videokarta podporována. Pokud ano, zkuste tuto proceduru:

- nainstalujte server s normální VGA;
- jděte na ftp://ftp.XFree86.org/pub/XFree86/current/binaries, přejděte do správného podadresáře a stáhněte archívy `X_version_bin.tgz`, `X_version_set.tgz` a všechny servery. Kromě jiných programů první archív obsahuje nejnovější `SuperProbe`;
- rozbalte `X_version_bin.tgz` do dočasného adresáře, přejděte do něj a spusťte `./SuperProbe`. Pokud je vaše videokarta rozpoznána, je šance, že ji budete moci nastavit. Jinak hodně štěstí;
- nainstalujte servery a `X_version_set.tgz` z /usr/X11R6/, pak spusťte `XF86Setup`.

Vždycky mi to fungovalo, ale nemusí to být jednoduché. Pamatujte si prosím, že se většinou X11 nespustí, protože jste zvolili špatné parametry vašeho monitoru! Začněte s konzervativním nastavením, tj. 800x600 a 256 barev, pak ho zvedněte. *Varování*: tyto operace jsou nebezpečné a váš monitor se může poškodit!

Pokud vaše karta není podporována, můžete: 1) čekat na další verzi XFree86, 2) zakoupit komerční server X, 3) zakoupit podporovanou videokartu. *Quartum non datur.*

Numerická klávesnice

Výše jste viděli, jak rozchodit některé speciální klávesy. Vzorový soubor `.Xmodmap` pracuje dobře, pokud chcete použít Xjed, ale dělá numerickou klávesnici nefunkční. Pak budete potřebovat jiný konfigurační soubor, kterému budeme říkat `.Xmodmap.num`:

```
! Definitions can be found in <X11/keysymdef.h>

keycode 77  = Num_Lock
keycode 112 = KP_Divide
keycode 63  = KP_Multiply
keycode 82  = KP_Subtract
keycode 86  = KP_Add
keycode 79  = KP_7
keycode 80  = KP_8
keycode 81  = KP_9
keycode 83  = KP_4
keycode 84  = KP_5
```

```
keycode 85  = KP_6
keycode 87  = KP_1
keycode 88  = KP_2
keycode 89  = KP_3
keycode 90  = KP_0
keycode 91  = KP_Decimal
```

Ujistěte se, že váš soubor /etc/X11/XF86Config neobsahuje tyto tři řádky:

```
ServerNumLock
Xleds
XkbDisable
```

V případě, že ano, tak je zakomentujte. Pro znovuzapnutí numerické klávesnice spustíte příkaz `xmodmap .Xmodmap.num`.

Grafické přihlášení s xdm

Abyste byli přivítáni grafickým přihlášením, upravte soubor /etc/inittab, který by měl zahrnovat řádek podobný tomuto:

```
x:5:respawn:/usr/bin/X11/xdm -nodaemon # též kdm nebo gdm
```

kde 5 je runlevel odpovídající X11. Upravte řádek, který definuje výchozí runlevel (obvykle 2 nebo 3), a změňte jej:

```
id:5:initdefault:
```

Počet barev je specifikován v /etc/X11/xdm/Xserver:

```
:0 local /usr/X11R6/bin/X :0 -bpp 16 vt07  # první server X, 65k barev
:1 local /usr/X11R6/bin/X :1 -bpp 32 vt08  # druhý server X, true colour
```

Pokud již máte `.xinitrc`, zkopírujte jej do `.xsession` a učiňte jej spustitelným pomocí `chmod +x .xsession`. Nyní spusťte příkaz `telinit 5` a je to hotovo.

Správce oken - Window Manager

Jakmile X pracuje, jsou zde nekonečné možnosti konfigurace; závisí to na správci oken, který používáte, jsou jich na výběr desítky. Většinou se to dělá úpravami v jednom nebo více souborech textovýcg ve vašem domovském adresáři; v jiných případech nemusíte nic upravovat a použijete applet nebo dokonce nabídku.

Některé příklady:

- **rodina fvwm**: zkopírujte /etc/X11/fvwm/system.fvwmrc (nebo podobný) do vašeho domovského adresáře, přičemž použijte příslušný název, prohlédněte si jej a začněte experimentovat. Může trvat hodně času, než to bude vypadat přesně tak, jak chcete;
- **WindowMaker**: obsahuje více konfiguračních souborů, které jsou uloženy v `$HOME/GNUstep`, a skvělý konfigurační applet;
- **KDE**, **Gnome**, **xfce** a další: nic se zde neupravuje ručně, všechno lze udělat skrze nabídky.

Krátce: pokud nechcete upravovat konfigurační soubory, zvolte něco jako `icewm`, `fvwm*`, `blackbox` atd.; pokud ano, je volba aktuálně omezena na KDE, Gnome, WindowMaker, a Xfce. Opravte mě, pokud se mýlím.

Je důležité mít dobrý `.xinitrc`. Příklad:

```
#!/bin/sh
# $HOME/.xinitrc

usermodmap=$HOME/.Xmodmap
xmodmap $usermodmap

xset s noblank  # vypnutí spořiče
xset s 300 2    # spořič se spustí za 5 min.
xset m 10 5     # nastavení rychlosti myši

rxvt -cr green -ls -bg black -fg white -fn 7x14 \
  -geometry 80x30+57+0 &

if [ "$1" = "" ] ; then  # implicitně
  WINMGR=wmaker
else
  WINMGR=$1
fi

$WINMGR
```

Ačkoliv to nevypadá, že by to bylo striktně vyžadováno, udělejte jej spustitelným pomocí `chmod +x .xinitrc`.

Výše uvedený `.xinitrc` vám umožní vybrat správce oken: vyzkoušejte

```
$ startx startkde # nebo jiný správce oken
```

Implicitní hodnoty pro aplikace X11

Najděte adresář implicitních nastavení (měl by to být /usr/X11R6/lib/X11/app-defaults). Některé aplikace sem ukládají konfigurační soubory.

Přidávání písem

Poslední verze XFree86 (řekněme, > 3.3.4) používají X Font Server, který sám podporuje písma PostScript Type 1 a True Type, takže můžete použít spousty písem dostupných na webu. Existuje pro to jednoduchá procedura.

Předpokládejme, že jste stáhli sadu písem Type 1, např. Freefont (ftp://ftp.gimp.org/pub/gimp/fonts/freefonts-0.10.tar.gz). Aby je viděl server písem, rozbalte archív do /usr/X11R6/lib/X11/fonts/. Pak upravte /etc/X11/fs/config, přidejte položku pro nový adresář a restartujte server písem.

Pokud si vytváříte svou vlastní sadu písem, budete potřebovat doplnit soubory `fonts.dir` a `fonts.scale`; to se dělá pomocí nástroje `type1inst`, který je k dispozici na adrese http://http://goblet.anu.edu.au/~m9305357/type1inst.html.

Co se týká písem True Type, seskupte je do adresáře, který si vyberete a vytvořte `fonts.dir` pomocí `ttmkfdir > fonts.dir`, zahrnutého v archívu Freetype; http://www.freetype.org. Pak proveďte to samé. Například, pokud chcete používat stejná písma jako ve Windows, řekněme, /mnt/win/windows/fonts, přejděte do tohoto adresáře, spusťte `ttmkfdir`, upravte /etc/X11/fs/config a restartujte server písem.

To všechno začalo na původním serveru písem X True Type: /http://www.dcs.ed.ac.uk/home/jec/programs/xfsft/.

Uživatelské konfigurace

Když máte upravené konfigurační soubory, zkopírujte je do /etc/skel jak jste viděli v části Konfigurace software.

Vytváření balíčků .rpm

`rpm` je tak skvělá metoda pro udržení balíčků pod kontrolou, že nejsem ochoten instalovat z archívů .tar.gz. Pouze ve velmi málo speciálních případech ano (např. zabezpečení). Kdykoliv instalujete tar archív, zvažte jeho převod na archív .rpm a pak jej znovu naistalujte; podívejte se do RPM HOWTO. Také pokud používáte nejnovější verze `gcc`, může být vhodné vložit toto do vašeho `/etc/rpmrc`:

```
optflags: i386 -O2 -mpentiumpro
```

Inovace

Pokud inovujete počítač, proveďte zálohu jako obvykle a uložte několik dalších souborů. To mohou být /etc/X11/XF86Config, /usr/bin/fax, veškerý obsah /usr/local, konfigurace jádra, celý adresář /etc a všechnu poštu v /var/spool/mail.

Nyní můžete inovovat (ve vyjímečných případech kazit!) aplikace, se kterými se dodává vaše distribuce, a přidat další balíčky. Udržujte si jejich seznam.

Konfigurační software a dokumentace

Existuje více programů, které zjednodušují nastavení a konfiguraci systému Linux. Některé se staly standardem: Red Hat, Caldera a další distribuce obsahují aplikace jako `setup`, `printtool`, `netcfg`, `usertool`, atd., zatímco SuSE dodává všeobecný konfigurační program YaST. Další užitečné programy jsou:

- **The Dotfile Generator**: pěkná aplikace X s moduly pro konfiguraci balíčků jako `emacs`, `bash`, `procmail` a dalších. Jeho stránky jsou na adrese http://www.imada.ou.dk/~blackie/dotfile;
- **Linuxconf**: poslední konfigurační nástroj. Umí dělat všechno; jak v konzole, tak pod X. Podívejte se na http://www.solucorp.qc.ca/linuxconf.

Dokumenty o konfiguraci systému Linux se objevují všude. Jedním z nejlepších je TrinityOS, http://www.ecst.csuchico.edu/~dranch/LINUX/index-linux.html. Otravujte autora, aby udělal svůj dokument v hezčích formátech.

Opravdu dobrá stránka je http://dotfiles.com. Přesně, co říkají – sada konfiguračních souborů.

Na konec

Copyright

Zpětná vazba

Možná více než jiná HOWTO, toto potřebuje a vítá vaše doporučení, kritiku a příspěvky. Zpětná vazba není jen vítána – je nutná. Pokud si myslíte, že něco chybí nebo je špatně, napište mi prosím. Pokud máte distribuci jinou než Red Hat/Mandrake a vaše konfigurační soubory jsou odlišné nebo umístěné v jiných adresářích, dejte mi prosím vědět a já vaše tipy zařadím. Mým záměrem je zjednodušení života se systémem Linux jak jen to bude možné.

Linux obsahuje mnoho balíčků, takže není možné zahnout pokyny pro všechny. Držte se při vašich požadavcích a doporučeních „nejpřijatelnějších“ programů---Nechám to na vašem uvážení.

Záruka

Tento dokument je poskytován „tak jak je“. Opravdu se ho snažím psát co nejpřesněji, ale informace zde obsažené používáte na své riziko. V žádném případě neodpovídám za případná poškození vyplývající z použití tohoto dokumentu.

Chtěl bych poděkovat autorům všech ostatních HOWTO a autorům a správcům manuálových stránek, které jsem nestydatě „vykrádal“, a všem ostatním lidem, kteří mi provádí zpětnou vazbu.

Doufám, že vám tento dokument bude užitečný. Vždycky, když provádím novou instalaci systému Linux, skutečně dělám...

Užijte si jej,

Guido =8-)

KAPITOLA 11

CD-Writing HOWTO, Winfried Trümper *winni@xpilot.org*, v2.9.3, 23. července 2000

Vypalujeme CD

Tento dokument vysvětluje, jak zapisovat na CD-ROM v systému Linux.

Úvod

Mnoho lidí používá Linux pro vypalování CD-ROMů, protože je to spolehlivé a jednoduché. Žádné modré obrazovky při vypalování a žádná starost ohledně správné kombinace hardware a software. Jakmile je to správně nastaveno, prostě to funguje. CD-writing HOWTO popisuje nastavování, umisťování dat na média a zmiňuje zajímavé aplikace zaslané laskavými čtenáři.

Copyright, licence a pojmy

Dostupnost

Jako vydavatel tohoto dokumentu většinou shrnuji to, co mi napsali jiní lidé. Nejsem vývojář softwaru a ani expert na hardware, takže se na specifické problémy s hard- nebo softwarem ptejte někoho jiného. Co má vždy smysl, je oznámení řešení problémů, které ještě nejsou popsány v tomto HOWTO, mně.

Ročně dostávám několik set e-mailů ohledně CD-Writing HOWTO. Takže prosím buďte trpěliví, jelikož nemohu vždy odpovědět do hodiny. Samozřejmě, čtu vše hned a dávám si to do mé CDR-fronty. Než se na něco zeptáte, ujistěte se prosím, že máte nejnovější verzi tohoto dokumentu; je vždy k dispozici na adrese http://www.guug.de/~winni/linux/.

Doporučená četba

Možná budete potřebovat příručku k vaší distribuci Linuxu, abyste se dozvěděli něco o instalaci nového jádra. Tuto problematiku neovládám pro jiné distribuce Linuxu, než používám já.

CD-R FAQ je obecný FAQ o zapisovatelných kompaktních discích (CD-R), zapisovacích mechanikách a požadovaném softwaru. Většina zapisovacích mechanik může být použita také pro čtení CD-ROMů, možná byste si mohli přečíst *Linux CD-ROM HOWTO*, *Linux SCSI HOWTO* a *Linux Kernel HOWTO*.

Terminologie ... lasery na maximum ... pal!

CD-ROM znamená *Compact Disc Read Only Memory*, médium úložiště využívající optický laser pro čtení mikroskopických dírek na barevném lesknoucím se disku. Díry reprezentují bity informace a jsou tak maličké, že se jich na disk vejde několik miliard. Proto je CD vysokokapacitní médium.

Termín *CD-R* je zkrácená podoba *CD-ROM recordable* a značí, že jde o CD, které na svém povrchu nemá tyto mikroskopické díry. Takže je prázdné. CD-R má uvnitř speciální chemický film, do kterého se dírky vypalují. To se dělá přidáním energie laseru, který normálně pouze snímá dírky, který pak tyto dírky vytváří. Tato akce může být na CD-R provedena **pouze** jednou. Některé oblasti můžete ponechat pro pozdější zápis, čímž se vytváří takzvané *multi-session CD*.

CD-ROM rewritable (krátce: *CD-RW*) bylo vyvinuto odstraněním omezení média CD-R. S vypalovačkou CD-RW může laser dělat obojí – vytvářet dírky do média a také vracet médium do jeho původního stavu. To je možné, protože laser ve skutečnosti nedělá dírky do média, které by se ztrácely v oblacích kouře. Dobrou analogií pro techniku je hra ledního hokeje: jízdou po ledu na něm hráči (laser) vytváří škrábance. Vzorek na ledu (médiu) je zápisem toho, co se stalo na ledu v průběhu jedné třetiny. Mezi třetinami čistící vůz Zamboni jezdí po ledu a vyplňuje škrábance rozpuštěním horní vrstvičky ledu. (Zamboni je název druhu čistícího vozu na stadionech ledního hokeje). Tímto způsobem je vzorek na ledu odstraněn a může začít nová třetina. Vědecký termín pro odpaření, kondenzaci, rozpuštění a zmrazení je „změna fáze“, což dává vypalovačkám CD-RW název „zařízení pro změnu fáze „.

Toto HOWTO se zabývá otázkou zápisu na média CD-R a CD-RW. Vítejte na palubě, kapitáne.

Adaptor vs. Adapter

Nejčastěji používaným ve zdrojích jádra je adapter (adapter: 4283, adaptor: 154). Ještě důležitější jsou parametry možností modulů a aliasy, jako u „scsi_hostadapter“. Takže z důvodu konzistentnosti terminologie v příkladech konfigurace a textu dokumentu používám tuto konvenci bez ohledu na správnost terminologie.

Podporované zapisovací jednotky CD

Zapisovací jednotky USB aktuálně nejsou podporovány. Až na toto můžete bezpečně předpokládat, že nejnovější zapisovací jednotky IDE/ATAPI a SCSI v systému Linux pracují. Novější jednotky jsou většinou kompatibilní s MMC a jsou proto podporovány. Pokud verze SCSI určité zapisovací jednotky pracuje, verze IDE/ATAPI bude pravděpodobně pracovat také a naopak. Samozřejmě, někteří lidé chtějí mít příjemný pocit po přečtení konkrétního modelu jejich zapisovací jednotky v nějakém seznamu kompatibilních jednotek. To je důvod, proč jsem nemohl vynechat následující seznam z HOWTO. Zde je kompletní shrnutí podporovaných jednotek:

```
Acer:           CDRW 4432A, CDRW 6206A, CD-R/RW 6X4X32, 8432A
BTC:            BCE 621E (IDE)
```

Compro:	CW-7502, CW-7502B
Creative:	MK 4211, RW 4224E,
Delta:	OME-W 141
Dysan:	CRW-1622
Elite:	Elite b444.41
Goldstar:	CED-8041B
Grundig:	CDR 100 IPW
Guillemot:	Maxi CD-R 4X/8X
HP:	SureStore 4020i, SureStore 6020i, C4324, C4325 CD-writer+ 7100, 7200i, 7500e, 8100i, 8110i, 8200i Plus, 8250i, 9100i, 9110i, 9200e, 9210, 9300i, 9310i
Hi-Val:	CDD 2242, CDD-3610,
Iomega:	ZIPCD 4x650
JVC:	XR-W 2001, XR-W 2010, XR-W 2040, XR-W 2042, XR-RW 2224, YR 2626
Kiss:	CDRW (model nezveden)
Kodak:	PCD 200, PCD 225, PCD 260, PCD 600
Matsushita:	matsushita je dostupná pro panasonic, podívejte se tam
Memorex:	CRW-620, CDR-622, CRW-1622, CRW-2224, CDRW-4420
Microboards:	PlayWrite 2000, PlayWrite 4000 RW, PlayWrite 4001 RW
MicroNet:	MasterCD Plus 4x4, MasterCD Plus 4x6
Mitsubishi:	CDRW-226
Mitsumi:	CR-2401-TS, CR-2600 TE, CR-2801 TE, CR-4801 TE, CR-4802 TE, CR-4804 TE
Nomai:	680.RW
Olympus:	CDS 615E, CDS 620E
Optima:	DisKovery 650 CD-R
OTI:	CDRW 965, CDRW 975 (Socrates 1.0)
Panasonic:	CW-7285, CW-7502, CW-7503, CW-7582
Philips:	CDD-521/10, CDD-522, CDD-2000, CDD-2600, CDD-3600, CDD-3610, CDD 4201 PCA 267cr, PCA 460 RW, PCRW 404, Omniwriter 26, Omniwriter 26A, CDRW800
Pinnacle:	RCD-100, RCD-1000, RCD-5020, RCD-5040
Pioneer:	DW-S114X
Plasmon:	CDR 480, CDR 4220, RF-4100, RF-4102, CDR 4400
Plextor:	CDR PX-24 CS, PX-412 C, PX-R412 C PX-R 810Ti, PX-R 820T, PX-W 4220Ti, PX-W 8220T, PX-W 8432T Plexwriter RW 4/2/20
Procom:	PCDR 4
REC:	820s
Ricoh:	RO-1420C+, MP 1420C, MP 6200S, MP 6201S, MP 7040A, MP-7060A
Samsung:	SW-204
Sanyo:	CRD-R24S
Smart and Friendly:	CD-RW 226, CD-R 1002, CD-R 1002/PRO, CD-R 1004, CD-R 2004, CD-R 2006 PLUS, CD-R 2006 PRO, CD-RW 2224, CD-R 4000, CD-R 4006, CD-R 4012, CD-RW 4424A CD-R 8020, CD-R 8220
Sony:	CDRX 100E, CDRX 120E, CDRX 140S-RP, CDU 920S, CDU 924, CDU 926S, CDU 928E, CDU 948S

```
Taiyo Yuden:    EW-50
TEAC:           CD-R50S, CD-R55S, CDR-55S, CDR-55K,
                CDR-56S-400, CD-R56S-600, R56S-614
Traxdata:       CRW 2260,
                CDR 4120, CDR 4120 Pro, CDRW 4260, CDRW 4424, CDR 4800
Turtle Beach:   2040R
Waitec:         wt 2036, wt 2444ei
WPI (Wearnes):  CDRW-622, CDR-632P
Yamaha:         CDR-100, CDR 102, CDR-200, CDR-200t, CDR-200tx
                CDR-400, CDR-400c, CDR-400t, CDR-400tx, CDR-400Atx
                CDW-2216E, CRW-2260, CRW-2260t,
                CRW-4250tx, CRW-4260t, CRW-4260tx, CRW-4261, CRW-4416S,
                CRW-6416S, CRW-8424E
```

Tabulka 1: Zapisovací CD mechaniky podporované systémem Linux

Detailní seznam modelů, které byly zjištěny jako funkční nebo nefunkční v různých unixových systémech je k dispozici online na adrese http://www.guug.de:8080/cgi-bin/winni/lsc-orig.pl.

Pokud váš hardware není podporován, můžete stále vytvářet obraz CD v systému Linux. Možná to tak budete chtít udělat, protože většina vypalovacího software pro DOS nepracuje s rozšířeními RockRidge (unixové souborové systémy na CD-ROM). V dalším kroku můžete vypálit obraz na CD-R v softwaru pro DOS nebo Macintosh.

Podporované funkce

Existují dvě třídy nástrojů: ovladače hardwaru a software pro formátování dat. Ovladače hardwaru podporují následující funkce:

Podporovaná funkce	cdwrite-2.1	cdrecord-1.6	cdrdao
IDE/ATAPI	ano	ano	ano
Paralelní Port	ne	ano	ano
CD-RW	ne	ano	ano
Audio CD	ano	ano	ano
Datový CD-ROM	ano	ano	částečně
Multisession	částečně	ano	ne
TAO (track-at-once)	ano	ano	ano
DAO (disk-at-once)	ne	částečně	ano
Packetový zápis	ne	ne	ne

`cdwrite` je neudržovaný software uvedený pouze pro úplnost. Používejte místo něj prosím `cdrecord`, jelikož podporuje širší paletu hardwaru a má výrazně více funkcí. Hlavní výhodou cdrdao je schopnost vytváření hudebních CD bez dvou sekund ticha mezi stopami (zápisem v režimu disk-at-once; DAO.

Nástroje, klasifikované jako formátovače dat (data-formatters) organizují data na médiu (vkládají na něj systém souborů).

Funkce	mkisofs	mkhybrid	mkvcdfs
ISO 9660	ano	ano	ne
RockRidge	ano	ano	ne
El Torito	ano	ano	ne
HFS	ne	ano	ne

```
Joliet          ano             ano             ne
Multisession    ano             ano             ne
CD-Extra        ano             ano             ne
Video-CD        ne              ne              ano
```

Největším rozdílem mezi souborovými systémy ISO 9660 a ReiserFS nebo Ext -2: je, že jakmile jsou zapsány, nemůžete upravovat soubory. Další omezení souborového systému ISO-9660 zahrnují:

- pouze 8 povolených úrovní podadresářů (počítáno od adresáře nejvyšší úrovně CD)
- maximální délka názvů souborů: 32 znaků
- kapacita 650 MB

RockRidge je rozšíření, které umožňuje delší názvy souborů a hlubší hierarchii adresářů než souborový systém ISO-9660. Při čtení CD-ROM s rozšířením RockRidge pod systémem Linux se objevují všechny známé vlastnosti souborů, jako jsou vlastník, skupina, oprávnění a symbolické odkazy (jako unixový souborový systém). Tato rozšíření nejsou dostupná, když čtete CD-ROM pod DOSem nebo heterogenní skupinou operačních systému rodiny Windows.

El Torito lze použít pro vytváření zaveditelných CD-ROM. Aby tato funkce pracovala, musí ji podporovat BIOS vašeho PC. Nepřesně řečeno, prvních 1.44 (nebo 2.88, pokud je podporováno) Mbajtů CD-ROM obsahuje obraz vámi zadané diskety. Tento obraz je BIOSem zpracováván jako disketa a je z něj spuštěn systém (v důsledku toho, když spouštíte z této virtuální diskety, nemusí být vaše původní jednotka A: (/dev/fd0) přístupná).

HFS umožňuje počítačům Macintosh načítat CD-ROM jako by to byl svazek HFS (nativní souborový systém MacOS).

Joliet přináší (mezi jinými věcmi) dlouhé názvy souborů pro novější varianty Windows (95, 98, NT). Autor samozřejmě nezná žádný nástroj, který by umožňoval dlouhé názvy souborů v samotném DOSu nebo Windows 3.11.

Video-CD mohou být přímo přehrávána na jednotkách DVD.

V části 2.8 je uveden seznam dostupnosti zmíněného softwaru.

E-mailové diskusní skupiny

Pokud se chcete připojit k vývojovému týmu (se záměrem jim aktivně *pomáhat*), odešlete e-mail na cdwrite-request@other.debian.org a napište do těla zprávy slovo `subscribe`.

Nastavení systému Linux pro vytváření CD-ROMů

Tato část se týká následujících druhů zapisovacích mechanik: SCSI, IDE/ATAPI a zařízení pro paralelní port. Zapisovací mechaniky USB do května 2000 nebyly podporovány. Zapisovací mechaniky jiné než SCSI vyžadují ovladače pro kompatibilitu, což umožní, aby se objevovaly jako skutečná zařízení SCSI. Na jednu stranu je taková unifikující strategie jednoduchá („všechno je SCSI"), protože na úrovni aplikace můžete sdílet své znalosti s ostatními uživateli bez ohledu na druh jejich zapisovací mechaniky. Na druhou stranu, musíte překonfigurovat aplikace jako přehrávače hudebních CD nebo nástroj pro připojování, aby odpovídaly změně názvu ovladače. Například, pokud jste dříve prováděli přístup k vaší zapisovací jednotce ATAPI skrze soubor zařízení /dev/hdc, po aktivování ovladačů kompatibility s SCSI k ní budete muset přistupovat skrze /dev/scd0.

Jakmile jste úspěšně nastavili váš hardware a zbytek vašeho systému Linux, můžete pomocí příkazu `cdrecord -scanbus` zobrazit seznam zařízení na vašich sběrnicích SCSI. Cílem této části je provést vás nastavením vašeho systému Linux, abyste nakonec viděli něco takového:

```
shell> cdrecord -scanbus
Cdrecord release 1.7a1 Copyright (C) 1995-1998 Jörg Schilling
scsibus0:
      0,0,0) 'Quantum ' 'XP34300          ' 'F76D' Disk
      0,1,0) 'SEAGATE ' 'ST11200N         ' '8334' Disk
      0,2,0) *
      0,3,0) 'TOSHIBA ' 'MK537FB/         ' '6258' Disk
      0,4,0) 'WANGTEK ' '5150ES SCSI 36   ' 'ESB6' Removable Tape
      0,5,0) 'EXABYTE ' 'EXB-8500-85QUE   ' '0428' Removable Tape
      0,6,0) 'TOSHIBA ' 'XM-3401TASUNSLCD' '3593' Removable CD-ROM
      0,7,0) *
scsibus1:
      1,0,0) 'Quantum ' 'XP31070W         ' 'L912' Disk
      1,1,0) *
      1,2,0) *
      1,3,0) 'TEAC    ' 'CD-R55S          ' '1.0H' Removable CD-ROM
      1,4,0) 'MATSHITA' 'CD-R   CW-7502   ' '4.02' Removable CD-ROM
      1,5,0) *
      1,6,0) 'YAMAHA  ' 'CDR400t          ' '1.0d' Removable CD-ROM
      1,7,0) *
```

Výpis 1: Detekování zařízení na vaší sběrnici SCSI

Příklad poskytl Jörg Schilling a zobrazuje celkem čtyři zapisovací mechaniky. Všimněte si, že -scanbus také vypisuje ostatní zařízení, např. normální jednotky CD-ROM a jednotky pevných disků. Poslední sloupec uvádí popis zařízení SCSI, podle kterého nemůžete zřetelně odlišit obyčejné jednotky CD-ROM od těch zapisovacích. Ale identifikace produktu (prostřední sloupec) často obsahuje zmínky o funkci ve tvaru R, -R nebo -RW.

Rychlý start

Tato část je pokusem o rychlý a jednoduchý popis konfigurace. Ne všechna možná nastavení jsou popsána, ale jděte na to a v každém případě to vyzkoušejte. Nejdříve ze všeho se podívejte na verzi jádra operačního systému Linux vypsanou příkazem „uname -r“. Může to být něco jako 2.0.X nebo 2.2.Y, kde X je vyšší než 36 a Y je vyšší než 11. Pokud máte starší verze nebo vývojové jádro, je to na vás. Instalace nového jádra je asi tolik práce jako oprava toho starého, takže jsem odstranil všechny pokyny, které jsou zapotřebí pro špatná jádra.

Níže uvedený seznam obsahuje sadu příkazů, kterými byste měli začít. Příkazy vytváří soubory zařízení v /dev, pokud ještě neexistují.

```
test 'whoami' = 'root' || echo "Pro spuštění těchto příkazů musíte být root."
cd /dev/
umask -S u=rwx,g=rwx,o-rwx
[ -f loop0 ] \
    || ./MAKEDEV loop \
    || for i in 0 1 2 3 4 5 6 7; do mknod loop$i b 7 $i; done
[ -f sg0 -o -f sga ] \
    || ./MAKEDEV sg \
    || for i in 0 1 2 3 4 5 6 7; do mknod sg$i  c 21 $i; done
```

Výpis 2: Vytváření souborů zařízení

Přístup k hardwaru je v systému Linux obvykle implementován pomocí souborů zařízení. Takže než budete dělat něco dalšího, ujistěte se, že tyto soubory v adresáři /dev existují. Pořád mi nikdo neuměl sdělit důvod, proč to nebylo zautomatizováno pomocí technik jako jsou souborové systémy zařízení (device filesystem; devfs). Devfs je k dispozici roky, přináší bezpečnější (!) a mnohem čistší pojmenování zařízení a vytváří položky v /dev automaticky. Někteří prominenti argumentují tím, že devfs není perfektní řešení, ale nepřišli s ničím lepším, ani ničím srovnatelným a v neposlední řadě není nyní k dispozici nic a nic se netestuje. Začněme pomocí devfs, takže mohu odstranit výše uvedené příkazy z tohoto dokumentu (http://www.atnf.CSIRO.AU/~rgooch/linux/kernel-patches.html).

Další věcí, kterou je potřeba zajistit, je to, aby bylo jádro systému Linux vybaveno nezbytnými ovladači. Následující příkazy kontrolují přítomnost různých souborů ovladačů ve spuštěném jádře systému Linux. Obvykle by měl příkaz „cdrecord -scanbus“ zapnout automatické načítání všech ovladačů. V případě, že ovladač pak v jádře není, je to oznámeno a je ručně načten modularizovaný ovladač (modul) skrze insmod.

```
test 'whoami' = 'root' || echo "Pro provedení těchto příkazů musíte být root."
cdrecord -scanbus > /dev/null
if ! (pidof kerneld || test -f "/proc/sys/kernel/modprobe"); then
    echo "Ani kerneld, ani kmod neběží, takže moduly nelze načíst automaticky
".
fi
report_no_autoload() {
    echo "Příště si ověřte, že je načten modul $1."
}
if test ! -f "/proc/scsi/scsi"; then
    report_no_autoload scsi_mod  &&  insmod scsi_mod
fi
if ! grep "^........ sg_" /proc/ksyms > /dev/null; then
    report_no_autoload sg  &&  insmod sg
fi
if ! grep "^........ sr_" /proc/ksyms > /dev/null; then
    report_no_autoload sr_mod  &&  insmod sr_mod
fi
if ! grep "^........ loop_" /proc/ksyms > /dev/null; then
    report_no_autoload loop  &&  insmod loop
fi
if ! grep iso9660 /proc/filesystems > /dev/null; then
    report_no_autoload iso9660  &&  insmod iso9660
fi
echo "Následující část je potřeba pouze pro zapisovací mechaniky IDE/ATAPI."
if ! grep ide-scsi /proc/ide/drivers > /dev/null; then
    report_no_autoload ide-scsi  &&  insmod ide-scsi
fi
cdrecord -scanbus
```

Výpis 3: Testování ovladačů

Pokud insmod nadává na chybějící soubory modulu, přečtěte si prosím následující kapitolu. Pokud jste v textovém režimu (konsole), může načítání modulů vyvolat zobrazení některých zpráv na obrazovce. Pokud jste v grafickém režimu (X11, KDE, Gnome), můžete se k těmto zprávám vrátit příkazem dmesg.

Existuje několik způsobů načtení modulů při příštím spuštění systému Linux:

```
(1) Vložení relevantního příkazu insmod do spouštěcí sekvence
    (skript příkazového interpretu s názvem rc.local nebo jeho ekvivalent).
(2a) Spuštění kerneld nebo kmod a
(2b) jejich konfigurace v /etc/modules.conf (abych byl přesnější,
     konfigurujete nástroj modprobe, který je volán démony)
```

Lidé se zapisovací mechanikou SCSI mohou přeskočit zbytek této části, protože cdrecord bude pravděpodobně stejně jejich hardware detekovat. Pokud ne, pak mi prosím pošlete e-mail s nějakými informacemi o vašem nastavení, abych mohl vylepšit část o zapisovacích jednotkách SCSI.

Nyní k lidem se zapisovacími mechanikami pro IDE/ATAPI. Jak bylo napsáno v předchozí kapitole, musíte načíst ovladač kompatibility ide-scsi. Ale tento ovladač má přístup k vaší zapisovací mechanice pouze pokud zde není jiný ovladač, který to již dělá. Jinými slovy, musíte říci vašemu ovladači IDE, aby ponechal vaši zapisovací mechaniku nerozpoznanou, aby ji mohl převzít ovladač ide-scsi.

```
hda = sběrnice IDE/konektor 0 zařízení master
hdb = sběrnice IDE/konektor 0 zařízení slave
hdc = sběrnice IDE/konektor 1 zařízení master
hdd = sběrnice IDE/konektor 1 zařízení slave
```

Výše uvedená tabulka zobrazuje vztah názvů souborů zařízení a umístění zařízení na sběrnicích IDE. Název souboru zařízení reprezentující vaši zapisovací mechaniku musí být předán ovladači v jádře systému Linux. Příklad: hdb=ide-scsi. Takové nastavení by mělo být přidáno do lilo.conf nebo chos.conf pokud je ovladač staticky zakompilován do vašeho jádra, což se zdá být nejobvyklejším nastavením. Pokud potřebujete předat jádru více než jeden parametr, pak je oddělte mezerami (jak bylo zobrazeno v příkladu chos). Následující dva výpisy zobrazují příklady konfigurací obsahujících více řádku než relevantní řádek append. Všimněte si prosím, že položky append- a cmdline- jsou určeny pro jednotlivé druhy jádra (tj. nepřidávejte je přímo na začátek).

```
image=/boot/zImage-2.2.14
  label=Linux
  read-only
  append="hdb=ide-scsi"
```

Výpis 4: Příklad konfigurace pro lilo (/etc/lilo.conf)

```
linux "Linux 2.1.14" {
        image=/boot/zImage-2.0.37
        cmdline= root=/dev/hda5 readonly hdb=ide-scsi
}
```

Výpis 5: Příklad konfigurace pro chos (/etc/chos.conf)

Pokud je ovladač pro IDE/ATAPI CD-ROM načten jako modul, pak se výše uvedené pro vás nebude lišit, ale ujistěte se, že jste vložili řádek options z následujícího výpisu. Poslední tři řádky výpisu jsou obecně doporučovány pro lepší automatizaci načítání požadovaných modulů.

```
options ide-cd ignore=hdb            # říká modulu ide-cd, aby ignoroval hdb
alias scd0 sr_mod                    # načtení sr_mod při přístupu k scd0
#pre-install ide-scsi modprobe imm    # odkomentujte pouze pro některé jednotky
ZIP
pre-install sg     modprobe ide-scsi # načíst ide-scsi před sg
pre-install sr_mod modprobe ide-scsi # načíst ide-scsi před sr_mod
pre-install ide-scsi modprobe ide-cd # načíst ide-cd   před ide-scsi
```

Výpis 6: Příklad konfigurace pro /etc/modules.conf

Pokud je vaše zapisovací mechanika jedinou mechanikou CD-ROM připojenou k vašemu počítači, pak si pamatujte, že máte přístup k jednotce CD-ROM v zapisovači pomocí souboru zařízení /dev/scd kde =0,..,8. Možná budete chtít změnit symbolický název zařízení cdrom, aby ukazoval na nový soubor zařízení. Níže uvedený výpis ukazuje příkaz, kterým toho dosáhnete pro scd0.

```
cd /dev && rm cdrom && ln -s scd0 cdrom
```

Výpis 7: Nastavení cdrom jako symbolického názvu pro scd0

Pokud jsou vaše zapisovací mechanika a jednotka CD-ROM dvě různá zařízení, pak symbolický odkaz cdrom neměňte.

Speciální poznámky k zapisovacím mechanikám SCSI

Ujistěte se prosím, že je vaše zapisovací jednotka rozpoznána BIOSem vaší karty hostitelského adaptéru SCSI. Každý hostitelský adaptér SCSI zkoumá po zapnutí sběrnici SCSI a hlásí všechna nalezená zařízení připojená ke sběrnici. Zpráva zahrnuje SCSI ID zařízení a označení produktu. Nemá smysl nic provádět, dokud není vaše zapisovací jednotka uvedena v této zprávě.

Pokud chcete připojit vaše zařízení SCSI přes paralelní port (nepleťte si to s jednotkami IDE pro paralelní port), potřebujete speciální aktivní kabel a speciální ovladač jádra. O této možnosti se dozvíte více na adrese http://www.torque.net/parport/parscsi.html.

Speciální poznámky k zapisovacím mechanikám pro paralelní port

Promiňte, ale nic o tom nevím. Přečtěte si prosím http://www.torque.net/parport/paride.html nebo váš lokální soubor /usr/src/linux/Documentation/paride.txt.

Kompilace chybějících modulů jádra (volitelně)

Tuto část nemusíte číst, pokud je váš hardware úspěšně rozpoznán a nastaven výše popsanými kroky konfigurace.

Jádro systému Linux může být vybaveno ovladači pro různé funkce. Ovladače do jádra můžete zakompilovat staticky nebo je můžete zkompilovat jako modul pro načítání na vyžádání. Druhá metoda je preferována pro ovladače, které nejsou absolutně nutné pro uvedení vašeho systému Linux do života, protože pak bude vaše jádro menší a rychlejší. Samozřejmě, některé ovladače jsou nezbytné pro spuštění systému a neměli byste je kompilovat jako modul. Příklad: pokud je váš systém uložen na pevném disku IDE, musíte mít ovladač pro pevné disky IDE v jádře – nikoliv jako modul.

Existují tři různé druhy zapisovacích mechanik: SCSI, IDE/ATAPI a externí zapisovací mechaniky, které pracují přes paralelní port. Tabulka zobrazuje jak nastavit jádro systému Linux pro tyto druhy hardware. První sloupec tabulky je část konfigurační nabídky jádra, kde můžete nalézt nastavení. Druhý sloupec je popis funkce (převzatý také z konfigurační nabídky jádra). Třetí sloupec udává název vyplývajícího modulu. Sloupce s názvy SCSI, IDE a PP obsahují nezbytné volby pro asociovaný hardware (PP = paralelní port).

Odd.	Popis	Modul	SCSI	IDE	PP
BLOCK	Enhanced IDE/MFM/RLL...			Y	
BLOCK	IDE/ATAPI CDROM	ide-cd		M	
BLOCK	SCSI emulation support	ide-scsi		M	
BLOCK	Loopback device	loop	M	M	M
PARIDE	Parallel port IDE device	paride			Y/M
PARIDE	Parallel port ATAPI CD-ROMs				M
PARIDE	Parallel port generic ATAPI				M
PARIDE	(vyberet vhodný nízkoúrovňový ovladač)				Y
SCSI	SCSI support	scsi_mod	Y/M	Y/M	
SCSI	SCSI CD-ROM support	sr_mod	Y/M	Y/M	
SCSI	Enable vendor-specific		Y	Y	
SCSI	SCSI generic support	sg	Y/M	Y/M	
SCSI	(vyberet vhodný nízkoúrovňový ovladač)		Y		
FS	ISO 9660 CDROM filesystem	iso9660	Y/M	Y/M	Y/M
FS	Microsoft Joliet cdrom...	joliet	Y	Y	Y

Tabulka 2: Výběr ovladače pro různé druhy zapisovacích mechanik

Y znamená ano a znamená to, že byste měli vložit tohoto démona do jádra. M znamená modul a to znamená, že byste měli nebo budete muset zkompilovat tuto funkci jako modul. Y/M vám dává na výběr (pořadí indikuje volby s menším počtem potenciálních problémů). Prázdná nastavení není potřeba měnit a jejich ponechání zvyšuje šanci, že výsledné jádro bude pracovat (pokud předtím pracovalo...). Zejména v prostředích, kde jsou zařízení SCSI i ATAPI, je lepší sestavovat většinu věcí jako moduly.

Kompilace zařízení loopback je nepovinná. Umožňuje vám otestovat obraz před jeho zápisem na médium. Pokud chcete číst CD-ROMy, potřebujete podporu pro souborový systém ISO 9660. Tento ovladač automaticky zahrnuje rozšíření RockRidge. Rozšíření Microsoft Joliet CD-ROM musí být přidáno do souborového systému ISO 9660 explicitně. V některém případě budete potřebovat pro váš hardware nízkoúrovňový ovladač. Nízkoúrovňový ovladače je ten, který spolupracuje přímo s hardwarem. Pro SCSI a paralelní port existuje mnoho nízkoúrovňových ovladačů.

Instalace výsledného jádra systému Linux přesahuje rámec tohoto HOWTO. Podívejte se, prosím, do dokumentace vaší distribuce systému Linux.

Uživatelé systému RedHat Linux, vězte, že musíte zakompilovat funkce „Ramdisk support“ a „Initial ramdisk“. Kromě toho musíte vygenerovat nový ramdisk s novými moduly spuštěním příkazu jako „mkinitrd – preload ide-cd initrd-2.2.14.img 2.2.14“.

Získání uživatelského softwaru pro vypalování CD-R

Detailnější seznam nástrojů týkajících se produkování CD-ROMů je k dispozici na adrese http://www.fokus.gmd.de/research/cc/glone/employees/joerg.schilling/private/cdb.html.

Nástroje příkazového řádku

Jeden z následujících balíčků je potřeba pro generování obrazů CD-R (vyžadováno pouze pro datové CD-ROMy):

ftp://tsx-11.mit.edu/pub/linux/packages/mkisofs/ (mkisofs)

ftp://ftp.ge.ucl.ac.uk/pub/mkhfs (mkhybrid)

Pro zápis obrazů na CD-R potřebujete jeden z následujících softwarových balíčků:

ftp://ftp.fokus.gmd.de/pub/unix/cdrecord/ (cdrecord)

http://www.ping.de/sites/daneb/cdrdao.html (cdrdao)

http://www.munich-vision.de/vcd/ (mkvcdfs)

Nevěřte manuálové stránce starých verzí `mkisofs`, která tvrdí, že potřebujete verzi 1.5 programu `cdwrite`. Použijte pouze cdrecord a budete v pohodě. Pamatujte si prosím, že novější verze cdrecord jsou dodávány s rozšířenou verzí mkisofs a některé další nástroje jsou v podadresáři misc/ (readcd, isosize) a nikde jinde.

Grafická uživatelská rozhraní (volitelně)

Rozhraní (Front-end) jsou v Linuxu opravdovým rozhraním. To znamená, že pořád musíte nainstalovat nástroje příkazového řádku, ale budete k nim mít přístup lépe vypadajícím způsobem.

X-CD-Roast je balíček programů určený pro jednoduché vytváření CD v Linuxu. Kombinuje nástroje příkazového řádku jako `cdrecord` a `mkisofs` s hezkým grafickým uživatelským rozhraním.

http://www.fh-muenchen.de/home/ze/rz/services/projects/xcdroast/e_overview.html

BurnIT je rozhraní napsané v jazyce JAVA pro cdrecord, mkisofs a cdda2wav-0.95, čímž tvoří kompletní balíček pro vypalování CD na unixové platformě. Je k dispozici na adrese

http://sunsite.auc.dk/BurnIT/

CD-Tux je textově orientované rozhraní pro programy mkisofs a cdrecord. „Vytváří jednoduše použitelné prostředí pro skoro všechno, co byste mohli dělat s CD v plných barvách pomocí (i nechvalně) známé knihovny NCURSES. A všechno tohle dělá spustitelný soubor menší než 75k.

http://www.datadictator.co.za/cdtux/

Vypalování CD-R

Vytváření CD-ROMů se v systému Linux skládá ze dvou kroků:

- zabalení požadovaných dat (souborů, hudby nebo obojího) do souborů se speciálními formáty
- zápis dat ze souborů na CD-R pomocí utility `cdrecord`

Tato kapitola detailně popisuje kroky pro datové a hudební CD.

Vytváření CD-ROMů (jen data)

Pamatujte si, že shromažďování dat, která chcete umístit na CD, obvykle trvá déle než by jeden čekal. Uvědomte si, že chybějící soubory nelze přidat na CD, pokud je zapsáno a ukončeno. To platí i ohledně CD-RW, které může být přepsáno pouze celé. Použití funkce multi-session není volbou pro jednotlivé soubory, protože zabírá moc místa pro nový kompletní obsah (table of contents; TOC). UDF na Linuxu ještě není připraveno.

Také pamatujte na to, že je určité množství volného místa na CD použito pro uložení informací o souborovém systému ISO-9660 (obvykle několik MB). 620 MB dat se vždy na 650MB CD-R vejde.

Vytváření obrazu pozdějšího CD-ROM

Než může být nějaké médium úložiště (disketa, pevný disk nebo CD) použito, musí na něm být systém souborů (DOSovsky řečeno: musí být zformátováno). Systém souborů odpovídá za organizaci a začleňování souborů, které by měly být uloženy na médiu.

Obvyklé utility pro vytváření souborových systémů na oddílech pevného disku na ně zapisují prázdný systém souborů, který je pak připojen a podle potřeby zaplněn soubory uživatelů. Na CD se dá zapsat jen jednou, takže když jsme psali o prázdném systému souborů, bylo by zformátováno a zůstalo úplně prázdné navždy. To platí i pro přepisovatelná média, jelikož nemůžete libovolně měnit sektory; musíte vymazat celý jejich obsah.

Takže to, co potřebujeme je nástroj, který vytváří systém souborů při kopírování souborů na CD. Tento nástroj se nazývá `mkisofs`. Příklad využití:

```
mkisofs  -r   -o cd_image   private_collection/
              '----------'  '------------------'
                    |              |
          zapsat výstup do  vzít adresář jako vstup
```

Volba '-r' nastavuje oprávnění pro všechny soubory, aby byly veřejně dostupné na CD a zapíná rozšíření RockRidge. Pravděpodobně nechcete tuto volbu použít dokud nevíte, co opravdu dělá (pozn.: bez '-r' připojovací body převezmou oprávnění z `private_collection`!).

`mkisofs` se pokusí mapovat všechny názvy souborů na formát 8.3 používaný systémem DOS pro zajištění nejvyšší možné kompatibility. V případě konfliktu názvů (různé soubory mají stejný název 8.3) se použijí v názvech souborů čísla a informace o zvoleném názvu je vypsána na STDERR (obvykle na obrazovku). Nepanikařte: V Linuxu nikdy neuvidíte tyto divné názvy souborů 8.3, protože Linux používá rozšíření Rock Ridge, která obsahují informace o původním souboru (oprávnění, název souboru, atd.).

Nezapomeňte použít volbu -J (rozšíření MS Joliet) nebo použít mkhybrid pokud chcete generovat CD-ROM přátelštější pro Windows. Pro CD-ROMy HFS používané na počítačích Macintosh, byste si měli spíše přečíst manuálovou stránku větší sestřičky mkisofs „mkhybrid", kde najdete detaily k různým volbám.

Nyní byste se mohli divit, proč není výstup `mkisofs` přímo odeslán na zapisovací zařízení. Existují tři důvody:

- `mkisofs` neví nic o ovládání zapisovacích jednotek CD.
- Mohli byste chtít obraz před jeho vypálením otestovat.
- Na pomalých počítačích by to nebylo spolehlivé.

Existuje metoda zápisu na CD-R najednou, která bude popsána níže.

Někdo by také mohl přemýšlet o vytváření dalšího oddílu a zápisu obrazu do tohoto oddílu místo do souboru. Jsem proti této strategii, protože pokud zapíšete kvůli překlepu na špatný oddíl, můžete přijít o celý systém Linux. Čti: mně se to stalo... Kromě toho je to mrhání místem na disku, protože obraz CD jsou dočasná data, která mohou být po vytvoření CD odstraněna. Samozřejmě, použití základních oddílů šetří čas pro odstraňování souborů o velikosti 650 MB.

Testování obrazu CD

Linux umí připojovat soubory jako by to byly oddíly disku. Tato funkce je užitečná pro kontrolu, že rozvržení adresáře a oprávnění k přístupu k souborům obrazu CD odpovídá vašim požadav-

kům. Ačkoliv je dnes médium velmi levné, proces zápisu pořád nějakou dobu trvá a mohli byste chtít třeba alespoň ušetřit čas pomocí rychlého vyzkoušení.

Pro připojení souboru `cd_image` vytvořeného výše na adresář `/cdrom` zadejte příkaz

```
mount -t iso9660 -o ro,loop=/dev/loop0 cd_image /cdrom
```

Nyní se můžete podívat na soubory v adresáři `/cdrom` – objevují se přesně jako by byly na skutečném CD. Pro odpojení obrazu CD jednoduše zadejte `umount /cdrom` (Varování: na jádrech systému Linux před verzí 2.0.31 nemusí být poslední soubor v `/cdrom` plně čitelný. Použijte prosím novější jádro jako např. 2.0.36. Volba -pad pro cdrecord se týká pouze hudebních CD a volba -pad pro mkisofs vyžaduje cestu, kterou je těžší zadávat než inovovat na bezchybné jádro).

> **POZNÁMKA:** Některé archaické verze `mount` neumí pracovat se zařízeními loopback. Pokud máte takovou starou verzi `mount`, pak inovujte svůj systém Linux. Někteří lidé také doporučovali doplnění informací o tom, jak získat nejnovější nástroje pro připojování, do tohoto HOWTO. Vždycky to odmítám. Pokud distribuce vašeho systému Linux obsahuje zastaralý `mount`, nahlašte to jako chybu. Pokud není distribuce Linuxu snadno inovovatelná, nahlašte to jako chybu.

Pokud bych přidal všechny informace, které jsou potřeba pro odstranění chyb ve špatně navržených distribucích systému Linux, bylo by toto HOWTO o hodně větší a bylo by těžší jej číst.

Zápis obrazu CD na CD

Tato část zahrnuje pouze zápis dat na CD v režimu TAO, protože je to nejpoužívanější režim pro data. Více informací o rozdílech režimů TAO a DAO najdete v kapitole o hudebních CD-R. Pokud používáte režim DAO s nástrojem cdrdao, pak nezapomeňte přidat fiktivní hudební stopu na konec souboru TOC (viz README).

Už toho moc nezbylo. Pokud jste to ještě nezkoušeli, tak teď je ten správný čas pro příkaz

```
cdrecord -scanbus
```

To vám řekne, ke kterému zařízení SCSI patří vaše zapisovací jednotka CD. Všechny ostatní metody odhadování informací, které dobře popisuje cdrecord, byly z tohoto HOWTO odstraněny.

Před předvedením posledního příkazu mi dovolte, abych vás varoval, že zapisovací jednotky CD chtějí být krmeny konstantním proudem dat. Takže proces zápisu obrazu CD na CD nesmí být přerušen, jinak vytvoříte poškozené CD. Datový proud je jednoduché přerušit odstraněním velmi velkého souboru. Příklad: pokud odstraníte starý obraz CD o velikosti 650 Mbajtů, musí jádro inovovat informace o 650,000 bloků na pevném disku (za předpokladu, že váš systém souborů má velikost bloku 1 Kbajt). To zabere nějaký čas a velmi pravděpodobně to zpomalí činnost disku na moc dlouho a vznikne tak několikasekundová časová mezera v proudu dat. Čtení pošty, procházení na Internetu nebo dokonce kompilace jádra samozřejmě na moderních počítačích obecně neovlivní proces zápisu.

Pamatujte si, prosím, že žádná zapisovací jednotky nemůže vrátit svůj laser a pokračovat od původního bodu na CD, ve kterém byla přerušena. Proto silné vibrace nebo jiné mechanické rázy pravděpodobně zničí CD, které vytváříte.

Když už jste duševně připraveni, oblečte se do černé róby, vynásobte identifikátor SCSI zapisovací jednotky CD jeho verzí SCSI a zapalte tolik svíček, vyslovte dvě verze ASR-FAQ (diskusní skupina alt.sysadmin.recovery) a nakonec napište:

```
shell> SCSI_BUS=0   # převzato z výpisu 1 "scsibus0:"
shell> SCSI_ID=6    # převzato z výpisu 1 "TOSHIBA XM-3401"
shell> SCSI_LUN=0
shell> cdrecord -v speed=2 dev=$SCSI_BUS,$SCSI_ID,$SCSI_LUN \
                   -data  cd_image

# stejné jako výše, ale kratší:
shell> cdrecord -v speed=2 dev=0,6,0  -data  cd_image
```

Pro lepší čitelnost jsou parametry zapisovací jednotky uloženy do tří proměnných prostředí s názvy: SCSI_BUS, SCSI_ID, SCSI_LUN.

Pokud používáte cdrecord pro přepis CD-RW, musíte přidat volbu „blank=...“ pro odstranění starého obsahu. Přečtěte si prosím manuálovou stránku, kde se dozvíte více o různých metodách vymazání CD-RW.

V době, kdy všichni kromě mě měli 400MHz počítač, lidé zadávali výstup z mkisofs přímo do cdrecord:

```
shell> IMG_SIZE='mkisofs -R -q -print-size private_collection/  2>&1 \
| sed -e "s/.* = //"'
shell> echo $IMG_SIZE
shell> [ "0$IMG_SIZE" -ne 0 ] && mkisofs -r  private_collection/  \
|cdrecord  speed=2  dev=0,6,0
            tsize=${IMG_SIZE}s  -data  -
#       nezapomeňte na --^         ^-- čti data ze STDIN
```

První příkaz je prázdné spuštění pro zjištění velikosti obrazu (aby to fungovalo, potřebujete mkisofs z distribuce cdrecord). Je potřeba zadat všechny parametry, které použijete při konečném spuštění (např. -J nebo -hfs). Vaše zapisovací jednotka možná nepotřebuje znát velikost obrazu, který má být zapsán, takže můžete toto spuštění vypustit. Vypsaná velikost musí být předána jako parametr tsize do cdrecord (je uložena v proměnné prostředí IMG_SIZE). Druhý příkaz je sekvence mkisofs a cdrecord, spojená rourou.

Vytváření hudebních CD

Vytváření hudebních CD je velmi podobné krokům popsaným výše pro datová CD. Můžete si vybrat mezi dvěma technikami: DAO nebo TAO. TAO (stopa najednou – track at once) se méně hodí pro hudbu, protože uslyšíte lupnutí mezi jednotlivými stopami. TAO byla popsána jako první, protože je o něco jednodušší a DAO není ještě k dispozici pro všechny jednotky.

Hlavní rozdíl oproti zapisování datových CD-R je formát obrazů. ISO-9660 (nebo jakýkoliv systém souborů, kterému dáváte přednost) nelze použít, protože žádný přehrávač hudebních CD neumí pracovat se systémy souborů. Namísto toho musí být hudební data zapsána jako „16bitové stereo vzorky v kódování PCM při 44100 vzorcích/sekundu (44,1 kHz)“.

Jedním nástrojem pro konverzi vašich zvukových souborů do požadovaného formátu je sox. Jeho používání je jednoduché:

```
shell> sox  killing-my-software.wav  killing-my-software.cdr
```

Tento příkaz by zkonvertoval píseň killing-my-software z formátu WAV do CDR s hudebním formátem. Podívejte se na manuálovou stránku sox, kde najdete detailní informace o formátech a rozšířeních systému souborů, které sox zná. Protože výstup ruční konverze zabírá moc místa na disku, byla vytvořena funkce vestavěná do cdrecord pro hudební formáty WAV a AU. Takže pokud

vaše hudební soubory mají přípony .wav nebo .au (a vzorkování „stereo, 16 bitů, 44,1 kHz“), můžete je použít jako hudební stopy bez ruční konverze do formátu CDR. cdrecord samozřejmě vyžaduje, aby velikost hudebních dat byla celé číslo násobek 2352 a aby byla větší než 705 600 bajtů, což není u některých souborů WAV splněno. Pro použití takových souborů v sox je potřeba doplnit hudebními daty na 2352 bajtů.

Zapisování hudebních CD (TAO)

Hudební CD se skládá z hudebních stop, které jsou v režimu TAO uspořádány jako samostatné obrazy. Takže pokud chcete mít na vašem CD deset stop, musíte vytvořit deset obrazů.

Cdrecord zapisuje obrazy CD jako hudební stopy pokud je uvedena volba -audio. Další volby jsou shodné s těmi, které se používají při vytváření datových CD (pokud nemáte velmi speciální požadavky). Tyto tři příklady dělají všechny to samé, ale čtou stopy z různých formátů zvukových souborů:

```
shell> cdrecord -v speed=2 dev=0,6,0  -audio  track1.cdr track2.cdr...
shell> cdrecord -v speed=2 dev=0,6,0  -audio  track1.wav track2.wav...
shell> cdrecord -v speed=2 dev=0,6,0  -audio  track1.au  track2.au...
```

Tímto vytvoříte hudební CD, které bude mít 2sekundové pauzy mezi hudebními stopami. Jedním zvláštním formátem, který není přímo čitelný v cdrecord, je MPEG Layer 3. Pro konverzi souborů v tomto formátu na formát CDR můžete použít příkaz „mpg123 –cdr - track1.mp3 > track1.cdr“. Volba --cdr zajišťuje, aby byla stopa kódována v požadovaném formátu (viz výše). Starší verze mpg123 vyžadují -s namísto jednoduchého – pro zápis na standardní výstup. Jiný směr konverze z WAV do MPEG může být proveden pomocí LAME pro soubory WAV (extrahujte stopu z hudebního CD pomocí cdda2wav a zakódujte ji pomocí LAME do MP3).

Pro vytvoření CD-R z více souborů MP3 můžete použít následující sekvenci příkazů:

```
for I in *.mp3
do
    mpg123 --cdr - "$I" | cdrecord -audio -pad -nofix -
done
cdrecord -fix
```

V závislosti na rychlosti vašeho počítače možná budete chtít zpomalit zápis na „speed=1“ (volba cdrecord). Pokud používáte „speed=4“, musí být váš počítač schopen přehrávat soubor MP3 čtyřnásobnou rychlostí. mpg123 spotřebovává mnoho času procesoru! Pokud si nejste jisti, zkuste příkaz spustit naprázdno s -dummy (ponechá laser vypnutý).

DAO

Pokud se chcete zbavit mezer mezi hudebními stopami, musíte místo výše popsaného režimu TAO použít zápis DAO (celý disk najednou – disk-at-once). Podpora DAO je aktuálně lepší v cdrdao. Podívejte se prosím na detailní informace na jeho domovské stránce.

Pokud připravujete CD v režimu DAO, pak použijete jednolitý obraz (zvukový soubor) a budete řídit informace o stopách pomocí konfiguračního souboru.

```
CD_DA
TRACK AUDIO
FILE "live.wav" 0 5:0:0
INDEX 3:0:0
TRACK AUDIO
```

```
FILE "live.wav" 5:0:0 5:0:0
TRACK AUDIO
FILE "live.wav" 10:0:0 5:0:0
INDEX 2:0:0
```

Smíšené CD-ROMy

O tomto tématu se toho nedá říct mnoho. Prostě indikujte druh (následujících) obrazů volbami -data a -audio. Příklad:

```
cdrecord  -v dev=0,6,0  -data  cd_image  -audio  track*.cdr
```

Drahý Winfriede,...

Toto je část, která se obvykle nazývá „často kladené dotazy s odpověďmi“. Pokud máte problém s vaším partnerem, dětmi nebo psem, pošlete je, pokud se vztahují k vytváření CD-R nebo jsou jinak zábavné.

Jak citlivý je proces vypalování?

Zkuste to. Použijte volbu -dummy pro spuštění cdrecord naprázdno. Dělejte všechno, co byste jinak dělali, a sledujte, jestli to proces vypalování přežije.

Pokud jste cdrecord nakrmili přímo z mkisofs, pak náročné diskové procesy jako např. aktualizace databáze *locate*, snižují maximální rychlost toku dat a mohou poškodit CD. Je lepší zkontrolovat, že takové procesy nejsou spuštěny přes `cron`, `at` nebo `anacron`, pokud vypalujete CD-R na starších počítačích.

Má fragmentace souboru špatný vliv na propustnost?

Fragmentace souborů je obvykle tak malá, že její vliv není pozorovatelný. Samozřejmě můžete jednoduše vytvořit patologické případy fragmentace, které sníží propustnost vašich pevných disků pod 100 kbajtů/sekundu. Ale nedělejte to. :-) Ano, soubory na pevném disku se během let stanou fragmentovanými. Čím prázdnější disk, tím rychlejší systém souborů je. Vždy ponechte 10 % nebo 20 % volného místa a mělo by vše běžet s ohledem na vytváření CD dobře.

Pokud si nejste jisti, podívejte se na zprávy vypisované při spouštění. Procento fragmentace je hlášeno při kontrole systémů souborů. Tuto hodnotu můžete zkontrolovat pomocí velmi nebezpečného příkazu

```
shell> e2fsck -n  /dev/sda5        # '-n' je důležité!
[další řádky vynechány -- chyby ignorujte]
/dev/sda5: 73/12288 files (12.3% non-contiguous)
```

V tomto případě fragmentace vypadá velmi vysoká – ale v systému souborů je pouze 73 velmi malých souborů. Takže tato hodnota *není* alarmující.

Existuje experimentální nástroj, který se jmenuje e2defrag a slouží pro defragmentaci systémů souborů extended-2. Aktuální verze nepracuje dostatečně spolehlivě, aby se dala použít i jen pro privátní prostředí. Pokud chcete opravdu defragmentovat váš systém souborů, vytvořte záložní kopii (lepší jsou dvě kopie), procvičte si obnovování dat, pak vytvořte nový systém souborů (který zničí ten starý) a obnovte data. Toto je aktuálně nejbezpečnější technika.

Je možné ukládat obraz CD na systém souborů UMSDOS?

Ano. Jediný systém souborů, který není dostatečně rychlý a spolehlivý pro vytváření CD-ROMů, je *network filesystem* (*NFS*). Já sám jsem používal UMSDOS pro sdílení diskového prostoru mezi Linuxem a DOS/Win na PC (486/66) určeném pro vytváření CD-ROMů.

Neexistuje nějaký způsob pro obejití omezení ISO-9660?

Ano. Na CD můžete umístit systém souborů, který budete chtít. Ale jiné operační systémy než Linux nebudou umět s tímto CD pracovat. Zde je návod:

- Vytvořte prázdný soubor o velikosti 650 MB.

```
dd if=/dev/zero of="empty_file" bs=1024k count=650
```

- Vytvořte v tomto souboru systém souborů ext2

```
shell> /sbin/mke2fs  -b 2048  empty_file
empty_file is not a block special device.
Proceed anyway? (y,n) y
```

- Připojte tento prázdný soubor pomocí zařízení loopback (potřebujete jej připojit; viz výše).

```
mount -t ext2 -o loop=/dev/loop1 empty_file /mnt
```

- Zkoírujte soubory do /mnt a pak soubor odpojte.
- Použijte `cdrecord` na empty_file (který již není prázdný) jako by to byl obraz ISO-9660.

Pokud chcete vytvořit položku v `/etc/fstab` pro takové CD, pak vypněte kontrolu souboru zařízení při spouštění systému. Např.:

```
/dev/cdrom  /cdrom  ext2  defaults,ro  0 0
```

První 0 znamená „zazálohovat pomocí dump“ (záloha), druhá (=důležitá) znamená „při startu nekontrolovat chyby „ (fsck by selhal při kontrole chyb na CD).

Jak přečíst stopy z hudebních CD?

Existuje více softwarových balíčků. Nejnovějším je „cdparanoia“ a může být stažen z

http://www.xiph.org/paranoia/

Nebo můžete vyzkoušet kombinaci „cdda2wav“ a „sox.

`cdda2wav` vám umožňuje dosáhnout specifického intervalu (nebo celé stopy) z vašeho hudebního CD a převést jej do souboru .wav. `sox` konvertuje soubory WAV zpátky do (hudební CD) formátu cdda, aby je bylo možno zapsat na CD-R pomocí `cdrecord`. `sox` nepotřebujete nutně, pokud používáte nejnovější verzi `cdrecord`, protože má vestavěnou podporu souborů .au a .wav.

Jak zkontrolovat zařízení SCSI po spuštění?

Soubor `drivers/scsi/scsi.c` obsahuje informace

```
/*
 * Usage: echo "scsi add-single-device 0 1 2 3" >/proc/scsi/scsi
 * with  "0 1 2 3" replaced by your "Host Channel Id Lun".
 * Consider this feature BETA.
 *     CAUTION: This is not for hot plugging your peripherals. As
```

```
 *     SCSI was not designed for this you could damage your
 *     hardware !
 * However perhaps it is legal to switch on an
 * already connected device. It is perhaps not
 * guaranteed this device doesn't corrupt an ongoing data transfer.
 */
```

Pamatujte si, že by toto mělo být použito pouze pokud přidáváte zařízení SCSI na konec řetězce. Vložení nových zařízení SCSI do stávajícího řetězce ruší přidělování názvů zařízení (adresář /dev) a může zničit kompletní obsah vašeho pevného disku.

Některé verze jádra vůbec nemají rády znovuprohledání sběrnice SCSI a váš systém může solidně zamrznout, když to zkoušíte. Byli jste varováni.

Je možné vytvořit kopii datového CD 1:1?

Ano. Ale měli byste vědět, že když se vyskytnou chyby při čtení originálu (z důvodu prachu nebo škrábanců), projeví se to špatnou kopií. Pamatujte si, že obě metody selžou na hudebních CD! Na hudebních CD musíte použít cdrdao nebo cdda2wav.

První případ: máte zapisovací jednotku CD a oddělenou jednotku CD-ROM. Provedením příkazu

```
cdrecord -v dev=0,6,0 speed=2 -isosize /dev/scd0
```

načtete datový proud z jednotky CD-ROM připojené jako `/dev/scd0` a zapíšete jej přímo na zapisovací jednotku.

Druhý případ: nemáte oddělenou jednotku CD-ROM. V tomto případě musíte použít zapisovací jednotku nejprve pro načtení CD-ROMu:

```
dd if=/dev/scd0 of=cdimage
```

Tento příkaz přečte obsah CD-ROMu v zařízení /dev/scd0 a zapíše jej do souboru „cdimage“. Obsah tohoto souboru je ekvivalentem toho, co produkuje `mkisofs`, takže můžete postupovat tak, jak bylo popsáno výše v tomto dokumentu (což znamená předat soubor cdimage jako vstup cdrecord). Pokud chcete vidět ukazatel průběhu a další umělecké věci, také můžete použít `sdd` od Jörga Schillingse.

V případě, že nastávají chyby, nainstalujte nejnovější verzi cdrecord, která obsahuje nástroj nazvaný „readcd“ (je pod misc/). Umožňuje stejný výsledek jako sdd, ale čte sektory na CD-ROM v případě chyb několikrát.

Může Linux načítat CD-ROMy Joliet? (zastaralá odpověď)

Ano. Novější jádra (2.0.36 a nadcházející 2.2) mají vestavěnou podporu pro formát Joliet. Pamatujte si, že musíte použít ve vašem /etc/fstab obě volby: klíčová slova iso9660 a joliet (druhý je opravdu rozšířením). Více informací najdete na adrese http://www-plateau.cs.berkeley.edu/people/chaffee/joliet.html.

Jak mám načítat/připojovat CD-ROMy pomocí zapisovací jednotky?

Stejně jako to děláte u běžných jednotek CD-ROM. Není v tom žádný trik. Pamatujte si, že musíte použít zařízení scd (SCSI CD-ROM) pro připojení CD-ROMů pro čtení, i když máte CD-ROM

ATAPI (vzpomeňte si, že jste konfigurovali vaše zařízení ATAPI tak, aby se chovalo jako SCSI). Příklad /etc/fstab:

```
/dev/scd0  /cdrom  iso9660  ro,user,noauto  0  0
```

Jak uložit více dat na CD-R?

Použijte `bzip2` místo jiných komprimačních programů jako `gzip` nebo `pkzip`. Ušetří vám to až 30 % místa na disku pro větší soubory (>100kb). Můžete jej stáhnout na adrese

http://www.muraroa.demon.co.uk/

Místo vytvoření skutečného hudebního CD můžete volitelně převést vaše hudební soubory WAV na hudební soubory MP3 a uložit je na souborový systém ISO-9660 jako běžné soubory. Obvykle MPEG III poskytuje kompresi 1:10. Samozřejmě, většina přehrávačů CD neumí načítat soubory... to je nevýhoda. Na druhou stranu, proč nespouštět hudbu na vaší příští párty z pevného disku? 18 Gbajtů stačí na 3000-4000 titulů. :-)

Software pro komprimaci do MPEG III je k dispozici na adrese

http://www.sulaco.org/mp3/

Přehrávač MPEG III je k dispozici na adrese

http://www.mpg123.org/

Pro záznam řeči byste mohli chtít vyzkoušet snížit jeho velikost pomocí `shorten` nebo „GSM lossy speech compression“:

ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/

http://kbs.cs.tu-berlin.de/~jutta/toast.html

Jak vytvářet spustitelné CD-ROMy?

Musíte mít spustitelnou disketu 1,44 MB. Vytvořte přesný obraz této diskety spuštěním příkazu

```
dd if=/dev/fd0 of=boot.img bs=18k
```

Umístěte obraz této diskety do adresáře, který obsahuje sadu vašich souborů (nebo do jeho podadresáře, jak budete chtít). Řekněte mkisofs o tomto souboru volbou ‘-b’ a také použijte ‘-c’. Detailnější informace najdete v souboru README.eltorito v distribuci mkisofs.

Zajímavá aplikace pro vlastní spustitelná CD je nezavirovaný systém DOS nebo Windows. Šetří to peníze za pevné disky (pokud máte síť a používáte sambu pro umísťování uživatelských dat na souborový server). V německém počítačovém časopisu c't o tom byl článek v čísle 11/99, strana 206 (http://www.heise.de/).

Detaily o spustitelném CD-ROMu RedHat jsou k dispozici na adrese http://members.bellatlantic.net/~smithrod/rhjol-technical.html.

Jak lze zapisovat na CD-ROMy jako na pevný disk?

Existuje souborový systém *overlay* dostupný pro Linux, který je připojen přes CD-ROM a brání všem operacím zápisu. Nové a upravené soubory jsou uloženy jinde, ale pro uživatele to vypadá, jako by se obsah CD-ROM měnil. Více informací najdete na adrese http://home.att.net/~artnaseef/ovlfs/ovlfs.html.

Pokud to vašim potřebám nevyhovuje: počkejte si, až bude systém souborů UDF podporován Linuxem nebo pomozte při jeho vývoji (viz. http://trylinux.com/projects/udf/. V tomto okamžiku je podporováno z důvodu omezení v ovladačích CD-ROM v jádře Linuxu pouze čtení médií CD.

Je možné použít více zapisovacích jednotek najednou?

Ano. Je hlášeno, že pracují alespoň 3 zapisovací jednotky na plné rychlosti (6x) na PC s 233 MHz, jedinou sběrnicí SCSI a jádrem 2.2.12. Potřebujete novou verzi jádra Linuxu (2.2.12 nebo vyšší).

Co Solaris, *BSD, AIX, HP-UX, atd.? Je má varianta Unixu podporována?

Pouze kapitola 2 je určena pouze pro Linux. Kapitoly 3 a 4 můžete použít i když máte operační systém jiné rodiny než Linux. Podívejte se prosím do souborů README.NetBSD, README.aix, README.hpux, README.next, README.solaris, README.sunos, README.vms nebo README.xxxBSD v distribuci cdrecord.

Pravděpodobně ano. Zkompilujte cdrecord pro vaši platformu a spusťte příkaz „cdrecord -scanbus“. Přečtěte si soubor README.* pro váš Unix distribuovaný se zdroji cdrecord. Samozřejmě, ne všechny varianty Unixu umí číst rozšíření RockRidge, Joliet nebo HFS na vašem nově vytvořeném CD-R.

Kam trvale uložit místní konfiguraci?

Máte dvě možnosti. Buď použijete vestavěný konfigurační soubor pro cdrecord, nebo použijete příkazový interpret jako v ukázce dole. Tento skript příkazového interpretu načte konfigurační soubor, který vypíše možnosti a parametry pro cdrecord řádek po řádku. Názvy jsou úplně stejné jako na příkazové řádce, ale bez počáteční čárky. Komentáře jsou povoleny. Příklad:

```
# buďte upovídaní
v
# nastavte rychlost zapisovací jednotky
speed=2
# parametry zařízení ve tvaru BUS,ID,LUN
dev=0,6,0
```

Konfigurační soubory pro obal patří do /etc/cdrecord/ a musí být uvedeny na příkazovém řádku. Příklad: pokud se chcete odkazovat na konfiguraci /etc/cdrecord/mywriter.cfg, pak můžete spustit příkaz „cdrecord.sh mywriter.cfg -audio track1...“. Všechno za mywrite.cfg je předáno cdrecord.

```
#! /bin/bash

CFGDIR="/etc/cdrecord"

CFG="$1"
shift
ARGS_LEFT="$@"

if [ ! -f "$CFGDIR/$CFG" ]
then
    echo "Konfigurační soubor $CFGDIR/$CFG nebyl nalezen. Konec."
    exit 1
fi
```

```
while read LINE
do
    case $LINE in
        \#*|"") continue;;
    esac
    old_IFS="$IFS"
    IFS="$IFS="
    set -- $LINE
    IFS="$old_IFS"
    O_NAME="$1"
    O_VALUE=""
    while shift
    do
        case $1 in
            "") continue;;
        esac
        O_VALUE="$1"
    done

    if [ -z "$O_VALUE" ]
    then
        O_CDRECORD="$O_CDRECORD -$O_NAME "
        continue
    fi
    O_CDRECORD="$O_CDRECORD $O_NAME=$O_VALUE "

done < "$CFGDIR/$CFG"

set -x    #DEBUG
exec cdrecord $O_CDRECORD $ARGS_LEFT
echo "Spuštění programu selhalo."
```

Jak získat informace o CD?

Někde před prvními 32 k CD je umístěn blok informací o CD. Tyto informace můžete načíst pomocí následujícího skriptu shellu:

```
#! /bin/bash

RD=/dev/cdrom
for i in 32768,7   32776,32 32808,32 32958,128 33086,128 33214,128 \
         33342,128 33470,32 33581,16 33598,16  33615,16  33632,16
do
    old_IFS="$IFS"
    IFS=","
    set -- $i
    IFS="$old_IFS"
    OFFSET=$1
    LENGTH=$2
    echo "*'dd if=$RD bs=1 skip=$OFFSET count=$LENGTH   2> /dev/null'#"
done
```

Co přepisování?

Když přepisujete média CD-RW, uveďte parametr blank=fast u cdrecord. To je vše. Detailní informace o tomto parametru najdete v manuálových stránkách pro cdrecord.

Jak vytvářet multi-session CD?

Nejprve ze všeho musí být obraz pro multi-session CD formátován pomocí souborového systému ISO-9660 s rozšířením RockRidge. A musíte použít volbu -multi u cdrecord pokud chcete přidat další data. Takže alespoň u prvního zápisu musíte zadat volbu -multi.

Některé zapisovací jednotky nemají podporu pro CD-ROM XA mode 2 nebo session-at-once (SAO), takže je potřeba zadat přepínač -data u cdrecord na příkazovém řádku.

Generování obrazů pro druhý a následující záznamy je trošku komplikovanější. Mkisofs musí vědět, kde začíná volné místo na CD-R. Tuto informaci je možno získat pomocí volby -msinfo u cdrecord (viz. níže uvedený příklad).

```
shell> NEXT_TRACK='cdrecord -msinfo dev=0,6,0'
shell> echo $NEXT_TRACK
shell> mkisofs -R -o cd_image2 -C $NEXT_TRACK -M /dev/scd5
         private_collection/
```

Více informací najdete v souboru README.multi, který je distribuován s cdrecord.

Mám použít adaptér SCSI dodaný se zapisovací jednotkou?

Hlášeno e-mailem: Většina dokumentů k zapisovacím jednotkám doporučuje použít oddělenou sběrnici SCSI, pokud zapisujete z jednotky CD-ROM na zapisovací jednotku a já sám jsem to viděl v následující variantě:

Karta Adaptec 2940UW SCSI, 24x SCSI CD-ROM a zapisovací jednotka 4x4 SCSI. Když jsem dostal zapisovací jednotku, byla s ní dodána karta ISA SCSI, která měla pracovat pouze s jedním zařízením. Myslel jsem, že ji zahodím a použiji mou lepší kartu Adaptec pro všechna zařízení. Všiml jsem si, že to bylo náchylnější na podtečení vyrovnávací paměti při zápisu rychlostí 4x, ale jakmile jsem zkusil zapojit kartu ISA SCSI, neměl jsem žádné problémy. Znám další dva lidi (oba používají karty adaptec 2940), kteří se setkali s přesně stejnými symptomy, obvykle při zápisu z jednotky CD-ROM na zapisovací jednotku. I když jsem se nikdy nesetkal s problémem při vypalování z pevného disku na zapisovací jednotku na stejné sběrnici.

Jak pálit přes síť?

Obvykle je přenos souborů přes FTP dostatečně rychlý, i když jen přes 10Mb ethernet, pro krmení zapisovací jednotky se čtyřnásobnou (4x) rychlostí. Klienta FTP a cdrecord můžete propojit skrze fifo. Nejprve vytvořte fifo s názvem cdimage:

```
mkfifo cdimage
ftp other.host.org
get cdimg cdimage
```

Pak zpracujte cdimage jako obyčejný soubor, tj. spusťte následující příkaz:

```
cdrecord dev=0,1,0 speed=2 cdimage
```

Váš klient FTP si všimne, že cdrecord chce číst ze souboru a začne přenášet data z hostitele FTP.

Slyším lupnutí nebo cvaknutí na konci každé stopy.

Abyste se lupnutí zbavili, musíte použít režim DAO (disk-at-once).

Jak se dá nastavit, aby mohl uživatel vypalovat CD bez toho, aby musel být root?

Můžete přidat bit setuid u spustitelného souboru cdrecord. Samozřejmě zde může být bezpečnostní riziko. Jednoduché nastavení oprávnění pro soubory zařízení nepomůže, protože cdrecord spouští privilegované příkazy skrze obecná rozhraní SCSI.

```
which cdrecord
chown root.root /usr/bin/cdrecord
chmod 4111 /usr/bin/cdrecord
```

Kde získám standardy „Yellow Book“ a „Orange Book“?

Tištěné specifikace můžete získat od společnosti Philips a jsou drahé.

Hledal jsem informace o vypalování Video-CD pod Linuxem.

Zde můžete nalézt nástroje pro vytváření videí MPEG a Video-CD: http://www.mainconcept.de/

http://www.johanni.de/munich-vision/vcd/

Korektní lidé se zmiňují o nástrojích Berkeley a jiných strategiích YUV. Jejich použití je komplikované, zabírá mnoho času a místa na pevném disku a neumožňují hudební stopy. Doporučuji použít aplikace koncepčně vyšší úrovně než výše uvedené.

Co je jednodušší nastavit, IDE nebo SCSI?

Zapisovací jednotky SCSI se trochu jednodušeji nastavují pro zápis na CD pod Linuxem. A je hlášeno, že mají lepší opravy chyb. Jestli to vyvažuje vyšší cenu, nelze obecně říci.

Jak mohu přepálit CD pomocí {cdrecord,cdrdao}?

Přepalování CD-R není nic speciálního. Je na vaše vlastní riziko, že budou data umístěna na CD-ROM, ale to je vše. V softwaru pro Linux neexistují žádná omezení 650 Mbajtů.

Co cdrecord udělá, když se zastaví vstup z roury?

Dokončí zápis. Takže můžete prostě spojit váš oblíbený nástroj pro zálohování s cdrecord pomocí roury, jako v „bru -size=640m -f - | cdrecord dev=0,1,0 speed=2 -“. Musíte věnovat speciální péči zálohovacímu nástroji, pokud záloha zabírá více CD-R.

Existuje ekvivalent ignore=hdX pro emulaci ide-scsi?

Neznám žádný způsob, ale někdo by mohl tuto funkci přidat do zdrojů jádra Linuxu.

Kolikrát mohu znovu použít CD-RW než se poškodí?

Dobrá otázka.

Který formát bych měl zvolit pro na platformě nezávislý CD-ROM?

CD-ROM by měl být čitelný pro všechny systémy pouze při použití holého formátu ISO 9660. To znamená stupidní názvy souborů 8+3 ze starého MS-DOSu a bez rozšíření HFS (Macintosh), Joliet (Microsoft) nebo RockRidge (novější Unixy). Neexistuje rozšíření pro delší názvy souborů, které by byly čitelné všemi operačními systémy.

Je možné multi-session pro hudební stopy?

Přehrávače hudebních CD umí pracovat pouze s hudebními stopami uloženými v první session. Jinými slovy, nemůžete přidávat hudební stopy pomocí dalších sessions. Samozřejmě, zápis datových stop do druhé session je efektivně skryje před přehrávači hudebních CD. Tímto způsobem zabráníte tomu, abyste měli tichou stopu na vašem CD ve smíšeném režimu (smíšení hudby a dat).

Jaké hardwarové prostředky potřebuji? Stačí staré Pentium?

Otázka závisí na vašich požadavcích. Pokud potřebujete důvod pro zakoupení nového počítače, je zde odpověď od mezinárodní asociace výrobců počítačů: Pro cokoliv, co chcete dělat, potřebujete procesor s 800 MHz. Protože nepůjde do stávající základní desky, potřebujete také novou základní desku. Nejjednodušším řešením je zakoupení typické kompletní nabídky, kterou vidíte v reklamách v televizi. Ignorujte prosím zbytek této části.

Nyní případ, kdy chcete rozumnou odpověď: Vytvořil jsem úspěšně více CD-ROMů na „486“ s 66 Mhz. Ačkoliv to MS už nepovažuje za PC a doporučuje na něm spouštět jejich verzi CE (viděno na CeBITu), Linux běží stejně dobře na předchůdcích procesoru Pentium a dokonce na nich umí vytvářet CD. Jednoduše můžete zjistit zda výkon vašeho hardwaru stačí pro vytváření CD-ROMů tím, že to vyzkoušíte. Prostě přidejte na příkazovém řádku přepínač -dummy pro spouštění cdrecord a laser zůstane vypnutý. Sledujte proces vypalování.

Odstraňování problémů

Pamatujte, že vadná CD můžete stále používat jako podložky pod nápoje :-)

Nepracuje pod Linuxem

Nejprve prosím zkontrolujte, že zapisovací jednotka pracuje v softwaru, se kterým byla dodána (=pod jiným operačním systémem). Konkrétně:

- Najde řadič zapisovací jednotku jako zařízení SCSI?
- Najde software ovladače zapisovací jednotku?
- Je možné vytvořit CD pomocí dodaného softwaru?

Pokud nepracuje ani s dodaným softwarem, je zde konflikt hardwaru nebo vadný hardware. Pokud pracuje a používáte loadlin pro spouštění systému Linux, pak je problém v loadlin. Loadlin dělá „teplý start“ s většinou hardwaru již inicializovanou a to může zmást jádro Linuxu.

Chybová zpráva: No read access for ‘dev=0,6,0’

Pod Linuxem jsou některé verze knihovny C nekompatibilní (chybné), takže aplikace sestavená pro jednu verzi nebude pracovat s jinou. Zde je příklad pro chybu vyvolanou předkompilovanými binárními soubory:

```
[root@Blue /dev]# cdrecord -eject dev=0,6,0
cdrecord: No such file or directory. No read access for 'dev=0,6,0'.
```

Řešením je instalace novější knihovny C.

Nepracuje pod DOSem a spol.

Zkuste použít Linux. Instalace a konfigurace ovladačů SCSI pro DOS je peklo. Linux že je příliš komplikovaný? Ha!

Chyby SCSI v průběhu vypalování

Tyto chyby jsou nejpravděpodobněji způsobeny

- chybějící funkcí odpojování/znovupřipojování na sběrnici SCSI
- nedostatečně chlazeným hardwarem
- vadným hardwarem (mělo by být detekováno v 5.1.)

Za různých okolností se zařízení SCSI odpojují a znovu připojují (elektronicky) ze sběrnice SCSI. Pokud tato funkce není k dispozici (zkontrolujte řadič a parametry jádra), mohou mít některé zapisovací jednotky problémy v průběhu vypalování nebo zakončování CD-R.

Zejména ovladač SCSI NCR 53c7,8xx má tuto funkci implicitně vypnutou, takže byste ji měli nejdříve zkontrolovat:

```
NCR53c7,8xx SCSI support                  [N/y/m/?] y
   always negotiate synchronous transfers [N/y/?] (NEW) n
   allow FAST-SCSI [10MHz]                [N/y/?] (NEW) y
   allow DISCONNECT                       [N/y/?] (NEW) y
```

Chyby médií

Pokud cdrecord hlásí chyby médií ve tvaru „Sense Key: ... Medium Error, Segment ...“, pak médium není prázdné. Pokud používáte CD-RW, zkuste přepnout blank=fast na spolehlivější blank=all. Pokud používáte CD-R, pak se ujistěte, že CD-R nikdy dříve nevidělo zapisovací jednotku, nebo vyzkoušejte CD od jiného výrobce.

Nově vytvořená CD nejsou v některých přehrávačích čitelná

Někteří lidé hlásili problémy s přehráváním jimi vytvořených CD. Velmi staré audio přehrávače nebo autopřehrávače mohou mít problémy s CD-R, ačkoliv je to extrémně vzácné. Celkem časté jsou problémy s CD-RW, protože neodráží laserový paprsek tak dobře jako CD-R a lisované „stříbrné“ disky.

Můj skener přestal pracovat poté, co jsem zavedl modul ide-scsi

Vložením emulace hostitelského adaptéru SCSI se mění názvy zařízení SCSI. Pokud byl váš skener předtím /dev/sg0, mohl by být nyní /dev/sg1 nebo /dev/sg2. Výborní vývojáři jádra v minulosti nemysleli, že je to úplně na mrtvici a zakázali řešení jako devfs. Ale to je jiný příběh. Nejprve byste měli vyzkoušet nastavení odkazu /dev/scanner tak, aby ukazoval na platné generické zařízení SCSI. Příklady:

```
cd /dev
ls -l scanner     # zobrazuje aktuální nastavení
ln -sf sg2 scanner
# otestuj scanner
ln -sf sg1 scanner
# otestuj the scanner
# atd.
```

Vývojáři aplikací by měli hodně myslet na podporu tohoto nebezpečného a k chybám náchylného schématu přidělování názvů. Zvažte prosím alespoň použití přechodných řešení jako jsou parametry SCSI používané nástrojem cdrecord.

Zásluhy

Mnoho díků patří čtenářům tohoto HOWTO, kteří přispívali aktivně do jeho obsahu. Jelikož nemám přístup k zapisovací jednotce mnoho let, byly pro mne zprávy o nastaveních ze skutečné praxe a zkušenosti opravdu cenné.

Doug Alcorn <doug@lathi.net>

pomáhal vylepšit manipulaci s novějšími jádry

Kalle Andersson <kalle@sslug.dk>

Jak zapisovat hudební CD přímo z MP3.

Alan Brown <alan@manawatu.net.nz>

Rick Cochran <rick@msc.cornell.edu>

podněty k odpojení/znovupřipojení v ovladači ncr implicitně vypnutém

Robert Doolittle <bob.doolittle@sun.com>

dobré argumenty pro vypuštění cdwrite z tohoto HOWTO

Markus Dickebohm <m.dickebohm@uni-koeln.de>

Thomas Duffy <tduffy@sgi.com>

Hlavní vyčištění syntaxe a pravopisu

Dave Forrest <dforrest@virginia.edu>

Oprava problémů s pojmy adaptéru

Jos van Geffen <jos@tnj.phys.tue.nl>

oznámil problém v kapitole 4.

Bernhard Gubanka <beg@ipp-garching.mpg.de>

Upozornil na potřebu nové verze mount pro práci se zařízením loopback

Stephen Harris <sweh@mpn.com>

Přispěl poznámkou k vytváření hudebních CD

Janne Himanka <shem@oyt.oulu.fi>

Odkaz na záplatu jádra pro čtení CD-ROMů Joliet

Stephan Noy <stnoy@mi.uni-koeln.de>

Informace a zkušenosti s vytvářením hudebních CD

Don H. Olive <don@andromeda.campbellsvil.edu>

URL nástroje mkhybrid

Jesper Pedersen <jews@imada.ou.dk>

Pierre Pfister <pp@uplift.fr>

Pomáhali při vývoji návodu na kopie 1:1.

Daniel A. Quist <dquist@cs.nmt.edu>

Informace o IDE CD-R a novějších verzích jádra

Martti.Rahkila@hut.fi

Oznámil problém s přednastavenými zapisovacími jednotkami při spouštění pomocí loadlin.

Dale Scheetz <dwarf@polaris.net>

Joerg Schilling <schilling@fokus.gmd.de>

Mnoho informací o cdrecord

Martin Schulze <joey@Infodrom.North.DE>

Podal informace o diskusní skupině cdwrite

Gerald C Snyder <gcsnyd@loop.com>

Testoval zápis CD se systémem souborů ext2 (viz. 4.4)

Art Stone <stone@math.ubc.ca>

Měl nápad k umisťování systémů souborů jiných než ISO-9660 na CD

The Sheepy One <kero@escape.com>

Doporučil používat vadné CD-ROMy jako tácky pod nápoje

Erwin Zoer <ezoer@wxs.nl>

Kromě toho bych rád poděkoval následujícím lidem za nahlášení chyb v pravopisu: Bartosz Maruszewski <B.Maruszewski@zsmeie.torun.pl>, Alessandro Rubini <rubini@prosa.it>, Ian Stirling <ian@opus131.com>, Brian H. Toby.

Konec Linux CD-Writing HOWTO. (Už můžete přestat číst)

ČÁST V

GNU general public licence – český překlad

Preambule

Licence pro většinu programového vybavení jsou navrženy tak, že vám odebírají právo jeho volného sdílení a úprav. Smyslem Obecné veřejné licence GNU je naproti tomu zaručit volnost sdílení a úpravy volného programového vybavení – pro zajištění volného přístupu k tomuto programovému vybavení pro všechny jeho uživatele. Tato Obecná veřejná licence GNU se vztahuje na většinu programového vybavení nadace Free Software Foundation a na jakýkoli jiný program, jehož autor se přikloní k jejímu používání. (Některé další programové vybavení od Free Software Foundation je namísto toho pokryto Obecnou knihovní veřejnou licencí GNU.) Můžete ji rovněž použít pro své programy.

Pokud mluvíme o volném programovém vybavení, máme na mysli volnost, nikoliv cenu. Naše Obecná veřejná licence je navržena pro zajištění toho, že můžete volně šířit kopie volného programového vybavení (a účtovat si za tuto službu, pokud chcete), že obdržíte zdrojový kód anebo jej můžete získat, pokud chcete, že můžete tento software měnit nebo jeho části použít v nových programech; a že víte, že tyto věci smíte dělat.

Abychom mohli vaše práva chránit, musíme vytvořit omezení, která zakáží komukoli vám tato práva odepírat nebo vás žádat, abyste se těchto práv vzdal. Tato omezení se promítají do jistých povinností, kterým musíte dostát, pokud šíříte kopie dotyčného programového vybavení anebo ho modifikujete.

Například, šíříte-li kopie takového programu, ať již zdarma nebo za poplatek, musíte poskytnout příjemcům všechna práva, která máte sám. Musíte zaručit, že příjemci rovněž dostanou anebo mohou získat zdrojový kód. A musíte jim ukázat tyto podmínky, aby znali svá práva.

Vaše práva chráníme ve dvou krocích: (1) autorizací programového vybavení, a (2) nabídkou této licence, která vám dává právoplatné svolení ke kopírování, šíření a modifikaci programového vybavení.

Kvůli ochraně každého autora i nás samotných chceme zaručit, aby každý chápal skutečnost, že pro volné programové vybavení nejsou žádné záruky. Je-li programové vybavení někým jiným modifikováno a posláno dále, chceme, aby příjemci věděli, že to, co mají, není originál, takže jakékoliv problémy vnesené jinými se neodrazí na reputaci původních autorů.

Konečně, každý volný program je neustále ohrožen softwareovými patenty. Přejeme si zamezit nebezpečí, že redistributoři volného programu obdrží samostatně patentová osvědčení a tím učiní program vázaným. Abychom tomu zamezili, deklarovali jsme, že každý patent musí být buď vydán s tím, že umožňuje volné užití, anebo nesmí být vydán vůbec.

Přesná ustanovení a podmínky pro kopírování, šíření a modifikaci jsou uvedeny dále.

Ustanovení a podmínky pro kopírování, distribuci a modifikaci

Tato licence se vztahuje na kterýkoliv program či jiné dílo, které obsahuje zmínku, umístěnou v něm držitelem autorských práv, o tom, že dílo může být šířeno podle ustanovení Obecné veřejné licence GNU. V dalším textu znamená „Program“ každý takový program nebo dílo a „dílo založené na Programu“’ znamená buď Program samotný anebo každé jiné dílo z něj odvozené, které podléhá autorskému zákonu: tím se míní dílo obsahující Program nebo jeho část, buď doslovně anebo s modifikacemi, popřípadě v překladu do jiného jazyka. (Nadále je překlad zahrnován bez omezení pod pojmem „modifikace“ .) Každý uživatel licence je označován jako „vy“ .

Jiné činnosti než kopírování, šíření a modifikace nejsou pokryty touto licencí; sahají mimo její rámec. Akt spuštění programu není omezen a výstup z Programu je pokryt pouze tehdy, jestliže obsah výstupu tvoří dílo založené na Programu (nezávisle na tom, zda bylo vytvořeno činností Programu). Posouzení platnosti předchozí věty závisí na tom, co Program dělá.

1. Smíte kopírovat a šířit doslovné kopie zdrojového kódu Programu tak, jak jste jej obdržel, a na libovolém médiu, za předpokladu, že na každé kopii viditelně a náležitě zveřejníte zmínku o autorských právech a absenci záruky; ponecháte nedotčené všechny zmínky vztahující se k této licenci a k absenci záruky; a dáte každému příjemci spolu s Programem kopii této licence.

   Za fyzický akt přenesení kopie můžete žádat poplatek a podle vlastního uvážení můžete nabídnout za poplatek záruční ochranu.

2. Můžete modifikovat vaši kopii či kopie Programu anebo kterékoliv jeho části, a tak vytvořit dílo založené na Programu, a kopírovat a rozšiřovat takové modifikace či dílo podle platné podmínky sekce 1, za předpokladu, že splníte všechny tyto podmínky:

   a) Modifikované soubory musíte opatřit zřetelnou zmínkou uvádějící, že jste soubory změnil a datum každé změny.

   b) Musíte umožnit, aby jakékoliv vámi publikované dílo či rozšiřované dílo, které obsahuje zcela nebo jen zčásti Program nebo jakoukoli jeho část, popřípadě je z Programu nebo jeho části odvozeno, mohlo být jako celek bezplatně poskytnuto každé třetí osobě v souladu s ustanoveními této licence.

   c) Pokud modifikovaný program pracuje tak, že čte interaktivně povely, musíte zjistit, že při nejběžnějším způsobu jeho spuštění vytiskne nebo zobrazí hlášení zahrnující příslušnou zmínku o autorském právu a uvede, že neexistuje žádná záruka (nebo popřípadě, že záruku poskytujete vy), a že uživatelé mohou za těchto podmínek Program redistribuovat, a musí uživateli sdělit, jakým způsobem může nahlédnout do kopie této licence. (Výjimka: v případě, že sám program je interaktivní, avšak žádné takové hlášení nevypisuje, nepožaduje se, aby vaše dílo založené na Programu takové hlášení vypisovalo.)

   Tyto požadavky se vztahují k modifikovanému dílu jako celku. Pokud lze identifikovat části takového díla, které zřejmě nejsou odvozeny z Programu a mohou být samy o sobě rozumně považovány za nezávislá a samostatná díla, pak se tato licence a její ustanovení nevztahují na tyto části, jsou-li šířeny jako nezávislá díla. Avšak jakmile tyto části rozšiřujete jako části celku, jímž je dílo založené na Programu, musí být rozšiřování tohoto celku podřízeno ustanovení této licence tak, že povolení poskytnutá dalším uživatelům se rozšíří na celé dílo, tedy na všechny jeho části bez ohledu na to, kdo kterou část napsal.

   Smyslem tohoto paragrafu tedy není získání práv na dílo zcela napsané vámi ani popírání vašich práv vůči němu; skutečným smyslem je výkon práva na řízení distribuce odvozených nebo kolektivních děl založených na Programu.

   Pouhé spojení jiného díla, jež není na Programu založeno, s Programem (anebo dílem založeným na Programu) na paměťovém nebo distribučním médiu neuvazuje toto jiné dílo do působnosti této licence.

3. Můžete kopírovat a rozšiřovat Program (nebo dílo na něm založené, viz sekce 2 v objektové anebo spustitelné podobě podle ustanovení sekcí 1 a 2 výše, pokud splníte některou z následujících náležitostí:

a) Doprovodíte jej zdrojovým kódem ve strojově čitelné formě. Zdrojový kód musí být rozšiřován podle ustanovení sekcí 1 a 2, a to na médiu běžně používaném pro výměnu programového vybavení; nebo

b) Doprovodíte je písemnou nabídkou nejméně na tři roky, podle níž poskytnete jakékoli třetí straně, za poplatek nepřevyšující vaše výdaje vynaložené na fyzickou výrobu zdrojové distribuce, kompletní strojově čitelnou kopii odpovídající zdrojovému kódu, jenž musí být šířen podle ustanovení sekcí 1 a 2 na médiu běžně používaném pro výměnu programového vybavení; nebo

c) Doprovodíte jej informacemi, které jste dostal ohledně nabídky na poskytnutí zdrojového kódu. (Tato alternativa je povolena jen pro nekomerční šíření a jenom tehdy, pokud jste obdržel program v objektovém nebo spustitelném tvaru spolu s takovou nabídkou, v souladu s položkou b výše.)

Zdrojový kód k dílu je nejvhodnější formou díla z hlediska jeho případných modifikací. Pro dílo ve spustitelném tvaru znamená úplný zdrojový kód veškerý zdrojový kód pro všechny moduly, které obsahuje, plus jakékoli další soubory pro definici rozhraní, plus dávkové soubory potřebné pro kompilaci a instalaci spustitelného programu. Zvláštní výjimkou jsou však ty programové komponenty, které jsou normálně šířeny (buď ve zdrojové nebo binární formě) s hlavními součástmi operačního systému, na němž spustitelný program běží (tj. s překladačem, jádrem apod.). Tyto komponenty nemusí být šířeny se zdrojovým kódem, pokud ovšem komponenta sama nedoprovází spustitelnou podobu díla.

Je-li šíření objektového nebo spustitelného kódu činěno nabídkou přístupu ke kopírování z určitého místa, potom se za distribuci zdrojového kódu počítá i nabídnutí ekvivalentního přístupu ke kopírování zdrojového kódu ze stejného místa, byť přitom nejsou třetí strany nuceny ke zkopírování zdrojového kódu spolu s objektovým.

4. Nesmíte kopírovat, modifikovat, poskytovat sublicence anebo šířit Program jiným způsobem než výslovně uvedeným v této licenci. Jakýkoli jiný pokus o kopírování, modifikování, poskytnutí sublicence anebo šíření Programu je neplatný a automaticky ukončí vaše práva daná touto licencí. Strany, které od vás obdržely kopie anebo práva v souladu s touto licencí, však nemají své licence ukončeny, dokud se jim plně podřizují.

5. Není vaší povinností tuto licenci přijmout, protože jste ji nepodepsal. Nic jiného vám však nedává možnost kopírovat nebo šířit Program nebo odvozená díla. V případě, že tuto licenci nepřijmete, jsou tyto činnosti zákonem zakázány. Tím pádem modifikací anebo šířením Programu (anebo každého díla založeného na Programu) vyjadřujete své podřízení se licenci a všem jejím ustanovením a podmínkám pro kopírování, modifikování a šíření Programu a děl na něm založených.

6. Pokaždé, když redistribuujete Program (nebo dílo založené na Programu), získává příjemce od původního držitele licence právo kopírovat, modifikovat a šířit Program v souladu s těmito ustanoveními a podmínkami. Nesmíte klást žádné překážky výkonu zde zaručených příjemcových práv. Nejste odpovědný za vymáhání dodržování této licence třetími stranami.

7. Jsou-li vám z rozhodnutí soudu, obviněním z porušení patentu nebo z jakéhokoli jiného důvodu (nejen v souvislosti s patenty) uloženy takové podmínky (ať již příkazem soudu, smlouvou nebo jinak), které se vylučují s podmínkami této licence, nejste tím osvobozen od podmínek této licence. Pokud nemůžete šířit Program tak, abyste vyhověl zároveň svým závazkům vyplývajícím z této licence a jiným platným zůvazkům, nesmíte jej v důsledku toho šířit vůbec. Pokud by například patentové osvědčení nepovolovalo bezplatnou redis-

tribuci Programu všemi, kdo vaším přičiněním získají přímo nebo nepřímo jeho kopie, pak by jediný možný způsob jak vyhovět zároveň patentovému osvědčení i této licenci spočíval v ukončení distribuce Programu.

Pokud by se za nějakých specifických okolností jevila některá část tohoto paragrafu jako neplatná nebo nevynutitelná, považuje se za směrodatnou rovnováha vyjádřená tímto paragrafem a paragraf jako celek se považuje za směrodatný za jiných okolností.

Smyslem tohoto paragrafu není navádět vás k porušování patentů či jiných ustanovení autorského práva, anebo tato ustanovení zpochybňovat; jediným jeho smyslem je ochrana integrity systému šíření volného programového vybavení, které je podloženo veřejnými licenčními předpisy. Mnozí lidé poskytli své příspěvky do širokého okruhu programového vybavení šířeného tímto systémem, spolehnuvše na jeho důsledné uplatňování; záleží na autorovi/dárci, aby rozhodl, zda si přeje šířit programové vybavení pomocí nějakého jiného systému a žádný uživatel licence nemůže takové rozhodnutí zpochybňovat.

Smyslem tohoto paragrafu je zevrubně osvětlit to, co je považováno za důsledek plynoucí ze zbytku této licence.

8. Pokud je šíření či použití Programu v některých zemích omezeno buď patenty anebo autorsky chráněnými rozhraními, může držitel původních autorských práv, který svěřuje Program do působnosti této licence, přidat výslovně omezení pro geografické šíření, vylučující takové země, takže šíření je povoleno jen v těch zemích nebo mezi těmi zeměmi, které nejsou tímto způsobem vyloučeny. Tato licence zahrnuje v tomto případě takové omezení přesně tak, jako bylo zapsáno v textu této licence.
9. Free Software Foundation může čas od času vydávat upravené nebo nové verze Obecné veřejné licence. Takové nové verze se budou svým duchem podobat současné verzi, v konkrétních věcech se však mohou lišit s ohledem na nové problémy či zájmy.

   Každé nové verzi je přiděleno rozlišující číslo verze. Pokud Program specifikuje číslo verze, která se na něj vztahuje, a „všechny následující verze“ , můžete se podle uvážení řídit ustanoveními a podmínkami buďto oné konkrétní verze anebo kterékoliv následující verze, kterou vydala Free Software Foundation. Jestliže Program nespecifikuje číslo verze této licence, můžete si vybrat libovolnou verzi, kterou kdy Free Software Foundation vydala.
10. Pokud si přejete zahrnout části Programu do jiných volných programů, jejichž distribuční podmínky jsou odlišné, zašlete autorovi žádost o povolení. V případě programového vybavení, k němuž vlastní autorská práva Free Software Foundation, napište Free Software Foundation; někdy činíme výjimky ze zde uvedených ustanovení. Naše rozhodnutí bude vedeno dvěma cíli; zachováním volné povahy všech odvozenin našeho volného programového vybavení a podporou sdílení a opětovného využití programového vybavení obecně.

ZÁRUKA SE NEPOSKYTUJE

11. VZHLEDEM K BEZPLATNÉMU POSKYTNUTÍ LICENCE K PROGRAMU SE NA PROGRAM NEVZTAHUJE ŽÁDNÁ ZÁRUKA, A TO V MÍŘE POVOLENÉ ZÁKONEM. POKUD NENÍ PÍSEMNĚ STANOVENO JINAK, POSKYTUJÍ DRŽITELÉ AUTORSKÝCH PRÁV POPŘÍPADĚ JINÉ STRANY PROGRAM „TAK, JAK JE“, BEZ ZÁRUKY JAKÉHOKOLIV DRUHU, AŤ VÝSLOVNÉ NEBO VYPLÝVAJÍCÍ, VČETNĚ, ALE NIKOLI JEN, ZÁRUK PRODEJNOSTI A VHODNOSTI PRO URČITÝ ÚČEL. POKUD JDE O KVALITU A VÝKONNOST PROGRAMU, LEŽÍ VEŠKERÉ RIZIKO NA VÁS. POKUD BY SE U PROGRAMU PROJEVILY ZÁVADY, PADAJÍ NÁKLADY ZA VŠECHNU POTŘEBNOU ÚDRŽBU, OPRAVY ČI NÁPRAVU NA VÁŠ VRUB.

12. V ŽÁDNÉM PŘÍPADĚ, S VÝJIMKOU TOHO, KDYŽ TO VYŽADUJE PLATNÝ ZÁKON, ANEBO KDYŽ TO BYLO PÍSEMNĚ ODSOUHLASENO, VÁM NEBUDE ŽÁDNÝ Z DRŽITELŮ AUTORSKÝCH PRÁV ANI ŽÁDNÁ JINÁ STRANA, KTERÁ SMÍ MODIFIKOVAT ČI ŠÍŘIT PROGRAM V SOULADU S PŘEDCHOZÍMI USTANOVENÍMI, ODPOVĚDNÝ ZA ŠKODY, VČETNĚ VŠECH OBECNÝCH, SPECIÁLNÍCH, NAHODILÝCH NEBO NÁSLEDNÝCH ŠKOD VYPLÝVAJÍCÍCH Z UŽÍVÁNÍ ANEBO NESCHOPNOSTI UŽÍVAT PROGRAMU (VČETNĚ ALE NIKOLI JEN, ZTRÁTY NEBO ZKRESLENÍ DAT, NEBO TRVALÝCH ŠKOD ZPŮSOBENÝCH VÁM NEBO TŘETÍM STRANÁM, NEBO SELHÁNÍ FUNKCE PROGRAMU V SOUČINNOSTI S JINÝMI PROGRAMY), A TO I V PŘÍPADĚ, ŽE TAKOVÝ DRŽITEL AUTORSKÝCH PRÁV NEBO JINÁ STRANA BYLI UPOZORNĚNI NA MOŽNOST TAKOVÝCH ŠKOD.

Jak uplatnit tato ustanovení na vaše nové programy

Pokud vyvinete nový program a chcete, aby byl veřejnosti co nejvíce k užitku, můžete toho nejlépe dosáhnout tím, že jej prohlásíte za volné programové vybavení, které může kdokoliv redistribuovat a měnit za zde uvedených podmínek.

K tomu stačí připojit k programu následující údaje. Nejbezpečnější cestou je jejich připojení na začátek každého zdrojového souboru, čímž se nejúčinněji sdělí vyloučení záruky; a v každém souboru by pak měla být přinejmenším řádka s „copyrightem" a odkaz na místo, kde lze nalézt úplné údaje.

```
řádka se jménem programu a nástinem toho, co dělá
Copyright (C) 19yy  jméno autora

Tento program je volné programové vybavení; můžete jej šířit a
modifikovat podle ustanovení Obecné veřejné licence GNU, vydávané Free
Software Foundation; a to buď verze 2 této licence anebo (podle vašeho
uvážení) kterékoli pozdější verze.

Tento program je rozšiřován v naději, že bude užitečný, avšak BEZ
JAKÉKOLI ZÁRUKY; neposkytují se ani odvozené záruky PRODEJNOSTI anebo
VHODNOSTI PRO URČITÝ ÚČEL.
Další podrobnosti hledejte v Obecné veřejné licenci GNU.

Kopii Obecné veřejné licence GNU jste měl obdržet spolu s tímto
programem; pokud se tak nestalo, napište o ni Free Software Foundation,
Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Připojte rovněž informaci o tom, jak je možné se s vámi spojit elektronickou a papírovou poštou.

Pokud je program interaktivní, zařiďte, aby se při startu v interaktivním módu vypsalo hlášení podobné tomutu:

```
Gnomovision verze 69, Copyright (C) 19yy jméno autora
Gnomovision je ABSOLUTNĚ BEZ ZÁRUKY; podrobnosti se dozvíte zadáním
show w. Jde o volné programové vybavení a jeho šíření za jistých podmínek
je vítáno; podrobnosti získáte zadáním show c.
```

Hypotetické povely `show w` a `show c` by měly zobrazit příslušné pasáže Obecné veřejné licence. Odpovídající povely ovšem nemusí být právě `show w` a `show c`; mohou to být třeba stisky tlačítka na myši nebo položky v menu - cokoliv, co se do vašeho programu hodí.

Pokud je to nutné, měl byste také přimět svého zaměstnavatele (jestliže pracujete jako programátor) nebo představitele vaší školy, je-li někdo takový, k tomu, aby podepsal „zřeknutí se autorských práv“ . Zde je vzor; jména pozměňte:

```
Yoyodyne, a.s., se tímto zříká veškerého zájmu o autorská práva k programu
`Gnomovision' (překladač s nakladačem) napsaného Jamesem Hackerem.

Tomáš Složitý --- podpis}, 1. dubna 1989
Tomáš Složitý, více než prezident
```

Tato Obecná veřejná licence neumožňuje zahrnutí vašeho programu do jiných než volných programů. Je-li váš program knihovnou podprogramů, můžete zvážit, zda je užitečné umožnit sestavování i vázaných aplikačních programů s vaší knihovnou. V takovém případě použijte Obecnou knihovní licenci GNU namísto této licence.

Dotazy a připomínky ohledně stránky posílejte na hubicka@paru.cas.cz

ČÁST VI

České sdružení uživatelů operačního systému LINUX (CZLUG)

CZLUG - Czech Linux User Group

České sdružení uživatelů operačního systému Linux (CZLUG) je nezisková organizace, která spojuje lidi se společným zájmem - zájmem o operační systém Linux. Odkaz na základní informace o tomto sdružení lze nalézt na internetové stránce, která je „výchozím rozcestníkem“ informací o Linuxu v České republice: http://www.linux.cz.

Linux je unixový operační systém, který se v posledních letech velmi dynamicky rozvíjí. Od prvních pokusů studenta informatiky Linuse Torvaldse v létě roku 1991, které se jen s velkou dávkou obrazotvornosti daly nazvat pokusy na novém operačním systému, uplynula relativně krátká doba. Svou prací chtěl Linus asi setřít propastný rozdíl mezi drahými unixovými pracovními stanicemi, se kterými se setkával ve škole, a levnými PC. Dnes můžeme s jistotou říci, že se mu povedlo splnit vytčený cíl. Linux je mladý operační systém, postavený na starých, léty ověřených unixových principech. Silně konkuruje komerčně šířeným Unixům a můžeme jej najít na velkém množství počítačových architektur. Procesory Intel (ona levná PC) jsou jen jednou z mnoha hardwarových platforem, kde se Linux provozuje. Geometrickou řadou roste počet aplikací pro Linux i počet jeho uživatelů. Na celém světě vznikají sdružení příznivců tohoto systému a CZLUG je jedním z takových sdružení.

Je asi všeobecně známo, že Linux se šíří pod licencí GPL (General Public License), vydané Free Software Foundation (Nadací pro volně šířený software). Licence neklade mezi autory programů a uživatele jejich práce žádné překážky. Uživatel může získat programy například od zprostředkovatele jen za manipulační poplatek nebo cenu média, tedy neplatí licenční poplatky. Není přitom omezován ve způsobu použití díla - může jej dále modifikovat, dále šířit a není mu zabráněn přístup ke zdrojovým textům díla. Firmy zabývající se distribucí software mohou bez omezení poskytovat zákazníkům placené služby, které souvisejí s údržbou, školením a dalšími běžnými komerčními aktivitami spojenými s existencí softwarového díla šířeného podle GPL.

Domnívám se, že z výše uvedeného důvodu nemá dobrovolné sdružení uživatelů Linuxu ve svém poslání jen podporování určitého operačního systému, ale též šíření myšlenky ideálního vztahu mezi autorem softwarového díla a uživatelem, jako je tomu v případě programů s licencí GPL. Linux je jen jedním příkladem takového programu. Stovky, možná tisíce dalších aplikací (zvláště pro Linux) se šíří podobným způsobem. Proto CZLUG deklaroval už při svém vzniku úzkou spolupráci s Českou nadací pro podporu free software, která je českou alternativou americké Free Software Foundation.

Členem CZLUG se může stát kdokoli zaplacením členského příspěvku na kalendářní rok. Členský příspěvek pro rok 2001 je 250 Kč nebo 170 Kč pro studenty. Touto více méně symbolickou částkou dáváte najevo svou podporu s propagací Linuxu. Údaje o členech sdružení jsou zaneseny do databáze. Členové dostávají přednostně pozvánky na akce CZLUG, případně mohou dostat se slevou CD či jiný materiál vytvořený za podpory CZLUG. Protože CZLUG je registrovanou neziskovou organizací, má své stanovy, podle nichž členové na valných shromážděních volí výbor a mohou určit další směry činnosti CZLUG a propagace Linuxu. Sídlem CZLUG je adresa Fakulty informatiky MU, Botanická 68a, 602 00 Brno. Další informace včetně stanov naleznete na adrese http://www.linux.cz/czlug.

Tuto knihu, Linux - dokumentační projekt, vydalo nakladatelství Computer Press ve spolupráci s Českým sdružením uživatelů operačního systému Linux, které ji doporučuje svým členům jako základní studijní a referenční materiál.

7. 9. 1998 Petr Olšák

Rejstřík

.orig, soubor 830
.rpm, balíček 936
/dev, adresář 160
/etc 158
/etc/exports 851, 875
/etc/hosts.allow, soubor 651
/etc/hosts.deny, soubor 652
/etc/inetd.conf, soubor 647
/etc/inittab, soubor 208
/etc/passwd, soubor 218
/etc/services, soubor 643
/etc/skel, adresář 219
/etc/slip.login, soubor 696
/etc/slip.tty, soubor 697
/proc, souborový systém 162
/proc, souborový systém 302
/usr, souborový systém 160
/var, souborový systém 161
3Com 653

A

A, záznam 338
absolutní cesta 29
active, soubor 600
address resolution 257
Address Resolution Protocol, zkráceně ARP 266
adduser 218
administrativa 786
adresáře
- aktuální 29
- odstraňování 29
- vytváření 29
adresáře Unix 26
adresářová struktura
- základy 155
adresy elektronické pošty 515
agent pro přenos pošty (MTA) 153
AIX, systém 866
aktivní okno 38
aktivní skupina 589
aktuální adresář 29
alert, volba 322
Alexandri, systém 897
aliasy 782
Allied Telesis 653
alokace odkládacího prostoru 190
ambice 235
AMD 653
analýza poštovních statistik 548
Ansel 653
Ant 893
Apache 2.0 902
Apache Jmeter 895
Apache Software Fondation 888
- dynamický obsah 888
- statický obsah 888
Apache, server 887
- moduly 892
- XML 897
aplikační program 149, 235
Appletalk, software 700
Apricot 653
Architektura firewallu 715
ARCNet 699
ARP tabulky 318
- kontrola 318
asynchronní mapa 365
Asynchronous Control
Character Map, volba 365
at, příkaz 152
Athena systém 43
- posuvné lišty 43
ATM 702
atributy X Windows 40
ATT 653
automatické detekce 275
automatické odpovídání 782
automatické připojování 911
autonomní systém 267
autorská práva 745, 786
Avalon 896
AX25 702

B

badblocks, příkaz 173, 181, 186
balíčky 931
bash(1) 920
bash, příkaz 47
Batik 899
bdflush, program 187, 198
Berkeley Socket Library 259
bezdiskové bootování 281

bezpečnost systému 262
bg, příkaz 52
BGP (Border Gateway Protocol) 271
binární soubory
- instalace 302
BIND, program 335
bloková zařízení s náhodným přístupem 165
blokové zařízení 276
bloky v textu 69
bootování 199
Bourne shell 23
brána 256, 269, 281, 311
broken_suid, volba 858
BSD klient, poštovní klient 781
BSD, systém 866
buffer 65

C

C News 569
- doručování 569
- instalace 571
C, jazyk 73
Cabletron 653
cache, volba 335
caching-only, kofigurace 339
Caldera 469
cancel, zpáva 581
case-sensitive 23
cat, příkaz 24, 60
CD
- hudební 652
- testování obrazu 950
- vypalování 939
cd, adresář 29
cd, příkaz 29
CD-ROM 170
CGI, skripty 889
Challenge Handshake Authentication Protocol (CHAP) 357, 367
- soubor hesel 368
character devices 829
chat, program 258
- automatické vytáčení 360
checkgroups, zpráva 581
chfn, příkaz 220
chmod, příkaz 58, 219
chown, příkaz 219
chsh, příkaz 159
chsh, příkaz 220
cklock, příkaz 233
cmp, příkaz 62
CNAME, záznam 339
Cocoon, projekt 898
Cogent 653
Compaq Tru 64 Unix, systém 867
compressed SLIP 258
COMSAT, démon 438
config, soubor 490
contact, pole 337
control.ctl, soubor 607
cowslip, hostitel 351
cp, soubory 31
- kopírování 31
cpio, program 224, 230
CPU, cykly 864
crack, program 215
Crimson 899
cron, příkaz 152
Crystal Lan 653
ctlinnd, příkaz 610
cyklické procházení okna 39
cylindr disku 167

Č

časové údaje 230
časové zóny 230
čtení pošty 776

D

Danpex 653
DAO 953
datagram 257
date, příkaz 233
dávkové zpracování článků 575
daylight savings time 231
dd, program 189
DDI (Device Driver Interface) 261
debugfs 186
DEC 653
DECNet 702
default, příkaz 355
defaultdomain 758
dekomprese 823
deliminace 347
deluser, program 221
démon 235
df, příkaz 59
DHCP 667
DHCPD 667
dial, soubor 497
diff, příkaz 62
dig, nástroj 345
Digi 653

dip (Dialup IP) 689
dip nástroje 350
diplogoin, program 356
directors, soubor 762
directory, adresář 846
directory, volba 334
diskety 169
disková oblast 173
- logická 174
- primární 174
- typy 175
diskový prostor 189
- nároky 190
- přidělování 189
- rozvržení 191
disky 165
- cylindr 167
- pevné disky 166
- řadič 167
- sektor 167
- stopa 167
- typ IDE 169
- typ SCSI 169
display X Windows 41
DLink 653
DNS
- databáze 330
- fungování 327
- vytváření dotazů 329
dns, volba 323
dnswalk, nástroj 345
Domain Name System, DNS 273
DOMAIN, makro 529
doména 327
dostupná skupina 589
driver (ovladač 20)
du, příkaz 59
dummy 313
dump, program 184, 224
dumpe2fs, program 188
Dvips 930

E

e2fsck, program 185
echo, služba 438
editor zpráv 780
- nastavení 780
efax(1) 929
EGP (External Gateway Protocol) 271
elektronická pošta 510
Element Construction Set 895
elm, program 519, 780
- globální nastavení 522
elm, program 776, 782
emacs(1) 927
Emacs, editor 64
- užití pod X Windows 66
Emacs, editor 71
- efektivita práce 75
- konfigurace 75
- nápověda 72
- pracovní módy 72
- vnitřní funkce 71
e-mailové diskusní skupiny 943
Embperl, projekt 900
Enyx 654
EQL 670
EQL, zařízení 656
Ethernet 253, 652
- automatická detekce 284
- instalace 284
ethernetové rozhraní 309
Ethernetový kabel 10base2 707
etwork Control Protocol (NCP) 357
Exim, program 551
- překlad 554
- spuštění 552
exin, program 754
expire, volba 338
expire.ctl, soubor 606
exports, soubor 463
ext, souborový systém 178
ext2, souborový systém 178
ext2-defrag, program 187
eznet 931

F

-F, parametr 28
falšování news 586
Fast CGI 889
FDDI (Fiber Distributed Data Interface) 255, 703
fdformat, program 172
fdisk, příkaz 174, 175
FEATURE, makro 529
fidopath, soubor 765
fiktivní rozhraní 313
file, příkaz 61
file, program 159
files, volba 323
filtrování 378
- způsoby 380
filtrování pošty 932
filtrování zpráv 784
filtry 25

firewall 377 , 859
- konfigurace jádra 379
- příklad 407
- testování konfigurace 405
flooding 565
FOP, aplikace 898
forwarders, volba 335
fragmentované disky 167
fragmentování
- nízkoúrovňové 171
- vysokoúrovňové 171
Frame Relay 255, 703
free, příkaz 195
From, údaj 536
fsck, příkaz 173, 183, 185
FSSTND, norma 155
Fujitsu 654

G

gated, démon 271
gcc, překladač 823
generíátor serveru 889
geometrie pevného disku 167
geometrie X Windows 40
getty, program 151, 159, 202, 205, 206, 211, 213, 297
Ghostscript 930
GNU - general public licence 12, 971
GNU libc 453
goto, příkaz 354
gpm, balíček 832
grafické uživatelské rozhraní 152
grep, příkaz 61
group, mapa 456
group, stránka 158
gunzip, příkaz 63
gzexe, program 190
gzip, příkaz 63, 190

H

handshake fáze 485
hardwarové hodiny 232
HDLC, High-Level Data Link Control Protocol 357
hdparm, balíček 832
head, příkaz 61
HINFO, záznam 339
historie
- Linux 10
- Unix 9
hlášení jádra systému 18
hlavní zaváděcí záznam (master boot record, MBR) 173
hodiny
- hardwarové 232
- softwarové 232
hodnoty metrik 271
host, nástroj 345
host.conf 322
hostcvt, nástroj 345
hostname resolution 257
hosts, soubor 305
HP 654
hpfs, souborový systém 179
HP-UX, systém 867
hudební CD - vytváření 952
HUP, signál 851

I

IBM 654
ICL 654
ICMP (Internet Control Message Protocol) 272
ICMP datagramy 387
- účtování 420
ID skupiny 219
IDE disky 825
identifikační číslo uživatele 217
if, příkaz 354
ifconfig 314
ifconfig, příkaz 290
IIS 902
incoming.conf, soubor 604
inetd, démon 641, 772
inetd, superserver 433
informační příkazy 61
init, proces 151, 158, 202, 205
inkrementální zálohy 225
inn.conf, soubor 598
inovace jádra 822
insecure_locs, volba 868
instalace jádra Linuxu 828
Intel 654
Internet News (INN) 595
- čtení zpráv 597
- globální parametry 598
- instalace balíku 597
- spuštění 609
Internet Protocol 256
interpret příkazů 215
interrupt request number (IRQ) 276
IP adresa 265, 303
IP aliasing 658
IP Firewall Chains 387

IP maškaráda 425, 664
- konfigurace 428
- konfigurace jádra 427
IP sítě 267
IP účtování 415
- konfigurace jádra 415
- podle adresy 416
IPCHAINS 665
ipchains, firewall 859
ipchains, nástroj 380, 388, 404, 420
- kompatibilita 399
- použití tříd 392
- pravidla 391
IPCHAINS, program 680
IPFWADM 665, 722
ipfwadm, nástroj 379, 381
- parametry 385
- výpis dat 420
IPFWADM, program 680
IPIP Zapouzdření 661, 676
iptables, nástroj 380, 399, 404, 420
- použití 400 pravidla 400
IPv6 666
IPX 468
- konfigurace 470
- statické směrování 473
IPX směrovače 472
ipx_configure, příkaz 470
ipx_interface, příkaz 471
ipx_route, příkaz 473
IRIX, systém 867
ISDN (Intergrated Services Digital Network) 685
iso9660, souborový systém 179

J

jádro Linuxu 817
- inovace 822, 837
- instalace 828, 837
- nastavení 824
- opravy 829
- překlad 828
jádro systému 14, 235
- hlášení 18
jádro systému Linux 150
- části 150
JAMES (Java Apache Mail Enterprise Server) 896
Java servlety 890
jed(1) 928
jednoduché zálohování 225
jednouživatelský režim 204
- zavádění systému 208
Jetspeed, webový portál 896
joe(1) 928
Jserv 895
JSSI 895
Jyve, projekt 896

K

kabely, kabeláž 706
kbd, balíček 831
kermit, program 289
kernel hacking 827
klávesnice 906
klient 36
knihovna resolveru 321
knihovna soketů 259
kompimované zálohy 230
konfigurace brány 311
konfigurace jádra 277
- hlavní verze jádra 278
- vedlejší verze jádra 278
konfigurace resolveru 321
konfigurace softwaru 920
konfigurace systému 905
kontrola jmenného serveru 342
kořenová doména 327
KTI 654

L

l18a 926
ladění 508
lanmanager 261
laser 938
libc, knihovna 823
Library General Public Licence 12
lilo(8) 912
lilo.conf, soubor 821
linoxové poštovní konference 243
Linux
- části operačního systému 149
- historie 10
- jádro systému 14, 150
- programové vybavení 11
- přihlášení 16
- sítě 260
- síťová zařízení 283
Linux Intranet Server 711
Linux Journal 242
LInux LOader, LILO 200
Linux Magazine 242
Linux Standard Base 245
Linux Usenet Newsgroups 242
Linux v sítích 627

LinuxConf, program 668
LOADLIN.EXE 912
locals, soubor 758
Log4j, balíček 897
logická disková oblast 174
login, program 151, 162, 211, 213, 214, 297
logovací soubory 508
lokální adresa 266
lokální rozhraní 307
lokální uživatelé 556
loopback interface 266
lpr, program 166
ls [-aRF], adresář 28
ls(1) 926
ls, příkaz 27
ls, program 166

M

machine1, machine2 846
Macromate 654
mail transport agent 514
mail user agent 514
mailx, program 775
make clean 832
make výstup 837
makefile 827
makra 531
manipulace se soubory 31
manuální vytváření účtů 219
mapovače portů (portmapper) 439
maps/, soubor 769
maska podsítě 268
Masn, projekt 900
master boot record (MBR) 173
maximální odkládací priorita 485
maximální velikost okna 40
Maximum Receive Unit, MRU 357, 365
Maximum reliability, služba 403
Maximum throughput, služba 403
Maximum Transfer Unit, MTU 281
média 165
- CD- ROM 170
- diskety 169
- disky 165
- páska 171
Meta, klíč 68
metody útoků 376
metrik 271
MFM/RLL 825
mgetty, démon 297 - 299
Microsoft IIS 902
minicom(1) 929
Minimum cost, služba 404
Minimum delay, služba 403
minimum, pole 338
minix, souborový systém 178
mkdir, adresář 30
mkfs, příkaz 173, 180
mkswap, příkaz 194
Mod_Perl, projekt 900
mode, příkaz 355
modem 507
moderatormailer 598
moduly 823, 836
- distribuované s jádrem 836
moduly pro Apache 892
- vytváření 892
moduly pro ověřování 892
more, příkaz 60
Motif systém 43
- posuvné lišty 43
mount, příkaz 158, 182
mountd 858
msdos, souborový systém 179
multi, volba 322
multicast 667
multicast 682
mutt, program 774, 780.
mv, příkaz
MX, záznam 339
myš 910

N

nabídkové lišty X Windows 42
náhodné umístění (random placement) 38
Name Service Switch 454
named, program 333
named.boot, soubor 334
nastavení času 232
NCPFS 469
ncpmount, příkaz 475
NCR NE2000/1000 654
neautorizovaný přístup 376
NetBEUI 704
NetBios 704
netfilter, firewall 859
netfilter, systém 397, 430
Netgear 652
netperm-table 727
Netscape 783
netstat, příkaz 310
netstat, příkaz 316
NetWare Directory Service (NDS) 469
NetWare, vzdálený svazek 475
- přihlašovací heslo 477
- připojení 475

- seznam serverů 478
- správa tiskové fronty 481
Network Address Translation, NAT 425
Network File Systém (NFS) 459
- démony 462
Network Information Center (NIC) 273
Network Information Systém (NIS) 447
- bezpečnost 452
- nastavení 451
- provozování serveru 451
- stínová hesla 458
networks, soubor 305
nevyžádaná pošta 539
New Media 654
newsgroup, soubor 600
newsgroup, zpráva 581
NFS 843
- konfigurační soubory 8486
- nastavení serveru 845
- softwarové vybavení 845
- úvod 844
- zabezpečení 845
nfs, souborový systém 179
NFSD 854
NFSv2, server 465
NFSv3, server 465
NIS (Network Information System) 273
nis, nisplus, volba 323
nn, program, konfigurace 617
NNTP protokol 585, 586
- instalace 592
nntpd démon 585
nntpsend.ctl, soubor 602
no_root_squash, adresář 847
no_subtree_chceck, adresář 846
NORTON COMMANDER 26
nospoof, volba 322
notebook 916
notifikátory 750
Novell 282
Novell 467
nprint, příkaz 480
NS, záznam 338
nsswitch.conf 323
null-modemový kabel 706
Null-modemový sériový kabel 622
nulování počitadel 422

O

odkládací prostor 194
- přidělování 196
- sdílení 196
- využití 195
odposlouchávání 376
okna 38
- aktivní 38
- cyklické procházení 39
- maximální velikost 40
- překrývání 39
- přemisťování 38
- správce 38
- transformace do ikony 39
- vytváření 38
- vyzvednutí 39
- zasunutí 39
- změna velikosti 39
opravy jádra Linuxu 829
optionxx, adresář 846
order, volba 322
origin, pole 337
ORO, balíček 894
otazník, znak 46
ovladač (driver) 20

P

pablo, hostitel 488
paket 252
- přepínání 252
paketové protokoly 504
paketové rádio 256
Parallel Line IP 286
partiční tabulka 173
páska 171
passwd, mapa 456
passwd, příkaz 220
passwd, program 158
Password Authentication Protocol (PAP) 357, 367
patch, výstup 837
PC záplata 281
periodicky rozesílání informace 785
Perl, projekt 900
pevný disk 19, 166
- dělení na diskové oblasti 176
- geometrie 167
- fragmentace 167
PHP 900
PicoServer 896
pine(1) 929
pine, poštovní klient 781
ping, příkaz 308
PLIP
- kabel 621
- ovladač 286
- protokol 279

- rozhraní 311
plusdomain 758
podpora sítí 825
podpora síťových zařízení 826
podsítě 268
- maska 268
- vnitřní dělení 268
podsítě 304
point-to-point, spojení 311
pomalé jádro Linuxu 832
POP Klient 932
pořadové číslo úlohy 51
port, soubor 496
porting 9
portmapper 439
portmapper, démon 849, 856
porty 259
POSIX 10
pošta 153, 729
- čtení 766
- filtrování 9342
- hardware 753
- použití 777
- schránka 153
- testování 777
- trvale připojené počítače 748
poštovní aliasy 537
poštovní hlavička 512
poštovní klient 780
poštovní konference 559, 783
poštovní přenosový agent
(mail transport agent) 514
poštovní schránka
- formáty 752
poštovní uživatelský agent (mail user agent) 514
poštovní zpráva 512
posuvné lišty X Windows 43
PPP ovladač 288
PPP protokol 357
- authentikace 367
- v Linuxu 358
PPP server 371
pppd, démon 358
- spuštění 358, 360
překlad jádra Linuxu 818, 828
překlad síťových adres 425
překrývání oken 39
přemisťování oken 38
přepínání paketů 252
přesměrování vstupu 48
přesměrování výstupu 48
přihlášení do systému 16
přihlašovní přes terminály 211
příkazový interpret 23
příkazový řádek 65
primární disková oblast 174
primary, volba 334
připojovací body 911
privátní IP sítě 349
programovací módy 73
protokol 251
Protokol Rose 704
proudové protokoly 504
pseudoznaky 45
PTR, záznam 339
PureData, SEEQ, SMC 654
pushing článku na server 587
pwd, adresář 29
Python, jazyk 901

Q

qmail, program 1.03 754
- rozbalení 755
- testování 758

R

R, příkaz
rádio, podpora 282
RAM 19
random placement (náhodné umístění) 38
rcphosts, soubor 758
rdev, program 183, 201, 202
Redundantní internetové připojení 716
referenční implementace 586
refresh, pole 338
Regexp, balíček 894
relativní cesta
resolv.conf, soubor 325
resolver 321
- knihovna 321
- konfigurace 321, 635
- robustnost 326
retry, pole 338
reverzní hledání 322, 332
reverzní mapování 332
RFC 822 515)
RIP, Routing Information Protocol 271
rlogin, program 153
rm, příkaz 32
rmdir, adresář 30
rmgroup, zpráva 581
ro, adresář 846
robustnost resolveru 326
root, souborový systém 157
roura 48

route, příkaz 309
routed, program 639
routers, soubor 765
rozlišování adresy (address resolution) 257
rozlišování názvu hostitele (hostname resolution) 257
rozšířená disková oblast 174
r-příkaz 440 - vypnutí 440
rsize, volba 853
rw, adresář 846

Ř

řadič disku 167
řídicí zprávy 581
řešení běžných chyb 821
řetězce
- náhrady 69
- vyhledávání 69
řízení přístupu 214
řízení úloh 50

S

S, příkaz 531
SAGE: The System Administrators Guild 623
samba 920
Scheme, jazyk 73
Schneider 654
schránka na poštu 153
SCSI support 825
secondary, volba 335
sektor disku 167
sémantika sad pravidel 533
send, příkaz 354
sendmail 8.x 773
sendmail, program 245, 524, 754
- instalace 525
- konfigurace nastavení 535
- spuštění 546
sendmail.cf, soubor 526, 531
sendmail.mc, soubor 526
- parametry 528
senduuname, zpráva 583
sendys, zpráva 583
sequence number check 486
Serial Line Internet Protocol 258
seriál, pole 337
sériová zařízení
- konfigurace 289
- soubory 292
- úvod 290
sériový hardware 293
Server Pages Foundation Classes 895
setfdprm, program 158, 170
setserial, příkaz 293
shaper 667
shaper 682
shell (příkazový interpret) 23
shutdown -h now, příkaz 202
shutdown -r now, příkaz 203
shutdown, příkaz 202, 203, 214
sítě
- historie 251 TCP/IP 252
sítě v Linuxu 260
síťové souborové systémy 153
síťový hardware, konfigurace 275
slattach, program 348, 689
slave, volba 335
sleep, příkaz 354
Slide, systém správy 894
SLIP linka 347
SLIP ovladač 288
SLIP server 694
slovníček 235
smail 3.1, program 761
smail 3.2, program 754
směrovací tabulka 270
směrování pošty 516
směrování zpráv 556
SMPT 774
SNMP 891
SOAP 899
SOCKS proxy server 715, 730
- instalace 730
softwarové hodiny 232
sokety 259
Solaris, systém 868
soubor hostitelů 251
souborový systém 177, 236, 826
- galerie 178
- užití 180
- vytvoření 177, 180
soubory Unix 26
sound 827
spam 539
spell, příkaz 62
spoofing 366
správa paměti 193
správa poštovní fronty 547
správa tiskové fronty 481
správce oken 934
správce okna 38
SQUID proxy server 726
ssh, démon 441
ssh-keygen, progra, 442

standardy souborového systému 244
statistika rozhraní 317
- zobrazení 317
stínová hesla 214, 458
stopa disku 167
sťové news 563
STRIP (Standard Radio IP) 704
Strust, projekt 894
stty, příkaz 295
su, program 221
subdoména 327
sudo, program 184
Sun Intel 654
Sun Lance 654
SunOS, systém 868
superserver inetd 433
SVGATextMode 914
swapoff, příkaz 195
swapon -a, příkaz 158
swapon, příkaz 195
sync, prgram 187
sys, soubor 572
syslog, program 152, 172
systémová statistika 59
systémový program 236
sysv, souborový systém 179
sysvinit, program 206

T

Taglibs 894
tail, příkaz 61
tail, program 221
TAO 953
tar, program 224, 226
Taylor UUCP, program 483
- historie 483
- konfigurační protokoly 487
- nízkoúrovňové protokoly 504
- poskytování účtů 501
- směrování pošty 517
- spuštění příkazů 499
- úvod 487
Taylorovy konfigurační soubory 491
Tcl, projekt 901
TCP/IP
- úvod 252 konfigurace 301
TCP/IP Firewall 375
tcpd, wrapper 435
tečková adresa 257
tečková notace 257
telinit, příkaz 208
telnet 149
teorie řízení úloh 54
terminálové programy 289
terminály 211
TeX 930
TFTP (Trivial File Transfer Protocol) 263
tftp 435
timegrade 495
timeo, volba 460
tin, program 614
- konfigurace 614
TIS FWTK
- instalace 727
- konfigurace 727
TIS server 726
tisk 154
tisk do front 479
tiskárna 913
tisková fronta 154
tlačítka X Windows 42
token 255
Tomcat 894
top, příkaz 195
touch, příkaz 58
tr, příkaz 63
transformace okna 39
Transmission Control Protocol 258
transparentní proxy 665
transportní agent 853
transports, soubor 866
trim, volba 322
trn, progra, 616
- konfigurace 616
Trumpet Winsock 733
trvalé vytáčení 372
tune2fs, program 187, 188
tunelované sítě 676
- konfigurace 676
tunelový hostitel 678
- konfigurace 678
Turbine, systém 896
typogeafické konvence 817
typy diskových oblastí 175

U

účtování 281
UDP paket 733
ukončení práce s počítačem 17
umount, příkaz 184
umount, příkaz 185
umsdos, souborový systém 179
UMSDOS, soubory 955
universal time 231

univerzální čas 231
Unix 9
- adresáře 26
- bash, příkaz 47
- doplňování jmen souborů 47
- doplňování příkazů 47
- historie 9
- pseudoznaky 45
- práce se soubory 57
- přesměrování výstupu 48
- příkazový interpret (shell) 23
- příkazy 23
- roura 48
- řízení úloh 50
- služby 151
- soubory 26
- standardní vstup 48
- standardní výstup 48
- virtuální konzoly 55
update, program 203
úplné zálohování 225
uptime , příkaz 59
Usenet
- historie 563
USENET, duskusní skupina 784
usenetové mapy 516
User Datagram Protocol 258
user ID 217
useradd 218
userdel, program 221
util-linux, balíček 831
UUCICO, program 485
UUCP sítě 260
uživatelské ID 219
uživatelské jméno 217
uživatelský poštovní agent (MUA) 154
uživatelský účet 217
- správa 217
- změna vlastností 220
- zablokování 221
- zrušení 220

V

V IPC systém 825
vadný blok 172
vadný sektor 172
vale, server 306
velké jádro Linuxu 832
Velocity, systém šablon 894
version, zpráva 583
vipw, příkaz 219
virtualdomains, soubor 758
virtuální hostitelé 281 890
virtuální konzoly 55
virtuální paměť 193
virtuální spojení 213
vlager, počítač 304
volání systému 236
volba, parametr 28
výkon pevného disku 898
vymazání pravidel 423
vypalování CD
- dostupnost 939
- podporované funkce 942
vypnutí počítače 17
vyrovnávací paměť 197
vytváření oken 38
výzva příkazového řádku 23
vyzvednutí okna 39
vzdálená práce s poštou 775

W

w, příkaz 60
wait, příkaz 354
Watchdog, projekt 895
WaveLan, karta 705
wc, příkaz 62
WD 654
WebDAV 893
who, příkaz 60
Windows Manager 934
Windows System 933
wrapper tcpd 435
wsize, volba 853
X
X Windows
- atributy 40
- display 41
- geometrie 40
- klient 36
- nabídkové lišty 42
- okna 38
- posuvné lišty 43
- společné vlastnosti 41
- spuštění 35
- ukončení 35
X.25 705
Xalan, procesor 888
Xang, projekt 899
XClock, program 36
xdm 934
xdm, proces 214
Xerces, projekt 897
Xerox 467

Xerox Network Specification (XNS) 467
Xfree86 933
xia, souborový systém 178
XML 897
XNS (Xerox Networking System) 260
xterm -ls 214
xterm, program 37, 203

Z

zabezpečené transakce 891
zablokování služby 376
záchranná disketa 908
záchranná zaváděcí disketa 204
základy adresářové struktury 155
zálohování 222
- inkrementální 225
- jednoduché 225
- komprimace 230
- média 224
- úplné zálohování 225
- víceúrovňové 228
zámkový soubor 291
zapnutí počítače 13
zasunutí okna 39
zavaděč 199
zaváděcí sektor 173
zaváděcí sektor 200
- hlavní 200
zcat, příkaz 63
zdrojový záznam 330
Zenith 654
Zip 909
změna velikosti okna 39
znaková zařízení 165, 276, 827
zpětná vazba 786
zpětnovazebné rozhraní 637
zvuk 827
zvuková karta 909