

Linux księga eksperta

**Połączone przez
Elizabeth**

Str. 2.

Skrócony spis treści

Wstęp.....	31
Część pierwsza Wstęp	
Rozdział 1. Wstęp do Linuxa.....	33
Rozdział 2. Rodzaje Linuxa.....	51
Rozdział 3. Instalacja Linuxa.....	63
Rozdział 4. LILO.....	89
Rozdział 5. Podsumowanie instalacji.....	111
Część druga Poznawanie Linuxa	
Rozdział 6. Od czego zacząć?.....	123
Rozdział 7. Podstawowe polecenia Linuxa.....	141
Rozdział 8. System plików.....	161
Rozdział 9. Prawa dostępu do plików i katalogów.....	181
Rozdział 10. Programy użytkowe projektu GNU.....	193
Rozdział 11. bash.....	209
Rozdział 12. pdksh.....	225
Rozdział 13. tcsh.....	239
Rozdział 14. Programowanie w języku powłoki.....	255
Rozdział 15. FTP oraz Telnet.....	275
Część trzecia Edycja i skład tekstu	
Rozdział 16. Edytory tekstu: vi i emacs.....	289
Rozdział 17. groff.....	305
Rozdział 18. gेनq oraz gtbl.....	315
Rozdział 19. TeX i LaTeX.....	327
Rozdział 20. Drukowanie.....	343
Rozdział 21. Linux i multimedia.....	349
Część czwarta Graficzne interfejsy użytkownika	
Rozdział 22. Instalacja i konfiguracja XFree86.....	365
Rozdział 23. Wabi.....	389
Rozdział 24. Ghostscript i Ghostview.....	395
Część piąta Linux dla programistów	
Rozdział 25. gawk.....	409
Rozdział 26. Programowanie w języku C.....	429
Rozdział 27. Programowanie w C++.....	447
Rozdział 28. Perl.....	463
Rozdział 29. Podstawy języków Tcl i Tk.....	479

Rozdział 30. Inne kompilatory.....	491
Rozdział 31. Smalltalk/X.....	497

Część szósta Linux dla administratorów

Rozdział 32. Podstawy administrowania systemem.....	521
Rozdział 33. Urządzenia.....	535
Rozdział 34. Procesy.....	557
Rozdział 35. Użytkownicy i konta.....	565
Rozdział 36. Obsługa urządzeń SCSI.....	579
Rozdział 37. Praca w sieci.....	587
Rozdział 38. SLIP i PPP.....	603
Rozdział 39. UUCP.....	617
Rozdział 40. Konfigurowanie poczty.....	633
Rozdział 41. Konfigurowanie grup dyskusyjnych.....	649
Rozdział 42. Bezpieczeństwo w sieci.....	657
Rozdział 43. NFS.....	665
Rozdział 44. NIS i YP.....	671
Rozdział 45. Kopie zapasowe.....	679
Rozdział 46. cron i at.....	689

Część siódma Konfiguracja węzła internetowego

Rozdział 47. Konfiguracja węzła internetowego.....	699
Rozdział 48. Konfigurowanie FTP i anonimowego FTP.....	705
Rozdział 49. Konfiguracja węzła WAIS.....	721
Rozdział 50. Konfigurowanie usługi Gopher.....	737
Rozdział 51. Konfigurowanie węzła WWW.....	757
Rozdział 52. Skrypty CGI.....	773
Rozdział 53. Podstawy języka HTML.....	781
Rozdział 54. Podstawy języków Java i JavaScript.....	803
Rozdział 55. Projektowanie spójnych stron WWW.....	809

Część ósma Programowanie dla zaawansowanych

Rozdział 56. Zarządzanie kodem źródłowym.....	817
Rozdział 57. Jądro systemu.....	835
Rozdział 58. Tworzenie sterowników urządzeń.....	845
Rozdział 59. Projekt Wine.....	857
Rozdział 60. HylaFAX.....	873
Rozdział 61. Gry.....	881
Rozdział 62. Adabas-D i inne bazy danych.....	901
Rozdział 63. StarOffice.....	915
Rozdział 64. Program Lone-Tar firmy Lone Star Software.....	923

Dodatki

Dodatek A Węzły FTP i listy dyskusyjne poświęcone Linuxowi.....	937
Dodatek B Komercyjni dystrybutorzy Linuxa.....	945
Dodatek C Projekt dokumentacji Linuxa.....	951
Dodatek D Publiczna licencja GNU.....	953
Dodatek E Informacje o prawach autorskich.....	961
Dodatek F Zawartość płyt CD-ROM.....	965
Skorowidz.....	967

Spis treści

O autorze.....	28
Dedykacja.....	28
Podziękowania.....	28
Wstęp.....	29

Część pierwsza Wstęp

Rozdział 1. Wstęp do Linuxa.....	33
Linux – co to takiego?.....	34
Kernel – jądro systemu.....	34
Oprogramowanie GNU.....	35
X.....	35
Współpraca z systemami DOS i Windows.....	36
Protokół TCP/IP.....	36
Historia Linuxa.....	37
Co Linux może zrobić dla Ciebie.....	38
Naucz się UNIX-a dzięki Linuxowi.....	38
Powierz swoją firmę Linuxowi.....	38
Serwery internetowe.....	39
Co jest potrzebne do instalacji Linuxa.....	39
Minimalne wymagania sprzętowe.....	40
Płyta główna i procesor.....	40
Dyski twarde.....	41
Karta graficzna i monitor.....	42
Mysz.....	43
Napędy taśmowe.....	43
Napędy CD-ROM.....	43
Dyski wyjmowalne.....	44
Drukarki.....	44
Modemy.....	44
Terminal.....	44
Karty multiport.....	45
Karty sieciowe.....	45
Prawa autorskie.....	45
Uzyskiwanie pomocy.....	46
Grupy dyskusyjne w sieci Usenet.....	47
O czym mówi ta książka?.....	49
Podsumowanie.....	49

6 Linux – księga eksperta

Rozdział 2. Rodzaje Linuxa.....	51
Skąd wziąć Linuxa?.....	51
Linux na płycie CD-ROM.....	52
Węzły FTP.....	53
Używanie FTP do ładowania plików.....	53
Węzły FTP udostępniające Linuxa.....	56
World Wide Web.....	56
Poczta elektroniczna.....	57
Systemy BBS (Bulletin Board Systems).....	58
Co to jest dystrybucja?.....	58
Dystrybucja Linuxa a zestawy dysków.....	59
Aktualizacja istniejącego systemu linuxowego.....	61
Podsumowanie.....	61
Rozdział 3. Instalacja Linuxa.....	63
Praca z Linuxem.....	63
Instalacja bez użycia stacji dysków.....	65
Dyskietki startowe.....	65
Wybór właściwych dysków boot i root.....	66
Tworzenie dysków boot i root.....	68
Krótki przewodnik po instalacji.....	70
Instalacja tekstowa i graficzna.....	70
Konfigurowanie dysku twardego.....	70
Formatowanie partycji.....	71
Konfiguracja sieci Ethernet.....	71
Konfiguracja myszki.....	72
Konfiguracja X.....	72
Wybór instalowanego oprogramowania.....	72
Użycie programu LILO.....	73
Podział dysku twardego na partycje.....	73
Partycja wymiany.....	74
Zakładanie partycji.....	75
Użycie systemu plików UMSDOS.....	76
Instalacja partycji linuxowych.....	77
Program fdisk.....	77
Zakładanie partycji linuxowych.....	78
Udostępnianie partycji wymiany programowi instalacyjnemu.....	79
Tworzenie partycji z systemem plików.....	80
Instalacja Linuxa.....	82
Konfiguracja startu systemu.....	83
Przeglądanie zainstalowanego oprogramowania.....	83
Rozwiązywanie problemów.....	84
Instalacja oprogramowania.....	84
Dysk twardy i kontroler dysku twardego.....	84
Konflikty pomiędzy urządzeniami.....	85
Problemy z urządzeniami SCSI.....	87
Uruchamianie Linuxa.....	87
Podsumowanie.....	88
Rozdział 4. LILO	89
Instalacja programu LILO.....	90

Problemy z dyskami twardymi.....	91
Plik Makefile.....	92
Aktualizacja LILO.....	92
Linux i dyski twarde.....	93
Sektor startowy.....	94
Uruchamianie systemu.....	96
Dysk twardy przeznaczony dla Linuxa.....	96
Program BOOTACTV.....	97
DOS i Linux.....	98
Używanie programu BOOTLIN.....	99
Parametry startowe.....	100
Program instalujący plik mapowania sektora startowego.....	101
Parametry podawane w wierszu poleceń.....	102
Parametry podawane w pliku konfiguracyjnym.....	103
Pliki obrazów sektora startowego.....	105
Tablica parametrów dysku.....	106
Wyłączanie i usuwanie programu LILO.....	107
LILO – rozwiązywanie problemów.....	108
Podsumowanie.....	109
Rozdział 5. Podsumowanie instalacji.....	111
Uruchamianie Linuxa.....	111
Procedura uruchamiania awaryjnego.....	112
Program dmesg.....	112
Zmiana rozmiaru partycji.....	112
Instalacja dodatkowego oprogramowania.....	113
RPM.....	113
installpkg.....	116
Inne programy instalujące pakiety oprogramowania.....	116
Systemy z kilkoma urządzeniami CD-ROM.....	118
Zmieniarzy dysków CD-ROM.....	118
Nagrywarki CD-ROM.....	118
Biblioteki na płytach CD-ROM.....	119
Zmienianie płyt CD-ROM.....	119
Podsumowanie.....	120
Część druga Poznawanie Linuxa	
Rozdział 6. Od czego zacząć?	123
Uruchamianie i zamykanie systemu.....	123
Polecenia zamykające Linuxa.....	124
Co to znaczy zalogować się?.....	125
Dlaczego nie powinieneś używać konta root.....	126
Twoje pierwsze logowanie.....	127
Hasła.....	129
Zakładanie nowego konta.....	132
Wylogowywanie się.....	135
Wypróbowywania nowego konta.....	135
Komunikaty o błędach.....	136
Ścieżki przeszukiwania.....	136
Polecenie who.....	137
Terminale wirtualne.....	138
Polecenia i programy.....	139

Podsumowanie.....	139
Rozdział 7. Podstawowe polecenia Linuxa.....	141
Jak działają polecenia Linuxa.....	141
Opcje polecień.....	142
Inne parametry.....	144
Przekierowanie wejścia i wyjścia.....	145
Konwencje używane do opisu polecen.....	148
Sześć podstawowych reguł opisywania polecen.....	148
Pomoc dostępna podczas pracy.....	149
Strony man.....	149
Słowa kluczowe na stronach man.....	150
help – polecenie interpretera polecen bash.....	151
Symbole wieloznaczne – „*” oraz „?”.....	152
Zmienne środowiskowe.....	152
Procesy i ich włączanie.....	154
Polecenie ps.....	154
Wyłączanie procesów: polecenie kill.....	156
Polecenie su, czyli jak stać się kimś innym.....	158
Program grep.....	158
Podsumowanie.....	159
Rozdział 8. System plików.....	161
Pliki – informacje ogólne.....	161
Popularne typy plików.....	162
Nazwy plików.....	163
Katalogi – informacje ogólne.....	163
Katalogi nadrzędne i podkatalogi.....	163
Katalog główny.....	164
Jak nazywane są katalogi.....	164
Katalog domowy.....	165
Poruszanie się po systemie plików.....	165
Polecenie pwd – gdzie to ja jestem.....	165
Absolutne i relatywne ścieżki dostępu.....	166
Idziemy na spacer: polecenie cd.....	167
Wszędzie dobrze, ale w domu najlepiej.....	168
Tworzenie i usuwanie plików.....	169
Polecenie cat.....	169
Tworzenie katalogów.....	171
Przenoszenie i kopianie plików.....	172
Przenoszenie i kopianie plików za pomocą symboli wieloznacznych.....	174
Przenoszenie katalogów.....	175
Usuwanie plików i katalogów.....	175
Usuwanie katalogów.....	176
Kompresja.....	177
Ważne katalogi systemu Linux.....	178
/.....	178
/home.....	178
/bin.....	178
/usr.....	178
/usr/bin.....	178

/usr/spool.....	179
/dev.....	179
/usr/sbin.....	179
/sbin.....	179
/etc.....	179
Podsumowanie.....	179
Rozdział 9. Prawa dostępu do plików i katalogów.....	181
Posiadanie plików i katalogów.....	182
Użytkownicy i posiadanie.....	182
Grupy.....	183
Zmiana przynależności do grupy.....	184
Prawa dostępu.....	185
Ustawienia zmiennej UMASK.....	186
Modyfikowanie praw dostępu do plików.....	187
Modyfikowanie praw dostępu do katalogów.....	190
Podsumowanie.....	191
Rozdział 10. Programy użytkowe projektu GNU.....	193
Aktualnie dostępne oprogramowanie podlegające licencji GNU.....	194
acm.....	194
Autoconf.....	194
bash.....	194
bc.....	194
BFD.....	195
Binutils.....	195
Bison.....	195
Kompilator języka C.....	195
Biblioteki dla języka C.....	195
Biblioteki dla języka C++.....	196
Calc.....	196
Chess.....	196
CLISP.....	196
Common Lisp.....	196
cpio.....	196
CVS.....	197
dc.....	197
DejaGnu.....	197
Diffutils.....	197
ecc.....	197
ed.....	197
Elib.....	197
GNU Emacs.....	198
GNU Emacs 19.....	198
es.....	198
Fileutils.....	198
find.....	198
finger.....	198
flex.....	199
Fontutils.....	199
gas.....	199
gawk.....	199
gdb.....	199
gdbm.....	200

Ghostscript.....	200
Ghostview.....	200
gmp.....	200
GNats.....	200
Aktualnie dostępne oprogramowanie podlegające licencji GNU	
GNU Graphics.....	200
GNU Shogi.....	201
gnuplot.....	201
GnuGo.....	201
gperf.....	201
grep.....	201
Groff.....	201
gzip.....	201
hp2xx.....	202
indent.....	202
Ispell.....	202
m4.....	202
make.....	202
mtools.....	202
MULE.....	203
NetFax.....	203
NetHack.....	203
NIH Class Library.....	203
nvi.....	203
Octave.....	203
Oleo.....	204
p2c.....	204
patch.....	204
PCL.....	204
perl.....	204
ptx.....	204
rc.....	205
RCS.....	205
recode.....	205
regex.....	205
Scheme.....	205
screen.....	205
sed.....	206
Shellutils.....	206
Smalltalk.....	206
Superopt.....	206
tar.....	206
Biblioteka Termcap.....	207
TeX.....	207
Texinfo.....	207
Textutils.....	207
Tile Forth.....	207
time.....	207
tput.....	208
UUCP.....	208
uuencode/uudecode.....	208

wdiff.....	208
Podsumowanie.....	208
Rozdział 11. bash.....	209
Po co komu interpreter poleceń.....	209
Co to jest interpreter poleceń powłoki?.....	209
Jak uruchamia się interpreter poleceń.....	211
Najczęściej używane powłoki.....	211
Powłoka Bourne Again Shell.....	212
Dokańczanie poleceń.....	212
Symbole wieloznaczne.....	214
Przywoływanie wydanych poleceń.....	215
Aliases.....	217
Przekierowanie wejścia.....	218
Przekierowanie wyjścia.....	219
Ciagi poleceń.....	220
Znak zachęty.....	220
Zarządzanie zadaniami.....	222
Dostosowywanie interpretera bash.....	222
Polecenia powłoki bash – podsumowanie.....	223
Zmienne powłoki bash.....	223
Podsumowanie.....	224
Rozdział 12. pdksh.....	225
Powłoka pdksh (Public Domain Korn Shell).....	225
Dokańczanie poleceń.....	226
Symbole wieloznaczne.....	227
Przywoływanie wydanych poleceń.....	228
Aliases.....	230
Przekierowanie wejścia.....	231
Przekierowanie wyjścia.....	232
Ciagi poleceń.....	232
Znak zachęty.....	233
Zarządzanie zadaniami.....	233
Skróty klawiaturowe.....	234
Dostosowywanie powłoki pdksh.....	236
Polecenia wewnętrzne powłoki pdksh.....	236
Zmienne powłoki pdksh.....	237
Podsumowanie.....	238
Rozdział 13. tcsh.....	239
Powłoka tcsh.....	239
Dokańczanie poleceń.....	240
Symbole wieloznaczne.....	240
Przywoływanie wydanych poleceń.....	241
Aliases.....	243
Przekierowanie wejścia i wyjścia.....	244
Ciagi poleceń.....	245
Znak zachęty.....	245
Zarządzanie zadaniami.....	246
Skróty klawiaturowe.....	247
Inne przydatne możliwości.....	248
Poprawianie pomyłek powstających w trakcie pisania.....	248
Prepolecenia.....	249

Polecenia wykonywane przy zmianie katalogu.....	249
Monitorowanie sesji.....	250
Dostosowywanie tcsh.....	251
Polecenia wewnętrzne interpretera tcsh – podsumowanie.....	252
Zmienne powłoki tcsh.....	252
Podsumowanie.....	253
Rozdział 14. Programowanie w języku powłoki.....	255
Tworzenie i uruchamianie programów powłoki.....	256
Używanie zmiennych.....	257
Nadawanie wartości zmiennej.....	257
Odczytywanie wartości zmiennej.....	258
Parametry pozycyjne i inne zmienne wewnętrzne powłoki.....	258
Cudzysłowy.....	259
Polecenie test.....	261
Odpowiedniki polecenia test w powłoce tcsh.....	263
Instrukcje warunkowe.....	264
Polecenie if.....	264
Polecenie case.....	266
Instrukcje iteracyjne.....	267
Pętla for.....	267
Pętla while.....	268
Pętla until.....	269
Polecenie shift.....	270
Polecenie select.....	270
Instrukcja repeat.....	271
Funkcje.....	272
Podsumowanie.....	273
Rozdział 15. FTP oraz Telnet.....	275
FTP.....	275
Konfiguracja serwera FTP.....	276
Obsługa FTP.....	277
Połączenia FTP.....	278
Polecenia FTP.....	278
Tryby przesyłania plików.....	280
Anonimowy dostęp do FTP.....	280
Protokół TFTP (Trivial file transfer protocol).....	282
Używanie programu Telnet.....	283
Używanie programu Telnet w systemach graficznych.....	284
TN3270 i inne.....	284
Podsumowanie.....	285
Część trzecia Edycja i skład tekstu	
Rozdział 16. Edytory tekstu: vi i emacs.....	289
Co to jest edytor tekstu?.....	289
Funkcje edycyjne.....	290
Wstawianie i usuwanie tekstu.....	290
Odczytywanie i zapisywanie plików.....	290
Wyszukiwanie tekstu.....	290

Kopiowanie i przenoszenie tekstu.....	291
Bufory edycyjne.....	291
Edytor vi.....	291
Uruchamianie edytora vi.....	291
Tryby pracy edytora vi.....	292
Wstawianie tekstu.....	293
Zamykanie edytora vi.....	293
Przesuwanie kurSORA.....	294
Usuwanie tekstu.....	295
Kopiowanie i przenoszenie tekstu.....	295
Wyszukiwanie i zastępowanie tekstu.....	296
Preferencje.....	297
Podsumowanie najważniejszych poleceń.....	298
Edytor emacs.....	299
Uruchamianie edytora emacs.....	299
Klawisze Control oraz Meta.....	300
Przesuwanie kurSORA.....	300
Zamykanie edytora emacs.....	301
Usuwanie tekstu.....	301
Praca z wieloma plikami.....	301
Kopiowanie i przenoszenie tekstu.....	302
Wyszukiwanie i zastępowanie tekstu.....	302
Tryby edycji.....	303
Pomoc dostępna podczas pracy.....	303
Podsumowanie najważniejszych poleceń.....	303
Podsumowanie.....	304
Rozdział 17. groff.....	305
Osadzanie poleceń.....	306
Określanie wyglądu znaków.....	306
Rozmiar czcionki i odstępy między wierszami.....	307
Czionki.....	308
Wcięcia i długość wiersza.....	309
Inne polecenia.....	310
Makropolecenia.....	310
Makropolecenia mm.....	311
Akapity i nagłówki.....	311
Wyliczenia.....	311
Zmiana rodzaju czcionki.....	312
Przypisy.....	313
Podsumowanie.....	313
Rozdział 18. geqn oraz gtbl.....	315
geqn.....	315
Uruchamianie geqn.....	316
Równania.....	316
Indeks górny i dolny.....	317
Ułamki.....	318
Pierwiastek kwadratowy.....	318
Sumy, teoria zbiorów i całki.....	318
Nawiasy i kolumny.....	319
Macierze.....	319
Cudzysłowy.....	320
Zmiana czcionki.....	320

Używanie programu geqn.....	321
gtbl.....	321
Uruchamianie programu gtbl.....	322
Opcje.....	322
Format.....	322
Dane.....	325
Przykłady.....	325
Podsumowanie.....	326
Rozdział 19. TeX i LaTeX.....	327
Edycja a skład tekstu.....	327
TeX.....	328
Proste formatowanie tekstu.....	328
Czcionki.....	330
Kontrolowanie odstępów.....	331
Układ strony.....	332
Grupowanie.....	333
Symbole matematyczne.....	333
Wstawianie wykresów.....	334
Makropolecenia.....	335
LaTeX – rozszerzenie systemu TeX.....	336
Definicja dokumentu systemu LaTeX.....	337
Pakiety.....	338
Znaki narodowe.....	339
Struktura dokumentu systemu LaTeX.....	339
Dodawanie innych elementów strukturalnych.....	340
Praca z tabelami i wykresami.....	340
VirTeX i IniTeX.....	341
Podsumowanie.....	341
Rozdział 20. Drukowanie.....	343
Konfiguracja drukarki.....	343
Nazwy portów.....	344
Sterowniki drukarek.....	345
Polecenia służące do drukowania.....	346
Podsumowanie.....	347
Rozdział 21. Linux i multimedia.....	349
Karty dźwiękowe.....	349
Używanie wbudowanego głośniczka.....	350
Konfiguracja karty dźwiękowej.....	351
Problemy z instalacją i konfiguracją karty dźwiękowej.....	357
Programy obsługujące kartę dźwiękową.....	358
vplay, vrec, splay oraz srec.....	358
WAVplay.....	359
Sound Studio.....	359
MixViews.....	359
Dżojstiki.....	360
Podsumowanie.....	360

Część czwarta Graficzne interfejsy użytkownika

Rozdział 22. Instalacja i konfiguracja XFree86.....	365
Co to jest XFree86?.....	366
Dystrybucja oprogramowania XFree86.....	367
Wybór serwera X.....	368
Ręczna instalacja XFree86.....	369
Instalowanie XFree86 za pomocą skryptu.....	370
Zmienna środowiskowa PATH.....	370
Konfigurowanie XFree86.....	371
Gdzie umieścić pliki Xconfig i XF86Config.....	372
SuperProbe.....	373
Użycie programów ConfigFX86 i fx86config.....	375
Pliki Xconfig i XF86Config.....	376
Ścieżki dostępu.....	376
Ustawienia klawiatury.....	377
Definiowanie myszki.....	379
Model monitora.....	380
Karty graficzne.....	382
Serwer XFree86.....	383
Testowanie konfiguracji XFree86.....	385
Plik .xinitrc.....	385
Podsumowanie.....	387
Rozdział 23. Wabi.....	389
Co potrafi Wabi?.....	390
Instalacja Wabi.....	391
Uruchamianie aplikacji systemu Windows 3.X.....	392
Podsumowanie.....	394
Rozdział 24. Ghostscript i Ghostview.....	395
Skąd można wziąć Ghostscript?.....	396
Pakiet Ghostscript.....	398
Konfigurowanie pakietu Ghostscript do współpracy z X.....	399
Przekierowanie wyjścia programu Ghostscript.....	400
Zmiana rozmiaru papieru.....	401
Zmienne środowiskowe programu Ghostscript.....	402
Ghostview.....	403
Obsługa programu ghostview.....	404
Czcionki Ghostview.....	405
Zasoby X używane przez Ghostview.....	405
Podsumowanie.....	406
Część piąta Linux dla programistów.....	407
Rozdział 25. gawk.....	409
Ogólnie o języku gawk.....	410
Pliki, rekordy i pola.....	410
Kojarzenie wzorców i akcji.....	412
Proste wzorce.....	413
Porównania i arytmetyka.....	414
Łańcuchy znaków i liczby.....	416
Formatowanie wyjścia.....	416
Zmiana separatora pól.....	419

Symbole specjalne.....	419
Wywoływanie programów w języku gawk.....	421
BEGIN oraz END.....	421
Zmienne.....	422
Zmienne wewnętrzne.....	423
Instrukcje strukturalne.....	424
Instrukcja if.....	424
Pętla while.....	425
Pętla for.....	426
next oraz exit.....	427
Tablice.....	427
Podsumowanie.....	428
Rozdział 26. Programowanie w języku C.....	429
Język C.....	429
Kompiletor GNU C.....	430
Uruchamianie GCC.....	430
Opcje kompilatora GCC.....	430
Opcje dotyczące optymalizacji.....	431
Opcje współpracy z debuggerem i programem profilującym.....	432
Wyszukiwanie błędów – debugger gdb.....	432
Kompilowanie kodu przeznaczonego do debugowania.....	433
Podstawowe polecenia gdb.....	433
Przykładowa sesja gdb.....	434
Inne narzędzia dla programistów.....	438
xgdb.....	438
calls.....	439
cproto.....	441
indent.....	442
gprof.....	443
f2c oraz p2c.....	444
Podsumowanie.....	444
Rozdział 27. Programowanie w C++.....	447
Język C++.....	447
Dlaczego C++?.....	448
Hermetyzacja danych.....	448
Dziedziczenie.....	449
Polimorfizm.....	449
Klasy i metody.....	449
Opcje kompilatora GCC.....	450
Opcje współpracy z debuggerem i programem profilującym.....	451
Opcje specyficzne dla języka C++.....	451
Wyszukiwanie błędów w aplikacjach C++.....	452
Wyszukiwanie błędów w funkcjach wirtualnych.....	453
Wyszukiwanie błędów w funkcjach obsługi wyjątków.....	454
Polecenia gdb specyficzne dla C++.....	455
Biblioteki klas GNU C++.....	455
Strumienie.....	455
Łańcuchy znaków.....	456
Liczby przypadkowe.....	456

Analiza danych statystycznych.....	457
Listy.....	457
Klasy Plex.....	458
Stosy.....	459
Kolejki.....	459
Zbiory.....	460
Podsumowanie.....	461
Rozdział 28. Perl.....	463
Język Perl.....	464
Tworzenie i uruchamianie programów w języku Perl.....	464
Dane.....	465
Zmienne.....	465
Liczby.....	466
Łańcuchy znaków.....	466
Operatory plikowe.....	468
Tablice.....	469
Elementy strukturalne.....	471
Instrukcja blokowa.....	471
Instrukcja if.....	471
Instrukcja unless.....	473
Instrukcja for.....	473
Instrukcja foreach.....	474
Instrukcja while.....	475
Instrukcja until.....	475
Funkcje.....	476
Przekazywanie argumentów do funkcji.....	476
Zwracanie wartości.....	477
Operatory.....	477
Konwertowanie programów do języka Perl.....	477
Podsumowanie.....	478
Rozdział 29. Podstawy języków Tcl i Tk.....	479
Czym jest Tcl?.....	479
Czym jest Tk?.....	480
Język Tcl.....	482
Zmienne i przypisywanie wartości.....	482
Podstawienie polecenia.....	483
Użycie cudzysłów.....	483
Instrukcja if.....	484
Instrukcja for.....	485
Polecenie while.....	486
Polecenie switch.....	486
Komentarze.....	487
Tk – nowe możliwości Tcl.....	488
Ramki.....	488
Przyciski.....	488
Menu i przyciski menu.....	489
Listy rozwijane.....	490
Paski przewijania.....	490
Podsumowanie.....	490
Rozdział 30. Inne kompilatory.....	491
Ada.....	492

COBOL.....	492
DSP.....	492
Eiffel.....	493
FORTRAN.....	493
LISP.....	493
Modula-3.....	494
OGI.....	494
Scheme.....	494
Scilab.....	495
Podsumowanie.....	495
Rozdział 31. Smalltalk/X.....	497
Co to jest Smalltalk/X.....	497
Instalacja Smalltalk/X.....	498
Uruchamianie systemu Smalltalk/X.....	498
Środowisko Smalltalk/X.....	499
Przeglądarki (Browsers).....	500
Przeglądarka systemowa (System Browser).....	500
Przeglądarka hierarchii klas (Class Hierarchy Browser).....	505
Klasy implementujące daną metodę (Implementors).....	505
Klasy wysyłające dany komunikat (Senders).....	506
Przeglądarka wprowadzonych zmian (Changes Browser).....	507
Przeglądarka katalogów (Directory Browser).....	508
Workspace (obszar roboczy).....	509
File Browser (przeglądarka plików).....	509
Projekty (Projects).....	511
Narzędzia (Utilities).....	511
Goodies.....	512
Gry i programy demonstracyjne (Games & Demos).....	513
Edycja za pomocą przeglądarki.....	513
Okno Inspector.....	514
Okno Debugger.....	515
Podsumowanie.....	517
Część szósta Linux dla administratorów	
Rozdział 32. Podstawy administrowania systemem.....	521
Konto root.....	522
Uruchamianie i zamknięcie systemu.....	522
Uruchamianie systemu z dyskietki.....	523
Uruchamianie systemu za pomocą programu LILO.....	524
Zamykanie systemu.....	524
Montowanie systemów plików.....	525
Montowanie dyskietki.....	526
Tworzenie nowego systemu plików.....	526
Odmontowywanie systemów plików.....	526
Sprawdzanie systemów plików.....	527
Plik wymiany.....	527
Kompresja danych – programy gzip i compress.....	528
Program tar.....	529
Kopie zapasowe.....	531

Konfiguracja systemu.....	531
Zmiana nazwy systemu.....	532
Dyskietka startowa.....	532
Zapomniałeś hasła użytkownika root?.....	532
Informacja dnia.....	532
Podsumowanie.....	533
Rozdział 33. Urządzenia.....	535
Urządzenia znakowe i blokowe.....	535
Główne i poboczne numery urządzeń.....	536
Polecenie mknod.....	537
Drukarki.....	538
lpd – rezydentny program drukujący.....	538
Proces drukowania.....	539
Plik /etc/printcap i katalogi kolejki.....	539
Dodawanie drukarki za pomocą polecenia mknod.....	541
Zarządzanie drukarkami – program lpc.....	543
Zarządzanie kolejkami drukowania za pomocą programów lpq i lprm.....	545
Terminal.....	546
Używanie kart multiport.....	546
Dodawanie terminali podłączonych przez port szeregowy.....	547
Proces logowania.....	548
/sbin/getty i /etc/gettydefs.....	549
Pliki konfiguracyjne terminala: /etc/ttys i /etc/inittab.....	550
Definicje terminali – plik /etc/termcap.....	552
Dodawanie terminalu.....	553
Polecenia stty oraz tset.....	554
Resetowanie terminalu.....	555
Modemy.....	555
Podsumowanie.....	556
Rozdział 34. Procesy.....	557
Co trzeba wiedzieć o procesach.....	557
Typy procesów.....	558
Użycie polecenia ps.....	558
Dane wyświetlane przez program ps.....	558
Interpretery polecen uruchamiane przy logowaniu.....	559
Uwagi dla użytkownika root.....	559
Przydatne opcje programu ps.....	560
Uwagi dla administratorów.....	562
Polecenie kill.....	562
Usuwanie procesów potomnych.....	563
Co można, a czego nie można usunąć.....	563
Podsumowanie.....	563
Rozdział 35. Użytkownicy i konta.....	565
Konto administratora.....	566
Konta użytkowników – plik /etc/passwd.....	567
Identyfikatory użytkowników.....	568
Hasła.....	569
Numeryczny identyfikator użytkownika.....	569
Identyfikator grupy.....	570
Komentarz.....	570
Katalog domowy.....	570

Powłoka domyślna.....	571
Użytkownicy systemowi.....	571
Dodawanie nowego konta.....	572
Usuwanie kont.....	573
Grupy.....	574
Grupy systemowe.....	576
Dodawanie nowej grupy.....	576
Dodawanie użytkowników do grupy.....	577
Usuwanie grupy.....	577
Polecenie su.....	578
Podsumowanie.....	578
Rozdział 36. Obsługa urządzeń SCSI.....	579
Nowe standardy SCSI.....	580
Obsługiwane urządzenia SCSI.....	581
Sterowniki SCSI.....	581
Dyski twarde.....	581
Napędy CD-ROM.....	582
Sterowniki SCSI.....	583
Napędy taśmowe.....	583
Inne urządzenia.....	583
SCSI – rozwiązywanie problemów.....	583
Podsumowanie.....	585
Rozdział 37. Praca w sieci.....	587
TCP/IP.....	587
Wymagania sprzętowe.....	589
Pliki konfiguracyjne.....	590
Zanim zaczniesz.....	590
Adres IP.....	591
Maska podsieci.....	591
Adres sieci.....	592
Adres rozgłoszenia.....	592
Adres bramki sieciowej.....	592
Adres serwera nazw.....	592
Konfigurowanie interfejsu pozornego.....	593
Pliki konfiguracyjne – szczegóły.....	593
Pliki rc.....	593
/etc/hosts.....	595
/etc/networks.....	596
/etc/host.conf.....	597
resolv.conf.....	597
/etc/protocols.....	597
/etc/services.....	598
/etc/hostname lub /etc/HOSTNAME.....	598
Testowanie konfiguracji i rozwiązywanie problemów.....	599
Polecenie netstat.....	599
Wyświetlanie danych o połączeniach.....	599
Statystyki interfejsu sieciowego.....	600
Tablica kierowania przepływem danych (routing table).....	600
ping.....	601

Podsumowanie.....	601
Rozdział 38. SLIP i PPP	603
Konfigurowanie interfejsu pozornego.....	603
Konfiguracja protokołu SLIP.....	604
Protokół SLIP.....	604
dip.....	606
Konfiguracja protokołu PPP.....	608
Zakładanie konta PPP.....	608
Nawiązywanie połączenia – program chat.....	609
Uruchamianie pppd.....	610
Testowanie konfiguracji.....	611
PPP a bezpieczeństwo.....	611
Używanie systemu DNS z protokołami SLIP i PPP.....	614
Podsumowanie.....	615
Rozdział 39. UUCP	617
Konfiguracja protokołu UUCP.....	618
Konfigurowanie Taylor UUCP.....	618
Konfigurowanie HDB UUCP.....	622
Połączenie UUCP.....	624
Komunikacja bezpośrednią.....	625
Skrypty logowania.....	625
Modyfikowanie harmonogramu dostępu.....	627
UUCP a bezpieczeństwo.....	627
Używanie UUCP.....	628
Przesyłanie poczty.....	630
Przesyłanie plików.....	630
Sprawdzanie transferu.....	631
Podsumowanie.....	632
Rozdział 40. Konfigurowanie poczty.....	633
Jak działa poczta elektroniczna.....	634
Konfigurowanie programu sendmail.....	634
Plik sendmail.cf.....	635
Położenie tablic konfiguracyjnych.....	637
Tworzenie pliku sendmail.cf.....	639
Używanie programu sendmail w wersji 8.....	640
smail.....	641
Konfigurowanie programu smail.....	641
Konfigurowanie smail z protokołem UUCP.....	642
Ustawianie lokalnych nazw domenowych.....	643
Ustawianie domeny lokalnej dla poczty wychodzącej.....	643
Inne nazwy UUCP.....	643
Ustawianie serwera UUCP.....	644
Konfigurowanie smail z protokołem TCP.....	644
Modyfikacja zachowania programu smail.....	645
Podsumowanie.....	647
Rozdział 41. Konfigurowanie grup dyskusyjnych.....	649
Usenet i grupy dyskusyjne.....	650
NNTP.....	651
Instalowanie serwera NNTP.....	652
Konfigurowanie nntpd.....	653

Konfiguracja przeglądarek grup dyskusyjnych.....	654
Konfigurowanie programu trn.....	654
Konfigurowanie programu tin.....	655
Podsumowanie.....	656
Rozdział 42. Bezpieczeństwo w sieci.....	657
Słabe hasła.....	657
Bezpieczeństwo plików.....	658
Dostęp przez modem.....	658
Modemy oddzwaniające.....	658
Problemy z modemami.....	659
Jak modem obsługuje połączenie.....	660
UUCP.....	660
Dostęp poprzez sieć lokalną.....	661
Śledzenie intruza.....	661
Przygotowywanie się na najgorsze.....	662
Podsumowanie.....	662
Rozdział 43. NFS	665
Konfiguracja NFS w systemie Linux.....	666
Konfiguracja serwera linuxowego.....	666
Konfigurowanie klienta.....	668
Podsumowanie.....	670
Rozdział 44. NIS i YP	671
Konfiguracja domeny NIS.....	672
Programy rezydentne obsługujące NIS.....	673
Konfigurowanie komputera-nadzorcy NIS.....	673
Konfigurowanie nadzorców zastępczych.....	675
Klienci NIS.....	676
Podsumowanie.....	676
Rozdział 45. Kopie zapasowe.....	679
Po co tworzyć kopie zapasowe.....	679
Nośniki.....	680
Harmonogram tworzenia kopii zapasowych.....	681
Inwentaryzowanie zapisanych danych.....	683
Używanie programu tar do tworzenia kopii zapasowych.....	684
Podsumowanie.....	687
Rozdział 46. cron i at.....	689
Używanie programu cron.....	689
Tworzenie pliku crontab.....	690
Zarządzanie plikami crontab.....	692
Bardziej złożone polecenia.....	693
Program at.....	694
Podsumowanie.....	695

Część siódma Konfiguracja węzła internetowego

Rozdział 47. Konfiguracja węzła internetowego.....	699
Podłączanie się do Internetu.....	700
Usługi, których potrzebujesz.....	701
Połączenie bezpośrednie za pomocą bramki.....	702
Połączenie za pomocą czujnej bramki.....	702
Usługodawcy internetowi.....	702
Podsumowanie.....	703
Rozdział 48. Konfigurowanie FTP i anonimowego FTP.....	705
Co to jest FTP?.....	706
Używanie FTP.....	706
Powiązania pomiędzy FTP a TCP/IP.....	710
Konfiguracja FTP.....	712
Konfigurowanie programu ftpd.....	713
Konta FTP.....	714
Konfigurowanie katalogów.....	715
Ustawianie praw dostępu.....	716
Testowanie systemu.....	716
Bezpieczniejsze FTP.....	717
Zabezpieczanie anonimowego FTP.....	718
Podsumowanie.....	719
Rozdział 49. Konfiguracja węzła WAIS	721
Kompilowanie i instalacja programu freeWAIS.....	724
Konfigurowanie programu freeWAIS.....	726
Uruchamianie serwera freeWAIS.....	729
Tworzenie własnych indeksów dla programu WAIS.....	730
Pliki indeksowe programu WAIS.....	730
Polecenie waisindex.....	731
Inne możliwości systemu WAIS.....	734
Podsumowanie.....	735
Rozdział 50. Konfigurowanie usługi Gopher.....	737
Gopher i Linux.....	738
Konfigurowanie usługi Gopher.....	740
Plik gopherd.conf.....	741
Plik gopherdlocal.conf.....	743
Konfigurowanie pliku Makefile.....	745
WAIS i Gopher.....	749
Katalogi systemu Gopher.....	750
Uruchamianie usługi Gopher.....	753
Daj światu znać o sobie.....	755
Podsumowanie.....	755
Rozdział 51. Konfigurowanie węzła WWW	757
Oprogramowanie serwera WWW.....	758
Rozpakowywanie plików z oprogramowaniem serwera WWW.....	758
Kompilowanie oprogramowania serwera WWW.....	759
Konfiguracja oprogramowania serwera WWW.....	760
Uruchamianie serwera WWW.....	766
Apache.....	767
Użycie programu make z pakietem Apache.....	768

Edycja pliku konfiguracyjnego.....	770
Opcje serwera Apache httpd.....	771
Konfigurowanie serwera Apache w niewielkim węźle WWW.....	771
Podsumowanie.....	772
Rozdział 52. Skrypty CGI.....	773
Co to jest CGI?.....	773
CGI i HTML.....	774
CGI i Perl.....	776
Podsumowanie.....	778
Rozdział 53. Podstawy języka HTML.....	781
Programy do tworzenia dokumentów HTML.....	782
Tworzenie stron WWW w systemie Windows.....	783
Tworzenie dokumentów HTML w systemie Linux.....	786
Konserwacja dokumentów HTML.....	787
Podstawy języka HTML.....	789
Jak wygląda język HTML?.....	789
Początek dokumentu HTML.....	791
Akapity.....	793
Hiperłącza.....	795
Wyliczenia.....	797
Zmiana kroju pisma.....	798
Podstawy języka HTML	
Inne znaczniki.....	799
Podsumowanie.....	801
Rozdział 54. Podstawy języków Java i JavaScript.....	803
Co będzie Ci potrzebne?.....	804
Język Java.....	805
JavaScript i HTML.....	807
Podsumowanie.....	808
Rozdział 55. Projektowanie spójnych stron WWW.....	809
Dostępność systemu.....	810
Utrzymywanie porządku na stronach WWW.....	811
Umieszczaj najważniejsze informacje na początku strony.....	811
Dzielenie dokumentu na wiele stron.....	812
Ikony.....	812
Prawidłowe stosowanie odnośników.....	813
Prawidłowe stosowanie znaczników HTML.....	813
Podsumowanie.....	814
Część ósma Programowanie dla zaawansowanych	
Rozdział 56. Zarządzanie kodem źródłowym.....	817
Program make.....	818
Przykładowy plik makefile.....	818
Format pliku makefile.....	819
Tworzenie kilku wersji programu.....	821
Wymuszanie aktualnienia.....	822

Makropolecenia.....	823
Reguły przyrostkowe.....	825
RCS.....	826
Delty.....	827
Tworzenie pliku RCS.....	828
Odzyskiwanie pliku RCS.....	829
Słowa kluczowe.....	829
Uzyskiwanie informacji o wersji z pliku RCS.....	830
Dostęp do plików RCS.....	831
Porównywanie wersji i łączenie poprawek.....	831
Praca z programem make i systemem RCS.....	832
Podsumowanie.....	833
Rozdział 57. Jądro systemu.....	835
Uaktualnianie i instalowanie nowych składników jądra systemu.....	836
Kompilowanie kodu źródłowego jądra systemu.....	837
Dodawanie sterowników urządzeń.....	840
Uaktualnianie bibliotek.....	840
Kompilator języka C dla systemu Linux.....	841
Opcje współpracy z debugerem i programem profilującym.....	843
Wyszukiwanie błędów – debuger gdb.....	843
Podsumowanie.....	844
Rozdział 58. Tworzenie sterowników urządzeń.....	845
Sterowniki urządzeń.....	846
Przerwania.....	847
Struktura sterownika urządzenia dla systemu Linux.....	848
Pliki nagłówkowe.....	849
Otwieranie urządzenia.....	849
Zamykanie urządzenia.....	850
Funkcje strategii.....	851
Funkcje write().....	851
Funkcje read().....	852
Podprogramy start i ioctl.....	852
Używanie nowego sterownika.....	853
Podsumowanie.....	854
Rozdział 59. Projekt Wine.....	857
Obecny stan projektu Wine.....	858
Konfiguracja Wine.....	858
Wymagania systemowe.....	858
Skąd można załadować Wine.....	859
Jak zainstalować Wine.....	859
Jak skonfigurować Wine przed komplikacją.....	860
Wstępna konfiguracja parametrów czasu wykonania za pomocą skryptu Configure.....	860
Automatyczna konfiguracja.....	862
Tworzenie pliku wykonywalnego.....	862
Używanie programu Wine.....	862
Parametry konfiguracyjne.....	862
Opcje dostępne z wiersza poleceń.....	863
Debuger programu Wine.....	865
Zasady działania programu Wine.....	866
Ładowanie programów do pamięci.....	866
Biblioteka Wine.....	867

Gdzie kończy się Wine, a zaczyna MS-Windows?.....	868
Ograniczenia programu Wine.....	869
Działające oprogramowanie.....	869
Używanie programu winestat do analizy programów systemu Windows.....	869
Najważniejsze braki programu Wine.....	871
Oprogramowanie, które prawdopodobnie nigdy nie będzie działać.....	871
Podsumowanie.....	872
Rozdział 60. HylaFAX.....	873
Instalacja programu HylaFAX.....	874
Kompilacja programu HylaFAX.....	874
Dodawanie modemów.....	875
Wysyłanie faksu.....	878
Opcje programu sendfax.....	878
Okładki.....	879
Odbieranie faksu.....	879
Podsumowanie.....	880
Rozdział 61. Gry.....	881
Które gry zainstalowałeś?.....	881
Gry dla systemu X Window.....	881
Gry dostępne w menu głównym menedżera xdm.....	883
Spider.....	883
Gry dla systemu X Window	
Puzzle.....	883
GNU Chess.....	884
Xtetris.....	884
Xlander.....	885
Xmahjongg.....	885
Xvier.....	885
Ico.....	885
Maze.....	886
Xeyes.....	886
Xgas.....	886
Xlogo.....	887
Xroach.....	887
Xhextris.....	887
Xbombs.....	887
Xpaint.....	888
Xfractint.....	888
Gry działające w trybie tekstowym.....	889
Tekstowe gry przygodowe.....	890
Gry słowne.....	891
Gry karciane.....	892
Gry planszowe.....	892
Symulatory.....	894
Gry „wideo”.....	894
Gry matematyczne i programy użytkowe.....	895
Inne gry.....	897
Programy demonstracyjne i użytkowe.....	898
Podsumowanie.....	899

Rozdział 62. Adabas-D i inne bazy danych.....	901
Bazy danych kompatybilne z dBASE.....	901
Co to jest xBase?.....	902
Co to jest FlagShip?.....	903
Instalowanie programu FlagShip.....	905
Używanie programu FlagShip.....	906
Przenoszenie istniejących aplikacji.....	907
dbMan.....	908
Adabas-D.....	910
Instalowanie programu Adabas-D.....	910
LINCKS.....	912
Inne bazy danych.....	913
Podsumowanie.....	913
Rozdział 63. StarOffice.....	915
Instalacja pakietu StarOffice.....	916
Uruchamianie StarOffice.....	917
StarWriter.....	917
StarCalc.....	919
StarImpress.....	919
Importowanie i eksportowanie plików.....	920
Podsumowanie.....	921
Rozdział 64. Program Lone-Tar firmy Lone Star Software.....	923
Co to jest Lone-Tar?.....	924
Interfejs programu Lone-Tar.....	924
Instalacja programu Lone-Tar.....	926
Tworzenie kopii zapasowych za pomocą programu Lone-Tar.....	927
Weryfikowanie plików.....	931
Przywracanie wcześniejszych wersji plików.....	931
Programy użytkowe i środowisko:	
dostosowywanie programu Lone-Tar do własnych potrzeb.....	933
Podsumowanie.....	934
Dodatki	
Dodatek A Węzły FTP i listy dyskusyjne poświęcone Linuxowi.....	937
Węzły FTP.....	937
Co to jest FTP?.....	938
Łączenie się z systemem zdalnym i ładowanie plików.....	938
Używanie programu ftpmail.....	940
Węzły FTP poświęcone Linuxowi.....	941
Grupy dyskusyjne w sieci Usenet.....	942
Dodatek B Komercyjni dystrybutorzy Linuxa.....	945
Dystrybucja Debian.....	946
Płyty CD-ROM Yggdrasil Plug-and-Play oraz Biblia Linuxa.....	946
Linux na płytach CD-ROM firmy Nascent.....	947
CD-ROM Unifix 1.02.....	947
Fintronic Linux Systems.....	947
InfoMagic Developer's Resource CD-ROM kit.....	947
Linux Quaterly CD-ROM.....	948
Linux Systems Labs.....	948

Sequoia International Motif Development Package.....	948
Takelap Systems Ltd.....	948
Trans-Ameritech Linux Plus BDS CD-ROM.....	949
Caldera OpenLinux.....	949
Dodatek C Projekt dokumentacji Linuxa.....	951
Dodatek D Publiczna licencja GNU.....	953
Publiczna licencja GNU, wersja 2, czerwiec 1991.....	953
E.1 Preambuła.....	954
E.2 Warunki licencji GNU: kopiowanie, dystrybucja i modyfikowanie plików.....	955
Jak zastosować zasady licencji GNU do własnych programów.....	959
Dodatek E Informacje o prawach autorskich.....	961
Dodatek F Zawartość płyt CD-ROM.....	965
Nagradzany system operacyjny.....	965
Pulpit.....	965
Serwer internetowy.....	965
Platforma edukacyjna.....	966
Oprogramowanie.....	966
Skorowidz.....	967

O autorze

Dr Tim Parker napisał ponad 40 książek, poruszających wiele problemów związanych z komputerami. Jest także edytorem technicznym magazynu *SCO World Magazine*, ale często współuczestniczy w tworzeniu innych czasopism, na przykład *UNIX Review*, *Canadian Computer Reseller*, *UNIQUE: The UNIX Systems Information Source*, *Windows NT Systems* i innych. Dr Parker jest najczęściej publikowanym autorem tekstów dotyczących systemów UNIX-owych. Prócz komputerów pasjonuje go również lataanie, nurkowanie oraz jazda na swym ulubionym motocyklu.

Dedykacja

Książkę tę poświęcam Yvonne. Cóż więcej mogę powiedzieć? *Amor vincit omnia*.

Podziękowania

Podobnie jak było w przypadku poprzednich wydań książki „Linux – Księga eksperta”, również ta książka powstała przy współudziale wspaniałych ludzi z wydawnictwa Sams. Nad opracowaniem tego tomu, z którego jesteśmy dumni, pracowali ze mną **Jane Brownlow** i **Mark Cierzniaik**. Dziękujemy również osobom, które zajęły się techniczną stroną projektu, podsuwając nam doskonałe pomysły pozwalające ulepszyć książkę, oraz osobom zajmującym się korektą tekstu, które dbały o to, by każde zdanie było prawidłowe. Bez tych ludzi, pracujących przy bardzo napiętym harmonogramie, książka ta na pewno byłaby na o wiele niższym poziomie.

Dziękujemy również pracownikom firmy **Caldera**, którzy udostępnili nam nie tylko różne wersje dystrybucji Linuxa, ale także niektóre aplikacje. Dziękujemy firmom **Lone Star** i **Cactus**, które udostępniły program Lone Tar. Ja osobiście dziękuję moim rodzicom, którzy potrafiili znieść kilka miesięcy, w czasie których nie mogłem ich odwiedzać. Na koniec dziękuję **Yvonne** za to że potrafiła zrozumieć, dlaczego znów spędżam poranki, dni i wieczory przed komputerem zamiast przeznaczać je na spotkania z nią.

Wstęp

Ogromna popularność systemu Linux bardzo nas zaskoczyła. Kiedy w roku 1994 zaczynaliśmy pisać pierwszą wersję książki *Linux Unleashed*, nie przypuszczaliśmy, że Linux zdobędzie świat szturmem. Wiedzieliśmy oczywiście, że to wspaniały system operacyjny i że jest doskonałym sposobem na to, by pobawić się UNIX-em na zwykłym peccie. Wiedzieliśmy też, że eksperymentatorzy i programiści zakochają się w Linuxie od pierwszego wejrzenia, jednak naprawdę nie spodziewaliśmy się, że ukaże się aż tyle wydań tej książki. Jest to, jak dotąd, piąta edycja *Linux Unleashed*. Dwie z nich poświęcone były dystrybucjom Red Hat i Slackware, ponieważ objęły one wszystkie wersje systemu Linux. Poza tym, moja książka pod tytułem „*Linux System Administrator's Survival Guide*” (opublikowana przez wydawnictwo Sams) cieszy się dużą popularnością od trzech lat.

Skąd bierze się tak ogromna popularność Linuxa? Jedną z najważniejszych przyczyn jest chyba jego powszechna dostępność. Linux jest bowiem systemem bezpłatnym, a jeśli już trzeba za niego zapłacić, to bardzo niewiele. Jest też dostępny w setkach węzłów sieci Internet oraz na tysiącach płyt CD-ROM, które można kupić w księgarniach i kioskach. Drugą przyczyną jest jego atrakcyjność jako systemu UNIX-owego. Cokolwiek nie mówiliby ludzie z Redmond, UNIX był zawsze wydajniejszy niż Windows (ostatnie wersje Windows nie dadzą się nawet uruchomić na komputerze z procesorem 80386!). Wydaje się, że trend ten będzie się utrzymywał, ponieważ UNIX potrafi po prostu lepiej wykorzystać możliwości szybkich procesorów. Wielozadaniowość i wielowątkowość zostały stworzone dla systemu UNIX, a Windows tylko go naśladuje.

Innym powodem popularności Linuxa jest liczba dostępnych aplikacji. Wejdź na dowolną stronę WWW z oprogramowaniem dla tego systemu, a przekonasz się sam. Co więcej – aplikacje te nie są tylko prostymi grami czy też pisany bez żadnego zaangażowania programami dla zapewnienia podstawowej funkcjonalności systemu. Są to profesjonalnie wykonane aplikacje, które śmiało mogą konkurować z programami sprzedawanymi za ogromne pieniądze, przeznaczonymi dla innych systemów operacyjnych. Linux jest również popularny ze względu na mechanizm wsparcia w kłopotach: jeśli natkniesz się na jakiś problem, posydasz wiadomość do jednej z grup dyskusyjnych i bardzo szybko otrzymujesz pomoc od innych użytkowników. Popularność Linuxa wynika też z faktu, że używanie go to czysta przyjemność. Czego można chcieć więcej od systemu operacyjnego?

Ta edycja Linux Unleashed zawiera sporo nowego materiału. Otrzymaliśmy pocztą elektroniczną mnóstwo uwag i sugestii, które wzięliśmy sobie do serca, opracowaliśmy więc dokładniejsze opisy instalacji i konfiguracji, dodaliśmy także kilka nowych rozdziałów o poszczególnych aplikacjach. Wiele rozdziałów napisaliśmy od nowa, aby były bardziej czytelne i zrozumiałe. W wyniku tych prac, mamy nadzieję, powstała książka, którą czyta się z przyjemnością od początku do końca.

Część pierwsza

Wstęp

W tej części:

- υ Wstęp do Linuxa
- υ Rodzaje Linuxa
- υ Instalacja Linuxa
- υ Używanie programu LILO
- υ Podsumowanie instalacji

Rozdział 1.

Wstęp do Linuxa

Tim Parker

W tym rozdziale:

- υ Linux – co to takiego?
- υ Historia Linuxa
- υ Co Linux może zrobić dla Ciebie
- υ Co jest potrzebne do instalacji Linuxa
- υ Minimalne wymagania sprzętowe
- υ Prawa autorskie
- υ Uzyskiwanie pomocy
- υ Grupy dyskusyjne w sieci Usenet
- υ O czym mówi ta książka?

Widziałeś już pewnie wiele książek i czytałeś wiele artykułów na temat Linuxa. Istnieją grupy dyskusyjne z setkami nowych wiadomości każdego dnia, CD-ROM-y z Linuxem sprzedawane są tysiącami, coraz więcej jest też użytkowników Windows próbujących wyrobić sobie zdanie o tym dziwnym systemie. Pomimo że system ten jest tak popularny, wciąż są tysiące, jeśli nie miliony użytkowników ciekawych Linuxa, ale bojących się kilku pierwszych kroków: instalacji oraz konfiguracji systemu. Pomóc Wam – to cel tej książki. Poprowadzimy Was krok po kroku przez instalację, pokażemy jak używać Linuxa i wprowadzimy w cudowny świat UNIX-a.

Jednak zanim zaczniemy, powiedzmy kilka słów o wymowie nazwy Linux. Są dwie szkoły wymawiania samogłoski *i* w tym wyrazie. Ponieważ pierwsza wersja Linuxa została opracowana przez programistę o imieniu Linus [wym. *lajnys*], wiele osób uważa, że prawidłowa wymowa tej samogłoski to [*ai*], tak jak w jego imieniu [*lainyks*]. Z drugiej

strony, Linux został napisany po to, żeby zastąpić system podobny do UNIX-a o nazwie Minix (z krótkim i zbliżonym do polskiego y), więc reszta społeczności linuxowej tak właśnie [*lyniks*] wymawia tę nazwę. Która wersja jest lepsza? Autorzy tego systemu używają tej drugiej, ale większość Amerykanów woli pierwszą. Wybierz tę, która Ci bardziej odpowiada. I tak każdy będzie wiedział, o co chodzi.

Linux - co to takiego?

Linux, jeśli ktoś jeszcze się nie zorientował, jest bezpłatnym, wielozadaniowym systemem operacyjnym, pozwalającym na jednoczesną pracę wielu użytkowników, który zachowuje się tak, jak UNIX. Linux został napisany specjalnie dla komputerów klasy PC (z procesorami firmy Intel) i wykorzystuje ich architekturę dla zapewnienia użytkownikowi wydajności porównywalnej z wydajnością stacji roboczych pracujących pod kontrolą systemu UNIX. Opracowano też sporo implementacji Linuxa dla innych platform sprzętowych; działają one bardzo podobnie jak wersja dla PC, na której skoncentrujemy się w tej książce.

Na początek spójrzmy na ten system całościowo. Co otrzymujesz po zainstalowaniu Linuxa? Oto krótka lista. Większość z jej pozycji zostanie omówiona dokładniej w następnych rozdziałach.

Kernel - jądro systemu

Linux jest wielozadaniowym, wielużytkownikowym systemem operacyjnym, który działa tak samo jak UNIX, zarówno pod względem zachowania się jądra systemu, jak i obsługi urządzeń zewnętrznych. Linux posiada wszystkie zalety UNIX-a, a poza nimi jeszcze kilka nowych rozszerzeń zwiększających jego elastyczność. Kod źródłowy Linuxa oraz programów użytkowych jest dostępny za darmo.

Jądro systemu było pierwotnie napisane dla trybu chronionego procesorów Intel 80386. Procesory te zostały zaprojektowane z myślą o wielozadaniowości (choć większość z nich pracowała pod kontrolą systemu DOS, nie mającego z wielozadaniowością nic wspólnego), a Linux wykorzystuje ich zaawansowane możliwości. Mocną stroną procesora 80386, w porównaniu z poprzednimi produktami firmy Intel, jest zarządzanie pamięcią. Programowa emulacja koprocesora arytmetycznego pozwala Linuxowi działać na maszynach nie posiadających go.

Linux pozwala na współużytkowanie plików wykonywalnych (ang. *shared executables*), zatem gdy uruchomiona jest więcej niż jedna kopia aplikacji (przez jednego użytkownika lub przez kilku użytkowników pracujących z tym samym programem), wszystkie kopie używają tego samego obszaru pamięci. Pozwala to na bardzo wydajne wykorzystanie pamięci RAM.

Jądro Linuxa obsługuje również stronicowanie na żądanie (ang. *demand paging*), co oznacza, że tylko niezbędne części programu są wczytywane do pamięci operacyjnej.

Żeby jeszcze bardziej zoptymalizować wykorzystanie RAM-u, Linux używa jednolitego modelu pamięci. To rozwiązanie pozwala systemowi przeznaczyć całą wolną pamięć na pamięć podręczną dla dysku, co przyspiesza dostęp do często używanych programów i danych. Rozmiar pamięci podręcznej jest automatycznie zmniejszany, gdy wzrasta zużycie pamięci przez aplikacje.

Aby sprostać dużym wymaganiom pamięciowym przy małej ilości fizycznej pamięci RAM, Linux używa pliku lub partycji wymiany (ang. *swap space*). Pozwala to na zapisywanie obszarów pamięci w zarezerwowane miejsce na dysku twardym, które jest traktowane jako „przedłużenie” pamięci RAM. Ceną płaconą za powiększenie pamięci operacyjnej jest spowolnienie dostępu do danych.

Linux pozwala również na używanie bibliotek współużytkowanych dynamicznie (ang. *dynamic shared libraries*). Biblioteki te są wspólne dla wielu aplikacji, co umożliwia zredukowanie wielkości programów. Dla utrzymania kompatybilności z systemami nie pozwalającymi na dynamiczne dołączanie bibliotek, Linux umożliwia także ich statyczne dołączanie (ang. *statically linked libraries*).

Linux obsługuje wiele różnych systemów plików, w tym kompatybilne z systemem DOS i OS/2. Własny system Linuxa, zwany `ext2fs`, został zaprojektowany z myślą o optymalnym wykorzystaniu dysku twardego.

Linux to doskonały wybór, jeśli chcesz pisać nowe aplikacje czy eksperymentować z nowymi językami programowania. W skład dystrybucji Linuxa wchodzą m.in. kompilatory języków C, C++, Fortran, Pascal, LISP, Ada, BASIC oraz Smalltalk. Wiele z kompilatorów i innych narzędzi programistycznych pochodzi z projektu Free Software Foundation GNU.

Oprogramowanie GNU

Projekt GNU (rekurencyjny akronim: *Gnu to Nie UNIX – Gnu's Not UNIX*) został opracowany przez Free Software Foundation (FSF, Fundacja na rzecz darmowego oprogramowania) po to, aby zapewnić programistom dostęp do darmowego oprogramowania. Od momentu rozpoczęcia jego działalności powstało bardzo dużo pakietów programistycznych i narzędzi przeznaczonych do dystrybucji przez FSF. Większość oprogramowania GNU to odpowiedniki programów dostępnych komercyjnie (i bardzo drogich), często w stosunku do nich ulepszone.

Linux również zawiera wiele programów użytkowych GNU, takich jak wspomniane wcześniej języki programowania, debuggery, programy do przetwarzania tekstów, do drukowania i inne.

X

System X (czasem zwany także X Window) to graficzny interfejs użytkownika (GUI), zaprojektowany dla zapewnienia jednolitej obsługi i wyglądu aplikacji na różnych platformach. Wersja X rozprowadzana z Linuxem nazywa się XFree86 i jest implementacją

standardowego systemu X11R5 dla komputerów opartych na procesorach 80386. XFree86 zawiera również pewne elementy zapewniające kompatybilność z niektórymi innymi GUI, np. z Open Look.

XFree86 obsługuje wiele różnych kart graficznych pracujących w różnych konfiguracjach, oferując wygodny interfejs graficzny o wysokiej rozdzielcości. Dowolna aplikacja pracująca z X może być przekompilowana tak, by działała poprawnie pod kontrolą Linuxa. W skład systemu X wchodzi pewna ilość gier i programów użytkowych.

System XFree86 zawiera również biblioteki, narzędzia i programy użytkowe potrzebne przy tworzeniu własnych aplikacji. Pozwala to programistom na pisanie programów przeznaczonych do pracy w X bez konieczności inwestowania w drogie, specjalistyczne oprogramowanie czy biblioteki.

Współpraca z systemami DOS i Windows

Ponieważ Linux został opracowany dla komputerów PC, pewien stopień kompatybilności z systemem MS-DOS firmy Microsoft zapewnia sam system operacyjny. Częścią dystrybucji Linuxa jest emulator systemu MS-DOS, który pozwala na uruchomienie wielu aplikacji DOS-owych bezpośrednio pod kontrolą Linuxa. Nie należy jednak spodziewać się pełnej kompatybilności – niektóre programy używają bezpośredniej komunikacji z urządzeniami zewnętrznymi czy dyskami twardymi, co dla Linuxa jest nie do przyjęcia. Opracowany został też emulator Microsoft Windows (projekt WINE), który pozwala na uruchamianie w systemie Linux aplikacji przeznaczonych do pracy w systemie MS Windows. Bardziej zaawansowana wersja tego emulatorka, zwana WABI, dostępna jest za dodatkową opłatą. WABI pozwala na uruchamianie aplikacji MS Windows za pomocą interfejsu X.

Linux pozwala na bezproblemową wymianę plików z partycjami systemu DOS, używając ich bezpośrednio, jeżeli tak jest skonfigurowany. Oznacza to łatwą wymianę danych pomiędzy tymi systemami operacyjnymi.

Protokół TCP/IP

TCP/IP (Transmission Control Protocol / Internet Protocol) jest podstawowym systemem sieciowym używanym przez systemy UNIX i Linux. Jest to cała rodzina protokołów zaprojektowanych dla Internetu, musisz więc ich używać, jeśli chcesz korzystać z dobrodzieszczeń tej sieci. Jeżeli zamierzasz połączyć się z jakimiś maszynami UNIX-owymi, najprawdopodobniej również będziesz musiał użyć TCP/IP.

Wersja TCP/IP dla systemu Linux zawiera oprogramowanie i sterowniki takie same, jak wersje rozprowadzane komercyjnie, przeznaczone dla systemów UNIX. Dzięki temu możesz stworzyć własną sieć lokalną (Local Area Network, LAN), podłączyć się do istniejącej sieci Ethernet, bądź też podłączyć się do Internetu.

Praca w sieci jest mocną stroną Linuxa i będzie omówiona dość szczegółowo w dalszej części tej książki. W systemie linuxowym nie jest konieczne posiadanie dostępu do sieci,

ale jej instalacja jest prosta i tania, a jest to najlepszy sposób przesyłania danych pomiędzy komputerami. Możliwe jest również używanie sieci przez modem, więc uzyskanie dostępu do komputerów przyjaciół nie sprawia problemu.

Historia Linuxa

Linux został opracowany jako darmowa wersja UNIX-a. UNIX jest najszerzej używanym systemem operacyjnym na świecie i od dawna jest standardem dla wysoko wydajnych stacji roboczych i większych serwerów. Wielu użytkowników systemu UNIX, opracowanego w 1965 roku, to programiści, którzy stanowią potężne wsparcie techniczne dla tego systemu.

System UNIX należy zakupić osobno dla każdej platformy, na której będzie używany, ponieważ jest on produktem komercyjnym. Licencje na implementacje UNIX-a dla komputerów PC kosztują od kilkuset do kilku tysięcy dolarów. Aby udostępnić go szerszej rzeszy użytkowników, w ciągu ostatnich lat powstało wiele ogólnodostępnych, darmowych wersji (public domain) tego systemu.

Jednym z pierwszych klonów UNIX-a był Minix, napisany przez Andy'ego Tanenbau ma. Choć nie miał wszystkich zalet UNIX-a, był małym systemem operacyjnym, który mógł być używany na maszynach klasy PC. Aby rozszerzyć jego możliwości, wielu użytkowników zaczęło pracować nad ulepszoną wersją, wykorzystującą możliwości, jakie oferowała architektura procesora 80386. Jednym z pierwszych projektantów nowego systemu, znanego później jako Linux, był Linus Torvalds z Uniwersytetu w Helsinkach, który opracował wczesną wersję Linuxa w roku 1991. Pierwsza „prawie bezbłędna” wersja została udostępniona programistom na całym świecie w marcu 1992 roku.

Wkrótce nad Linuxem pracowało już wielu programistów. Pomyśl udoskonalenia nowego klonu UNIX-a okazał się „strzałem w dziesiątkę” i Linux zaczął rozrastać się w bardzo szybkim tempie, szczególnie od czasu, gdy programiści zaczęli adaptować do jego potrzeb programy i rozwiązania znane z komercyjnych wersji UNIX-a. Nowe wersje Linuxa lub programów użytkowych dla tego systemu pojawiają się teraz bardzo często. Zdarza się nawet, że nowa wersja pojawia się w tydzień po poprzedniej. Liczba programistów pracujących nad Linuxem rosła, a sam system stawał się coraz lepszy; dziś stoi on na bardzo wysokim poziomie i zawiera wszystkie narzędzia, które można znaleźć w komercyjnych wersjach UNIX-a.

Aby uniknąć jakichkolwiek opłat za Linuxa, programiści nie używali kodu zaczerpniętego z innych systemów UNIX-owych. Za używanie systemu Linux nie trzeba uiszczać żadnych opłat licencyjnych; podstawowym założeniem jego twórców jest jego powszechna dostępność. Kilka firm podjęło się zadania kompletowania i wydawania kolejnych wersji Linuxa, umieszczając je następnie na płytach CD-ROM i pobierając za nie symboliczne (zazwyczaj) opłaty.

Co Linux może zrobić dla Ciebie

Dlaczego powinieneś zwracać sobie głowę Linuxem? To dobre pytanie, ale trudno na nie odpowiedzieć. Jeżeli jesteś ciekawy, jak działa UNIX lub inny system operacyjny, Linux daje Ci wspaniałe środowisko do eksperymentowania, za bardzo niską cenę lub wręcz za darmo. Jeżeli oczekujesz od swojego sprzętu większej wydajności niż może dać Ci DOS lub Windows, Linux jest systemem dla Ciebie. Jeżeli potrzebujesz solidnego serwera, który połączy Twoją firmę z Internetem, Linux jest najtańszym i doskonałym rozwiązaniem. Ale jeśli nie zamierzasz uczyć się niczego nowego, nie chcesz wykorzystywać wszystkich możliwości swojego sprzętu i lubisz wesołe komunikaty systemu Windows na niebieskim tle, to Linux na pewno nie jest dla Ciebie.

Naucz się UNIX-a dzięki Linuxowi

Wyjaśnijmy sobie na samym początku jedną sprawę. Choć Linux nie nazywa się UNIX, to jest on pełną implementacją tego systemu. Spełnia wiele z norm spełnianych przez systemy UNIX, wiele z nich nawet w szerszym zakresie, i bardzo trudno byłoby nawet doświadczonemu użytkownikowi UNIX-a stwierdzić (bez użycia specjalistycznych narzędzi do identyfikacji systemu), czy pracuje pod kontrolą systemu Linux czy UNIX. Dlaczego Linux nie nazywa się UNIX? To tylko kwestia praw autorskich.

Ponieważ Linux od UNIX-a różni się tylko nazwą, jest on doskonałym sposobem na nauczenie się UNIX-a. Linux działa dobrze na większości pęcetów używanych w domach, czego na pewno nie można powiedzieć o komercyjnych wersjach UNIX-a; większość z nich wymaga szybkiego komputera z procesorem Pentium, a jego koszt to dopiero ułamek kosztów instalacji pełnego systemu. A przecież wszystko, czego nauczysz się w Linuxie, można bezpośrednio przenieść na platformę UNIX-ową. Aplikacje, które napiszesz dla Linuxa, zwykle również mogą być przekompilowane w systemie UNIX i będą działać znakomicie.

Uwzględniając to wszystko, można śmiało stwierdzić, że Linux jest naturalną drogą poznawania UNIX-a. UNIX jest jednym z najpotężniejszych z dostępnych systemów operacyjnych i jest powszechnie używany przez duże korporacje i firmy, które koncentrują się na badaniach naukowych i pracach projektowych. Powód jest prosty: UNIX pozwala projektantowi na bezproblemowe uzyskanie największej możliwej wydajności sprzętu. Jest on również najlepszym sposobem na połączenie w sieć wielu komputerów. Możliwe, że umiejętność pracy z systemem UNIX nie jest Ci potrzebna w tej chwili, ale kto wie, co zdarzy się w przyszłości? Nie wspominając już o tym, że znajomość UNIX-a świetnie prezentuje się w podaniu o pracę...

Powierz swoją firmę Linuxowi

Wykorzystując system Linux, można śmiało prowadzić firmę. UNIX jest najczęściej stosowanym w biznesie systemem operacyjnym, a Linux może robić wszystko to, co może robić UNIX. Połączenie serwerów linuxowych z innymi rodzajami klientów

sieciowych (takimi jak Windows, DOS, Macintosh czy inne rodzaje UNIX-a) tak, by mieć pod kontrolą całą firmę, nie nastręcza żadnych trudności.

W początkowym okresie rozwoju Linuxa rozwiązania takie uważane były za ryzykowne, ponieważ sam system operacyjny był jeszcze niezbyt stabilny. Firmy nie mogą pozwolić sobie na używanie serwera, który często odmawia współpracy. Od tamtej pory jednak Linux dojrzał, stał się stabilniejszy i przejął wszystkie zalety UNIX-a. Brakuje w nim tylko najbardziej profesjonalnych systemów zabezpieczeń, oferowanych przez komercyjne wersje UNIX-a. Dla systemu linuxowego ciągłe działanie przez lata bez żadnych wpadek i bez potrzeby restartowania systemu nie jest niczym nadzwyczajnym (choć raz na jakiś czas zaleca się zrestartować system, żeby usunąć tymczasowe katalogi i pliki, w których zapisywane są bieżące akcje systemu). Nie można tego powiedzieć o wielu innych systemach operacyjnych (jak często „wykładał” się Twój Windows w ciągu ostatnich kilku miesięcy?).

Tak naprawdę, różnica pomiędzy używaniem Linuxa i komercyjnego UNIX-a sprowadza się do dwóch rzeczy: Linux jest tańszy, za to UNIX posiada lepsze wsparcie techniczne. Oczywiście jeżeli pieniądze nie są problemem dla Twojej firmy (przynajmniej kwoty rzędu tysiąca dolarów, czyli koszt kompletnej platformy UNIX-owej), pierwsza z tych różnic przestaje mieć znaczenie. Wsparcie techniczne jest szczególnie ważne dla firm, więc często zasadne jest zapłacenie za komercyjną wersję UNIX-a tylko z tego powodu. Jednak prawdą jest również, że zazwyczaj poprzez Internet możesz otrzymać lepszą niż tę oferowaną przez dystrybutorów UNIX-a pomoc techniczną dla systemu Linux.

Serwery internetowe

Linux nadaje się też doskonale do obsługi serwera WWW, FTP czy poczty dla Twojego domu, biura i organizacji. Zawdzięcza to przede wszystkim swojemu podobieństwu do systemu UNIX. Wszystkie te usługi pierwotnie dostępne były tylko dla UNIX-a, więc ten system operacyjny jest idealnie dopasowany do tego typu zadań. Linux może być łatwo skonfigurowany jako serwer internetowy, o czym przekonasz się, czytając rozdział 47. „Zakładanie węzła internetowego”.

Czy Linux jako serwer Internetu jest lepszy niż systemy Windows NT, UNIX czy Macintosh? Lepszy to może niewłaściwe słowo – na pewno nie jest pod żadnym względem gorszy, a kosztuje mniej. Poza tym, Linux może pochwalić się ogromną ilością darmowego oprogramowania, które pomoże w udostępnianiu usług internetowych.

Co jest potrzebne do instalacji Linuxa

Linux jest systemem bardzo atrakcyjnym dla wielu użytkowników ze względu na to, że oferuje on środowisko stacji roboczej UNIX przy wykorzystaniu maszyn klasy PC. Jego wymagania sprzętowe nie są wygórowane, chyba że zamierzasz zajmować się profesjonalnym programowaniem lub intensywnie wykorzystywać GUI. Jeżeli masz gdzieś stare 386, możesz na nim zainstalować Linuxa – będzie działał całkiem przyzwoicie.

Zmieniasz właśnie sprzęt z 486 czy starszego Pentium i nie wiesz co zrobić ze starym komputerem? To doskonała platforma dla Linuxa.

Ten rozdział będzie traktował o podstawowym sprzęcie potrzebnym do instalacji Linuxa. Omówią minimalne wymagania sprzętowe i obsługę większości urządzeń zewnętrznych. Rozszerzanie systemu o nowy sprzęt omówione zostanie później, w rozdziale o administrowaniu systemem.

Minimalne wymagania sprzętowe

Sprzęt obsługiwany standardowo przez jądro systemu jest typowy dla komputera PC, gdyż właśnie dla użytkowników komputerów PC Linux został zaprojektowany. Niektóre rzadziej spotykane urządzenia są obsługiwane, o ile jakiś programista poświęcił swój czas i napisał dla nich sterownik, po czym udostępnił go użytkownikom Linuxa. Kiepsko jest również z obsługą różnych akcesoriów typu karty multiport itp., choć sytuacja poprawia się w miarę, jak Linux staje się popularniejszy.

Minimalne wymagania sprzętowe to procesor 80386 lub lepszy, 2 MB pamięci RAM, stacja dyskietek, dysk twardy 40 MB, karta grafiki i monitor – prawie każdy system jest lepiej wyposażony. Dla osiągnięcia zadowalającej wydajności przy pracy w trybie tekstowym, potrzeba 80386 z 8 MB RAM. Jeśli chcesz używać X lub Motif, szybki 80486 lub Pentium z 16 MB pamięci RAM będzie dobrym rozwiązaniem.

Przyjrzymy się teraz bardziej szczegółowo poszczególnym składnikom systemu.

Płyta główna i procesor

Sprzęt wymagany do zainstalowania Linuxa to typowa konfiguracja komputera PC. Procesor powinien być typu 80386 lub nowszy (lub też jeden z klonów Intel'a, jak AMD). W zasadzie Linux będzie działał nawet na wolnym procesorze 386, ale jego wydajność będzie raczej nieakceptowalna.

Do tworzenia własnych aplikacji zalecany jest model 80486DX lub nowszy, ze względu na intensywne wykorzystanie procesora przez kompilator i linker. To samo dotyczy użytkowników systemu X, którego obsługa zajmuje dużo czasu procesora. Możesz kompilować programy na 80386, podobnie jak możesz używać X, ale wydajność systemu spada znacznie poniżej granicy cierpliwości użytkownika.

Linux używa koprocesora arytmetycznego, jeżeli jest on dostępny w systemie (koprocesor jest wbudowany w procesory DX i Pentium). W przeciwnym przypadku Linux używa emulacji programowej o zupełnie rozsądnej wydajności. Obsługiwane są zarówno koprocesory firmy Intel, jak i produkowane przez inne firmy, choć zgłaszały były pewne problemy z układami firmy Weitek.

Linux obsługuje płyty główne w standardzie ISA (Industry Standard Architecture), EISA (Extended ISA) oraz PCI (Peripheral Component Interconnect), nie obsługuje natomiast standardu MCA (MicroChannel Architecture firmy Intel). Obsługiwane są również płyty standardu VESA Local Bus, pozwalające urządzeniom na bezpośredni dostęp do elementów płyty głównej, choć są one obecnie bardzo rzadko spotykane.

Wymagania co do ilości pamięci RAM zależą od planowanej wielkości systemu. Wersja minimalna potrzebuje 2 MB pamięci RAM, choć bardzo intensywnie wykorzystywany jest wtedy plik wymiany (ang. *swapping*). 4 MB RAM-u może już być uważane za rozsądne minimum zapewniające efektywną pracę, a większa ilość pamięci owocuje lepszą wydajnością. Dla programistów i użytkowników X 8 MB jest ilością minimalną, niezbędną do pracy (choć X działa już przy 4 MB, korzystając z pliku wymiany). 16 MB pamięci pozwala w znaczącym stopniu zredukować wymianę danych pomiędzy pamięcią i dyskiem.

Systemy linuxowe, które będą używane jednocześnie przez więcej niż jednego użytkownika, powinny dysponować większą ilością RAM-u. Każdy dodatkowy użytkownik to większe wymagania pamięciowe. Na przykład 8 MB RAM-u wystarczy dla dwóch użytkowników, nawet jeśli używają X. Dla ośmiu użytkowników wystarczy 16 MB RAM-u, choć nie będą oni mogli korzystać z X. W systemie tym regułą, której należy się trzymać, jest posiadanie 4 MB RAM-u dla każdego użytkownika, chyba że serwer linuxowy może przenieść obsługę X na terminal za pomocą mechanizmów klient-serwer.

Linux używa całej dostępnej pamięci RAM, nie narzucając żadnych ograniczeń wynikających z jego architektury (jak robi to DOS i niektóre inne systemy operacyjne).

Zalecane jest stworzenie tzw. partycji wymiany (ang. *swap partition*) w celu powiększenia ilości dostępnej w systemie pamięci. Jest to jakby wolniejsze przedłużenie pamięci fizycznej, a dane przechowywane na niej mogą być swobodnie wymieniane z danymi przechowywanymi w pamięci RAM. Partycja taka powinna zostać utworzona nawet w systemach posiadających dużą ilość RAM-u. Jej rozmiar zależy od ilości dostępnej fizycznej pamięci RAM, liczby użytkowników oraz typowego obciążenia systemu.

Dyski twardye

Linux może być uruchamiany z dyskietki, bez konieczności posiadania dysku twardego, ale jego funkcjonalność jest wtedy dość ograniczona. System ten został zaprojektowany z myślą o komputerach wyposażonych w dysk twardy i obsługuje wszystkie popularne typy kontrolerów, takie jak IDE (Integrated Device Electronic), EIDE (Extended IDE), ESDI (Enhanced Small Device Interface), RLL (Run Length Limited) oraz SCSI (Small Computer System Interface). Obsługuje też starsze, ośmiorozbitowe kontrolery, choć większość spotykanych dziś kontrolerów jest szesnastorozbitowa.

Linux nie jest wybredny, jeśli chodzi o producenta dysku twardego. Jeżeli DOS radzi sobie z danym typem dysku, to poradzi sobie i Linux. Wyjątkiem są dyski SCSI, które wymagają innego sposobu obsługi. Linux jest jednak ograniczony przez starsze wersje BIOS-u, które limitują maksymalną liczbę obsługiwanych sektorów, głowic i cylindrów. Ostateczną granicą rozmiaru dysku w takich starszych systemach jest 1024 kB, ale niektóre

mniejsze dyski również nie mogą być prawidłowo obsługiwane. Istnieją sterowniki programowe pozwalające obejść ten problem, lecz są one dość trudno dostępne. Jeżeli nie uda Ci się znaleźć takiego sterownika, a Twoja wersja BIOS-u nie obsługuje dużych dysków, to nowy dysk twardy o pojemności 8 GB zostanie rozpoznany przez BIOS jako dysk 1 GB, czyli zmarnuje się 7 GB wolnego miejsca. Czasem dobrym rozwiązaniem jest wymiana samego BIOS-u, ale dla starszych płyt głównych rzadko dostępne są jego nowe wersje.

Linux obsługuje większość urządzeń pracujących w standardzie SCSI. Na rynku dostępne są jednak kontrolery SCSI nie działające w systemie Linux. Maksymalny rozmiar dysku SCSI wciąż jest ograniczony przez starsze wersje BIOS-u do 1 GB. Inne wersje UNIX-a, np. SCO UNIX, w większości przypadków pozwalają ominąć te ograniczenia.

Wielkość dysku twardego wymagana przez Linuxa zależy od tego, które z pakietów oprogramowania chcesz zainstalować. Minimalna konfiguracja wymaga ok. 20 MB; jest to ilość wystarczająca dla podstawowych programów użytkowych, jeśli nie jest zainstalowany X Window. Jeśli chcesz zainstalować cały podstawowy system, włączając w to narzędzia programistyczne i X Window, same pliki zajmą ok. 250 MB miejsca na dysku twardym. Dodatkowo potrzebne jest miejsce na pliki użytkowników oraz na plik wymiany. Dobrą regułą jest zapewnienie dwa razy większej ilości miejsca niż wymaga sama instalacja.

Nie zapominaj o pozostawieniu na dysku wolnego miejsca na partycję (lub plik) wymiany. Choć jej minimalny rozmiar zależy od przeznaczenia i obciążenia systemu, prawie wszyskie dobrym rozwiązaniem jest od 40 do 64 MB.

Możliwe jest używanie więcej niż jednego dysku twardego, ale partycja linuxowa powinna być umieszczona na pierwszym z nich. Jeśli używasz systemu DOS lub Windows, one także muszą znajdować się na pierwszym dysku. Liczba dysków możliwych do zainstalowania w systemie zależy od BIOS-u i kontrolera. Kontrolery IDE zazwyczaj obsługują dwa dyski, EIDE do czterech, ESDI i RLL również po dwa. Kontrolery SCSI obsługują do siedmiu dysków twardych, a nowsze wersje nawet do piętnastu. SCSI to najelastyczniejsze, choć zarazem najdroższe rozwiązanie.

Dyski twarde sukcesywnie tanieją i często spotyka się już nośniki o dużej pojemności. Linux może dzielić taki dysk z trzema innymi systemami operacyjnymi (a nawet więcej, po zastosowaniu pewnych chwytów), więc jeśli chcesz używać DOS-u, Windows i Linuxa, po prostu zapewnij każdemu z systemów odpowiednią ilość miejsca.

Karta graficzna i monitor

Linux współpracuje z prawie wszystkimi typami kart graficznych nie wymagającymi specjalnych sterowników w systemie DOS. Obsługuje karty Hercules, CGA, EGA, VGA i SuperVGA. Niektóre karty o większych możliwościach również są w pełni wykorzystywane, np. Cirrus Logic, Diamond, ATI oraz Trident. Ponieważ na rynku oferowane są setki różnych typów kart graficznych, nie są dostępne sterowniki dla wszystkich trybów specjalnych tych kart, ale jeżeli karta obsługuje standardowe tryby VGA lub SVGA, może być używana z Linuxem.

System X nakłada własne wymagania na kartę graficzną. Choć może on pracować w trybach VGA lub SVGA, zalecane są karty obsługujące wyższe rozdzielczości. Dlatego przed zakupem nowego sprzętu warto upewnić się, czy dostępny jest odpowiedni sterownik lub czy karta jest kompatybilna z innym modelem, dla którego sterownik został opracowany.

Mysz

Linux nie wymaga myszy do pracy w trybie tekstowym, ale jest ona niezbędna do obsługi systemu X. Linux radzi sobie praktycznie z każdym typem myszy i trackballa, który jest obsługiwany przez DOS i Windows, włączając w to produkty firm Microsoft, Logitech, Mouse Systems i inne. Daje sobie radę zarówno z myszami szeregowymi, jak i magistralowymi.

Niektóre inne urządzenia wskazujące, takie jak piórka i joysticki, również są obsługiwane prawidłowo.

Napędy taśmowe

W Linuxie działać będą napędy taśmowe używające kontrolerów SCSI rozpoznawanych przez system. Inne napędy, używające dedykowanych kontrolerów, prawdopodobnie będą również działać prawidłowo, pod warunkiem, że odpowiednio skonfigurujesz zasoby sprzętowe używane przez te urządzenia (numer przerwania IRQ, kanału DMA i zakres pamięci).

Coraz popularniejsze stają się urządzenia wykorzystujące kontroler stacji dysków lub port równoległy – dla niektórych z nich dostępne są odpowiednie sterowniki. Większość z tych napędów dla zwiększenia objętości danych możliwych do zarchiwizowania na nośniku stosuje ich kompresję przed zapisaniem, nie jest więc niestety wykluczone, że maksymalną objętością, jaką uda Ci się uzyskać w systemie Linux, będzie fizyczna pojemność takiego nośnika.

Napędy CD-ROM

Ponieważ większość napędów CD-ROM korzysta z interfejsu EIDE lub SCSI, do ich obsługi niezbędny jest odpowiedni kontroler. Starsze modele były sterowane przez najróżniejsze karty, np. przez karty dźwiękowe. Dla tego typu urządzeń wymagany jest odpowiedni sterownik. Napędy CD-ROM używające interfejsu SCSI działają prawidłowo, o ile system potrafi rozpoznać kontroler SCSI.

Nie wszystkie formaty dysków CD-ROM są obsługiwane przez Linuxa. Na razie obsługuje on tylko płyty zapisane w standardzie ISO-9660. Choć jest to najpowszechniej używany standard, nie jest on jedyny i może zdarzyć się, że nie uda się zamontować systemu plików z dysku CD-ROM, jeśli był on zapisany dla systemu DOS lub Macintosh.

Dyski wyjmowalne

Działanie dysków wyjmowalnych w systemie Linux uzależnione jest od typu interfejsu używanego przez taki dysk. Obsługiwana jest większość urządzeń opartych na interfejsie SCSI (takich jak Iomega Jaz czy SyQuest), choć wymiana nośnika w czasie pracy systemu rzadko jest poprawnie rozpoznawana. Urządzenia Iomega Bernoulli i magneto-optyczne dyski LaserSafe Pro pracują bez żadnych dodatkowych sterowników, o ile da się sformatować nośnik. Obsługiwane są też niektóre inne wymienne dyski magnetyczne i magnetoptyczne.

Niektóre z urządzeń tego typu, szczególnie te nie bazujące na interfejsie SCSI, ale obsługiwane przez dedykowane karty sprzętowe, wymagają specjalnych sterowników. Tylko niewielka część takich urządzeń jest obsługiwana przez Linuxa, a i to jedynie dzięki programistom, którzy napisali odpowiedni sterownik dla siebie i udostępnili go dla innych użytkowników.

Drukarki

Praktycznie wszystkie drukarki korzystające z portu równoległego lub szeregowego są obsługiwane w szerokim zakresie jako urządzenia „nieinteligentne”. Dostępne są sterowniki dla niektórych popularniejszych drukarek, jak DeskJet i LaserJet firmy Hewlett-Packard, ale wiele drukarek nie doczekało się jeszcze sterowników przeznaczonych specjalnie dla nich. Jeżeli dla drukarki nie są dostępne żadne sterowniki, będzie ona pracować tylko w trybie tekstowym (ASCII).

Często można samodzielnie opracować interfejs do nieobsługiwanej drukarki, bądź to używając tablicy tłumaczącej (ang. *translation table*), bądź też pisząc własny sterownik.

Modemy

Linux radzi sobie z większością modemów asynchronicznych, a także z niektórymi urządzeniami synchronicznymi. Dostępna jest też obsługa modemów ISDN. Ogólnie rzecz biorąc, jeżeli modem jest obsługiwany przez systemy DOS i Windows, obsługuje go też Linux.

Linux potrafi wykorzystać wszystkie prędkości transmisji danych, nie wyłączając 56K (co prawda niezbędne są wówczas dodatkowe sterowniki). Do systemu można podłączyć więcej niż jeden modem – ich liczba ograniczona jest jedynie przez ilość portów szeregowych.

Terminale

Terminale tekstowe mogą być podłączane przez port szeregowy lub kartę multiport. Obsługiwana jest większość typów tych urządzeń, a każdy terminal, dla którego znane są kody kontrolne, może być łatwo zaadaptowany do pracy z Linuxem. Terminale graficzne

(w sensie UNIX-owym) używają prostej grafiki opartej na znakach ASCII i nie mogą obsługiwać systemu X.

Terminale X są również obsługiwane, choć nie wszystkie działają prawidłowo. Zazwyczaj aby dobrze wyświetlać grafikę, wymagają one szybkiego połączenia z serwerem (przez port szeregowy lub kartę sieciową). Komputer PC używający oprogramowania klienta X może także funkcjonować jako terminal X.

Karty multiport

Niektóre karty przeznaczone do pracy z UNIX-em będą działać w systemie Linux, ponieważ zostały dla nich opracowane sterowniki (przez dystrybutorów sprzętu lub użytkowników), ale przed zakupem takiej karty lepiej upewnić się, czy są one dostępne. Karty rozszerzające możliwości portów równoległych wymagają dodatkowych sterowników.

Niektóre karty multiport zamiast dedykowanego kontrolera używają interfejsu SCSI, jednak nawet dla takich rozwiązań potrzebne są odpowiednie sterowniki. Karty multiport przeznaczone do obsługi sieci często pracują bez dodatkowych sterowników, ponieważ każdy z portów jest wtedy widziany jako osobne urządzenie sieciowe.

Karty sieciowe

Podstawowym protokołem sieciowym używanym w systemie Linux jest TCP/IP. Inne protokoły są również dostępne, ale najczęściej używa się właśnie TCP/IP, ponieważ wchodzi on w skład standardowej dystrybucji i jest domyślnym protokołem sieciowym. Jego dodatkowym atutem jest możliwość udostępniania zasobów Internetu. TCP/IP to także najpopularniejszy protokół w sieciach Ethernet.

Na rynku dostępnych jest wiele kart sieciowych (Network Interface Cards, NIC) przeznaczonych do pracy w sieciach Ethernet. Najczęściej spotykane karty produkcji takich firm jak 3Com, Novell, Western Digital czy Hewlett-Packard, pracują w systemie Linux bezproblemowo, podobnie jak wiele kart Ethernet pochodzących od innych producentów.

Prawa autorskie

To, że Linux jest rozpowszechniany za darmo, nie oznacza, że nie jest objęty prawami autorskimi. Prawa te zostały zebrane w tzw. licencji GNU GPL (GNU General Public License). Pozwala ona na używanie i dystrybucję oprogramowania wraz z jego kodem źródłowym każdemu, kto ma na to ochotę.

Nie ma gwarancji na żadną z wersji Linuxa. Nawet jeśli kupisz Linuxa od kogoś i zapłacisz mu za pomoc techniczną, nie dotyczy to w żadnym stopniu autora oprogramowania. Programiści nie biorą odpowiedzialności za działanie napisanych przez nich programów.

Jeżeli Linux zniszczy Twoje najcenniejsze dane, masz pecha. Wziąłeś ryzyko na siebie. Mimo tej ponurej wizji, Linux dowiodł, że jest systemem bardzo stabilnym i nie zdarzają się przypadki zniszczenia danych, nie zawiadione przez błąd użytkownika. Z drugiej strony, jeżeli zamierzasz powierzyć Linuxowi jakieś naprawdę wartościowe dane, to powinieneś rozważyć zakup komercyjnej wersji UNIX-a, zawierającej gwarancję.

Zgodnie z licencją GNU GPL, dozwolone jest nawet sprzedawanie Linuxa, o ile znajdziesz się ktoś, kto będzie chciał za niego zapłacić. Można modyfikować kod programów i zmieniać zawartość pakietów, ale nie nabywa się przez to praw autorskich do oprogramowania. Poza tym, jednym z warunków GNU GPL jest obowiązek udostępniania kodu źródłowego programów, które mają być sprzedawane dla zysku, ta, aby inni programiści mogli je również modyfikować i sprzedawać.

Twórcy Linuxa nie pobierają żadnych opłat. W większości przypadków udostępnili oni ten system z zamiłowaniem do programowania i po to, aby podzielić się efektami swojej pracy z innymi programistami, potrafiącymi je docenić.

Uzyskiwanie pomocy

Linux nie posiada linii wsparcia telefonicznego. W pewnym sensie przy instalowaniu go jesteś skazany tylko na siebie. Z drugiej jednak strony, tysiące innych użytkowników Linuxa na całym świecie gotowych jest podać Ci pomocną dłoń, obojętne, czy jesteś zaawansowanym programistą, czy też jest to Twój „pierwszy raz”. Musisz tylko wiedzieć, gdzie trzeba szukać takiej pomocy. Dwa podstawowe źródła wsparcia w kłopotach to pliki dokumentacji i pomoc innych użytkowników Linuxa.

Zazwyczaj pierwszą dokumentacją systemu Linux, z jaką spotykają się użytkownicy, jest plik nazywany Linux INFO-SHEET. Jest to stosunkowo krótki dokument ASCII dostępny w sieci Usenet, w węzłach BBS i w grupach dyskusyjnych poświęconych Linuxowi. Plik INFO-SHEET zawiera skrócony opis całego systemu i jest regularnie aktualniany i posyłany do grup dyskusyjnych sieci Usenet.

Aby zapewnić powszechny dostęp do bardziej szczegółowej dokumentacji, powstał Projekt na rzecz dokumentacji Linuxa (Linux Documentation Project). Dokumentacja ewoluowała od paru notatek na temat instalacji sprzed kilku lat, do około tysiąca stron materiałów źródłowych, czasem lepszych, czasem nieco gorszych. Oto zestaw dokumentów, w które warto się zaopatrzyć:

- υ Linux Installation (Instalacja Linuxa) – o tym jak zainstalować i skonfigurować Linuxa,
- υ Linux User's Guide (Podręcznik użytkownika Linuxa) – przewodnik dla początkujących,
- υ Linux System Administrator's Guide (Podręcznik administratora Linuxa) – przewodnik po zagadnieniach związanych z administrowaniem systemem,

- υ Linux Network Administration Guide (Zarządzanie siecią w systemie Linux) – o instalacji i użytkowaniu sieci,
- υ Linux Kernel Hacker's Guide (Wprowadzanie poprawek do jądra systemu Linux) – przydatny, gdy zamierzasz sam modyfikować jądro systemu.

Oprócz tych dokumentów, istnieje kilkanaście mniejszych, traktujących o bardziej specyficznych aspektach Linuxa, nazywanych „How-to”. Wszystkie razem tworzą całkiem potężną dokumentację, obejmującą praktycznie wszystkie zagadnienia dotyczące Linuxa. Jednak niektóre z nich są nieaktualne, ze względu na to, że system znacznie się rozwinął od czasu, gdy zostały napisane.

Dokumentacja Linuxa została stworzona przez wiele osób, więc siłą rzeczy jej styl i wygląd nie są jednolite. Większość z tych dokumentów jest dostępna razem z dystrybucjami Linuxa. Można również kupić ich drukowaną wersję, wydaną przez Linux Systems Labs.

Za pośrednictwem grup dyskusyjnych oraz jako część dystrybucji Linuxa rozprowadzanych jest wiele dokumentów FAQ (Frequently Asked Questions, czyli odpowiedzi na często zadawane pytania). FAQ pomyślano są jako forma szybkiej pomocy, która ma zaoszczędzić Ci konieczności przedzierania się przez wiele stron dokumentacji. Jeden z tych dokumentów, nazywany META-FAQ, zawiera podstawowe informacje o Linuxie, o tym, jak można go zdobyć, oraz o pozostałych dokumentach. On również dostępny jest w grupach dyskusyjnych.

Istnieje też dokument zwany Linux Software Map (Przewodnik po oprogramowaniu dla Linuxa), który zawiera listę różnych aplikacji przeznaczonych dla systemu Linux. Niestety, lista ta jest niekompletna i brakuje w niej sporej ilości danych. Jeśli jednak jesteś ciekawy, jakie oprogramowanie dla Linuxa już napisano, możesz go przejrzeć. Plik ten jest co jakiś czas uaktualniany i można go znaleźć w sieci Usenet, w węzłach FTP i w wielu dystrybucjach Linuxa.

Informacje o systemie Linux osiągalne są też w postaci list e-mailowych, dostępnych dla każdego w Internecie. Więcej o tych listach dowiedzieć się możesz w sieci Usenet i w węzłach BBS.

Grupy dyskusyjne w sieci Usenet

Grupy dyskusyjne w Internecie są prawdziwą kopalinią informacji o Linuxie. Istnieje ich kilka; aby je obejrzeć powinieneś użyć swojej przeglądarki internetowej, szukając grup ze słowem „Linux” w nazwie. Pojawiają się węzły BBS poświęcone w całości lub częściowo Linuxowi, spotkać też można wyjątki z konwersacjami w sieci Usenet dotyczącymi Linuxa, przeznaczone dla tych, którzy nie mają do niej dostępu.

Usenet to zbiór list (grup) dyskusyjnych (ang. *newsgroups*), dostępnych dla użytkowników Internetu. Istnieje ponad 10000 różnych grup dyskusyjnych, generujących codziennie setki megabajtów wiadomości. Kilka spośród tych grup (obejmujących każdy chyba temat) poświęconych jest Linuxowi.

Dostęp do list dyskusyjnych możesz uzyskać za pomocą odpowiedniego oprogramowania, podłączając się bezpośrednio do Internetu lub też do lokalnego serwera grup dyskusyjnych. Inne sieci, takie jak CompuServe, America Online i Delphi, również umożliwiają korzystanie z grup dyskusyjnych (niekiedy za dodatkową opłatą).

Grupy dyskusyjne podzielone są na trzy kategorie: główne, dostępne dla wszystkich użytkowników, lokalne, zazwyczaj obejmujące tylko dany region geograficzny, oraz inne, które nie mogą być obsługiwane przez wszystkie serwery ze względu na panujące w nich reguły. Głównymi grupami dyskusyjnymi poświęconymi Linuxowi, działającymi w chwili pisania tej książki, są:

- ⦿ comp.os.linux.admin – poświęcona administrowaniu systemami linuxowymi,
- ⦿ comp.os.linux.advocacy – tu można wygłosić swoją opinię na tematy związane z Linuxem,
- ⦿ comp.os.linux.announce – ogłoszenia istotne dla społeczności linuxowej,
- ⦿ comp.os.linux.answers – pytania i rozwiązania problemów dotyczących Linuxa,
- ⦿ comp.os.linux.development – ogólnie o pracach nad Linuxem,
- ⦿ comp.os.linux.development.apps – o pracach nad aplikacjami linuxowymi,
- ⦿ comp.os.linux.development.system – o pracach nad systemem operacyjnym,
- ⦿ comp.os.linux.hardware – skupia się na obsłudze urządzeń zewnętrznych w systemie Linux,
- ⦿ comp.os.linux.help – pytania i porady na temat Linuxa,
- ⦿ comp.os.linux.misc – tematy dotyczące Linuxa nie zawierające się w tematyce innych grup,
- ⦿ comp.os.linux.networking – poświęcona w całości problemom sieci,
- ⦿ comp.os.linux.setup – o problemach związanych z instalacją i konfiguracją Linuxa.

Te grupy dyskusyjne powinny być osiągalne we wszystkich węzłach sieci Usenet, chyba że z jakichś powodów dostępu do nich zabroni administrator systemu. Inne grupy poświęcone Linuxowi często się zmieniają, głównie dlatego, że obejmują zasięgiem niewielki region albo są zdominowane przez osoby o bardzo jednostronnych poglądach; szczególnie grupy .alt (ang. *alternate* – inne) mają tendencję do pojawiania się i znikania. Jedną z takich grup aktywnych podczas pisania tej książki jest alt.uu.comp.os.linux.questions. Jeżeli masz dostęp do sieci Usenet, przeglądaj regularnie informacje o powstawaniu nowych i zakończeniu działalności istniejących grup o tematyce związanej z Linuxem. Tego typu wiadomości zazwyczaj są posyłane do wszystkich grup dyskusyjnych, ale od czasu do czasu bez rozwgłosu powstaje nowa grupa. Serwery pozwalające

na dostęp do sieci Usenet zwykle zapewniają też aktualną listę działających grup dyskusyjnych, którą można łatwo przeglądać.

Większość informacji w grupach dyskusyjnych poświęconych Linuxowi dotyczy problemów i zagadnień związanych z instalacją, konfiguracją, administracją i ogólnie użytkowaniem tego systemu operacyjnego. Przewija się przez nie wiele wartościowych informacji, warto więc przeglądać je regularnie. Najciekawsze z wiadomości dotyczących danego wątku są często zbierane razem i udostępniane w formie archiwów w węzłach FTP.

O czym mówi ta książka?

Celem tej książki jest przeprowadzenie Cię przez instalację i konfigurację Linuxa, pokazanie – z punktu widzenia administracji systemem – jak go używać i dostosować do własnych potrzeb oraz towarzyszenie Ci w drodze od nowicjusza do doświadczonego użytkownika systemów Linux i UNIX. W książce tej poruszamy wiele problemów, niektóre z nich omawiając szczegółowo, o innych zaś tylko wspominając, aby dać Ci pojęcie o tym, co jeszcze potrafi Linux.

Książka ta z pewnością nie zawiera wszystkich informacji o Linuxie. Nie sposób byłoby je zmieścić nawet na 10000 stron. Nie próbujemy na przykład nauczyć Cię programowania w C czy C++, choć możemy pokazać, jak poprawnie skonfigurować środowisko programistyczne i jak używać dołączonych kompilatorów. Na pewno nie podamy Ci rozwiązań wszystkich problemów, na które natkniesz się podczas używania Linuxa, ale damy odpowiednie podstawy i narzędzia, tak abyś mógł sam sobie z nimi radzić. Pokażemy także, gdzie możesz szukać rozwiązań swoich problemów.

Życzymy przyjemnej pracy z Linuxem i niniejszą książką. Dołożyliśmy wszelkich starań, by czytało się ją z przyjemnością i mamy nadzieję, że wiele się z niej nauczysz. Przede wszystkim – baw się dobrze. Linux to wspaniały system operacyjny!

Podsumowanie

Wyjaśniliśmy już podstawowe zagadnienia dotyczące Linuxa, czas przyjrzeć się różnym dystrybucjom tego systemu, jego instalowaniu, i zacząć poznawać system operacyjny. W następnych rozdziałach omówione zostaną szczegóły instalacji i podstawy konfiguracji systemu. Następna część pokaże Ci, jak używać systemu Linux. Jeżeli zainstalowałeś już Linuxa samodzielnie, możesz przejść do rozdziału „Poznawanie Linuxa”. W przeciwnym przypadku, kilka następnych rozdziałów pomoże Ci uporać się z prawidłową instalacją systemu.

Jeżeli chcesz dowiedzieć się, jak zainstalować Linuxa, zajrzyj do rozdziału 3. „Instalacja Linuxa”.

Podstawowe informacje o pracy z tym systemem operacyjnym znajdziesz w rozdziale 6. „Od czego zacząć”.

Instalacja i konfiguracja systemu X Window omówiona jest w rozdziale 22. „Instalacja i konfiguracja XFree86”.

Rozdział 32. „Podstawy administracji systemem”, omawia podstawowe zagadnienia związane z zarządzaniem istniejącym systemem linuxowym.

Rozdział 2.

Rodzaje Linuxa

Tim Parker

W tym rozdziale:

- υ Skąd wziąć Linuxa?
- υ Co to jest dystrybucja?
- υ Dystrybucja Linuxa a zestawy dysków
- υ Aktualizacja istniejącego systemu linuxowego

W tym rozdziale omówimy kilka zagadnień, z którymi powinieneś zapoznać się przed rozpoczęciem instalacji Linuxa. Jeśli nie masz jeszcze dostępu do wersji instalacyjnej systemu, albo chcesz zdobyć wersję inną niż posiadasz, pierwsza część tego rozdziału powie Ci, gdzie i jak możesz zaopatrzyć się w Linuxa. Dalsza część wyjaśnia, które zestawy dysków są potrzebne do zainstalowania systemu i jaka jest funkcja każdego z nich. Informacji tych będziesz potrzebował również przy rozszerzaniu funkcjonalności swojego systemu o nowe aplikacje.

Na rynku dostępnych jest mnóstwo wersji Linuxa: Slackware, Red Hat, Caldera, SLS (Softlanding Linux System), TAMU (Texas A&M University), Yggdrasil i jeszcze kilkanaście innych. Różnią się one mniej i bardziej znaczącymi szczegółami. Trudno z góry rozstrzygnąć, która wersja będzie najlepsza dla Ciebie. Większość różnic to drobiazgi, usprawniające m.in. instalację, dostęp do pomocy technicznej, obsługę sprzętu itp. Ponieważ możliwe jest samodzielne dobieranie i dopasowywanie składników pochodzących z różnych wersji (do pewnego stopnia), większość użytkowników wybiera jedną z wersji i trzyma się jej. Choć zawsze możesz zainstalować inną wersję Linuxa, powinieneś mieć naprawdę ważne powody, aby to robić.

Skąd wziąć Linuxa?

Linux jest dostępny w każdej chwili za darmo, jeśli tylko wiesz, gdzie go szukać. Ponieważ jest on rozprowadzany bez żadnej organizacji czy kontroli (tak jak ma to miejsce w przypadku komercyjnych wersji UNIX-a), nie ma nikogo odpowiedzialnego za aktualizację rozprowadzanych wersji. Twoim zadaniem jest znaleźć źródło oprogramowania i upewnić się, że wersja, którą otrzymujesz, jest kompletna i zawiera wszystko, co jest Ci potrzebne. Powinieneś również zwrócić uwagę na numer wersji, ponieważ dystrybutorzy często oferują ich kilka, a nie wszystkie są pierwszej świeżości.

Kopię wersji instalacyjnej Linuxa zdobyć można na kilka sposobów. Wybierz ten, który jest dla Ciebie najwygodniejszy albo też najtańszy. Najpopularniejszym sposobem jest zdobycie kopii na CD-ROM-ie; są one dostępne jako dodatki do wielu książek i czasopism. Inną metodą jest załadowanie go poprzez sieć z któregoś z węzłów FTP lub BBS. Można również zamówić Linuxa drogą pocztową.

Sposób, w jaki zdobędziesz kopię Linuxa, w pewnym stopniu przesądza o ilości dodatkowego oprogramowania. Wersje dostępne na dyskach CD-ROM często zawierają komplet oprogramowania, podczas gdy niektóre BBS-y i węzły FTP oferują jedynie niewielką część dystrybucji, która wystarcza do zainstalowania i użytkowania systemu w stopniu podstawowym. Mniejsze wersje mogą na przykład nie zawierać wszystkich dostępnych kompilatorów języków programowania czy systemu X Window.

Linux na płycie CD-ROM

Dystrybucje Linuxa na dyskach CD-ROM rozprowadzane są przez co najmniej kilkunastu producentów. Różnią się one zarówno numerami wersji, jak i liczbą dołączonych programów. Można zakupić jedno- i dwudyskowe wersje zawierające większość potrzebnych aplikacji, ale dostępne są też wersje pięcio- lub sześciodyskowe, zawierające ogromne ilości innych materiałów, takich jak dokumentacja pomocnicza, archiwa grup dyskusyjnych czy inne wersje jądra systemu.

Większość dystrybucji zawiera te same programy, różnica polega na ich organizacji, ilości oraz jakości dołączonej dokumentacji i programów narzędziowych. Poszczególne dystrybucje są stosunkowo tanie, więc wybierz tę, która najbardziej Ci odpowiada.

Upewnij się, że wersja, którą otrzymujesz, jest najświeższa (porównaj numery wersji na innych dostępnych kompaktach) i że zawiera interesujące Cię oprogramowanie dodatkowe. Na opakowaniu CD-ROM-u zazwyczaj znajdują się informacje o numerze dystrybucji (tak jak np. Slackware 3.4 czy Red Hat 5.0) i najważniejszych dołączonych aplikacjach (jak XFree86, kompilatory itp.). Czasem jednak informacje takie nie są zamieszczone i może okazać się, że zakupiłeś starą wersję systemu, lub że brakuje w niej istotnych dla Ciebie programów. Na szczęście dyski CD-ROM z dystrybucjami Linuxa nie są drogie.

Niektórzy dostawcy dołączają do dystrybucji programy lub dyski startowe znaczco ułatwiające proces instalacji. Dodanie skompilowanych gier, aplikacji i programów użytkowych również zwiększa atrakcyjność dystrybucji. Zwykle kosztują one nieco więcej, ale często w zamian za różnicę w cenie otrzymujesz fachowo napisaną instrukcję, a nawet pomoc techniczną. Dobry przykład może stanowić tu dystrybucja Caldera Linux.

Węzły FTP

FTP (File Transfer Protocol, protokół transmisji plików) jest protokołem powszechnie stosowanym w Internecie; umożliwia on dostęp do plików zlokalizowanych na serwerach zdalnych. Dystrybucje Linuxa oferowane są przez wiele węzłów FTP dostępnych dla każdego (nie musisz posiadać konta w takim węźle, po prostu logujesz się jako „guest” lub „anonymous” i używasz jako hasła swojego nazwiska lub adresu e-mailowego). Dostęp do nich wymaga tylko połączenia się z siecią Internet.

Żeby można było używać FTP, w systemie musi być obsługiwany protokół TCP/IP; nie jest ważne, czy jest to zwykły peçet pracujący w DOS-ie lub Windows, czy też stacja robocza UNIX lub Linux, byleby tylko korzystał z usług dostawcy Internetu udostępniającego usługę FTP. Oprogramowanie FTP musi działać po obu stronach połączenia. Aby załadować pliki z innego systemu, musisz uruchomić oprogramowanie do obsługi FTP w swoim systemie i połączyć się z odpowiednim oprogramowaniem w systemie zdalnym.

W Internecie działa duża liczba węzłów FTP. Ich zadaniem jest udostępnianie oprogramowania dla wszystkich chętnych. Wiele z nich jest tylko „odbiciami” (ang. *mirror*) innych węzłów, a więc udostępniają one dokładnie takie same pliki. Należy wybrać ten węzeł, z którym najłatwiej jest nawiązać połączenie, albo taki, który pozwoli na uzyskanie największej prędkości transmisji danych. Węzłem, z którego najczęściej ładowany jest Linux, jest sunsite.unc.edu. Zawiera on przejrzystą listę wszystkich dostępnych programów oraz starszych wersji jądra, sterowników i programów użytkowych. Wiele z płyt CD-ROM dostępnych na rynku to po prostu kopie zawartości tego węzła. Jeśli zamierzasz zdobyć wersję instalacyjną Linuxa poprzez FTP, jest to jeden z najlepszych punktów wyjścia.

Używanie FTP do ładowania plików

Łączenie się z węzłami FTP obsługiwanyimi przez Linuxa jest dość łatwe (pod warunkiem, że masz dostęp do Internetu). Możesz uruchomić program do obsługi FTP podając nazwę systemu, z którym zamierzasz się połączyć, jako parametr, lub podać ją po uruchomieniu tego programu. Jeśli jesteś podłączony bezpośrednio do Internetu, wprowadź polecenie `ftp` z nazwą serwera, z którym chcesz się połączyć, na przykład w ten sposób:

```
ftp sunsite.unc.edu
```

W tym przypadku używamy interfejsu tekstowego, choć istnieje kilka bardzo przyjemnych w obsłudze klientów FTP posiadających graficzny interfejs użytkownika, rozprowadzanych na przykład wraz z systemem X Window. Ich używanie jest o wiele łatwiejsze, przypomina rozwiązania znane z interfejsu Microsoft Windows. Wystarczy po prostu wybrać pliki, które nas interesują, i przeciągnąć je do katalogu w komputerze lokalnym.

Ponieważ wersja FTP oparta na interfejsie tekstowym jest najpopularniejsza, omówimy ją nieco dokładniej. Po połączeniu się z dostawcą Internetu powinieneś uruchomić program obsługi FTP. Jeśli używasz systemu Windows, okno DOS-owe jest często najłatwiejszym sposobem nawiązania połączenia. Wywołaj tryb MS-DOS i w wierszu poleceń programu FTP podaj nazwę serwera, z którym chcesz się połączyć.

Po uruchomieniu programu FTP system próbuje połączyć się z serwerem zdalnym. Jeśli proces ten zakończy się sukcesem, serwer poprosi o podanie identyfikatora użytkownika. Musisz wprowadzić prawidłowy identyfikator oraz hasło, chyba że serwer obsługuje anonimowy dostęp do FTP (jest tak w przypadku wszystkich węzłów udostępniających Linuxa). Po nawiązaniu połączenia, zanim zalogujesz się, często pojawia się krótki komunikat informujący o tym, że dany serwer obsługuje anonimowy dostęp do FTP (zwykle coś w rodzaju Guest login ok.). Oto przykładowy zapis logowania się do węzła sunsite.unc.edu:

```
ftp sunsite.unc.edu
331 Guest login ok, send your complete e-mail address as password.
Enter username (default: anonymous): anonymous
Enter password [tparker@tpci.com]:
|FTP| Open
230-           WELCOME to UNC and SUN's anonymous ftp server
230-           University of North Carolina
230-           Office FOR Information Technology
230-           sunsite.unc.edu
230 Guest login ok., access restrictions apply.
FTP>
```

Dostęp do anonimowego FTP uzyskuje się zazwyczaj, podając guest lub anonymous jako identyfikator użytkownika. Jeśli nie wiadomo, której z możliwości użyć, można spróbować obu. W większości przypadków serwer zapyta o hasło – w niektórych systemach nie musisz go podawać, a w innych należy wprowadzić swoje nazwisko lub adres e-mail. Dane te używane są tylko do celów statystycznych i ich podawanie nie stwarza żadnych problemów związanych z bezpieczeństwem.

Po zakończeniu procesu logowania zobaczysz symbol zachęty FTP>. Oznacza to, że system jest gotów do przyjmowania poleceń FTP. Czasem wyświetlany jest jeszcze krótki komunikat zawierający instrukcje dotyczące ładowania plików lub też ograniczeń nałożonych na Ciebie jako użytkownika anonimowego. Przykładowe komunikaty z węzła sunsite.unc.edu wyglądają następująco:

```
To get a binary file, type: BINARY and then: GET "File.Name" newfilename
To get a text file, type: ASCII and then: GET "File.Name" newfilename
Names MUST match upper, lower case exactly. Use the "quotes" as shown.
To get a directory, type: DIR. To change directory, type: CD "Dir.Name"
To read a short text file, type GET "File.Name" TT
For more, type HELP or see FAQ in gopher.
To quit, type EXIT or Control-Z.
```

```
230- If you e-mail to info@sunsite.unc.edu you will be sent help Σ
information
230- about how to use the different services sunsite provides.
230- We use the Wuarchive experimental ftpd. If you "get"
<directory>.tar.Z
230- or <file>.Z it will compress and/or tar it on the fly. Using ".gz" Σ
instead
230- of ".Z" will use the GNU zip (/pub/gnu/gzip*) instead, a superior
230- compression method.
```

Kiedy masz już dostęp do zasobów serwera, do wyświetlania zawartości plików i poruszania się po drzewie katalogów możesz używać poleceń Linuxa (czy UNIX-a). Aby otrzymać informacje o zawartości katalogu, użądź polecenia ls lub jego DOS-owego odpowiednika dir.

Zmiana katalogu bieżącego jest możliwa dzięki poleceniu `cd <nazwa_katalogu>`. Polecenie `cd ..` pozwoli Ci powrócić do katalogu nadzaknego (stojącego w hierarchii katalogów o jeden poziom wyżej od katalogu bieżącego). Inaczej niż w Linuxie, nie jest możliwe stosowanie skrótów klawiaturowych, więc musisz dokładnie wpisywać całe nazwy plików i katalogów.

Gdy przejrzyesz zawartość katalogów i znajdziesz plik, który chcesz pobrać, możesz zrobić to, wydając polecenie:

```
get nazwa_pliku
```

Polecenia `get` (załaduj, pobierz) i `put` (wyślij) mają punkt odniesienia w komputerze lokalnym, to znaczy wydając polecenie `get` kącesz swojemu oprogramowaniu pobrać plik z serwera i umieścić go w swoim komputerze, natomiast poleceniem `put` wysyłasz pliki ze swojego komputera do serwera zdalnego. Ważne jest, by dobrze zapamiętać, które polecenie działa w którą stronę, w przeciwnym wypadku może zdarzyć się, że przez przypadek pozbędziesz się jakichś ważnych danych.

Cudzysłów wokół nazwy pliku jest opcjonalny w większości wersji FTP, ale zabezpiecza on przed złym interpretowaniem polecenia przez serwer, warto więc go stosować.

Gdy wydajesz polecenie `get`, serwer przesyła dane do komputera, po czym wypisuje krótką informację o zakończeniu tego procesu. Podeczas trwania operacji przesyłania nawet dużych plików na ekranie nie są wyświetlane żadne informacje, musisz więc po prostu być cierpliwy. Oto co widzimy na ekranie po wydaniu polecenia `get`:

```
FTP> get "file1.txt"
200 PORT command successful.
150 BINARY data connection for FILE1.TXT (27534 bytes)
226 BINARY Transfer complete.
27534 byte received in 2.35 seconds (12Kbytes/s).
```

FTP posiada dwa tryby przesyłania danych: tryb `ASCII` (znaki 7 – bitowe) i `Binary` (znaki 8 – bitowe). Choć niektóre systemy przełączają się pomiędzy nimi automatycznie, warto jest ustawić odpowiedni tryb ręcznie, dzięki czemu masz pewność, że nie tracisz niepotrzebnie czasu. Pliki wchodzące w skład dystrybucji Linuxa muszą być przesyłane w trybie `Binary`. Zmiana trybu przesyłania danych na binarny (wymagany przy przesyłaniu wszystkich plików wykonywalnych) możliwa jest dzięki poleceniu:

```
binary
```

Do trybu ASCII wrócić można, używając polecenia `ASCII`. Gdy załadujesz plik binarny w trybie `ASCII`, nie będzie można go uruchomić. Ładowanie plików tekstowych w trybie `Binary` nie narusza ich zawartości, więc jest to tryb, którego można używać zawsze.

Do opuszczania programu FTP służą polecenia `quit` lub `exit`. Oba najpierw zamykają sesję na serwerze, po czym kończą program FTP na Twoim komputerze.

Pamiętaj, że archiwia zawierające Linuxa są dość spore i przesyłanie nawet niewielkiej dystrybucji za pomocą modemu asynchronicznego może zająć dłuższą chwilę. Jeśli używasz wolnego modemu (9600 bodów lub mniej), powinieneś wziąć pod uwagę inne

metody, gdyż ta zajmie Ci co najmniej kilka godzin. Niektóre serwery zaś ograniczają czas, przez jaki możesz pozostawać połączony.



Jeśli zamierzasz przesyłać wiele plików i nie masz ochoty wpisywać kolejno wszystkich ich nazw, użyj polecenia `mget`. Jego parametrem jest szablon pasujący do nazw plików, które chcesz załadować. Po wydaniu polecenia `mget` trzeba potwierdzić, czy na pewno chcesz załadować każdy z plików o nazwie zgodnej z zadanym szablonem. Przykładowo, polecenie `mget *.*` spowoduje załadowanie wszystkich plików z bieżącego katalogu, przy czym o potwierdzenie przesłania każdego z nich zostaniesz zapytany osobno. Jeśli jesteś pewny, że chcesz załadować wszystkie pliki, możesz ominąć te pytania używając polecenia `prompt`. Wyłączy ono konieczność potwierdzania i pliki będą przesyłane jeden po drugim.

Węzły FTP udostępniające Linuxa

Lista węzłów FTP udostępniających Linuxa powoli ulega zmianom, niemniej jednak adresy zebrane poniżej były dostępne w chwili, gdy książka ta szła do druku. Wiele z nich to kopie innych serwerów (ang. *mirrors*) i mają one dokładnie taką samą zawartość jak serwery główne (ang. *home sites*). Serwerami głównymi są: `tsx-11.mit.edu`, `sunsite.unc.edu`, oraz `nic.funet.fi`.

Węzeł, który znajduje się najbliżej Twojego miejsca zamieszkania, może być rozpoznany przez identyfikator kraju na końcu adresu (`uk` – Wielka Brytania, `fr` – Francja, `pl` – Polska itp.). Większość wersji FTP pozwala używać zarówno nazwy domenowej, jak i adresu IP serwera, choć nazwa nie może być poprawnie zinterpretowana przez lokalną bramkę internetową, użycie adresu IP jest najlepszą metodą, należy jednak upewnić się, że wszystkie cztery składniki adresu IP są wpisane prawidłowo. Tabela 2.1 zawiera adresy węzłów będących doskonałym źródłem materiałów dotyczących Linuxa.

Jeżeli natkniesz się na jakieś problemy przy łączaniu się z którymś z tych węzłów, spróbuj połączyć się z innym. Jeśli trudności nie ustępują, może to oznaczać problem z dostępem do Internetu.

World Wide Web

Również strony WWW mogą być pomocne przy zdobywaniu kopii dystrybucji Linuxa. Na przykład pod adresem `http://sunsite.unc.edu/mdw/linux.html` znajdziesz ofertę głównych węzłów FTP dostarczających Linuxa. Aby załadować wersję instalacyjną systemu, możesz użyć dowolnej przeglądarki internetowej, takiej jak Mosaic czy Netscape. Większość ze stron WWW poświęconych Linuxowi oferuje również dokumentację.

Stron WWW poświęconych tematyce związanej z systemem Linux jest bardzo dużo, o wiele więcej niż moglibyśmy tu wypisać. Aby je znaleźć, użyj jednej z wyszukiwarek

internetowych (jak np. `yahoo.com` czy `altavista.digital.com`), jako klucz podając słowo „Linux”. Otrzymasz ogromną liczbę adresów WWW. Aby zawęzić obszar poszukiwań, możesz dopisać `binaries` lub wpisać nazwę interesującej Cię wersji (np. `slackware`).

Tabela 2.1. Węzły FTP zawierające oprogramowanie dla Linuxa i ich adresy IP

Nazwa węzła	Adres IP	Katalog
<code>tsx-11.mit.edu</code>	18.172.1.2	<code>/pub/linux</code>
<code>sunsite.unc.edu</code>	152.2.22.81	<code>/pub/Linux</code>
<code>nic.funet.fi</code>	128.214.6.100	<code>/pub/OS/Linux</code>
<code>ftp.mcc.ac.uk</code>	130.88.200.7	<code>/pub/linux</code>
<code>fgbl.fgb.mw.tu-muenchen.de</code>	129.187.200.1	<code>/pub/linux</code>
<code>ftp.infdrnatik.twmuenchen.de</code>	131.159.0.110	<code>/pub/Linux</code>
<code>ftp.dfv.rwth-aachen.de</code>	137.226.4.105	<code>/pub/linux</code>
<code>ftp.informatik.rwth-aachen.de</code>	137.226.112.172	<code>/pub/Linux</code>
<code>ftp.ibp.fr</code>	132.227.60.2	<code>/pub/linux</code>
<code>kirk.bu.oz.au</code>	131.244.1.1	<code>/pub/OS/Linux</code>
<code>ftp.uu.net</code>	137.39.1.9	<code>/systems/unix/linux</code>
<code>wuarchive.wustl.edu</code>	128.252.135.4	<code>/systems/linux</code>
<code>ftp.win.tue.nl</code>	131.155.70.100	<code>/pub/linux</code>
<code>ftp.stack.erc.tue.nl</code>	131.155.2.71	<code>/pub/linux</code>
<code>ftp.ibr.cs.tu-bs.de</code>	134.169.34.15	<code>/pub/os/linux</code>
<code>ftp.denet.dk</code>	129.142.6.74	<code>/pub/OS/linux</code>

Poczta elektroniczna

Jeśli nie masz dostępu do usług FTP ani WWW, natomiast masz możliwość korzystania z poczty elektronicznej (e-mail), nadal masz szansę na zdobycie kopii Linuxa poprzez sieć. Jest to przydatne w systemach nie pozwalających na bezpośrednie połączenia FTP, ale dopuszczających obsługę poczty. Aby załadować Linuxa z węzła FTP za pośrednictwem poczty elektronicznej, użyj programu `ftpmail`.

Wszystkie wymienione wcześniej węzły FTP umożliwiają połączenia za pomocą `ftpmail`. Jeśli chcesz otrzymać instrukcję obsługi tego programu, wyślij wiadomość do użytkownika `ftpmail` na dowolnym z tych serwerów (na przykład do `ftpmail@sunsite.unc.edu`). Treść tego listu powinna zawierać jedynie słowo `help`. Inna zawartość może spowodować, że zostaniesz źle zrozumiany przez program `ftpmail` i nie otrzymasz instrukcji. Z tego powodu nie powinieneś również wysyłać standardowo dołączanego podpisu `itp`.

Po otrzymaniu Twojego zgłoszenia, `ftpmail` prześle Ci instrukcje dotyczące korzystania z tej usługi. W większości przypadków musisz po prostu wpisać polecenia, jakie

wydałbyś w programie `ftp`, jako treść swojego listu. Na przykład, by otrzymać informację o zawartości katalogu `/pub/Linux`, wyślij wiadomość o następującej treści:

```
open sunsite.unc.edu
cd /pub/Linux
ls
quit
```

Program obsługujący `ftpmail` po stronie serwera zinterpretuje te polecenia tak, jakby były wpisywane w programie `ftp`. Jeśli chcesz załadować plik pocztą elektroniczną, wyślij wiadomość o treści:

```
open sunsite.unc.edu
cd /pub/Linux
binary
get README
quit
```

Wiadomość o takiej zawartości spowoduje wysłanie do Ciebie pliku `README` z katalogu `/pub/Linux`. `ftpmail` jest wolniejszy od `ftp`, ponieważ trzeba czekać, aż e-mail pokoną drogę do serwera docelowego, zostanie zinterpretowany przez program `ftpmail`, odpowiedź zostanie odpowiednio sformatowana i w końcu wysłana do Ciebie. Mimo wszystko, jest to dobry sposób przesyłania plików dla osób nie mających dostępu do FTP. Ma on również tę zaletę, że nie wymaga logowania się do serwera, jeśli chcesz tylko obejrzeć zawartość katalogów. Może to być użyteczne, gdy chcesz okresowo sprawdzać, czy nie pojawiło się tam nowe oprogramowanie.

Jest jedna rzecz, na którą powinieneś zwrócić uwagę. Rozmiar plików, które zostaną przesłane, może przekroczyć maksymalny dopuszczalny rozmiar obsługiwany przez Twój system poczty. Niektóre z systemów podzielą w takim przypadku plik na kilka mniejszych, pozwalając Ci je potem poskładać, ale niektóre po prostu nie radzą sobie z dużymi plikami, co czyni program `ftpmail` zupełnie nieprzydatnym do przesyłania dystrybucji Linuxa.

Systemy BBS (Bulletin Board Systems)

Setki różnych węzłów BBS oferują dostęp do dystrybucji Linuxa i archiwów list dyskusyjnych. Niektóre z nich same uaktualniają oferowaną wersję Linuxa poprzez FTP, inne zrzucają ten obowiązek na swoich użytkowników.

Lista tych systemów byłaby bardzo dłużna i prawie natychmiast stałaby się nieaktualna. Najlepszym sposobem na otrzymanie aktualnego wykazu węzłów BBS zajmujących się rozprowadzaniem Linuxa jest wysłanie wiadomości do `healyzh@holonet.net` (Zane Healy) z prośbą o przesyłanie takowej. Jeśli nie masz dostępu do poczty elektronicznej, spróbuj zapytać w lokalnych systemach BBS (niektóre prowadzą własne listy, ale różnie bywa z ich dokładnością i aktualnością), a w razie niepowodzenia po prostu poproś kogoś mającego dostęp do Internetu o zdobycie tych informacji dla Ciebie.

Co to jest dystrybucja?

Słowo „dystrybucja” ma dla użytkowników Linuxa dwa znaczenia. Pierwsze z nich związane jest z dystrybutorem wersji Linuxa, na przykład zarówno Slackware, jak i Red Hat są dystrybucjami tego systemu. Jak wspomniano wcześniej, różnice między nimi nie są zbyt znaczące.

Drugie znaczenie dotyczy wersji Linuxa i oprogramowania, na przykład różnymi dystrybucjami są Slackware 2.3 i Slackware 2.4. Właściwie powinno się je określać jako wersje tej samej dystrybucji (Slackware), ale termin ten używany jest od dawna dla określenia nowej wersji jądra systemu i programów użytkowych, trudno więc to zmienić. Może to być nieco mylące w momencie, gdy ktoś pyta, jakiej dystrybucji używasz, ponieważ nie wiadomo, czy chodzi o dostawcę, czy numer wersji. Większość ludzi radzi sobie, podając obie informacje: „używam Slackware 2.0.33”.

Czy musisz koniecznie używać najnowszej wersji Linuxa? Najprawdopodobniej nie. Różnice pomiędzy kolejnymi wersjami są zazwyczaj drobne (szczególnie pomiędzy wersjami mającymi ten sam numer główny, jak 2.3 i 2.4). Dopóki nowa wersja nie zawiera jakichś niezbędnych sterowników lub czegoś bardzo Ci potrzebnego, raczej nie warto zwracać sobie nią głowy. Inaczej sprawa wygląda w przypadku wersji różniących się numerem głównym, na przykład 1.X i 2.X. Zwykle zawierają one poważne zmiany w jądrze systemu operacyjnego i są warte zainstalowania. Niestety, często wiąże się to z koniecznością przeinstalowania całego oprogramowania.



Nie daj się wciągnąć w manię uaktualniania systemu. Niektórym wydaje się, że koniecznie muszą mieć zainstalowaną najnowszą wersję Linuxa, ale tak naprawdę różnica pomiędzy wersjami np. 2.0.33 i 2.0.34 w przeważającej większości przypadków będzie niezauważalna. Prawdziwą motywacją do uaktualnienia oprogramowania może być większa wydajność nowej wersji, nowe, potrzebne Ci sterowniki, lub też zmiana głównego numeru wersji (co oznacza poważniejsze zmiany w kodzie źródłowym). Ciągłe instalowanie nowych wersji sprawi, że nie wystarczy Ci czasu na pracę czy zabawę z dopiero co zainstalowanymi programami, oraz stwarza konieczność częstego przeinstalowywania całego systemu.

Dystrybucja Linuxa a zestawy dysków

Dystrybucja to zbiór oprogramowania wystarczający do instalacji i używania systemu operacyjnego. Składa się ona z tzw. zestawów dysków (ang. *disk sets*), zawierających pogrupowane tematycznie aplikacje (choć w rzeczywistości zestawy dysków rzadko są rozprowadzane na dyskietkach).

Większość CD-ROM-ów i węzłów FTP zawiera to samo oprogramowanie, ale niektóre z nich używają własnego klucza nazywając zestawy dysków. Jako przykład omówiona

zostanie jedna z popularniejszych dystrybucji – Slackware. Dostępne w niej obecnie zestawy dysków przedstawiamy poniżej.

- υ **Zestaw A** – system w wersji podstawowej. Zawiera jądro systemu i najważniejsze programy użytkowe, takie jak interpreter poleceń (ang. *shell*, zwany dalej również powłoką systemu operacyjnego albo po prostu powłoką), edytor tekstów i kilka programów narzędziowych. Jest to jedyny zestaw mieszczący się w całości na jednej dyskietce. Pozwala to na instalację i używanie Linuxa bez twardego dysku.
- υ **Zestaw AP** – aplikacje, czyli różne edytory tekstów, wszystkie standardowe programy użytkowe pracujące w trybie tekstowym znane z UNIX-a, strony z dokumentacją (ang. *man pages*) i dodatki rozprowadzane zgodnie z zasadami licencji GNU, takie jak np. GhostScript.
- υ **Zestaw D** – narzędzia dla programistów. Zawiera kompilatory, biblioteki i narzędzia pomocnicze objęte licencją GNU. W zestawie tym znajduje się również kod źródłowy wielu bibliotek Linuxa, którego możesz użyć, by dostosować jądro systemu do własnych potrzeb.
- υ **Zestaw E** – edytor *emacs*.
- υ **Zestaw F** – FAQ (Frequently Asked Questions), czyli najczęściej zadawane pytania i krótkie, ale treściwe odpowiedzi oraz inne pliki z dokumentacją systemu.
- υ **Zestaw I** – dokumentacja dla oprogramowania objętego licencją GNU.
- υ **Zestaw IV** – Interviews. Jest to zestaw bibliotek, plików nagłówkowych i dokumentacji pomocny przy tworzeniu graficznego interfejsu użytkownika w języku C++.
- υ **Zestaw N** – oprogramowanie sieciowe. Obsługa protokołu TCP/IP, UUCP, programy do obsługi poczty i grup dyskusyjnych oraz inne programy użytkowe.
- υ **Zestaw OI** – zawiera ParcPlace Object Builder oraz Object Interface Library. Są to produkty komercyjne, udostępnione programującym w systemie Linux przez firmę ParcPlace.
- υ **Zestaw OO** – narzędzia do programowania zorientowanego obiektowo (Object Oriented Programming, OOP), między innymi kompilator GNU Smalltalk wraz z interfejsem X (STX).
- υ **Zestaw Q** – pliki źródłowe jądra Linuxa oraz pliki zawierające obrazy dysków startowych.
- υ **Zestaw T** – systemy formatowania tekstu TeX i LaTeX. System TeX jest dość powszechnie używany do komputerowego składu tekstu.
- υ **Zestaw TCL** – język Tcl, wraz z Tk, TclX i różnymi programami narzędziowymi.
- υ **Zestaw Y** – kolekcja gier.

- υ **Zestaw X** – XFree86, czyli system X i kilka programów zarządzających okienkami (ang. *window managers*).
- υ **Zestaw XAP** – aplikacje dla systemu X, takie jak programy do zarządzania plikami, GhostView, kilka bibliotek, gry i programy użytkowe.
- υ **Zestaw XD** – programy narzędziowe dla programistów korzystających z systemu X, biblioteki, obsługa PEX i narzędzia ułatwiające łączenie się z serwerem. Ten zestaw jest niezbędny, jeśli planujesz pisanie aplikacji wykorzystujących graficzny interfejs użytkownika oparty na systemie X.
- υ **Zestaw XV** – program zarządzający okienkami dla X. Zawiera biblioteki XView i menedżery Open Look. Mogą być one używane zamiast tych dostarczanych w zestawie X.

Choć zestaw A pozwala zainstalować Linuxa z dyskietki, do pełnej instalacji (na dysku twardym, zawierającej wszystkie standardowe programy użytkowe) powinieneś posiadać zestawy A, AP, D i F. Pozwolą one na pracę w trybie tekstowym. Jeżeli zamierzasz również używać systemu X, potrzebne będą też zestawy X oraz XAP. Programiści nie obejdą się bez zestawu D i ewentualnie XD.

Aktualizacja istniejącego systemu linuxowego

Jeśli Twój system pracuje już pod kontrolą Linuxa, możliwe, że właśnie nadszedł czas na jego modernizację. W większości przypadków będzie ona polegała na skopiowaniu nowego oprogramowania do odpowiedniego katalogu i przekompilowaniu jądra tak, by połączyć je ze sterownikami, których używasz. Czasem jednak konieczne będzie zainstalowanie całego systemu na nowo, zwłaszcza przy zmianie głównego numeru wersji lub rodzaju dystrybucji.

Większość wersji Linuxa nie oferuje możliwości automatycznej aktualizacji systemu (jak ma to miejsce np. w systemie Windows). Zamiast tego, należy wykonać kopię zapasową potrzebnych plików zapisanych na dysku twardym, zainstalować nową wersję systemu, po czym przywrócić pliki z kopii zapasowej. Na szczęście zwykle musisz tylko skopiować kilka nowych plików zawierających kod źródłowy jądra systemu do istniejących katalogów, a następnie przekompilować jądro. Do tematu tego wrócimy jeszcze w następnym rozdziale.

Podsumowanie

W tym rozdziale wyjaśniliśmy, czym jest dystrybucja Linuxa, gdzie można zaopatrzeć się w wersję instalacyjną, a także opisaliśmy poszczególne zestawy dysków wchodzące

w skład dystrybucji. Przejdziemy teraz do następnego zagadnienia, czyli instalacji i konfiguracji systemu. Poniżej przedstawiono rozdziały poruszające ten problem.

Rozdział 3. „Instalacja Linuxa” i rozdział 5. „Podsumowanie instalacji” zawierają informacje o procesie instalowania systemu.

Podstawowe informacje o używaniu systemu znajdziesz w części II, w rozdziale 6. „Od czego zacząć” oraz w rozdziale 11. „bash”.

O tym, jak skonfigurować serwer X Window, dowiesz się czytając rozdział 22. „Instalacja i konfiguracja XFree86”.

Rozdziały w części IV, m.in. rozdział 32. „Podstawy administracji systemem” traktują o administrowaniu serwerem linuxowym.

Rozdział 3.

Instalacja Linuxa

Tim Parker

W tym rozdziale:

- υ Praca z Linuxem
- υ Dyskietki startowe
- υ Krótki przewodnik po instalacji
- υ Podział dysku twardego na partycje
- υ Instalacja partycji linuxowych
- υ Instalacja Linuxa
- υ Konfiguracja startu systemu
- υ Przeglądanie zainstalowanego oprogramowania
- υ Rozwiązywanie problemów
- υ Problemy z urządzeniami SCSI

Praca z Linuxem

Najprawdopodobniej zainstalowałeś już Linuxa. Nawet jeśli system wydaje się działać prawidłowo, możesz nie być zadowolony z instalacji, czy to z powodu kiepskiej organizacji, czy też dlatego, że dopiero eksperymentowałeś i chciałbyś spróbować lepszej konfiguracji. Ten rozdział omawia zagadnienia związane z instalowaniem Linuxa po raz pierwszy i przeinstalowywaniem go od początku oraz z aktualizacją oprogramowania.

Powinieneś pamiętać o tym, że rozdział ten porusza zagadnienia związane z instalowaniem trzech głównych dystrybucji Linuxa: Slackware, Red Hat oraz OpenLinux (która jest oparta na Slackware). Zależnie od dystrybucji, którą instalujesz, a także od numeru wersji, program instalacyjny może się nieco różnić. Na szczęście pytania zadawane przez program instalacyjny są dość oczywiste.

Choć proces instalacji Linuxa jest prosty, możesz natknąć się na wiele drobiazgów sprawiających problemy. Nie ufaj zapewnieniom o łatwej instalacji, które często można znaleźć na opakowaniach dystrybucji systemu. Kilka kroków wymaga szczególnej cierpliwości, eksperymentowania i wiedzy o tym, co się dzieje – dopiero po uporaniu się z nimi Linux instalował się będzie bezboleśnie. Główne etapy instalacji Linuxa to:

- υ stworzenie dysków boot (startowego) i root (zawierającego system plików niezbędny do działania systemu)¹,
- υ podział dysku twardego na partycje,
- υ uruchomienie Linuxa z dyskietki,
- υ utworzenie pliku wymiany,
- υ utworzenie linuxowego systemu plików,
- υ instalacja oprogramowania,
- υ konfiguracja jądra systemu,
- υ konfiguracja startu systemu,
- υ uruchomienie Linuxa z dysku twardego.

Przyjrzyjmy się każdemu z tych kroków nieco bardziej szczegółowo. Instalowanie przebiega prawie tak samo w przypadku użycia dysku CD-ROM, jak i w przypadku dyskietki. Ponieważ instalacja z dysku CD-ROM jest najczęstszym przypadkiem, ten właśnie proces opiszemy dokładniej w tym rozdziale.

Jeśli instalujesz system z dyskietek, a dystrybucję załadowałeś z węzła FTP lub skopiowałeś ją z dysku CD-ROM, będziesz potrzebował jednej sformatowanej w systemie DOS dyskietki na każdy dysk w każdym z instalowanych zestawów dysków. Aby skopiować odpowiednie pliki na dyskietkę, możesz użyć standardowego polecenia `copy`. Pliki są numerowane, więc nie powinieneś mieć problemów ze zorientowaniem się w ich kolejności.

¹ Krok ten (i w ogóle używanie stacji dysków) w większości systemów można pominąć, używając programu `loadlin` znajdującego się na CD-ROM-ie z dystrybucją Red Hat, bądź też uruchamiając system wprost z dysku CD-ROM (tylko w przypadku komputerów z nowszymi płytami głównymi; *przyp. tłum.*).

Instalacja bez użycia stacji dysków

Jeśli na dysku twardym Twojego komputera znajduje się już system operacyjny DOS lub system umożliwiający pracę w oknie DOS-owym, jak Windows, możesz spróbować uruchomić instalację bezpośrednio z dysku CD-ROM. Uruchom system operacyjny i na dysku CD-ROM poszukaj pojedynczego pliku wykonywalnego (z rozszerzeniem .com, .exe lub .bat). Niektóre wersje dystrybucji Red Hat Linux zawierają na przykład plik wykonywalny o nazwie `RED HAT`, który obsługuje instalację. W takim przypadku wpisz w wierszu poleceń DOS-a: „`RED HAT`”²

Zostaniesz poproszony o podanie kilku informacji, które pomogą programowi instalacyjnemu wybrać najodpowiedniejszą wersję jądra. W tym momencie możesz też wydać specjalne polecenia dla instalacji.



Nie uruchamiaj instalacji bezdyskietkowej bezpośrednio z poziomu systemu Windows 3.1, Windows 95 czy OS/2. Uruchom system w trybie MS-DOS i dopiero wtedy wywołaj program instalacyjny.

Program instalacyjny znajdujący się na dysku CD-ROM poprowadzi Cię przez proces dobierania jądra systemu, wyświetlając szereg menu ułatwiających jego obsługę. Cały proces jest dość intuicyjny i raczej łatwy dla każdego, kto wie, jaki sprzęt jest zainstalowany w jego komputerze. Jeśli nawet nie jesteś tego pewny, najgorsze, co może się zdarzyć, to konieczność ponownego wykonania instalacji z innymi ustawieniami.

Jeśli program instalacyjny wykryje problemy związane z konfiguracją sprzętu, może zaoferować Ci poradę co do zmiany ustawień. Jest to jednak tylko porada, nic więcej. Nie musisz jej słuchać, jeśli nie chcesz, chociaż w niektórych przypadkach sugestie takie będą bardzo pomocne (np. rozwiązują problem konfliktu przerwań), toteż powinieneś zwrócić na nie uwagę (w dalszej części tego rozdziału podamy tabelę zawierającą standardowe przyporządkowania numerów przerwań).

Dyskietki startowe

Nawet jeśli instalujesz Linuksa z CD-ROM-u, nadal potrzebujesz dwóch dyskietek³ (o pojemności 1.2 lub 1.44 MB). Są to dyskietki nazywane *boot disk* (startowa, zawiera jądro systemu) i *root disk* (zawierająca system plików niezbędny do instalacji). Oba te dyski razem stanowią kompletną i bardzo małą implementację Linuksa. Mając je do dyspozycji, można już bawić się tym systemem, choć nie jest dostępna większość programów użytkowych.

² W wersji rozprowadzanej z książką funkcję tę pełni plik `autoboot.bat` (który uruchamia program `loadlin.exe`), znajdujący się w katalogu `\dosutils` (*przyp. wyd.*).

³ Jeśli nie ma możliwości uruchomienia systemu z dysku CD-ROM, w skład dystrybucji Red Hat wchodzi zwykle program `loadlin`, dzięki któremu można uruchomić program instalacyjny bez konieczności tworzenia dyskietek startowych (*przyp. thm.*).



Jeśli posiadasz nowszy komputer, możliwe, że w ogóle nie potrzebujesz dyskietek. Najnowsze wersje BIOS-u pozwalają uruchomić system bezpośrednio z dysku CD-ROM (szczególnie jeśli jest to urządzenie SCSI, ale jest to też możliwe w niektórych urządzeniach IDE). Jeśli Twój system umożliwia uruchomienie z dysku CD-ROM, wszystko, co musisz zrobić, to włożyć CD-ROM z Linuxem do napędu i zrestartować komputer. Oprogramowanie zawarte na dysku automatycznie wybierze prawidłowe obrazy dysków root i boot.

W większości przypadków dyski boot i root należy utworzyć w oparciu o tzw. pliki obrazów (ang. *images*). Są to gotowe wersje systemu, które trzeba tylko przenieść na dyskietki. Dystrybucje dostępne na płytach CD-ROM i w węzłach FTP zawierają specjalne katalogi z obrazami dysków boot i root dostosowanymi do różnych konfiguracji sprzętowych. Należy wybrać te z nich, które najlepiej pasują do posiadanego sprzętu, skopiować je na dyskietki i uruchomić z nich system.

Większość z tych czynności można wykonać pod kontrolą systemu DOS, ale do przeniesienia plików obrazów na dyskietki nie można użyć polecenia `copy`. Służy do tego specjalny program o nazwie `rawrite.exe`, dołączany do większości dystrybucji Linuxa.

Wybór właściwych dysków boot i root

CD-ROM-y zawierające dystrybucje Linuxa zazwyczaj posiadają w katalogu głównym podkatalogi `bootdsks.144` i `rootdsks.144` (dla dyskietek 3,5 – calowych) oraz `bootdsks.12` i `rootdsks.12` (dla dyskietek 5,25 cala), w których znajdują się odpowiednie pliki obrazów dla dysków boot i root. Dostępne wersje dysków boot, a właściwie zawartego na nich jądra systemu, są zazwyczaj opisane w pliku znajdującym się w jednym z tych katalogów. Również same pliki mają nazwy skorelowane z ich zawartością, np. plik obrazu o nazwie `scsi` zawiera jądro z wbudowaną obsługą kontrolerów SCSI; jeśli używasz komputera wyposażonego w taki kontroler, najprawdopodobniej właśnie tego pliku powinieneś użyć.

Liczba wersji dysku boot jest całkiem spora. Opiszemy teraz skrótnie sprzęt obsługiwany przez najpopularniejsze z nich.

- ⦿ **aztech.** Sterowniki dla dysków twardych IDE oraz SCSI, obsługa CD-ROM-ów w standardzie Aztech, włączając w to napędy Aztech, Okana, Orchid oraz Wearnes.
- ⦿ **bare.** Tylko sterowniki dla dysków twardych IDE (bez obsługi dysków CD-ROM).
- ⦿ **cdu31a.** Sterowniki dla dysków twardych IDE i SCSI oraz dla napędów CD-ROM Sony CDU31 i Sony CDU33a.
- ⦿ **cdu535.** Sterowniki dla dysków twardych IDE i SCSI oraz dla napędów CD-ROM Sony 535 i Sony 531.

- υ **idecd**. Sterowniki dla dysków twardych IDE oraz SCSI, obsługa CD-ROM-ów IDE oraz ATAPI.
- υ **mitsumi**. Sterowniki dla dysków twardych IDE oraz SCSI, obsługa napędów CD-ROM firmy Mitsumi.
- υ **sbpacd**. Sterowniki dla dysków twardych IDE oraz SCSI, obsługa napędów CD-ROM podłączanych za pośrednictwem karty Sound Blaster (rozprowadzanych głównie jako zestawy multimedialne).
- υ **scsi**. Sterowniki dla dysków twardych IDE oraz SCSI, obsługa CD-ROM-ów w standardzie SCSI.
- υ **scsinet1**. Sterowniki dla dysków twardych IDE oraz SCSI, obsługa CD-ROM-ów w standardzie SCSI oraz sterowniki do obsługi sieci Ethernet. Sterowniki SCSI obsługują m.in. następujący sprzęt: Adaptec 152X, 1542, 1740, 274X i 284X, Buslogic, EATA-DMA (jak np. DPT, NEC i karty AT&T), Seagate ST-02 oraz Future Domain TCC-8XX i 16XX. Karty kompatybilne z powyższymi oczywiście również będą działać poprawnie.
- υ **scsinet2**. Sterowniki dla dysków twardych IDE oraz SCSI, obsługa CD-ROM-ów w standardzie SCSI oraz sterowniki do obsługi sieci Ethernet. Sterowniki SCSI obsługują następujący sprzęt: NCR5380, NCR 53C7 i 8XX, Always IN2000, Pro Audio Spectrum 16, Qlogic, Trantor T128, T128F, T228, Ultrastor oraz 7000 FASST, a także karty kompatybilne z powyższymi.
- υ **xt**. Sterowniki dla dysków twardych IDE i kompatybilnych ze starszymi dyskami przeznaczonymi dla komputerów PC-XT.

W niektórych dystrybucjach rozszerzenie pliku zawierającego obraz dysku boot wskazuje na rodzaj dyskietki, jakiej należy użyć (rozszerzenie .144 wskazuje na dyskietkę 3,5 cala, .12 – na dyskietkę 5,25 cala). Nie można zapisać obrazu typu .144 na dysk 5,25 cala, ani obrazu typu .12 na dyskietkę 3,5 cala. W większości dystrybucji konwencja ta nie jest przestrzegana, ale w zamian wspomniane pliki umieszczane są w dwóch osobnych katalogach.

Mniej możliwości masz przy wyborze obrazu dysku root. W większości dystrybucji dostępne są cztery wersje tego dysku, choć czasem pojawiają się też inne pliki. Oto najczęściej spotykane pliki obrazów dysku root.

- υ **color**. Oferuje pełnoekranowy, kolorowy program instalacyjny systemu Linux.
- υ **tape**. Zaprojektowany z myślą o instalacji Linuxa z napędu taśmowego.
- υ **tty**. Prosta wersja programu instalacyjnego dla terminali, pozabawiona kolorów i grafiki.
- υ **umsdos**. Plik obrazu używany do instalacji UMSDOS, czyli wersji Linuxa pracującej w oparciu o istniejący system plików MS-DOS. Skrypt instalacyjny tworzy automatycznie wszystkie potrzebne podkatalogi. Nie jest to jednak sposób

tak wydajny, jak instalacja Linuksa na osobnej partycji, choć pozwala zachować istniejące systemy plików i partycje.

Informacja o typie dyskietki, dla jakiego przeznaczony jest każdy z plików obrazu, wpisana jest w jego nazwę – np. `color144` i `color12`.

Jeśli wersję instalacyjną otrzymałeś z węzła FTP lub BBS, pliki te mogą być zarchiwizowane i skompresowane – mają wówczas rozszerzenie `.gz`. Przed skopiowaniem na dyskietki należy rozpakować je za pomocą programu `gzip`.



Wersja `color` programu instalującego jest o wiele bardziej przyjazna dla użytkownika i przyjemniejsza w obsłudze niż `tty`. Jednak nie jest ona odporna na błędy popełnione przy wpisywaniu informacji i nie zawsze działa poprawnie. Warto spróbować jej użyć, chyba że dokładnie wiesz, w jaki sposób chcesz zainstalować system. Wersja kolorowa informuje użytkownika o tym, co aktualnie się dzieje, a także wymaga więcej interakcji (na przykład naciskania przycisku OK).

Po zdecydowaniu, których dyskietek boot i root chcesz użyć (jeśli nie jesteś pewny, wybierz dysk boot najbardziej odpowiadający konfiguracji sprzętowej Twojego systemu i dysk root w wersji `color` lub `tty`), można utworzyć odpowiednie dyskietki, kopując na nie pliki obrazów. Jeśli nawet wybierzesz nieprawidłową wersję plików obrazów, nie stanie się nic złego – co najwyżej będziesz musiał ponowić instalację, używając innych plików.

Tworzenie dysków boot i root

Dyskietki boot i root możesz stworzyć z poziomu systemu DOS lub Linux (UNIX). Jeśli w Twoim komputerze nie jest zainstalowany DOS i nie posiadasz dyskietki startowej MS-DOS, musisz zrobić kopie tych dwu dyskietek w innym systemie. Najpierw opiszymy tworzenie dyskietek pod kontrolą systemu MS-DOS, ponieważ jest to przypadek najczęściej spotykany.

Żeby utworzyć dyski boot i root, musisz użyć specjalnego programu kopiącego pliki obrazów na dyskietki. Jeśli pliki te są skompresowane (mają rozszerzenie `.gz`), powinieneś rozpakować je, używając programu `gzip`. Jeśli znajdują się one na dysku CD-ROM, należy najpierw skopiować je na dysk twardy, ponieważ w przeciwnym przypadku program `gzip` nie będzie w stanie zapisać zdekompresowanej wersji pliku na dysku CD-ROM. Nawet jeśli posiadasz pliki w wersji nieskompresowanej, warto skopiować je do jakiegoś tymczasowego katalogu na dysku twardym – dzięki temu nie będziesz musiał martwić się o wpisywanie pełnych ścieżek dostępu do plików.

Aby zdekompresować plik z rozszerzeniem `.gz`, należy wydać następujące polecenie:

```
gzip -d <nazwa_pliku.gz>
```

Opcja `-d` powoduje rozpakowanie pliku o nazwie `nazwa_pliku.gz`. Po zakończeniu tej operacji plik z rozszerzeniem `.gz` zostanie usunięty i na dysku pozostanie tylko wersja

rozpakowana (o tej samej nazwie, ale bez rozszerzenia .gz). Musisz zdekompresować plik obrazu zarówno dla dysku boot, jak i root. Przykładowo, by rozpakować pliki scsi.144 i color144, powinieneś wydać następujące polecenia:

```
gzip -d scsi.gz  
gzip -d color144.gz
```

Teraz pliki obrazów mogą zostać zapisane na dyskietkach. Dyskietki nie muszą być puste, ponieważ program rawrite nie używa formatu dysku zgodnego z MS-DOS. Program ten znajduje się zazwyczaj w jednym z podkatalogów katalogu głównego na CD-ROM-ie z dystrybucją Linuxa. Obie dyskietki muszą być dyskietkami o wysokiej gęstości (HD, High Density). W niektórych dystrybucjach możesz użyć jednej dyskietki 5,25 i jednej 3,5 – calowej, ale w większości systemów nie jest to zalecane. Dyski muszą być sformatowane za pomocą DOS-owego polecenia format. Dyskietka boot jest dyskiem startowym, więc musisz mieć możliwość wystartowania z niej systemu (innymi słowy, jeśli Twój system startuje tylko z 3,5 – calowej stacji dysków a:, to dysk boot musi być 3,5 – calowy).

Do zapisania plików obrazów na dyskietki należy użyć programu rawrite. Jest to program DOS-owy, który skopiuje zadany plik, blok po bloku, na dyskietkę. Po uruchomieniu program zapyta Cię o nazwę pliku obrazu, który ma zostać przeniesiony na dyskietkę, oraz o nazwę docelowej stacji dysków. Po zakończeniu kopowania pliku obrazu dyskietki są nieczytelne dla systemu MS-DOS. Powinieneś więc je opisać, aby uniknąć pomyłek.

Można również stworzyć dyski boot i root z poziomu systemu UNIX czy Linux. Potrzebne są do tego te same pliki, co w systemie MS-DOS, oraz program użytkowy dd. Jeśli pliki obrazów są skompresowane, rozpakuj je poleceniem

```
gunzip <nazwa_pliku.gz>4
```

Spowoduje ono usunięcie skompresowanej wersji pliku (z rozszerzeniem .gz) i utworzenie wersji rozpakowanej.

Potrzebna będzie również informacja o tym, jak w systemie, w którym pracujesz, nazywa się stacja dysków. Najczęściej jest to /dev/fd0 dla pierwszej stacji i /dev/fd1 dla drugiej, ale może być inaczej (na przykład /dev/rfd0 i /dev/rfd1). Skopij pliki obrazów na dyskietkę następującym poleceniem:

```
dd if=<nazwa_pliku> of=/dev/fd0 obs=18k
```

Program dd dokonuje konwersji formatu plików. Parametry if i of wyznaczają nazwy pliku źródłowego i docelowego. Wartość parametru obs określa wielkość bloku danych wyjściowych (w tym przypadku 18 KB).

Przykładowo, by skopiować pliki scsi oraz color144 na dyskietki w pierwszej stacji dysków, należy wydać następujące polecenia:

```
dd if=scsi of=/dev/fd0 obs=18k  
dd if=color144 of=/dev/fd0 obs=18k
```

⁴ Można również, podobnie jak w systemie MS-DOS, użyć polecenia gzip:
gzip -d <nazwa_pliku.gz>

Dwie utworzone dyskietki pozwalają Ci już uruchomić system Linux w minimalnej konfiguracji.

Krótki przewodnik po instalacji

Ten podrozdział zawiera krótki przewodnik po procedurze instalacyjnej różnych dystrybucji Linuxa (jako modelową przyjęto dystrybucję Red Hat, ale w większości pozostałych przypadków instalacja przebiega podobnie). Informacje w nim zawarte powinny wystarczyć do zainstalowania Linuxa w systemach o prostej konfiguracji sprzętowej, ale jeśli natkniesz się na jakieś problemy w czasie instalacji, powinieneś przejść do bardziej szczegółowego opisu w dalszej części tego rozdziału.

Po wystartowaniu systemu z dysku boot i załadowaniu dysku root (bądź też wystartowaniu komputera z dysku CD-ROM), rozpoczyna się procedura instalacji. Dystrybucja Red Hat Linux zawiera bardzo wygodny program instalacyjny, obsługiwany za pomocą menu, który jest o wiele odporniejszy na pomyłki, niż programy dostarczane z innymi dystrybucjami. W większości przypadków wystarczy czytać pojawiające się na ekranie komunikaty i wybierać opcje odpowiadające konfiguracji sprzętu. Wiele etapów instalacji nie wymaga Twojej interwencji – zostaną one przeprowadzone automatycznie.

Jeśli w trakcie instalowania systemu wystąpi jakiś błąd, na ekranie pojawi się ostrzeżenie i dalszą część instalacji będzie trzeba przeprowadzić ręcznie, ale jest to równie proste jak instalacja automatyczna (również dostępny jest łatwy w obsłudze system menu), choć trwa nieco dłużej.

Jeśli uda Ci się zainstalować Linuxa za pomocą instalacji automatycznej, co jest bardzo prawdopodobne, możesz przejść do następnych rozdziałów tej książki. Dalsza część tego rozdziału omawia bardziej szczegółowo najważniejsze etapy instalacji Linuxa. Jeśli natkniesz się na jakieś problemy, bądź też będziesz chciał zmienić szczegóły konfiguracji systemu, powinieneś przeczytać odpowiednie podrozdziały zawarte w dalszej części tego rozdziału. Przyjrzyjmy się w skrócie poszczególnym etapom instalacji automatycznej.

Instalacja tekstowa i graficzna

Red Hat Linux pozwala na wybór dwóch rodzajów instalacji: w trybie tekstowym i opartej na interfejsie X. Jeżeli wcześniej instalowałeś już kiedyś Linuxa albo inne systemy operacyjne, obie opcje są równie bezpieczne. Interfejs graficzny prezentuje się o wiele lepiej niż tekstowy, ale jeśli nie wiesz, jak używać interfejsu X, albo nie potrafisz skonfigurować systemu X (podając typ myszki i karty graficznej), powinieneś pozostać przy instalacji w trybie tekstowym. Prawdopodobnie użyją jej również weterani Linuxa, gdyż jest bardzo podobna do procedury instalacyjnej znanej z poprzednich wersji systemu – prowadzi użytkownika krok po kroku przez wszystkie etapy instalacji, wyświetlając dokładne informacje o każdym z nich.

Konfigurowanie dysku twardego

Jeśli jeszcze nie używałeś Linuxa albo chcesz założyć nową partycję linuxową, musisz użyć jednego z programów przeznaczonych do dzielenia dysku na partycje. Red Hat oferuje dwa takie programy: `fdisk` oraz `cfdisk`. Program `fdisk` jest podobny do programu o tej samej nazwie przeznaczonego dla systemu DOS – dokładniej omówimy go w podrozdziale „Instalacja partycji linuxowych”.

Program `cfdisk` jest zbliżony do programu `fdisk`, ale pracuje w trybie pełnoekranowym. Wiele osób uważa, że jest łatwiejszy w użyciu niż `fdisk`. Wybór zależy tylko od Ciebie, ponieważ programy te mają takie same możliwości.

Jeśli założyłeś nową partycję dla Linuxa, to aby można było jej używać, musisz ponownie uruchomić komputer (nie zapomnij o zapisaniu tablicy partycji na dysku, gdy wychodzisz z programu `fdisk` lub `cfdisk`!). Po ponownym uruchomieniu systemu rozpoczęź jeszcze raz instalację z dyskietek lub dysku CD-ROM, a nowe partycje zostaną prawidłowo rozpoznawane przez program instalacyjny.

Formatowanie partycji

Po zakończeniu dzielenia dysku na partycje Red Hat próbuje wykryć partycję wymiany (ang. *swap partition*). Program instalacyjny wyświetla okienko zawierające listę wszystkich takich partycji odnalezionych w systemie. Aby sformatować jedną z nich, wybierz ją z listy i naciśnij przycisk `yes`. Formatowanie może zająć nawet dłuższą chwilę, zależnie od wielkości partycji wymiany.

Po tym, jak partycja wymiany zostanie sformatowana i udostępniona dla jądra systemu, program instalacyjny wykryje wszystkie partycje z danymi Linuxa. Zostanie wyświetlona ich lista i również można będzie je sformatować. Jeśli jest to pierwsza instalacja Linuxa, powinieneś skorzystać z tej możliwości.

Konfiguracja sieci Ethernet

Po zakończeniu konfiguracji dysku twardego program instalacyjny daje możliwość zainstalowania sieci Ethernet. Jeśli posiadasz kartę sieciową i chcesz ją skonfigurować podczas instalacji, możesz to teraz zrobić. Jeśli nie posiadasz karty sieciowej albo zamierzasz odłożyć jej instalację na później, po prostu pomiń kilka następnych kroków.

Podczas instalacji Ethernetu musisz podać nazwę komputera i domeny, do której należy Twój system. Następnie należy wprowadzić adres IP oraz maskę podsieci (która jest generowana automatycznie przez program instalacyjny i prawdopodobnie nie wymaga modyfikacji). Mogą pojawić się też inne pytania, np. czy używasz bramki internetowej lub serwera NFS; w większości przypadków odpowiedzi podpowiadane przez program instalacyjny są prawidłowe.

Na pytanie o to, czy używasz bramki internetowej (ang. *gateway*), powinieneś odpowiedzieć twierdząco, gdy używasz osobnego komputera do łączenia się z Internetem lub z inną siecią. Jeśli nie planujesz takiego rozwiązania, odpowiedz przecząco. Jeśli korzystasz z bramki, musisz podać jej nazwę sieciową.

Zostaniesz również zapytany, czy Twój system będzie korzystał z usług serwera nazw (ang. *nameserver*). Jeśli w sieci działa serwer DNS (Domain Name System), który dokonuje konwersji nazw domenowych na adresy IP, powinieneś odpowiedzieć twierdząco i podać jego nazwę sieciową. W przeciwnym przypadku odpowiedz *no*. Można oczywiście dodać obsługę serwera nazw później, po zainstalowaniu systemu.

Konfiguracja myszki

Po informacji o serwerze Metro-X dostarczonym wraz z dystrybucją Red Hat Linux⁵ i wybraniu typu karty graficznej (jeśli nie jesteś pewien, wybierz standardową kartę VGA lub SVGA), program instalacyjny zadaje pytanie o typ podłączonej myszki (nie ma to żadnego związku z serwerem Metro-X – po prostu tak działa program instalacyjny). Wybierz mysz, która jest zainstalowana w systemie, lub model z nią kompatybilny. Należy również podać informację o porcie, do którego jest ona podłączona. W większości przypadków jest to port COM1 (`/dev/ttys0`) lub COM2 (`/dev/ttys1`). Wybierz opcję właściwą dla Twojego systemu; jeśli nie jesteś pewien – wybierz COM1, ponieważ jest to najczęściej spotykana konfiguracja.

Konfiguracja X

Podczas instalacji opartej na interfejsie graficznym, Red Hat próbuje ustalić rodzaj zainstalowanej karty graficznej na samym początku, aby mógł zostać uruchomiony system X. W przypadku instalacji tekstowej, proces ten odłożony jest aż do teraz.

W większości przypadków Red Hat próbuje skonfigurować system X, sprawdzając, jaki typ karty graficznej posiadasz. Prawdopodobnie będziesz miał możliwość wyboru, czy karta ma zostać wykryta automatycznie (ang. *autoprobe*), czy też podasz typ karty ręcznie (czasem jest to konieczne, gdy wykrywanie powoduje zawieszanie się systemu). Jeśli nie wiesz, czy konfiguracja automatyczna zadziała, po prostu spróbuj. W najgorszym przypadku będziesz musiał ponownie uruchomić system. Jeśli automatyczna konfiguracja nie działa prawidłowo, musisz podać wszystkie parametry karty graficznej.

Jeżeli karta zostanie wykryta automatycznie, program instalacyjny wyświetli informacje o rozpoznanym procesorze graficznym i przypuszczalnej ilości pamięci graficznej. W większości systemów dane te są prawidłowe, ale jeśli tak nie jest, możesz je zmienić.

Po zebraniu odpowiedzi na wszystkie pytania program instalacyjny uruchamia system X i rozpoczyna się instalacja oparta o interfejs graficzny.

⁵

Serwer ten nie znajduje się na dołączonym do książki dysku CD-ROM (*przyp. wyd.*).

Wybór instalowanego oprogramowania

W kolejnym etapie instalacji trzeba zdecydować, które z dołączonych do dystrybucji pakietów oprogramowania mają zostać zainstalowane w systemie. Można wybrać dowolną liczbę pakietów, ogranicza ją jedynie ilość dostępnego na dysku twardym miejsca. Po zakończeniu instalacji również istnieje możliwość dodania do systemu dowolnego pakietu oprogramowania lub usunięcia go.

Po wybraniu pakietów oprogramowania program instalacyjny umożliwia instalację tylko niektórych fragmentów pakietów. Taka opcja wymaga Twojej interwencji przy każdym instalowanym pakiecie. Jeśli z niej nie skorzystasz, wybrane pakiety zostaną w całości zainstalowane.

Na koniec program instalacyjny rozpoczęte kopowanie plików na dysk twardy. Na ekranie będą wyświetlane komunikaty o postępach w instalacji programów.

Użycie programu LILO

Po sformatowaniu partycji i skopiowaniu na dysk twardy wybranego oprogramowania wyświetlane jest pytanie, czy do zarządzania uruchamianiem systemu ma być używany program LILO. Jeśli dysk twardy jest w całości przeznaczony dla Linuxa lub jest podzielony pomiędzy systemy DOS i Linux, możesz używać LILO do uruchamiania dowolnego z tych systemów z dysku twardego.

Jeśli używasz innego systemu operacyjnego, jak np. UNIX lub OS/2, nie powinieneś używać programu LILO, a zamiast tego utworzyć dyskietkę startową. Program LILO jest opisany szczegółowo w rozdziale 4. „LILO”.

Podział dysku twardego na partycje

Dyski twarde podzielone są na partycje, czyli obszary przeznaczone dla poszczególnych systemów operacyjnych. Na jednym dysku mogą znajdować się maksymalnie cztery partycje podstawowe (ang. *primary partitions*; czasem nazywa się je również partycjami głównymi), które z kolei mogą być podzielone przez oprogramowanie systemu operacyjnego na dyski logiczne (ang. *logical drives*). Bardziej szczegółowe omówienie zagadnienia partycji znajdziesz w rozdziale 4.

Jeśli używasz Linuxa zainstalowanego na partycji DOS-owej (za pomocą dysku root obsługującego UMSDOS), nie musisz ponownie dzielić dysku na partycje – używane są partycje już istniejące. Ponieważ jednak UMSDOS jest, w porównaniu z linuxowym, kiepskim systemem plików, prawdopodobnie będziesz chciał utworzyć partycje specjalnie dla Linuxa. Aby dowiedzieć się więcej o systemie UMSDOS, przejdź do podrozdziału „Użycie systemu plików UMSDOS”.

Linux do poprawnego działania wymaga założenia dwóch partycji: jednej partycji wymiany⁶ oraz jednej przeznaczonej na oprogramowanie i dane. Partycja wymiany używana jest jako rozszerzenie fizycznej pamięci RAM i nie musi być duża. Natomiast partycja zawierająca system plików powinna być spora, ponieważ ma pomieścić całe oprogramowanie, które chcesz zainstalować. Możesz posiadać kilka partycji z danymi, ale jedna z nich musi być partycją startową (ang. *boot partition*), na której znajdzie się jądro systemu i podstawowe programy użytkowe.

Jeśli na Twoim dysku twardym jest już zainstalowany system operacyjny, musisz na nowo podzielić go na partycje, żeby przydzielić odpowiednią ilość miejsca Linuxowi. Proces ten jest destruktywny, co oznacza, że wszystkie dane zapisane na dysku twardym zostaną bezpowrotnie utracone. Koniecznie wykonaj ich kopię zapasową!

Podziału dysku na partycje dokonać można za pomocą programu `fdisk`. Jego funkcja jest taka sama jak w wersji dla DOS-u, ale obsługuje się go zupełnie inaczej (jest bardziej skomplikowany, ale ma większe możliwości). Wiele systemów UNIX-owych działających na komputerach klasy PC również używa tego programu.



Program `FIPS` działający w systemie MS-DOS umożliwia podział dysku na partycje bez utraty danych, o ile dane nie znajdują się w obszarze nowo tworzonej partycji. `FIPS` dostępny jest w większości węzłów FTP prowadzących Linuxa i z niektórymi dystrybucjami na płytach CD-ROM. Jednak nawet używając takiego programu powinieneś wykonać kopię zapasową ważnych danych.

Przed rozpoczęciem instalacji musisz zdecydować, ile miejsca chcesz przydzielić poszczególnym partycjom, ponieważ każda zmiana pociąga za sobą utratę danych zapisanych na dysku. Wielkość partycji wymiany zależy od ilości dostępnej pamięci RAM, liczby użytkowników i typu oprogramowania używanego w systemie.

Gdy na dysku, oprócz Linuxa, ma znajdować się jeszcze system DOS, musisz odpowiednio wyważyć proporcje. Minimalna wielkość partycji z danymi Linuxa to ok. 20 MB, ale by uruchomić system X, niezbędne staje się ok. 100 MB.

Partycja wymiany

Jak duża powinna być partycja wymiany? Niestety, nie ma jednej odpowiedzi prawidłowej dla wszystkich systemów. Ponieważ jest ona używana jako rozszerzenie pamięci RAM, im więcej RAM-u posiadasz, tym mniejsza partycja wymiany, jest niezbędna do pracy. Wielkość dostępnej dla systemu pamięci operacyjnej jest sumą wielkości fizycznej pamięci RAM i rozmiaru partycji wymiany. Na przykład, jeśli posiadasz 8 MB RAM-u i 16 MB partycji wymiany, Linux będzie zachowywał się tak, jakbyś posiadał 24 MB pamięci operacyjnej.

⁶ W systemach wyposażonych w większą ilość pamięci RAM partycja wymiany nie jest wymagana. Poza tym zamiast partycji możliwe jest użycie pliku wymiany (przyp. *tlum.*).

Linux używa partycji wymiany zapisując na niej strony pamięci RAM, gdy nie są akurat używane i wczytując je na powrót, gdy stają się potrzebne. Dlaczego więc nie stworzyć ogromnej partycji wymiany? Problem polega na tym, że dysk twardy jest tysiące razy wolniejszy niż pamięć RAM. Gdy partycja wymiany jest zbyt duża, system zamiast przyspieszać traci na wydajności.

Możliwe, że partycja wymiany nie jest wcale potrzebna w Twoim systemie. Na przykład, jeśli posiadasz 16 MB RAM-u i nie zamierzasz tworzyć własnych aplikacji czy uruchamiać X, system prawie w ogóle nie będzie używał partycji wymiany, ponieważ wszystkie potrzebne dane zmieszcza się w obszarze 16 MB. Mimo wszystko, nawet wtedy powinieneś ją na wszelki wypadek założyć.

Jeśli używasz systemu X, tworzysz własne aplikacje, lub też używasz jakichś „pamięciożernych” aplikacji (jak programy obsługi baz danych), nie obejdzieś się bez partycji wymiany nawet, gdy posiadasz dużą ilość pamięci RAM. Nawet 16 MB pamięci operacyjnej może okazać się ilością niewystarczającą dla systemu X.

Nie pozbywaj się partycji wymiany całkowicie, chyba, że masz naprawdę ogromną ilość pamięci RAM. Pamiętaj o tym, że brak dostępnej pamięci operacyjnej może doprowadzić do zawieszenia się systemu, co czasem może skończyć się nawet utratą danych.

Zakładanie partycji

Ponieważ wersje programu `fdisk` dla systemów DOS, OS/2, UNIX i Linux różnią się między sobą, nie będziemy szczegółowo opisywać procesu zakładania partycji. Program `fdisk` jest prosty w obsłudze i jeżeli używałeś wcześniej komputerów klasy PC, nie powinieneś mieć z nim żadnych problemów. Pamiętaj, że niszczy on wszystkie dane zapisane na dysku twardym. Zakładanie partycji wymiany i partycji z danymi systemu Linux można przeprowadzić zarówno w systemie DOS, jak i Linux; wersja DOS-owa programu `fdisk` jest nieco łatwiejsza w obsłudze.

Aby założyć partycję linuxową, usuń najpierw istniejące partycje (chyba że chcesz pozostawić je tak, jak są). Jeśli planujesz używać również DOS-u, powinieneś on znajdować się na pierwszej partycji, ponieważ tylko wtedy potrafi się poprawnie uruchomić (istnieje co prawda kilka sposobów na uruchomienie DOS-u z innej niż pierwsza partycji dysku trwałego za pomocą programu LILO, ale mimo wszystko lepiej pozostawić go na pierwszej partycji).

Następnie trzeba utworzyć dysk startowy systemu DOS, z którego będzie można sformatować nową partycję i przenieść na nią pliki systemowe. Zakładając, że stacja dysków nazywa się `a:`, powinieneś w tym celu użyć następującego polecenia systemu DOS:

```
format a: /s
```

Opcja `/s` powoduje przeniesienie na dyskietkę jądra systemu MS-DOS. Następnie skopiuj na tę dyskietkę programy użytkowe, takie jak `fdisk`, `format`, `sys` oraz `chkdsk`. Powinieneś także skopiować jakiś edytor tekstów, np. `edit`, oraz pliki `config.sys` i `autoexec.bat` pochodzące z Twojego systemu. Tak spreparowana dyskietka pozwala

sformatować nową partycję dla systemu DOS. Jeśli jest to nowa partycja DOS-owa, możesz również po prostu uruchomić od nowa program instalacyjny systemu DOS.

Oto co powinieneś zrobić, jeśli chcesz usunąć istniejącą partycję DOS-ową i na jej miejsce utworzyć mniejszą (po wykonaniu kopii zapasowej danych zapisanych na dysku twardym):

1. usuń istniejącą partycję systemu DOS,
2. utwórz nową partycję podstawową (o mniejszym rozmiarze) jako pierwszą partycję,
3. uaktywnij partycję DOS-ową,
4. uruchom system z dyskietki,
5. sformatuj partycję DOS-ową i skopiuj na nią jądro systemu DOS (`COMMAND.COM`),
6. przywróć pliki ze stworzonej wcześniej kopii zapasowej (ten krok możesz odłożyć na później).

Następnie załóż linuxową partycję wymiany o odpowiednim rozmiarze. Możesz to zrobić pod kontrolą systemu DOS⁷ lub po uruchomieniu Linuxa z dysków boot i root. Dla potrzeb tego rozdziału przyjęliśmy, że nową partycję zakładasz w systemie DOS, choć proces ten w obu przypadkach przebiega podobnie.

Większość wersji programu `fdisk` pozwala na podanie wielkości zakładanej partycji w megabajtach, przeliczając ją samodzielnie na ilość potrzebnych sektorów. Ustal wymaganą wielkość partycji wymiany, ale nie ustawiaj tej partycji jako aktywnej i nie formatuj jej. Może zawierać się ona na partycji rozszerzonej, ale lepiej, by była partycją podstawową (o ile na dysku można utworzyć jeszcze jedną partycję podstawową).

Na koniec utwórz partycję, na której przechowywane będą dane Linuxa, o rozmiarze dopasowanym do Twoich wymagań, lub też po prostu zajmującą pozostałą część dysku twardego. Tej partycji również nie należy formatować ani aktywować. Po uruchomieniu programu instalacyjnego Linuxa powinieneś zidentyfikować i sformatować założone partycje danych i wymiany.

Użycie systemu plików UMSDOS

UMSDOS (ang. *UNIX in MS-DOS*) umożliwia zainstalowanie Linuxa na istniejącej partycji DOS-owej (pozwala na to tylko starsze wersje systemu Linux). W takim przypadku Linux jest zmuszony do używania DOS-owego systemu plików, co ogranicza jego wydajność w porównaniu z systemem zainstalowanym na oddzielnej partycji. Z drugiej strony, pozwala to obejść problemy związane z podziałem dysku na partycje i formatowaniem ich. Jest to dobry sposób na szybką i bezproblemową instalację w przypadku,

⁷ Jeśli partycja wymiany zakładana jest z poziomu systemu MS-DOS, warto później zmienić jej identyfikator typu za pomocą linuxowego programu `fdisk`, ponieważ nie pozwala na to wersja DOS-owa tego programu (*przyp. tłum.*).

gdy chcesz tylko poeksperimentować z Linuxem przed zainstalowaniem jego pełnej wersji.

Musisz zdawać sobie sprawę, że UMSDOS nie pozwala na używanie jednocześnie DOS-u i Linuxa. Tworzy on linuxowy system plików na partycji DOS-owej, choć jest on zmodyfikowany tak, by można było używać długich nazw plików oraz praw dostępu niezbędnych do pracy Linuxa. Przy uruchamianiu systemu można wybrać, czy chcesz uruchomić Linuxa, czy system DOS. W DOS-ie nie można używać długich nazw plików i choć będzie możliwe poruszanie się po katalogach linuxowych, nazwy plików mogą być wyświetlane nieprawidłowo, ze względu na ich skrócenie.

Jedynym ograniczeniem wynikającym z użycia UMSDOS jest spadek wydajności systemu, spowodowany faktem, że DOS-owy system plików nie jest zaprojektowany równie dobrze jak system linuxowy. Zwykle nie jest to wielkim problemem, o ile nie używasz aplikacji intensywnie korzystających z systemu plików, takich jak na przykład X Window lub kompilatory. UMSDOS może być dobrym punktem wyjścia, a gdy już zdecydujesz się na zainstalowanie Linuxa na stałe, musisz tylko zrobić kopię zapasową plików linuxowych, zainstalować Linuxa na osobnej partycji, a następnie przywrócić skopiowane pliki.

Jeśli chcesz używać UMSDOS, musisz podczas instalacji systemu podjąć kilka dodatkowych kroków. Powinieneś, podobnie jak przy zwykłej instalacji, użyć dysków boot i root, ale będziesz potrzebował obrazu dysku root obsługującego system UMSDOS. Zazwyczaj odpowiedni plik nazywa się `umsds144` lub `umsds12`.

Gdy program instalacyjny zapyta, na której partycji chcesz zainstalować linuxowy system plików, podaj partycję DOS-ową. Dalsza część instalacji przebiega tak samo, jak w przypadku przeznaczenia dla Linuxa osobnej partycji.

Instalacja partycji linuxowych

Proces instalacji rozpoczyna się w momencie, gdy uruchamiasz system z dysku boot. Po załadowaniu jądra systemu odpowiedni komunikat informuje o konieczności zmiany dyskietki na dysk root. Po jej odczytaniu rozpoczęcie działanie programu instalacyjnego, albo też zostaniesz zapytany o identyfikator użytkownika. Zaloguj się jako `root`; na tym etapie nie jest wymagane żadne hasło (ponieważ nie zostało ono jeszcze podane).

Pierwszym krokiem instalacji jest podział dysku twardego na partie programem `fdisk` – to masz już za sobą. Jeśli posiadasz więcej niż jeden dysk twardy, możesz umieścić partie linuxowe na każdym z nich. Partycja startowa DOS-u musi być pierwszą partycją na pierwszym dysku twardym. Jeżeli chcesz uruchamiać Linuxa bezpośrednio, umieść partycję z jego systemem plików na pierwszym dysku twardym. Można też utworzyć partie z danymi na innych dyskach. Partycja wymiany może znajdować się na dowolnym dysku, ale zalecane jest, aby znajdowała się razem z podstawowym systemem plików na pierwszym dysku twardym.

Program fdisk

Wersja programu `fdisk` przeznaczona dla systemu Linux różni się od wersji DOS-owej, więc aby uniknąć pomyłek, dokładnie czytaj wszystkie wyświetlane komunikaty. Program ten uruchamia się w ten sam sposób, jak jego DOS-owy odpowiednik. Jeśli nie wyszczególnisz nazwy dysku, `fdisk` przyjmie, że chodzi Ci o pierwszy dysk w systemie. Możesz oczywiście podać, który dysk zamierzasz dzielić na partycje, na przykład jeśli ma to być drugi dysk IDE, powinieneś wydać polecenie:

```
fdisk /dev/hdb
```

Dyski IDE, ESDI i RLL nazywają się `/dev/hda`, `/dev/hdb` itp., natomiast dyski SCSI – `/dev/sda`, `/dev/sdb` itd. W systemie może być do siedmiu dysków SCSI, więc ostatni z nich nazywałby się `/dev/sdg` (niektóre karty kontrolerów umożliwiają obsługę jeszcze większej liczby urządzeń!).



Nie powinieneś używać linuxowego programu `fdisk` do tworzenia partycji dla innych systemów operacyjnych. Partycje dla DOS-u utwórz DOSowym programem `fdisk`, ponieważ utworzone przez wersję linuxową nie zostaną prawidłowo rozpoznane.

Jak wspomniano wcześniej, polecenia linuxowego programu `fdisk` różnią się od poleceń w wersji DOS-owej. Najważniejsze z nich to:

- d** usunięcie istniejącej partycji,
- l** wyświetlenie wszystkich dostępnych typów partycji,
- n** utworzenie nowej partycji,
- p** wyświetlenie aktualnej zawartości tablicy partycji,
- q** wyjście z programu bez zapisywania zmian,
- t** zmiana typu partycji,
- v** weryfikacja tablicy partycji,
- w** zapisanie zmian i wyjście z programu.

Powinieneś najpierw przejrzeć aktualną zawartość tablicy partycji, aby upewnić się, że odpowiednie wpisy są prawidłowe. Jeśli posiadasz partycję systemu DOS, powinna być ona widoczna. Jeśli założyłeś wcześniej partycje linuxowe pod kontrolą systemu DOS, one również powinny być widoczne, choć będą miały niewłaściwy typ.

Zakładanie partycji linuxowych

Aby założyć partycję wymiany, użyj polecenia `fdisk`, a następnie podaj numer sektora, od którego ma się ona rozpoczynać. Zazwyczaj powinna zaczynać się zaraz po partycji systemu DOS (czy też innego, używanego przez Ciebie systemu operacyjnego). Linuxowy program `fdisk` pozwala na podanie wielkości partycji zarówno przez wprowadzenie numeru sektora końcowego, jak i przez podanie jej rozmiaru w megabajtach. Drugi sposób jest o wiele wygodniejszy – rozmiar partycji należy wówczas wprowadzić w formacie `+XXM`, gdzie XX to liczba megabajtów, np. `+16M`. Można również podać wielkość partycji w kilobajtach.



Większość starych BIOS-ów obsługuje maksymalnie 1024 cylindry na jednym dysku twardym. Może się zdarzyć, że nie będziesz w stanie utworzyć partycji lub systemów plików w obszarze poza cylindrem 1023 (numerując od zera). Niektóre inne systemy operacyjne, jak na przykład SCO UNIX, pozwalają ominąć to ograniczenie. Linux obsługuje partycje leżące poza obszarem pierwszych 1024 cylindrów, ale nie może być z nich uruchamiany. Jeśli posiadasz dysk mający więcej niż 1024 cylindry, upewnij się, że podstawowa partycja Linuksa kończy się przed cylindrem 1023. Możesz stworzyć inne partycje leżące poza tym cylindrem i zamontować je w głównym systemie plików.

Program `fdisk` zapyta, czy chcesz utworzyć partycję podstawową (ang. *primary*), czy rozszerzoną (ang. *extended*). W przypadku partycji podstawowej musisz podać jej numer (od 1 do 4, pamiętając o tym, że partycja DOS-owa musi mieć numer 1). W większości przypadków powinieneś zakładać tylko partycje podstawowe, chyba że posiadasz naprawdę duży dysk twardy. Partycje rozszerzone przeznaczone są do zakładania na nich dysków logicznych. Partycja rozszerzona (ang. *extended*) nie ma nic wspólnego z linuxowym systemem plików o podobnej nazwie – Extended Filesystem.

Po założeniu nowej partycji należy określić jej typ. Niektóre wersje programu `fdisk` zapytają Cię o to zaraz po założeniu partycji, ale w innych musisz określić typ samodzielnie. Polecenie `l` powoduje wyświetlenie wszystkich dostępnych typów partycji i ich liczbowych identyfikatorów. Wybierz ten odpowiadający linuxowej partycji wymiany (Linux Swap Space), a następnie sprawdź tablicę partycji – w odpowiedniej pozycji powinien pojawić się właściwy rozmiar i identyfikator typu. Tak naprawdę, sam system Linux nie przejmuje się zbytnio typem partycji, ale inne systemy operacyjne i niektóre programy użytkowe wykorzystują te informacje, warto więc ustawić go poprawnie. Ułatwia to również późniejszą identyfikację istniejących partycji.

Następnie powinieneś utworzyć w taki sam sposób partycję dla danych systemu Linux. Jeśli chcesz do tego celu użyć całej pozostałej objętości dysku (jest to często spotykana konfiguracja – na dysku znajdzie się wtedy partycja systemu DOS, partycja wymiany i partycja z danymi systemu Linux), wpisz numer ostatniego sektora dysku twardego (`fdisk` poda Ci zakres dostępnych sektorów). Po utworzeniu partycji nie zapomnij ustawić jej identyfikatora na numer odpowiadający typowi „Linux native”.

Zanotuj gdzieś rozmiar partycji wymiany (w blokach) – informacji tej będziesz potrzebował później. Możesz ją znaleźć w tablicy partycji.

Po założeniu partycji linuxowych zapisz zmiany i zakończ działanie programu `fdisk`. Jeśli nie zapiszesz zmian, cały proces będziesz musiał powtórzyć od początku.

Udostępnianie partycji wymiany programowi instalacyjnemu

Program instalacyjny wymaga sporej ilości pamięci RAM. Jeśli w Twoim systemie jest jej mniej niż 4 MB, możesz podczas instalacji natknąć się na problemy, chyba że użyjesz jądra systemu obsługującego partycję wymiany (jeśli posiadasz 4 MB pamięci, partycja ta powinna mieć wielkość co najmniej 8 MB). Jeżeli przy instalacji Linuxa pojawiają się komunikaty o błędach pamięci, oznacza to właśnie zbyt małą ilość dostępnego RAM-u i konieczność użycia partycji wymiany.



W przypadku, gdy uruchomłeś partycję wymiany i nadal otrzymujesz komunikaty o błędach pamięci, powinieneś ją powiększyć. Najlepiej zrobić to od razu, ponieważ każda zmiana w tablicy partycji przeważnie pociąga za sobą konieczność przeinstalowywania całego systemu.

Jeśli posiadasz niewiele pamięci RAM, powinieneś załączyć obsługę partycji wymiany w procesie instalacji. Nawet jeśli masz dość pamięci, nie ma powodu, dla którego nie miałbyś zrobić tego w tej chwili. Wydaj następujące polecenie (*partycja* to nazwa partycji wymiany, a *rozmiar* to jej wielkość w blokach):

```
mkswap -c partycja rozmiar
```

Rozmiar partycji zapisałeś wcześniej, przy jej zakładaniu; jeśli tego nie zrobiłeś, uruchom program *fdisk* i sprawdź odpowiednie dane w tablicy partycji.

Przykładowo, jeżeli założyłeś partycję wymiany jako drugą partycję podstawową na pierwszym dysku twardym (nie SCSI) i ma ona wielkość 13565 bloków, powinieneś wydać polecenie:

```
mkswap -c /dev/hda2 13565
```

Opcja *-c* powoduje sprawdzenie, czy w obszarze partycji wymiany nie występują uszkodzone bloki. Wydłuża to nieco czas potrzebny na wykonanie tego polecenia, ale uszkodzony blok na partycji wymiany może spowodować zawieszenie się całego systemu, więc warto poświęcić chwilę na jej sprawdzenie. Jeżeli *mkswap* wykryje jakieś błędy, wygeneruje odpowiedni komunikat. Można go zignorować, ponieważ uszkodzone bloki nie będą używane. Jeżeli jednak jest ich zbyt dużo, może to oznaczać poważniejszy problem z dyskiem twardym.

Kiedy partycja jest już sformatowana, należy ją udostępnić dla jądra systemu poleciem *swapon*. Jako parametr powinieneś podać nazwę partycji wymiany, choć nie we wszystkich wersjach systemu jest to bezwzględnie wymagane. Polecenie to może wyglądać na przykład tak:

```
swapon /dev/hda2
```

Procedurę formatowania i załączania powinieneś powtórzyć dla każdej partycji wymiany, jeśli założyłeś ich więcej niż jedną. Natychmiast po wywołaniu polecenia `swapon` jądro systemu Linux zaczyna używać partycji wymiany jako rozszerzenia pamięci RAM.

Tworzenie partycji z systemem plików

Po skonfigurowaniu i uruchomieniu partycji wymiany przyszła pora na skonfigurowanie partycji, które zawierać będą dane systemu Linux. Etap ten przeprowadzany jest w niektórych dystrybucjach automatycznie, ale czasem musisz poradzić sobie sam. Ten podrozdział wyjaśni Ci, jak to zrobić.

Założyłeś już na dysku twardym partycję typu Linux native. Teraz, używając polecenia `mke2fs` (ang. *make filesystem*), możesz utworzyć na niej system plików. Dokładna postać tego polecenia zależy od tego, jaki typ systemu plików zamierzasz założyć. Najpopularniejszy jest Second Extended Filesystem, prawdopodobnie ze względu na jego ogromną elastyczność (nazwa może być nieco myląca – nie ma on nic wspólnego z partycjami rozszerzonymi, czyli po angielsku *extended*). Aby stworzyć system plików tego typu, należy wydać następujące polecenie (partycja oznacza nazwę partycji, a rozmiar – jej wielkość w blokach):

```
mke2fs -c partycja rozmiar
```

Dla przykładu, aby założyć system plików o rozmiarze 162344 bloków na partycji `/dev/hda3`, należy wydać polecenie:

```
mke2fs -c /dev/hda3 162344
```

Przy podawaniu rozmiaru partycji upewnij się, że jest to rozmiar w blokach, a nie w sektorach lub cylindrach. Zła wartość spowoduje albo wygenerowanie komunikatu o błędzie, albo sytuację, gdy tylko część partycji będzie dostępna.

Program `mke2fs` sprawdzi, czy partycja `/dev/hda3` nie zawiera uszkodzonych bloków (opcja `-c`), po czym założy na niej system plików. Jeśli jest to partycja o dużym rozmiarze, proces ten może potrwać nawet kilkanaście minut, nie powinieneś jednak rezygnować z opcji `-c`, chyba że masz pewność, że na dysku twardym nie występują żadne błędy.

Oprócz systemu Second Extended Filesystem, Linux może być też również zainstalowany z takimi systemami plików, jak Xia, Extended Filesystem oraz Minix Filesystem. System Xia jest równie dobry jak Second Extended, ale jest mniej popularny. Extended Filesystem to starsza wersja systemu Second Extended Filesystem, natomiast Minix jest kompatybilny z systemem plików systemu operacyjnego Minix (który został wyparty przez Linuxa). Możesz je utworzyć za pomocą następujących poleceń:

- `Extended – mke2fs`,
- `Minix – mkfs_minix`,
- `Xia – mkxfs`.

Wszystkie te polecenia mają składnię identyczną jak polecenie `mke2fs`, używane do tworzenia systemu Second Extended Filesystem. Maksymalny rozmiar systemu plików typu Minix to 64 MB.

Żadne z powyższych poleceń nie powoduje formatowania systemu plików; formatowanie możesz wykonać w procesie instalacji.

Instalacja Linuxa

Po założeniu i sformatowaniu partycji oraz utworzeniu systemu plików można przystąpić do instalacji oprogramowania. Ten etap również bywa zautomatyzowany, zależnie od programu instalacyjnego dołączonego do dystrybucji. Większość wersji Linuxa zawiera program użytkowy o nazwie `setup`, który ułatwi Ci instalację odpowiednich pakietów. By go uruchomić, po prostu wydaj polecenie

```
setup
```

Jeśli używasz dysku root w wersji `color`, program instalacyjny będzie posiadał pełnoekranowy interfejs w stylu okienkowym. Pozostałe wersje oferują interfejs tekstowy. W obu wersjach programy instalacyjne wykonują dokładnie te same zadania. Jednak niektórzy użytkownicy unikają wersji kolorowej, ponieważ odpowiadanie na wszystkie pytania w niej zadawane trwa nieco dłużej, a literówki powstałe przy wprowadzaniu danych nie dają się łatwo poprawić.

Którkolwiek wersję dysku root wybrałeś, zawsze należy dokładnie czytać informacje wyświetlane na ekranie. Choć standardowe odpowiedzi podsuwane przez program instalacyjny są zazwyczaj prawidłowe, czasem zdarza się konieczność zmiany oferowanych ustaleń.

Program instalacyjny wymaga podania różnych informacji. Można pozwolić mu instalować wszystkie dołączone pakiety oprogramowania bez konieczności potwierdzania, ale możliwości tej powinieneś użyć tylko wtedy, gdy dokładnie wiesz, co znajdziesz się na dysku twardym. Jeśli instalujesz Linuxa po raz pierwszy albo chcesz wybrać instalowane oprogramowanie czytając jego opis, wybierz opcję instalacji interaktywnej (ang. *verbose*). Spowoduje to wyświetlanie wszelkich informacji i pozwoli lepiej kontrolować proces instalacji.

Należy również podać, z jakiego źródła chcesz instalować oprogramowanie. Jeśli jest to dysk CD-ROM, powinien on zostać uaktywniony podczas startu systemu, o ile wybrane zostały właściwe sterowniki. Wybierz instalowanie z dysku CD-ROM; może pojawić się pytanie o rodzaj używanego napędu CD-ROM. Wybierz odpowiednią opcję (lub tę najbliższą prawdy). Jeśli źródłem oprogramowania jest partycja dysku twardego (linuxowa lub DOS-owa), musisz podać jej nazwę i ścieżkę prowadzącą do katalogu zawierającego dystrybucję.

Nastecną informacją, którą musisz podać, jest nazwa partycji docelowej. Najprawdopodobniej będzie to partycja linuxowa, którą dopiero co stworzyłeś, wpisz więc jej nazwę.

Może pojawić się pytanie, czy chcesz sformatować tę partycję. Powinieneś odpowiedzieć twierdząco (polecenie `mkfs` ani żadna z jego odmian nie formatuje partycji).

W końcu Linux wyświetli listę zestawów dysków, które można zainstalować – wybierz te, które są Ci potrzebne. Na koniec sprawdź jeszcze raz, czy wybrałeś wszystkie niezbędne zestawy i pozwól Linuxowi rozpocząć instalowanie. Dalej postępuj zgodnie z instrukcjami podawanymi na ekranie. Jeśli instalujesz system z dyskietek, co jakiś czas będziesz musiał je wymieniać.

Program instalacyjny zapyta również, czy chcesz utworzyć dysk startowy. Taki dysk pozwala na uruchomienie systemu w dowolnym momencie; jest przydatny szczególnie wtedy, gdy z jakiegoś powodu zawiedzie normalny proces uruchamiania systemu. Zawsze warto mieć go pod ręką. Nie jest on tym samym, co dysk boot z którego uruchomiłeś instalację (on przydatny jest tylko wtedy, gdy instalujesz system od początku).

Konfiguracja startu systemu

Ostatnim etapem instalacji jest skonfigurowanie dysku, z którego system będzie uruchamiany. Zazwyczaj Linux uruchamiany jest za pomocą programu zwanego `LILO` (Linux Loader). Potrafi on uruchomić system na kilka różnych sposobów, w zależności od tego, czy chcesz używać innych systemów operacyjnych, czy też nie. Większość użytkowników używa `LILO`, żeby uruchamiać Linuxa mając jednocześnie możliwość uruchomienia w razie potrzeby systemu DOS.

Instalacja `LILO` jest prosta – wszystkie potrzebne informacje są wyświetlane na ekranie, ale istnieje kilka drobiazgów, z których należy zdawać sobie sprawę. Dlatego programowi temu poświęcony jest cały następny rozdział, wyjaśniający zasadę działania `LILO` i pozwalający na jego prawidłową instalację. Jeśli jesteś niecierplwy, wybierz opcje podpowiadane przez program instalacyjny, ale nie pozwól na zapisanie informacji na główny sektor startowy (ang. *Master Boot Record*) dysku twardego. Jeśli to zrobisz, możesz mieć pewne problemy z uruchomieniem systemu DOS. Można za to bezpiecznie zapisać te informacje do sektora startowego (ang. *boot sector*) partycji linuxowej, a następnie ustawić aktywną partycję za pomocą programu `fdisk`.

Jeśli jednak nie jesteś pewny, czy chcesz instalować `LILO`, po prostu na razie zignoruj ten program. Masz przecież dyskietkę startową, z której możesz uruchomić system, a gdy lepiej zrozumiesz zasadę działania `LILO`, łatwo będzie Ci poprawnie go zainstalować.

Na koniec pozostało tylko uruchomić ponownie komputer z dysku startowego lub za pomocą `LILO`, jeśli go zainstalowałeś. Jeżeli system startuje poprawnie, możesz już używać Linuxa. Jeśli nie, obserwuj pilnie wyświetlane komunikaty i sprawdź jeszcze raz proces instalacji, by wykryć, który jego etap zakończył się niepowodzeniem. Dopóki posiadasz dysk startowy, powinieneś mieć możliwość uruchomienia systemu bez żadnych problemów.

Przeglądanie zainstalowanego oprogramowania

Po zainstalowaniu Linuxa można usunąć lub dodać niektóre zestawy dysków i inne oprogramowanie. Można również sprawdzić, czy programy zawarte w danym zestawie dysków zostały zainstalowane prawidłowo. Istnieje kilka programów użytkowych pozwalających zrobić to w wygodny sposób, najpopularniejszy z nich nazywa się `pkgtool`.

Po wpisaniu polecenia `pkgtool` wyświetcone zostanie menu pozwalające zainstalować nowe oprogramowanie, usunąć istniejące, lub też sprawdzić, które pliki z pakietu znajdują się w systemie.

Jeżeli chcesz obejrzeć zawartość pakietu, wybierz z menu głównego programu `pkgtool` opcję `view`, a następnie wybierz z listy interesujący Cię pakiet. Lista ta powinna zawierać wszystkie zestawy dysków, które wybrałeś podczas instalacji, oraz dodatkowe oprogramowanie zainstalowane później.

Czasem wyświetlanie listy zainstalowanego oprogramowania trwa dłuższą chwilę – bądź cierpliwy. Lista wyświetlana przez `pkgtool` zawiera również krótkie opisy programów i spis wszystkich plików wchodzących w skład pakietu.

Rozwiązywanie problemów

Podczas instalowania i konfigurowania Linuxa może wystąpić wiele różnego rodzaju problemów, ale większość z nich łatwo rozpoznać dzięki komunikatom o błędach. Z niektórymi, częściej wystającymi problemami łatwo jest również sobie poradzić, przyjrzymy się im więc bliżej.

Instalacja oprogramowania

Podczas instalowania Linuxa możesz natknąć się na kilka problemów. Jeśli otrzymujesz komunikat `device full` (dysk pełny), oznacza to, że na dysku docelowym zabrakło wolnego miejsca. W takim przypadku powinieneś powiększyć partycję linuxową, podzielić instalację pomiędzy kilka partycji, albo też zainstalować mniejszą liczbę składników. Jeśli komunikat ten pojawia się już przy instalacji podstawowego systemu, w grę wchodzi tylko pierwsze rozwiązanie.

Takie błędy, jak `read error` (błąd odczytu), `file not found` (nie odnaleziono pliku) i `tar: read error` (komunikat programu tar o błędzie odczytu), sygnalizują problem z nośnikiem albo niekompletny zestaw dysków. Zazwyczaj występują one tylko przy instalacji z dyskietek i oznaczają uszkodzenie jednej z nich. W takim przypadku możesz tylko wymienić uszkodzoną dyskietkę na nową.

Dysk twardy i kontroler dysku twardego

Podczas uruchamiania systemu Linux wypisuje wiele komunikatów. Jednym z najważniejszych jest informacja o sprawdzeniu partycji. Może ona wyglądać na przykład tak:

```
Partition check:  
hda: hda1 hda2 hda3  
hdb: hdb1 hdb2
```

W tym przypadku, pierwszy dysk (nie SCSI) podzielony został na trzy partycje, a drugi na dwie. Oczywiście w Twoim systemie będzie prawdopodobnie inaczej. Jeśli informacje o partycjach nie są wyświetlane, oznacza to, że albo kontroler nie został prawidłowo rozpoznany, albo też dyski twarde nie są dostępne. Problem ten może mieć różne przyczyny. Co należy zrobić w takim przypadku? Oto kilka porad.

- υ Sprawdź okablowanie wewnątrz komputera. Taśma dysku twardego powinna łączyć sterownik z każdym z dysków twardych. Upewnij się, że jest ona włożona w dobrą stronę (czerwony pasek na taśmie odpowiada nóżce 1 gniazda).
- υ Sprawdź, czy do dysku twardego podłączony jest przewód zasilający. Bez tego dysk nie będzie się obracał i nawet Linux nie będzie miał z niego pożytku.
- υ Sprawdź tablicę partycji, aby upewnić się, czy prawidłowo utworzyłeś partycję linuxową.

Jeśli dysk twardy nie działa z systemem Linux, ale działa prawidłowo w DOS-ie, winę prawdopodobnie ponosi sterownik zawarty w jądrze systemu. Niektóre z kontrolerów IDE zachowują się niezbyt prawidłowo (nie spełniają wszystkich norm zawartych w standardzie), w związku z czym sterownik zawarty w jądrze systemu nie potrafi się z nimi porozumieć. Spróbuj wybrać inną wersję jądra i sprawdź, czy problem został rozwiązany. Jeśli używasz jądra obsługującego SCSI i kontroler oraz dyski SCSI nie zostały prawidłowo rozpoznane, spróbuj użyć dostarczonych wraz ze sterownikiem programów użytkowych. Może się na przykład zdarzyć, że urządzenie ma niewłaściwie przypisany identyfikator SCSI.

Konflikty pomiędzy urządzeniami

Najczęściej spotykanym problemem jest niewłaściwe rozpoznawanie urządzeń podłączonych do komputera. Może on wystąpić w przypadku dysku CD-ROM, karty sieciowej, a nawet dysku twardego. Jest to zazwyczaj spowodowane konfliktami w ustawieniach numerów używanych przerwań IRQ, kanałów DMA (bezpośredni dostęp do pamięci) lub ustawień adresów I/O. Jeśli dwa urządzenia korzystają z tych samych ustawień, Linux i BIOS nie mogą ich rozpoznawać i komunikować się z nimi prawidłowo.

Jednym z symptomów takiego problemu może być zawieszanie się systemu w czasie, gdy próbuje on wykryć dołączone urządzenia, jak wyjaśniają informacje wyświetlane przez system Linux podczas uruchamiania. Jeśli na przykład wypisuje on komunikat o próbie inicjalizacji karty sieciowej i nigdy nie przechodzi poza ten punkt, prawdopo-

dobnie karta sieciowa jest w konflikcie z innym urządzeniem (uszkodzenia kart są raczej rzadkie i przeważnie nie powodują zawieszania się systemu).

Aby sprawdzić, czy w systemie nie występuje konflikt sprzętowy, powinieneś uruchomić jakiś program diagnostyczny pracujący w systemie DOS, na przykład `MSD` lub `Norton Info`. Takie programy potrafią pokazać aktualne ustawienia numerów przerwań IRQ, kanałów DMA i adresów I/O, pozwalając na wykrycie konfliktów. Mogą one być też przydatne do wyszukiwania dostępnych ustawień.

Innym sposobem rozwiązywania konfliktu sprzętowego jest sprawdzanie konfiguracji każdego urządzenia z osobna. Zazwyczaj konflikty powodowane są przez karty sieciowe, dźwiękowe, sterowniki napędów taśmowych, karty graficzne i inne karty. Do ustawienia zasobów używanych przez wiele z nich używa się zworek lub mikroprzełączników, powinieneś to sprawdzić w ich dokumentacji. Pomóc w wyizolowaniu problemu może wyjęcie kart, które nie są niezbędne (jak karta dźwiękowa), i ponowne uruchomienie systemu. Jeśli teraz system uruchamia się prawidłowo, winowajca został zlokalizowany.

Inne problemy wystąpić mogą w przypadku sterowników SCSI, dla których wymagane są specjalne ustawienia jądra systemu. Niektóre wersje jądra zostały skompilowane z domyślnymi ustawieniami dla kontrolerów czy dysków, gdy zaś ustawienia te zostaną zmienione, jądro systemu zawiesza się. Jest to dość częsty problem z wersjami jądra kompilowanymi dla niestandardowych urządzeń. Rozwiążanie można czasem znaleźć czytając dołączoną do jądra dokumentację.

Najczęściej spotykane w komputerach PC urządzenia (porty szeregowe COM, porty równoległe LPT i stacje dysków) oraz używane przez nie zazwyczaj przerwania IRQ, adresy I/O oraz kanały DMA, zebrane są w tabeli 3.1. Są to wartości domyślne, ale w wielu systemach mogą być zmienione przez użytkowników. Ponieważ w systemie DOS zwykle obsługiwane są tylko dwa porty szeregowe, porty trzeci i czwarty dzielą z nimi przerwania IRQ, ale mają różne adresy I/O. Obie stacje dysków używają tego samego adresu I/O, przerwania IRQ oraz kanału DMA.

Tabela 3.1. Najczęściej używane przez komputery PC urządzenia oraz ich IRQ, DMA i adresy I/O

Urządzenie	przerwanie IRQ	kanał DMA	adres I/O (szesnastkowo)
COM 1 (/dev/ttys0)	4	brak	3F8
COM 2 (/dev/ttys1)	3	brak	2F8
COM 3 (/dev/ttys2)	4	brak	3E8
COM 4 (/dev/ttys3)	3	brak	2E8
LPT 1 (/dev/lp0)	7	brak	378 – 37F
LPT 2 (/dev/lp1)	5	brak	278 – 27F
Stacja dysków a: (/dev/fd0)	6	2	3F0 – 3F7
Stacja dysków b: (/dev/fd1)	6	2	3F0 – 3F7

Karty sieciowe, sterowniki SCSI, karty dźwiękowe, graficzne i inne urządzenia muszą mieć przypisane różne numery przerwań IRQ, kanałów DMA i adresy I/O. Warunek ten może być trudny do spełnienia w mocno rozbudowanych systemach. Jeśli chcesz zdobyć więcej informacji na temat dostępnych zasobów sprzętowych oraz potencjalnych konfliktów, przejrzyj dokumentację sprzętu.

Problemy z urządzeniami SCSI

SCSI jest jednym z najbardziej elastycznych, ale i najtrudniejszych w konfiguracji (ze względu na możliwość wystąpienia konfliktów) interfejsów. Linux zazwyczaj dobrze radzi sobie z wyświetlaniem informacji o problemach z urządzeniami SCSI, ale mogą one nie wskazywać prawdziwej przyczyny konfliktu.

W tabeli 3.2 zebrane są najczęstsze błędy SCSI i ich najbardziej prawdopodobne przyczyny. Znajdź komunikat najbardziej zbliżony do tego, który generowany jest przez Linuxa, a następnie podejmij odpowiednie kroki.

Tabela 3.2. Najczęstsze błędy SCSI i ich prawdopodobne przyczyny

Błąd	Prawdopodobna przyczyna
Urządzenie SCSI ma przypisane wszystkie możliwe identyfikatory	Jedno lub więcej urządzeń ma przypisany ten sam numer ID co kontroler. Sprawdź i zmień numery ID. Kontroler powinien mieć numer 7.
Sense error	Powodem jest najczęściej brak terminatora lub jego złe działanie. Sprawdź, czy oba końce łańcucha SCSI są prawidłowo zakończone. Przyczyną problemów może być też uszkodzony kabel.
Timeout error	Problem spowodowany najczęściej konfliktem DMA, IRQ lub adresu I/O. Tego typu problemy omówione są szerzej w poprzednim podrozdziale.
SCSI adapter not detected	BIOS urządzenia SCSI został wyłączony lub sterownik SCSI nie został wykryty przez jądro systemu
Cylinders beyond 1024	Twój dysk ma więcej niż 1024 cylindry, a BIOS nie obsługuje tak dużych dysków. Linux potrafi używać cylindrów o numerach powyżej 1024, ale nie może być uruchamiany z partycji obejmującej cylindry powyżej 1024.
CD-ROM drive not recognized	Niektóre napędy CD-ROM wymagają, aby przy uruchamianiu systemu był załadowany dysk, tylko wtedy mogą być rozpoznane prawidłowo. Włóz płytę CD i uruchom ponownie komputer

Uruchamianie Linuxa

Jeśli zainstalowałeś Linuxa i system się nie uruchamia, problemem może być konfiguracja LILO lub błąd w tablicy partycji. Uruchom system z dyskietki startowej i sprawdź tablicę partycji, uruchamiając program fdisk. Upewnij się, że partycja linuxowa usta-

wiona jest jako aktywna. Jeżeli to nie pomaga, uruchom system z dyskietki i uruchom ponownie program **LILO**, by skonfigurować jeszcze raz sektor startowy. W rozdziale 4. znajdziesz więcej informacji o programie **LILO**.

Czasem Linux nie potrafi odnaleźć partycji z podstawowym systemem plików. W takiej sytuacji uruchom system z dyskietki trzymając wciśnięty klawisz **Shift** lub **Control**. Ukaże się menu, z którego będziesz mógł wybrać partycję z systemem plików. Problem ten może zazwyczaj być usuany przez ponowne uruchomienie **LILO**.

Podsumowanie

Jeśli instalowałeś już wcześniej Linuxa, większość informacji zawartych w tym rozdziale była Ci znana, choć niektórzy użytkownicy wykorzystujący instalację automatyczną tak naprawdę nie mają pojęcia, o co w niej właściwie chodzi. Poznanie tego procesu i kontrolowanie go pomaga zapobiegać problemom występującym podczas instalacji i mogącym pojawić się później.

Nastepnym krokiem jest skonfigurowanie programu **LILO** tak, by system uruchamiał się prawidłowo. Proces ten jest bardzo często źle rozumiany i, co za tym idzie, program jest konfigurowany nieprawidłowo. Tym problemom poświęcony jest następny rozdział. Jeżeli chcesz dowiedzieć się czegoś więcej na temat instalacji i na tematy pokrewne, przejrzyj wymienione niżej rozdziały.

Aby dowiedzieć się, jak używać nowego systemu, przeczytaj część drugą, zaczynając od rozdziału 6. „Od czego zacząć”.

Jeśli chcesz dowiedzieć się czegoś o instalowaniu i konfiguracji systemu X, zajrzyj do rozdziału 22. „Instalowanie i konfiguracja XFree86”.

Jeżeli chcesz dowiedzieć się, jak zarządzać swoim systemem, przeczytaj część czwartą, rozpoczynając od rozdziału 32. „Podstawy administracji systemem”.

Rozdział 4.

LILO

Kamrain Husain

W tym rozdziale:

- υ Instalacja programu LILO
- υ Linux i dyski twardye
- υ Sektor startowy
- υ Uruchamianie systemu
- υ Program instalujący plik mapowania sektora startowego
- υ Pliki obrazów sektora startowego
- υ Wyłączanie i usuwanie programu LILO
- υ LILO – rozwiązywanie problemów

Program LILO (ang. *LInux LOader*, czyli program ładujący Linuxa) służy do ładowania jądra systemu operacyjnego. Jest on programem bardzo elastycznym: potrafi uruchomić jądro Linuxa zapisane w dowolnym systemie plików, nie wyłączając dyskietki, również z poziomu innego systemu operacyjnego.

Ten rozdział poświęcony jest programowi LILO, obsłudze dysków twardych w systemie Linux, procesowi uruchamiania systemu (ang. *boot*), najpopularniejszym metodom uruchamiania systemu i ich powiązaniem z LILO. Powinien on pomóc Ci zainstalować oraz poprawnie i wydajnie używać programu LILO.

W systemach linuxowych spotkać można kilka wersji programu LILO. Większość nowszych wersji obsługuje jedną z dwóch struktur katalogów: bardziej tradycyjna (i starsza) struktura przechowywana jest w katalogu `/etc/lilo`, natomiast nowsza struktura porozrzucana jest po kilku katalogach, takich jak `/etc`, `/sbin` i `/boot`. Ponieważ wersja znajdująca się w całości w katalogu `/etc/lilo` jest popularniejsza, posłużymy się nią jako wersją modelową. Jeśli posiadasz wersję rozrzuconą po kilku kata-

logach, w przykładach należy podstawić odpowiednie ścieżki dostępu (jeżeli w systemie plików istnieje katalog `/etc/lilo`, oznacza to, że zainstalowana jest starsza wersja programu LILO).

Instalacja programu LILO

W większości systemów program LILO jest już zainstalowany i skonfigurowany. Jeśli działa on prawidłowo w Twoim systemie, możesz pominąć ten podrozdział, chyba że zamierzasz uaktualnić wersję tego programu.

Wiele wersji Linuxa zawiera procedurę szybkiej instalacji programu `LILO`, instalującą minimalną liczbę plików niezbędnych do jego uruchomienia, która jest opisana w pliku `QuickInst.old` albo `QuickInst.new` (choć nie we wszystkich wersjach Linuxa). Może ona być używana wyłącznie w przypadku pierwszej instalacji programu LILO lub, gdy jego stara wersja ma zostać całkowicie usunięta. Nie powinna być natomiast stosowana do uaktualniania wersji LILO, ponieważ utracone zostaną wszystkie informacje konfiguracyjne.



Zanim można będzie skompilować program LILO, należy odpowiednio skonfigurować jądro systemu, używając programu `makeconfig`. Aby komplikacja przebiegła poprawnie, wszystkie pliki nagłówkowe kodu źródłowego jądra systemu muszą znajdować się w katalogu `/usr/include/linux`. Proces instalacji i komplikacji powinien zostać uruchomiony pod kontrolą powłoki `sh` (Bourne Shell) albo kompatybilnej. Zgłaszały były problemy pojawiające się podczas komplikacji pod kontrolą powłoki Korn Shell, lepiej więc użyć powłoki `/bin/sh` lub `/bin/bash`.

Pełna instalacja LILO wymaga, by wszystkie pliki z jego archiwów (zazwyczaj nazywają się one `lilo.xxx.tar.gz`, gdzie `xxx` odpowiada numerowi wersji) były rozpakowane do katalogu o nazwie innej niż `/etc/lilo`, ponieważ procedura instalacyjna zgłasza błąd, gdy nazwy katalogu źródłowego i docelowego są identyczne. Po zdekompresowaniu plików do jakiegoś tymczasowego katalogu należy kolejno:

1. przejrzeć plik `makefile` i upewnić się, że zawarte w nim informacje dotyczące konfiguracji są prawidłowe (dokładniej temat ten zostanie omówiony w podrozdziale „Plik `makefile`”);
2. skompilować program `LILO`; jeśli chcesz używać starej struktury katalogów (`/etc/lilo`), wydaj polecenie

```
make -f Makefile.old
```

3. natomiast jeśli chcesz zainstalować nowszą strukturę:

```
make -f Makefile.new
```

4. skopiować pliki programu LILO do katalogu docelowego, wydając polecenie

```
make -f Makefile.old install
```

lub

```
make -f Makefile.new install
```

w zależności od tego, czy używasz starej, czy nowej struktury katalogów;

5. przejrzeć katalogi `lilo`; powinny znajdować się w nich takie pliki jak: `any_d.b`, `boot.b`, `chain.b`, `disktab`, `lilo` i `os2_d.b`.

Jeśli plików tych nie ma albo też podczas instalacji wyświetlane były komunikaty o błędach, powinieneś powtórzyć proces instalacji. Należy również sprawdzić, czy informacje zawarte w pliku `Makefile` są odpowiednie dla systemu. Po prawidłowym zainstalowaniu programu LILO można użyć go do skonfigurowania procesu uruchamiania systemu.

Problemy z dyskami twardymi

W niektórych systemach dyski twarde nie pozwalają na odczytanie informacji o swoich parametrach, takich jak ilość głowic, cylindrów i sektorów na ścieżkę – generowane są wtedy komunikaty typu „bad geometry”, a proces instalacji kończy się błędami dysku. Dotyczy to szczególnie dysków SCSI oraz dysków o pojemności większej niż 1GB.

W takim przypadku parametry dysku muszą zostać wpisane do pliku `disktab`. Dokładniejszy opis tego zagadnienia znajdziesz w podrozdziale „Tablica parametrów dysku” w dalszej części tego rozdziału. Zmodyfikuj zgodnie z opisem zawartość pliku `disktab`, dopisując do niego parametry dysku, a następnie przetestuj nową konfigurację, kopując ją na dyskietkę i uruchamiając z niej system. Aby to zrobić, musisz:

1. wejść do katalogu programu LILO (zazwyczaj jest to katalog `/etc/lilo`);

2. skopiować konfigurację LILO na dyskietkę za pomocą następującego polecenia:

```
echo image=nazwa_kernela | ./lilo -C - -b /dev/fd0 -v
```

gdzie `nazwa_kernela` to nazwa pliku zawierającego jądro systemu (jeśli nie chcesz utracić bieżącej zawartości pliku mapowania sektora startowego, powinieneś dodać opcję `-m`, a po niej nazwę nowego pliku mapowania);

3. ponownie uruchomić system z dyskietki.

Jeśli informacje konfiguracyjne są poprawne, LILO odczyta program uruchamiający system z dyskietki, po czym załaduje z dysku twardego jądro systemu. Parametry dysku są poprawne, jeżeli uruchomienie przebiegło pomyślnie i można poruszać się po systemie plików. Jeśli nie masz dostępu do systemu plików zapisanego na dysku twardym, oznacza to, że parametry dysku zostały podane nieprawidłowo i należy je poprawić.

Plik Makefile

Plik `Makefile`, dostarczany razem z programem LILO, zawiera informacje poprawne dla większości systemów, dla bezpieczeństwa należy jednak je przejrzeć. LILO używa danych zapisanych w pliku `Makefile` albo `/etc/lilo/config.defines`. Jeśli istnieje plik `config.defines`, plik `Makefile` jest ignorowany. W większości przypadków edycja pliku `Makefile` jest dobrym rozwiązańiem, ale jeśli zamierzasz intensywnie korzystać z programu LILO, lepiej wprowadzać modyfikacje do pliku `config.defines`, ponieważ nowe instalacje LILO nie naruszają danych w nim zapisanych.

Parametry definiowane w pliku `Makefile`, na które należy zwrócić szczególną uwagę, zebrane zostały w tabeli 4.1, wraz z objaśnieniami sposobu ich działania.

Tabela 4.1. Parametry programu LILO definiowane w pliku `Makefile`

Parametr	Znaczenie
<code>IGNORECASE</code>	Powoduje, że w nazwach plików zawierających jądro systemu nie rozróżniane są małe i wielkie litery; domyślnie włączone.
<code>NO1STDIAG</code>	Powoduje, że nie będą wyświetlane komunikaty o błędach odczytu przy ładowaniu programu zarządzającego uruchamianiem systemu; domyślnie wyłączone.
<code>NOINSTDEF</code>	Jeśli w wierszu poleceń pominięto opcję instalacji, nie zostanie zainstalowany nowy sektor startowy, ale zmodyfikowany istniejący; domyślnie wyłączone.
<code>ONE_SHOT</code>	Powoduje, że system nie uruchomi się samoczynnie w przypadku, gdy wcisnięty zostanie jakiś klawisz; domyślnie wyłączone.
<code>READONLY</code>	Zapobiega usunięciu istniejącego pliku mapowania sektora startowego; domyślnie wyłączone.

Aktualizacja LILO

Proces uaktualniania istniejącej wersji LILO przebiega prawie tak samo jak jego pierwsza instalacja. Jedyną różnicą jest fakt, że wykonane zostaną kopie zapasowe wszystkich istniejących plików konfiguracyjnych, o nazwach z rozszerzeniem `.old`, na przykład plik `chain.b` zostanie przemianowany na `chain.old`. Jeśli nowa wersja LILO zachowuje się poprawnie, pliki te mogą później zostać usunięte.

Uaktualnienie wersji LILO zawsze wiąże się z koniecznością odświeżenia sektora startowego, ponieważ trzeba dodać do niego informacje o nowych lokalizacjach plików i o nowym pliku mapowania sektora startowego. Odświeżanie sektora startowego polega na uruchomieniu programu `lilo`.

Linux i dyski twarde

Żeby zrozumieć zasadę działania LILO, musisz poznać „budowę” dysku twardego. Na pewno wiesz, że jest to w zasadzie system koncentrycznych ścieżek (ang. *track*), rozchodzących się od środka powierzchni dysku (talerza) w kierunku jego brzegu. Każda ścieżka podzielona jest na pewną liczbę sektorów (ang. *sector*).

Dysk twardy identyfikowany jest przez liczbę talerzy, a dokładniej liczbę głowic (ponieważ liczba talerzy może być większa, gdy nie wszystkie są używane do zapisywania danych, jak to ma zazwyczaj miejsce w przypadku pierwszego i ostatniego talerza), liczbę ścieżek na cal (mierzoną wzdłuż promienia) i liczbę sektorów przypadających na jedną ścieżkę. Znając pojemność jednego sektora, łatwo wyliczyć całkowitą pojemność dysku przez pomnożenie liczby sektorów na ścieżkę, liczby ścieżek i głowic oraz pojemności jednego sektora.

Linux jest dość ściśle powiązany z systemem DOS, więc zobaczymy najpierw, jak z dysku twardego korzysta DOS. Dysk zawierający jeden system operacyjny (czyli również dyskietka) posiada sektor startowy (ang. *boot sector*), po którym następuje obszar danych, zawierający między innymi blok administracyjny (informacje o rozkładzie plików itp.).

Sektor startowy jest pierwszym sektorem na dysku; jest on odczytywany podczas uruchamiania komputera w celu załadowania systemu operacyjnego. Sektor startowy zawiera krótki program, kierujący proces uruchamiania systemu do właściwych procedur inicjujących.

Blok administracyjny jest przeważnie częścią obszaru danych, choć nie jest bezpośrednio dostępny dla użytkowników. Zawiera on informacje o położeniu plików na dysku (w formacie głowica/ścieżka/sektor). W systemie DOS zapisane są one w tablicy alokacji plików FAT (ang. *File Allocation Table*), a na przykład UNIX i Linux używają do tego celu tabeli zawierających informacje o superblokach lub węzłach I-node (zwanych krócej tablicami I-node). Blok administracyjny zazwyczaj nie jest odczytywany aż do momentu rozpoczęcia procesu uruchamiania systemu operacyjnego.

Obszar danych używany jest do zapisywania plików (włączając w to programy inicjujące działanie systemu operacyjnego). Każdemu plikowi na dysku twardym odpowiada wpis w bloku administracyjnym, zawierający jego nazwę i fizyczne położenie na dysku. Zależnie od systemu operacyjnego, są tam również przechowywane inne informacje (identyfikator właściciela, atrybuty, data i czas utworzenia itp.).

Jeżeli posiadasz dysk twardy o większej pojemności, prawdopodobnie będziesz chciał założyć kilka partycji (najpewniej zawierających więcej niż jeden system operacyjny, na przykład DOS i Linux). Na dysku DOS-owym można utworzyć do czterech partycji podstawowych.



Niektóre systemy operacyjne pozwalają na utworzenie większej liczby partycji podstawowych, ale jeśli na danym dysku zamierzasz zainstalować system DOS, nie powinno być ich więcej niż cztery. Jeśli zachodzi taka potrzeba, można użyć partycji rozszerzonych. Są to partycje podstawowe, które mogą być dalej dzielone na mniejsze dys-

ki logiczne.

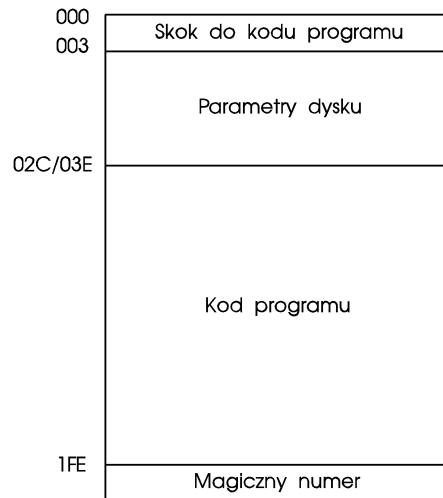
Tablica partycji zapisywana jest w sektorze startowym (ang. *boot sector*) każdego dysku twardego (ale nie każdego talerza). Sektor ten jest czasem nazywany głównym rekordem startowym (ang. *Master Boot Record*, MBR). Jedyna różnica pomiędzy sektorem startowym a głównym rekordem startowym polega na tym, że główny rekord startowy zawiera informacje o rozmieszczeniu partycji. Sektor startowy na dysku twardym zwyczaj nazywany jest głównym rekordem startowym, ale choć sektor startowy dyskietki nie zawiera informacji o rozmieszczeniu partycji, czasem używa się tego terminu również w odniesieniu do niego. Tablica partycji zapisywana jest także na początku każdej istniejącej partycji rozszerzonej. W systemie Linux sektor startowy jest tworzony za pomocą specjalnego programu instalującego pliki mapowania sektora startowego, nazywanego po angielsku *map installer*.

Jeśli na dysku twardym znajduje się kilka partycji, są one w systemie Linux nazywane odpowiednio `/dev/hda1`, `/dev/hda2` itd. Partycje rozszerzone oznaczane są jako `/dev/hda5`, `/dev/hda6` itd. (ich numerowanie rozpoczyna się od liczby 5, ponieważ dozwolone jest istnienie tylko czterech partycji podstawowych). Cały pierwszy dysk twardy nazywa się `/dev/hda`, drugi – `/dev/hdb` (partycje na drugim dysku twardym mają więc nazwy `/dev/hdb1`, `/dev/hdb2` itd.). Możliwe są również inne nazwy dysków twardych, zależnie od ich typu i rodzaju sterownika, np. dysk SCSI będzie się nazywał `/dev/sda`, a nie `/dev/hda` (nazwy typu `/dev/hdx` stosowane są zwykle dla dysków IDE i EIDE).

Sektor startowy

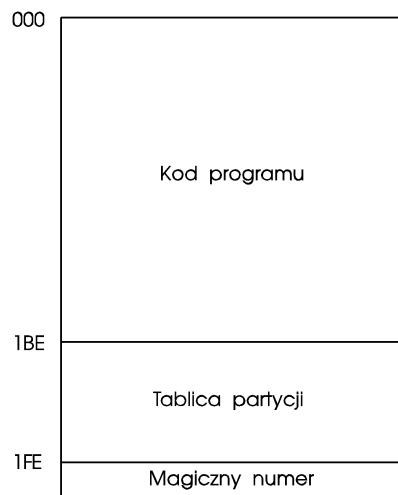
Aby zrozumieć proces uruchamiania Linuxa, trzeba najpierw przyjrzeć się sektorowi startowemu (ang. *boot sector*) systemu DOS. Rysunek 4.1 pokazuje jego typową strukturę. Kod programu to jedynie skok do procedur inicjalizujących działanie systemu operacyjnego. Parametry dysku zawierają tablicę alokacji plików FAT.

Rysunek 4.1.
Struktura sektora startowego systemu DOS



Sektor startowy zapisywany przez program LILO jest podobny do DOS-owego, tyle że sekcja zawierająca parametry dysku nie jest używana, poza tym granice pomiędzy sekcjami kodu są nieco inne, co pokazuje rysunek 4.2. Różnice te mogą powodować problemy w momencie, gdy sektor startowy pochodzący z LILO zostanie zapisany w głównym rekordzie startowym dysku DOS-owego, ponieważ system DOS nie jest wtedy w stanie poprawnie się załadować.

Rysunek 4.2.
Struktura sektora startowego przy zainstalowanym programie LILO



„Magiczny numer” (ang. *magic number*) występujący w obu wyżej wymienionych strukturach to dwubajtowa liczba, używana przez niektóre systemy operacyjne do sprawdzania, czy dany sektor jest rzeczywiście sektorem startowym.

Teoretycznie sektor startowy programu LILO może być używany do uruchamiania systemu DOS, ponieważ obszar tablicy partycji może zawierać DOS-ową tablicę FAT, ale

w praktyce przeważnie jest, niestety, inaczej. Zalecane jest używanie sektora startowego zapisanego na początku partycji linuxowej.



Ponieważ sektor startowy systemu DOS i zapisywany przez program LILO różnią się, system DOS należy zainstalować przed zainstalowaniem Linuxa. Dzięki temu DOS-owy sektor startowy zostanie na pewno zapisany na dysku twardym. Jeśli najpierw zainstalujesz Linuxa, zapisując na dysk twardy sektor startowy LILO, DOS nie będzie potrafił się uruchomić.

Sektor startowy LILO może zostać zapisany na dyskietce, w głównym rekordzie startowym dysku twardego, na początku partycji linuxowej, albo w obszarze sektora startowego partycji rozszerzonej. Nie można go zapisać na żadnej partycji systemu innego niż Linux, ani na innym niż pierwszy dysku twardym.

Zauważ, że choć DOS nie potrafi obsłużyć sektora startowego zapisanego na początku partycji rozszerzonej, potrafi to Linux, wykorzystując program `fdisk` lub program o nazwie `activate`.



Fakt, że program LILO potrafi zapisać sektor startowy w dowolnym miejscu, nawet niedostępnym dla systemu, jest częstą przyczyną problemów występujących podczas używania tego programu. Upewnij się, że zapisujesz sektor startowy tam, gdzie trzeba. Nawet jeśli zainstalowałeś już Linuxa, a teraz tylko wprowadzasz drobne zmiany, zawsze powinieneś mieć pod ręką dyskietkę startową.

Uruchamianie systemu

Podczas uruchamiania systemu odczytywany jest sektor startowy i – dzięki kodowi w nim zawartemu – sterowanie przekazywane jest do systemu operacyjnego. W przypadku systemu DOS, odczytywany jest główny rekord startowy, a następnie ładowany jest program `COMMAND.COM`, pełniący funkcję jądra systemu DOS.

Choć proces uruchamiania systemu zwykle konfigurowany jest podczas instalacji Linuxa, możliwa jest również późniejsza zmiana jego przebiegu. Do zagadnienia tego można podejść na kilka sposobów, zależnie od wymagań użytkownika i konfiguracji dysku twardego. Poniżej omówimy kilka typowych konfiguracji, aby pokazać, jak można modyfikować ten proces.

Dysk twardy przeznaczony dla Linuxa

Jeżeli dla Linuxa przeznaczony jest cały dysk twardy, albo też Linux ma być podstawowym systemem operacyjnym (a na dysku znajdują się inne systemy), to główny rekord

startowy (MBR) może zostać zastąpiony sektorem startowym programu LILO. Wtedy LILO od razu uruchomi system z głównego rekordu startowego, pomijając zawartość sektorów startowych innych partycji. Jednak w niektórych przypadkach możesz być zmuszony do bezpośredniego podania, z którego sektora startowego chcesz uruchomić system. Inaczej mówiąc, jeżeli ustawienia domyślne nie działają, to po wyświetleniu komunikatu `boot:` podczas uruchamiania systemu należy wpisać `boot=/dev/hda` (podstawiając za `/dev/hda` nazwę urządzenia zawierającego zmodyfikowany sektor startowy).



Jeśli zastąpisz MBR sektorem startowym programu LILO, a następnie zrezygnujesz z używania Linuxa, musisz przywrócić poprzednią wersję głównego rekordu startowego - dopiero po takim zabiegu możliwe będzie używanie innych systemów operacyjnych, na przykład systemu DOS.

Oto co należy zrobić, aby zainstalować LILO na dysku twardym przeznaczonym w całości dla systemu Linux.

1. Uruchom Linuxa w normalny sposób. Upewnij się, że na wypadek problemów masz pod ręką dyskietkę startową.
2. Na wszelki wypadek skopiuj aktualną wersję MBR na dyskietkę. W tym celu wydaj następujące polecenie:

```
dd if=/dev/hda of=/fd/MBR bs=512 count=1
```

gdzie `/dev/hda` to nazwa dysku, z którego system jest uruchamiany, zaś opcja `bs=512` powoduje, że używane są 512-bajtowe bloki.
3. Użyj programu `setup` albo programu instalacyjnego LILO, by zainstalować LILO w sektorze startowym.
4. Uruchom ponownie system z głównego rekordu startowego.

Teraz Linux powinien zostać uruchomiony automatycznie. Jeśli Linux nie uruchamia się, użyj dyskietki startowej, po czym spróbuj powtórzyć proces instalacji LILO albo przywrócić oryginalną, nagraną wcześniej na dyskietkę, wersję MBR za pomocą polecenia

```
dd if=/fd/MBR of=/dev/hda bs=446 count=1
```

Program BOOTACTV

Alternatywą dla LILO jest program BOOTACTV, który również zapisywany jest w sektorze startowym i pozwala na podanie nazwy partycji, z której ma zostać uruchomiony system. Wymaga to zapisania niekompatybilnej z systemem DOS wersji MBR, więc używać go należy tylko w przypadku, gdy Linux jest dominującym systemem operacyjnym, a LILO z jakiegoś powodu odmawia posłuszeństwa.

Jeśli do uruchamiania systemu używany jest program BOOTACTV, jego kopia zapisana jest w głównym rekordzie startowym. Po uruchomieniu pozwala on na podanie nazwy partycji, z której należy uruchomić system, a następnie przekazuje sterowanie do odpowiedniego sektora startowego.

Oto co należy zrobić, aby zainstalować program BOOTACTV.

1. Uruchom Linuxa jak zwykle. Upewnij się, że na wypadek problemów masz pod ręką dyskietkę startową.
2. Skopiuj, podobnie jak przy instalacji LILO, aktualną wersję MBR na dyskietkę. Wydaj w tym celu polecenie

```
dd if=/dev/hda of=/fd/MBR bs=512 count=1
```

3. Uruchom program `setup` albo program instalacyjny LILO, aby zainstalować LILO w sektorze startowym partycji linuxowej (NIE w głównym rekordzie startowym!).

4. Zainstaluj program BOOTACTV w głównym rekordzie startowym. Plik, który należy tam skopiować, nazywa się zazwyczaj `bootactv.bin` i powinien znajdować się w bieżącym katalogu, kiedy wydasz polecenie

```
dd if=bootactv.bin of=/dev/hda bs=446 count=1
```

zapisujące jego kopię w obszar MBR.

5. Uruchom system ponownie.

Po wykonaniu tych kroków powinien zgłosić się program BOOTACTV, pozwalając wybrać partycje, z której chcesz uruchomić system. Jeśli Linux ani żaden inny system operacyjny nie uruchamia się prawidłowo, uruchom ponownie Linuxa z dyskietki. Jeśli tylko Linux nie chce się uruchomić, oznacza to, że sektor startowy na partycji linuxowej jest niewłaściwy i powinieneś ponownie uruchomić program instalujący LILO. Jeżeli system nie uruchamia się z żadnej partycji, trzeba przywrócić poprzednią wersję MBR, usuwając program BOOTACTV, za pomocą polecenia:

```
dd if=/fd/MBR of=/dev/hda bs=446 count=1
```

Można również przeinstalować główny rekord startowy z poziomu systemu DOS (o ile posiadasz dyskietkę startową systemu DOS), za pomocą polecenia

```
fdisk /mbr
```



Jeżeli nie chcesz zmieniać zawartości głównego rekordu startowego, a posiadasz więcej niż jedną partycję linuxową, możesz zainstalować program BOOTACTV w sektorze startowym którejś z tych partycji, a następnie zmieniać aktywną partycję za pomocą programu `fdisk`. Wtedy w przypadku ponownego dzielenia dysku na partycje czy zmiany rozmiaru partycji trzeba będzie odświeżyć zawartość sektora startowego. Żeby np. zapisać program BOOTACTV na

czwartej partycji pierwszego dysku twardego, należy skopiować istniejącą zawartość MBR do sektora startowego tej partycji, a następnie zainstalować program **BOOTACTV**:

```
dd if=/dev/hda of=/dev/hda4 bs=512 count=1  
dd if=bootactv.bin of=/dev/hda4 bs=446 count=1
```

DOS i Linux

Większość systemów linuxowych współistnieje z DOS-em i używa głównego rekordu startowego systemu DOS. W takim przypadku odczytywany jest MBR i system uruchamia się automatycznie z partycji ustawionej programem **fdisk** jako partycja aktywna. Jest to jedna z najbezpieczniejszych metod uruchamiania systemu, ponieważ nie wymaga zmian w MBR, a także dlatego, że łatwo jest usunąć lub zmienić rozkład partycji bez obawy o kompatybilność z MBR.



Nowsze wersje systemu DOS (od 6.00 w górę) zamazują istniejącą wersję MBR, jeżeli są instalowane po zainstalowaniu systemu Linux. Uniemożliwia to uruchomienie Linuxa, choć DOS działa bezproblemowo. Poradzić sobie z tym problemem można, uruchamiając ponownie LILO lub ustawiając partycję linuxową jako aktywną.

Partycja aktywna może być zmieniona w każdej chwili za pomocą programu **fdisk** lub linuxowego programu **activate**; zazwyczaj potrafi to również program instalacyjny systemu Linux. Tylko jedna partycja na dysku twardym może być ustawiona jako aktywna. Niektóre systemy operacyjne, w tym Linux, pozwalają uruchomić komputer z innej niż aktywna partycji. Linux na przykład wyświetla komunikat **boot:** i czeka na wprowadzenie nazwy partycji, z której system ma zostać uruchomiony, albo też, po upływie zadanego czasu, uruchamia się z partycji aktywnej.

Aby uruchamiać Linuxa w ten właśnie sposób, po prostu zainstaluj LILO w sektorze startowym partycji linuxowej. Następnie uruchom program **fdisk** i ustaw tę partycję jako aktywną. Po zresetowaniu system uruchomi się z aktywnej partycji – czyli z partycji linuxowej.

Po usunięciu Linuxa lub zastąpieniu go innym systemem operacyjnym, sektor startowy nowego systemu operacyjnego zostanie zapisany na miejsce linuxowego, nie zmieniając zawartości głównego rekordu startowego.

Używanie programu **BOOTLIN**

Program **BOOTLIN**, również dość często używany do uruchamiania Linuxa, także nie zmienia zawartości MBR. Podczas uruchamiania systemu odczytywany jest główny rekord startowy, po czym użytkownik może podjąć decyzję, który system operacyjny chce załadować. Jest to w zasadzie proces identyczny z uruchamianiem systemu MS-DOS, tyle że w pliku **config.sys** lub **autoexec.bat** wywoływany jest program **BOOTLIN**, pozwala-

jący uruchomić program dający możliwość wybrania systemu operacyjnego. Przykładowo, uruchomiony może zostać program `BOOT.SYS`, który wyświetla menu, pozwalające użytkownikowi wybrać pomiędzy załadowaniem systemów DOS i Linux.

Oto, co powinieneś zrobić, aby zainstalować program `BOOTLIN`.

1. Uruchom Linuxa. Upewnij się, że masz pod ręką dyskietkę startową.
2. Skopiuj jądro systemu Linux na partycję DOS-ową, używając np. jednego z programów pakietu `mtools`. Możesz zrobić to również za pośrednictwem dyskietki.
3. W ten sam sposób na partycję DOS-ową skopiuj pliki `boot.sys` i `bootlin.sys`.
4. Dopusz do pliku `config.sys` wiersze uruchamiające programy `boot.sys` i `bootlin.sys`.
5. Upewnij się, że partycja DOS-owa jest ustawiona jako aktywna i uruchom system ponownie.

Podeczas uruchamiania systemu DOS program `boot.sys` wyświetli menu pozwalające na wybór systemu operacyjnego. Jeżeli natkniesz się na jakieś problemy, po prostu usuń odpowiednie odwołania z pliku `config.sys`, a wróćisz do pierwotnej konfiguracji.

Program `boot.sys` ma jedną bardzo ważną zaletę: nie zmienia zawartości sektorów startowych. Ułatwia to znaczowo instalowanie i usuwanie poszczególnych systemów operacyjnych. Możliwe jest równoczesne zmienianie aktywnej partycji i używanie programu `boot.sys`, dzięki czemu ładowany jest system operacyjny zapisany na aktywnej partycji, a następnie komputer oczekuje zadaną ilość czasu na potwierdzenie lub zmianę uruchamianego systemu operacyjnego. W takiej konfiguracji nie są wymagane żadne zmiany w obszarze MBR.

Parametry startowe

Bez względu na to, jak wygląda proces uruchamiania systemu, LILO zatrzymuje się na chwilę, by sprawdzić, czy wcisnięty jest klawisz `Shift`, `Control` lub `Alt` oraz czy aktywne są znaczniki `CapsLock` i `ScrollLock`. Jeśli żaden z powyższych klawiszy nie został wcisnięty, a znaczniki są aktywne, LILO wyświetla komunikat `boot:` i czeka chwilę na decyzję użytkownika. W tym momencie możliwe jest podanie nazwy pliku obrazu sektora startowego lub wcisnięcie klawisza `Enter`, powodujące uruchomienie systemu ustawionego jako domyślny (system domyślny, którym może być również DOS, zostanie uruchomiony również po upływie czasu określonego w pliku konfiguracyjnym).

Jeśli chcesz uruchomić inny niż domyślny system operacyjny, musisz po wyświetleniu zachęty `boot:` podać odpowiednią etykietę. Listę dostępnych etykiet możesz otrzymać po wcisnięciu klawisza `Tab` lub znaku zapytania (w zależności od ustawień klawiatury). Etykieta wybierana domyślnie zapisana jest w pliku `/etc/rc` lub `/etc/lilo.conf` (zależnie od wersji Linuxa) w wierszu rozpoczynającym się od tekstu

`BOOT_IMAGE=`

Oprócz tego, podczas uruchamiania systemu można również przekazać do jądra systemu kilka parametrów. Dostępne parametry różnią się nieco w zależności od wersji Linuxa, ale zazwyczaj obsługiwane są następujące:

- no387** wyłącza koprocesor arytmetyczny;
- root** pozwala na podanie urządzenia, z którego system ma zostać uruchomiony, np. `root=/dev/sda1`; wymaga podania pełnej ścieżki dostępu do urządzenia lub jego adresu szesnastkowego;
- ro** montuje główny system plików w trybie tylko do odczytu;
- rw** montuje główny system plików w trybie do odczytu i zapisu;
- single** uruchamia Linuxa w trybie administracyjnym (dostęp do systemu ma wówczas tylko administrator).

Parametr `root` pozwala na użycie liczby szesnastkowej określającej urządzenie. Liczby te przypisane są zgodnie z rodzajem urządzenia. Numery stacji dyskietek zaczynają się od 200, więc urządzenie `/dev/fd0` ma numer 200, `/dev/fd1` to 201 itd. Partycje pierwszego dysku twardego zazwyczaj przypisane mają numery rozpoczynające się od 301 (ponieważ nie ma partycji o numerze 0): `/dev/hda1` ma numer 301, `/dev/hda2` – 302 i tak dalej. Numery partycji drugiego dysku twardego zaczynają się od 340: `/dev/hdb1` to 341 itd. Numery urządzeń SCSI (również dysków twardych) zaczynają się od 801, np. `/dev/sda1` ma numer 801, `/dev/sda2` – 802 itd. Drugie urządzenie SCSI ma przypisane numery od 811, więc `/dev/sdb1` ma numer 811. Ponieważ dyskietki i dyski twarde to zwykle jedynie urządzenia, z których można uruchamiać system, powyższe zestawienie powinno wystarczyć w razie problemów (chyba że używasz dysków wymienionych).

Można podać równocześnie kilka parametrów, oddzielając je spacją. Oto przykładowy zestaw parametrów, które podane po wyświetleniu zachęty `boot:` spowodują uruchomienie jądra systemu o etykiecie `image5`, zapisanego na urządzeniu o nazwie `/dev/hda2`:

```
image5 root=/dev/hda2 single ro
```

System plików zostanie zamontowany w trybie tylko do odczytu i uruchomiony zostanie tryb administracyjny.

Program instalujący plik mapowania sektora startowego

Program instalujący plik mapowania sektora startowego (ang. *map installer*) uaktualnia zawartość sektora startowego i tworzy plik, który zawiera kod w nim zapisany. Zazwyczaj tę funkcję spełnia program `/etc/lilo/lilo`. Za każdym razem, gdy jest on uruchamiany,

sprawdza, czy w sektorze startowym nie występują błędy. Jeśli zostaną wykryte jakieś problemy, zmiany nie są zapisywane i program kończy działanie.

Jeżeli uda się prawidłowo zmienić zawartość sektora startowego, jego stara wersja jest zapisywana do pliku o nazwie `boot.hex_num` w katalogu `/etc/lilo` (`hex_num` jest szesnastkowym numerem urządzenia, na które zapisywany jest sektor startowy, zgodnie z informacjami podanymi w poprzednim podrozdziale). Jeśli zmieniony został sektor startowy partycji, stara wersja zapisywana jest do pliku o nazwie `part.hex_num`, gdzie `hex_num` jest również numerem szesnastkowym partycji.

Zachowanie programu instalującego plik mapowania może być modyfikowane przez podanie parametrów w wierszu poleceń albo przez modyfikację zawartości pliku `/etc/lilo/config`. Większość opcji może być ustawiona zarówno w wierszu poleceń, jak i w pliku konfiguracyjnym – zapoznajmy się z nimi, zanim przejdziemy do uruchamiania samego programu.

Parametry podawane w wierszu poleceń

LILO rozpoznaje wiele opcji podawanych w wierszu poleceń; zostały one zebrane w tabeli 4.2. Większość z nich ma swoje odpowiedniki w zmiennych konfiguracyjnych, omówionych w następnym podrozdziale.

Tabela 4.2. Opcje podawane w wierszu poleceń

Opcja	Znaczenie
<code>b dev</code>	Wymusza uruchomienie systemu z urządzenia <code>dev</code> . Jeśli nie podano żadnego urządzenia, używane jest urządzenie domyślne, którego nazwa określona jest przez wartość zmiennej konfiguracyjnej <code>boot</code> .
<code>c</code>	Włącza tryb kompaktowy, co powoduje, że pojedyncze żądania odczytu grupowane są w bloki – efektem jest przyspieszenie ładowania systemu. Opcja ta używana jest zazwyczaj w przypadku uruchamiania systemu z dyskietki. Odpowiada jej zmienna konfiguracyjna <code>compact</code> .
<code>C plik_konf</code>	Powoduje, że jako plik konfiguracyjny użyty zostanie plik <code>plik_konf</code> . Jeśli nie podano żadnego pliku, domyślnie używany jest <code>/etc/lilo/config</code> .
<code>d secs</code>	Pozwala na podanie czasu (w dziesiątych częściach sekundy), jaki należy od czekać przed rozpoczęciem ładowania systemu. Wartość ta może również zostać podana poprzez zmienną konfiguracyjną <code>delay</code> .
<code>f plik</code>	Powoduje użycie zbioru o nazwie <code>plik</code> jako pliku z parametrami dysku (ang. <code>disktab</code>). Domyślnie nazwą takiego pliku jest <code>/etc/lilo/disktab</code> .
<code>i sector</code>	Instaluje jądro systemu jako nowy sektor startowy. Opcji tej odpowiada zmienna <code>install</code> .
<code>I nazwa</code>	Wyświetla ścieżkę dostępu do pliku zawierającego obraz jądra systemu. Jeśli nie znajdzie takiego pliku, generowany jest komunikat o błędzie. Po nazwie można również użyć opcji <code>v</code> , by sprawdzić, czy zadany plik istnieje. Opcja ta używa korzysta z wartości zmiennej <code>BOOT_IMAGE</code> .
<code>l</code>	Generuje liniowe adresy sektorów zamiast adresów w domyślnym formacie sektorów.

	tor/główica/cylinder. Opcji tej odpowiada zmienna konfiguracyjna <code>linear</code> .
<code>m plik</code>	Powoduje użycie zbioru <code>plik</code> jako pliku mapowania sektora startowego. Domyślnie używany jest plik <code>/etc/lilo/map</code> .
<code>P fix</code>	Pozwala programowi LILO skorygować adresy sektor/główica/cylinder w oparciu o specjalny plik. Zmienna odpowiadająca tej opcji ma nazwę <code>fix-table</code> .
<code>P ignore</code>	Wymusza pominięcie korekcji adresów sektor/główica/cylinder. Opcji tej odpowiada zmienna: <code>ignore-table</code> .
<code>q</code>	Wyświetla nazwy aktualnie mapowanych plików.
<code>r katalog</code>	Wykonuje polecenie <code>chroot</code> w zadanym katalogu. Jest to niezbędne, jeśli system plików znajduje się w innym miejscu niż program LILO. Ponieważ efektem użycia tego polecenia jest zmiana katalogu bieżącego, w nazwach plików należy podawać bezwzględne ścieżki dostępu.

Tabela 4.2. cd. Opcje podawane w wierszu poleceń

Opcja	Znaczenie
<code>R opcje</code>	Zapisuje opcje do pliku mapowania. Będą one używane jako parametry startowe jądra systemu (pierwszym wyrazem musi być etykieta systemu).
<code>s plik</code>	Kopiuje oryginalną zawartość sektora startowego do pliku o nazwie <code>plik</code> , zamiast do pliku <code>/etc/lilo/boot.hex_num</code> .
<code>S plik</code>	Jak wyżej, z tym że usuwa istniejący plik.
<code>t</code>	Uruchamia tryb testowy, przeprowadzając cały proces instalacji za wyjątkiem zapisu pliku mapowania i sektora startowego. Opcji tej używa się w połączeniu z opcją <code>v</code> by sprawdzić, czy program będzie działał prawidłowo.
<code>u dev</code>	Przywraca poprzednią wersję sektora startowego urządzenia <code>dev</code> z kopii zapasowej. Jeśli nazwa urządzenia nie zostanie podana, przywracany jest sektor startowy urządzenia domyślnego lub, w razie niepowodzenia, bieżącego. Przed dokonaniem zapisu sprawdzany jest czas wykonania kopii zapasowej.
<code>U dev</code>	Jak wyżej, z tym że pomija sprawdzanie czasu utworzenia kopii zapasowej.
<code>v poziom</code>	Ustawia poziom interaktywności (więcej lub mniej wyświetlanych komunikatów).
<code>V</code>	Wyświetla informację o wersji programu i kończy działanie.

Parametry podawane w pliku konfiguracyjnym

Parametry programu instalującego plik mapowania mogą zostać również zapisane w pliku konfiguracyjnym. Domyślnie jest to plik o nazwie `/etc/lilo/config`. Zawiera on zestaw par `zmienna = wartość`, choć niektóre opcje nie wymagają podawania wartości. Pomiędzy zmienną, znakiem równości i wartością mogą znajdować się znaki białe. Tekst od symbolu `#` do końca wiersza traktowany jest jako komentarz.

W nazwach zmiennych nie są rozróżniane małe i wielkie litery, natomiast w ich wartościach są one istotne. Jednak dobrym zwyczajem jest pisanie wszystkich nazw małymi literami (konwencja przyjęta w UNIX-ie).

Opcje, które mogą być ustawione w pliku `/etc/lilo/config`, zebrane w tabeli 4.3.

Tabela 4.3. Opcje podawane w pliku konfiguracyjnym `/etc/lilo/config`

Opcja	Znaczenie
<code>alias=nazwa</code>	Pozwala uruchomić system operacyjny przez podanie etykiety zamiast nazwy pliku obrazu.
<code>append=tekst</code>	Dolączają tekstu do wiersza poleceń przesyłanego do jądra systemu. Opcja używana najczęściej do podawania informacji o urządzeniach nie wykrywanych automatycznie przez kernel.
<code>backup=plik</code>	Kopiuje oryginalną zawartość sektora startowego do pliku <code>plik</code> zamiast do <code>/etc/lilo/boot.hex_num</code> . Można tu również podać nazwę urządzenia (np. <code>/dev/null</code>).

cd. na następnej stronie

Tabela 4.3. cd. Opcje podawane w pliku konfiguracyjnym `/etc/lilo/config`

Opcja	Znaczenie
<code>boot=dev</code>	Pozwala na podanie urządzenia, z którego należy uruchomić system. Domyślnie używana jest aktualnie zamontowana partycja główna.
<code>compact</code>	Powoduje, że pojedyncze żądania odczytu są grupowane w bloki, co przyspiesza proces ładowania jądra. Opcja używana głównie w przypadku uruchamiania systemu z dyskietki.
<code>delay=secs</code>	Pozwala podać czas (w dziesiątych częściach sekundy), jaki należy od czekać przed rozpoczęciem ładowania systemu. Domyślnie ładowanie rozpoczyna się natychmiast.
<code>disktab=plik</code>	Podaje nazwę pliku z parametrami dysku. Domyślnie jest to plik <code>/etc/lilo/disktab</code> .
<code>fix-table</code>	Pozwala programowi LILO skorygować adresy sektor/głowica/cylinder. Opcja używana w systemach, które zmieniają te adresy.
<code>force-backup=plik</code>	Podobnie jak <code>backup</code> , z tym że usuwa istniejące pliki. Jeśli opcja ta jest podana, wszystkie inne opcje dotyczące tworzenia kopii zapasowej są ignorowane.
<code>install=sektor</code>	Instaluje plik obrazu w zadanym sektorze startowym. Jeśli nie podano żadnej wartości, używany jest <code>/etc/lilo/boot.b</code> .
<code>label=nazwa</code>	Zmienia etykietę systemu na podaną nazwę.
<code>linear</code>	Wymusza używanie liniowego adresowania sektorów zamiast formatu sektor/głowica/cylinder. Adresowanie liniowe jest niezależne od geometrii dysku, a adresy są tłumaczone w czasie rzeczywistym. Dyski startowe adresowane liniowo mogą nie działać prawidłowo w innych systemach.
<code>literal=tekst</code>	Podobnie jak <code>append</code> , z tym że nie dopuszcza stosowania żadnych

	dodatkowych opcji, używając tylko tych podanych jako <code>tekst</code> .
<code>map=plik</code>	Położenie pliku mapowania. Domyślnie jest to plik <code>/etc/lilo/map</code> .
<code>message=plik</code>	Używa zawartości pliku o nazwie <code>plik</code> jako informacji wyświetlanej przed pojawiением się komunikatu <code>boot:</code> . Plik taki nie może być dłuższy niż 64 kB. Jeśli treść informacji zostanie zmieniona, należy ponownie utworzyć plik mapowania.
<code>optional</code>	Sprawia, że dany system jest opcjonalny, tzn. jeśli nie może być zlokalizowany, to po prostu nie jest możliwe uruchomienie go. Opcja przydatna podczas testowania nowych konfiguracji jądra.
<code>password=hasło</code>	Ustawia hasło dla wszystkich konfiguracji. Jeśli dla któregoś systemu ustawiona jest opcja <code>restricted</code> , hasło będzie stosowało się tylko do niego.
<code>prompt</code>	Wymusza pojawienie się komunikatu <code>boot:</code> bez względu na to, czy naciśnięte są jakieś klawisze. Zazwyczaj opcja ta używana jest razem z opcją <code>timeout</code> .

Tabela 4.3. cd. Opcje podawane w pliku konfiguracyjnym `/etc/lilo/config`

Opcja	Znaczenie
<code>ramdisk=rozmiar</code>	Ustawia rozmiar opcjonalnego RAM-dysku. Wartość 0 wyłącza jego obsługę.
<code>read-only</code>	Montuje główny system plików w trybie tylko do odczytu.
<code>read-write</code>	Montuje główny system plików w trybie do odczytu i zapisu.
<code>restricted</code>	Zwalnia zabezpieczenie hasłem.
<code>root=dev</code>	Wskazuje urządzenie, na którym zapisany jest główny system plików. Jeśli użyta jest wartość <code>current</code> , będzie to urządzenie, na którym aktualnie zamontowany jest główny system plików (chyba że zostanie to zmienione opcją <code>-r</code> w wierszu poleceń).
<code>serial=parametry</code>	Pozwala sterować uruchamianiem systemu przez port szeregowy (w taki sam sposób, jak z konsoli). Format parametrów to: port, szybkość transmisji, parzystość, ilość bitów. Jeśli opcja ta jest ustawiona, automatycznie ustawiana jest również opcja <code>timeout</code> na wartość 20, chyba że wartość ta zostanie zmieniona w wierszu poleceń.
<code>timeout=secs</code>	Pozwala podać czas (w dziesiątych częściach sekundy), jaki należy czekać na wprowadzenie danych z klawiatury przed załadowaniem systemu. Opcja ta jest używana również do ograniczenia czasu wprowadzania hasła. Domyślną wartością jest nieskończoność.
<code>verbose=poziom</code>	Załącza wyświetlanie komunikatów diagnostycznych: im wyższy poziom, tym więcej komunikatów jest generowanych. Jeśli włączona jest również opcja <code>-v</code> , wyświetlana jest jeszcze większa liczba komunikatów.
<code>vga=tryb</code>	Ustawia tryb karty VGA używany podczas uruchamiania systemu. Dostępne wartości to: <code>normal</code> (80x25), <code>extended</code> lub <code>ext</code> (80x50), <code>ask</code> (pyta użytkownika podczas uruchamiania) i inne wartości opisujące tryby graficzne (aby otrzymać ich listę, uruchom <code>system z pa-</code>

parametrem `vga=ask` i wciśnij `Enter`, gdy zostaniesz zapytany o wartość). Wielkość liter nie jest tu istotna.

Jeśli któryś z parametrów nie jest podany ani w wierszu poleceń, ani w pliku konfiguracyjnym, używana jest wartość domyślna. Niektóre wartości są również zależne od jądra systemu (np. wielkość RAM-dysku, tryb VGA itp.).

Pliki obrazów sektora startowego

LILO potrafi uruchomić jądro systemu zapisane w różnych lokalizacjach, takich jak plik w głównym systemie plików, dowolnym innym zamontowanym systemie plików, na urządzeniu blokowym, stacji dysków, czy w końcu zapisane w sektorze startowym innej partycji czy dysku. Sposób uruchomienia systemu określony jest przez wpisy w pliku konfiguracyjnym.

W pliku konfiguracyjnym opisującym plik zawierający jądro systemu zdefiniować można wartości kilku zmiennych konfiguracyjnych, których znaczenie omówiliśmy już wcześniej. Możliwe jest określenie wartości następujących zmiennych: `alias`, `label`, `optional`, `password`, `ramdisk`, `read-only`, `read-write`, `restricted`, `root` i `vga`.

Aby uruchomić jądro systemu z pliku, jedyną informacją, którą należy podać w pliku konfiguracyjnym, jest nazwa pliku zawierającego jądro systemu, nazywanego dalej plikiem obrazu lub obrazem. Na przykład, aby możliwe było uruchomienie jądra zapisanego w pliku obrazu o nazwie `/linux_main`, trzeba do pliku konfiguracyjnego dodać następujący wiersz:

```
image=/linux_main
```

Aby uruchomić jądro systemu z innego urządzenia, należy wprost podać numery sektorów, które mają zostać odczytane. Istnieje kilka metod wprowadzania tej informacji. Należy podać sektor początkowy, a następnie liczbę sektorów, które należy odczytać (w formacie `start+długość`) albo też numer sektora końcowego (`start-koniec`). Jeśli podana jest tylko wartość sektora początkowego, odczytany zostanie tylko ten jeden sektor.

Przykładowo, aby załadować jądro systemu z dyskietki, zaczynając od sektora 1 i odczytując następne 512 sektorów, należy do pliku konfiguracyjnego dodać następujące wiersze:

```
image=/dev/fd0
range=1+512
```

Dla każdego pliku obrazu można utworzyć więcej niż jedną konfigurację, ponieważ LILO przechowuje informacje konfiguracyjne w osobnym pliku, a nie w samym pliku obrazu. Plik konfiguracyjny może więc zawierać następujące informacje:

```
image=/linux_main
label=linux-hda1
root=/dev/hda1
```

```

image=/linux_main
label=linux-hda3
root=/dev/hda3
image=/linux_main
label=linux-flop
root=/dev/fd0

```

Plik konfiguracyjny o takiej zawartości tworzy trzy różne konfiguracje dla jednego pliku obrazu (`/linux_main`), każdą z innym głównym systemem plików (odpowiednio `/dev/hda1`, `/dev/hda2` i `/dev/hda3`) i o innej etykiecie. Znaki białe (spacja, tabulator etc.) w pliku konfiguracyjnym są ignorowane, więc wcięcia są tylko dla wygody użytkownika.

Tablica parametrów dysku

LILO zazwyczaj potrafi uzyskać informacje o dyskach twardych i stacjach dyskietek odczytując jądro systemu operacyjnego. Niekiedy jednak nie jest to możliwe (szczególnie w przypadku niektórych sterowników SCSI i urządzeń nie zachowujących się ani jak urządzenia SCSI, ani IDE). Jeśli LILO nie może zdobyć informacji o parametrach dysków, generuje komunikat „bad geometry”.

Parametry dysku mogą być odczytane z pliku `/etc/lilo/disktab`. Jeśli taki plik istnieje, dane w nim zawarte mają pierwszeństwo przed tymi pochodząymi z autodetekcji. Plik `/etc/lilo/disktab` zawiera numer urządzenia (szesnastkowy), jego kod BIOS-owy oraz dane o geometrii dysku. Oto przykładowa zawartość pliku `disktab`:

```

# /etc/lilo/disktab - LILO disk parameter table
#
# Dev. num   Bios code   Secs/track   Heads/cyl   Cyls   Part. Offset
#
0x800        0x80       32           64          1714      0
0x801        0x80       32           64          1714      1001

```

Opisane są w nim dwie partycje dysku SCSI. Pierwsza z nich, `/dev/sda1`, ma numer 800, a druga, `/dev/sda2`, 801. Kod BIOS-owy obu tych partycji ma wartość 80 (przedrostek `0x` oznacza, że są to wartości podawane są w systemie szesnastkowym). Dysk ma 32 sektory na ścieżkę, 64 głowice na cylinder i 1714 cylindrów. Ponieważ obie partycje znajdują się na tym samym dysku twardym, parametry te w obu przypadkach są takie same.

Przesunięcie początku partycji (ang. *partition offset*) jest polem opcjonalnym. W tym przypadku pierwsza partycja rozpoczyna się od sektora 0, a druga od 1001. Informacja ta musi być podana, jeśli kernel nie może jej uzyskać inną drogą. Większość dysków twardych (a także dysków wymienialnych i optycznych) nie wymaga podania tych informacji, natomiast potrzebują jej niekiedy dyski CD-ROM.



Dane w pliku `/etc/lilo/disktab` nie muszą być dokładnie zgodne z faktycznym stanem rzeczy. Większość systemów i tak mapuje parametry dysku na 32 sektory na ścieżkę i 64 głowice, bez względu na to, czy jest tak rzeczywiście, czy nie (w taki sposób BIOS podaje systemowi informacje o dysku). Ilość cylindrów musi być co najmniej równa faktycznej (ale może być większa), aby zapobiec obcięciu dostępnego na dysku miejsca rozpoznawanego przez system operacyjny.

ny.

Niektóre wersje BIOS-u nie obsługują dysków o parametrach przekraczających pewne wartości. Jest to problem występujący w kontrolerach SCSI oraz IDE zaprojektowanych do pracy ze stosunkowo małymi dyskami (o pojemności mniejszej niż 1GB) i we wcześniej-szych wersjach BIOS-u. Czasem można ominąć to ograniczenie używając sterowników programowych, ale niektóre systemy nadal mogą mieć problemy z dostępem do plików położonych poza granicą 1 GB.

Wyłączanie i usuwanie programu LILO

Aby zapobiec uruchamianiu systemu za pomocą programu LILO, należy wyłączyć sektor startowy (zmieniając aktywną partycję programem `fdisk`) lub usunąć go całkowicie. Większość wersji LILO może zostać szybko wyłączona poleceniem

```
/etc/lilo/lilo -u
```

Jeśli używasz nowszej struktury katalogów, podstaw odpowiednią ścieżkę dostępu.

Jeżeli chcesz usunąć LILO z głównego rekordu startowego, musisz zapisać tam jakąś inną jego wersję. Przykładowo, jeżeli chcesz zastąpić MBR wersją zapisaną na DOS-owym dysku startowym, uruchom DOS-owy program `fdisk` z opcją `/mbr`.

Ponieważ LILO podczas instalacji tworzy kopię oryginalnej zawartości MBR, można wrócić do starej wersji (o ile nie została ona usunięta). Jeżeli chcesz przywrócić wersję głównego rekordu startowego zapisaną w pliku `boot.0800` (czyli pochodząca z pierwszego dysku SCSI), powinieneś wydać polecenie:

```
dd if=/etc/lilo/boot.0800 of=/dev/sda bs=446 count=1
```

Jeśli używasz innego urządzenia, podstaw jego nazwę zamiast `/dev/sda` i odpowiednio zmień nazwę pliku zawierającego oryginalny MBR.

LILO - rozwiązywanie problemów

Jeżeli LILO z jakichś powodów nie potrafi uruchomić się prawidłowo, wyświetla komunikaty o błędach. Powinny one wystarczyć do identyfikacji problemu. Najczęściej występujące komunikaty i rozwiązania sygnalizowanych przez nie problemów przedstawione są w tabeli 4.4.

Tabela 4.4. Komunikaty o błędach generowane przez program LILO

Komunikat	Polskie tłumaczenie	Rozwiążanie
-----------	---------------------	-------------

Can't put the boot sector on logical partition X	Nie można zapisać sektora startowego na partycji logicznej X	Program LILO próbował zapisać sektor startowy na partycji logicznej, a MBR domyślnie może uruchamiać tylko partieje główne. Można obejść ten problem używając opcji <code>-b</code> i podając bezpośrednio nazwę partycji, z której system ma zostać uruchomiony, bądź też używając zmiennej konfiguracyjnej <code>boot=urządzenie</code> .
Got bad geometry	Zła geometria dysku	Kontroler dysku (przeważnie SCSI) nie pozwala na automatyczną detekcję jego geometrii. Właściwe dane podać należy w pliku <code>/etc/lilo/disktab</code> .
Invalid partition table, entry X	Błąd w tablicy partycji, wpis X	Adres sektor/głowica/cylinder i adres liniowy pierwszego sektora partycji nie zgadzają się. Zdarza się to zazwyczaj gdy system operacyjny tworzy partycje nie rozpoczynające się od początku ścieżki. Spróbuj użyć opcji <code>fix-table</code> .
First sector doesn't have a valid boot signature	Pierwszy sektor nie posiada znacznika sektora startowego	Pierwszy sektor prawdopodobnie nie jest prawidłowym sektorem startowym. Sprawdź nazwę urządzenia i ewentualnie uruchom jeszcze raz program LILO, by zainstalować sektor startowy.
Cylinder number too big	Zbyt duży numer cylindra	Plik położony jest poza 1024 cylindrem i LILO nie ma do niego dostępu ze względu na ograniczenia BIOS-u.

Tabela 4.4. cd. Komunikaty o błędach generowane przez program LILO

Komunikat	Polskie tłumaczenie	Rozwiązanie
XXX doesn't have a valid LILO signature	XXX nie pochodzi z programu LILO	XXX został zlokalizowany, ale nie jest zgodny z programem LILO. Jeśli XXX to sektor startowy, powinieneś użyć opcji <code>-I</code> lub <code>install</code> programu LILO, by zainstalować wersję zgodną z programem LILO.
XXX has an invalid stage code	Nieprawidłowy kod kontrolny	Wpis został uszkodzony. Uruchom ponownie LILO.
Kernel XXX is too big	Jądro systemu jest za duże	Kernel jest większy niż 512 kB i LILO nie umie sobie z nim poradzić. Usuń kilka niepotrzebnych sterowników i przekompiluj jądro.
Partition entry not found	Partycja nie odnaleziona	Partycja nie występuje w tablicy partycji.
Sorry, don't know how to handle device XXX	Przykro mi, ale nie potrafię obsługiwać urządzenia XXX	LILO nie potrafi ustalić parametrów dysku. Należy je podać w pliku <code>/etc/lilo/disktab</code> .

Podsumowanie

Ten rozdział zawiera wszystkie informacje potrzebne do tego, by zainstalować i poprawnie używać programu LILO oraz tworzyć własne sektory startowe systemu Linux. LILO jest programem dość elastycznym i potrafi z łatwością obsłużyć kilka różnych konfiguracji, dzięki czemu łatwo dostosować proces uruchamiania systemu do własnych potrzeb.

Choć program LILO używany jest prawie wyłącznie podczas pierwszej instalacji systemu i po wprowadzeniu zmian do kernela, powinieneś znać podstawowe zasady jego działania, tak abyś wiedział, w jaki sposób wykorzystuje on dyski twardye i sektory startowe. Ma to szczególne znaczenie, gdy używasz kilku systemów operacyjnych.

Aby dowiedzieć się więcej o pracy z systemem linuxowym, przejdź do rozdziału 6. „Od czego zacząć”.

Jeśli chcesz skonfigurować system X i pracować z interfejsem graficznym, przeczytaj rozdział 22. „Instalacja i konfiguracja Xfree86”.

O administrowaniu systemem dowiesz się więcej z rozdziału 32. „Podstawy administracji systemem”.

Rozdział 5.

Podsumowanie instalacji

Tim Parker

W tym rozdziale:

- υ Uruchamianie Linuxa
- υ Instalacja dodatkowego oprogramowania
- υ Systemy z kilkoma urządzeniami CD-ROM

Po zainstalowaniu systemu i skonfigurowaniu programu LILO wszystko powinno działać prawidłowo. Niestety, systemy komputerowe mają tendencję do niewłaściwych zachowań, nawet jeśli Ty robisz wszystko jak należy. W tym rozdziale zajmiemy się kilkoma zagadnieniami związanymi z tym problemem: sprawdzaniem, czy system jest poprawnie zainstalowany, oraz używaniem narzędzi do instalacji nowego oprogramowania. Zakończymy go kilkoma słowami o używaniu w systemie więcej niż jednego dysku CD-ROM.

Uruchamianie Linuxa

Jeśli przeczytałeś dwa poprzednie rozdziały, to zainstalowałeś już Linuxa (prawdopodobnie z dysku CD-ROM) oraz program LILO. Po załączeniu lub zresetowaniu system powinien uruchamiać Linuxa automatycznie (lub dawać możliwość uruchomienia go, zależnie od konfiguracji). Jeżeli tak nie jest, problem prawdopodobnie tkwi w konfiguracji LILO.

Najczęstszą przyczyną niepowodzeń jest fakt, że w obszarze Master Boot Record lub na początku partycji startowej nie zostały zapisane instrukcje systemu Linux, pozwalające na jego uruchomienie. Rozdział poświęcony programowi LILO wyjaśnia, jak usunąć taki problem.

Czasem zdarza się, że system uruchamia się prawidłowo, ale nie działają niektóre urządzenia. Rzadko ma to miejsce w przypadku dysku twardego, stacji dysków czy CD-ROM-u, ale jest to dość częsty problem z kartami dźwiękowymi, sieciowymi, zewnętrznymi sterownikami SCSI oraz takimi urządzeniami jak drukarki czy skanery. Przeważnie jest on spowodowany tym, że jądro systemu nie potrafi porozumieć się z danym urządzeniem.

Procedura uruchamiania awaryjnego

Co dzieje się, gdy zainstalowałeś Linuxa i program LILO (albo zapomniałeś to zrobić) i resetujesz komputer? Albo ładuje się system operacyjny, widzisz niewesoły komunikat typu „No OS” czy „Wrong disk. Replace and press enter” generowany przez BIOS. Czy w takiej sytuacji musisz przeprowadzać instalację od początku? Na szczęście nie, o ile tylko posiadasz dyski boot i root, które utworzyłeś na początku instalacji.

Uruchom komputer z dysku boot. System powita Cię znajomym komunikatem `boot:`. Teraz należy powiedzieć Linuxowi, że powinien załadować jądro systemu z dysku twardego, a nie z dyskietki, podając lokalizację jądra systemu. Zazwyczaj można to zrobić, podając nazwę partycji, na przykład `/dev/hda1`. Polecenie, jakie musisz wydać by uruchomić system z tej partycji, to:

```
boot ro root=/dev/hda1
```

W ten sposób informujesz LILO, że powinien znaleźć jądro systemu na partycji `/dev/hda1`. Należy oczywiście użyć właściwej dla systemu nazwy partycji, a jądro systemu musi się na niej znajdować.

Po uruchomieniu systemu z dyskietki, należy ponownie uruchomić program LILO, dzięki czemu zawartość sektora startowego zostanie odświeżona i system będzie uruchamiał się z dysku twardego.

Program dmesg

Jeśli posiadasz jakieś urządzenia, które nie są prawidłowo rozpoznawane po uruchomieniu komputera, powinieneś przejrzeć komunikaty diagnostyczne generowane przez system podczas uruchamiania. Można to zrobić, używając polecenia `dmesg`. Jeśli urządzenie nie zostało prawidłowo rozpoznane, zobaczysz jedną lub więcej informacji dotyczących go. Przykładowo, jeśli karta sieciowa nie została rozpoznana prawidłowo, możesz zobaczyć następujące komunikaty:

```
loading device 'eth0'  
ne.c:v1.10 9/23/94 Donald Becker (becker@cesdis.gsfc.nasa.gov)  
NE*000 ethercard probe at 0x300: 00 00 6e 24 1e 3e  
eth0: NE2000 not found at 0x300  
eth0 not loaded
```

Jak wspomniano wcześniej, błędy te spowodowane są prawie zawsze podaniem złych informacji konfiguracyjnych. W powyższym przykładzie, prawidłowym adresem I/O karty był adres 330H, a nie, jak podano, 300H. Ponowne uruchomienie programu instalacyjnego umożliwia zmianę tych parametrów.

Zmiana rozmiaru partycji

Jeżeli chcesz zmienić rozmiary partycji na dysku twardym, z pomocą mogą przyjść Ci programy komercyjne, takie jak np. Partition Magic firmy PowerQuest Corporation. Istnieje też darmowe narzędzie spełniające znakomicie to zadanie. Jest nim FIPS, program

działający pod kontrolą systemów DOS i Windows. Pozwoli on zmienić wielkość partycji bez utraty zapisanych na niej danych. Przed uruchomieniem tego programu należy zdefragmentować system plików.

Jeśli zainstalowałeś już Linuxa, możesz uruchomić system z DOS-owej dyskietki startowej i uruchomić program FIPS z płyty CD-ROM zawierającej dystrybucję Linuxa. Możesz też skopiować ten program na dyskietkę startową pod kontrolą systemu Linux. Nie uruchamiaj programu FIPS z katalogu linuxowego, nawet pod kontrolą emulatora systemu DOS, ponieważ może to mieć nieprzewidziane i katastrofalne wręcz skutki.

Instalacja dodatkowego oprogramowania

Twój system linuxowy jest już w pełni sprawny, więc teraz pewnie chciałbyś zainstalować jakieś ciekawe programy. Istnieje kilka źródeł oprogramowania, takich jak płyty CD-ROM, strony WWW i węzły FTP. Czasem dodatkowe oprogramowanie rozprowadzane jest również przez dystrybutorów. W niektórych przypadkach do pakietu dołączany jest program instalacyjny, po którego uruchomieniu cały proces instalacji zostanie przeprowadzony automatycznie. Jest tak zazwyczaj w przypadku aplikacji komercyjnych; kilku takim aplikacjom przyjrzymy się bliżej w części ósmej.

Aby zainstalować inne aplikacje, szczególnie te pochodzące z CD-ROM-ów z dystrybucją Linuxa, należy użyć programu narzędziowego, takiego jak `installpkg`, `pkgtool` lub `pkgadd`. Programy tego typu mają różne nazwy, w zależności od wersji Linuxa, powinieneś więc zatrzymać się w dokumentacji dołączonej do dystrybucji, której używasz. Poniżej przedstawimy dwa przykłady instalacji oprogramowania, w oparciu o dwa różne programy. Pierwszy z nich to RPM (Red Hat Package Manager), a drugi – `installpkg` (dostępny z wieloma wersjami Linuxa).

RPM

RPM jest programem dołączonym do kilku różnych wersji Linuxa, nie tylko do wersji Red Hat. Jest to program oparty na interfejsie tekstowym, pozwalający instalować, budować, sprawdzać, aktualniać i odinstalowywać pakiety oprogramowania (rozumiane jako archiwa zawierające pliki programu i wszystkie dodatkowe informacje o pakiecie, takie jak jego nazwa, wersja i krótki opis). RPM posiada dziesięć różnych trybów pracy, a każdy z nich posiada oddzielny zestaw opcji. Nie ułatwia to posługiwania się tym programem, choć w większości zastosowań jest on całkiem prosty w użyciu. Tryby pracy RPM i opis ich zastosowania zebrano w tabeli 5.1.

Nie będziemy zbyt szczegółowo omawiać wszystkich dostępnych opcji, ponieważ w większości przypadków ich użycie jest oczywiste, a poza tym są one doskonale opisane w dokumentacji. Podamy tylko kilka przydatnych w praktyce przykładów, które przybliżą nieco zastosowanie tego programu. Aby zainstalować pakiet, należy podać jego nazwę. Składnia polecenia instalującego pakiet jest następująca:

```
rpm -i [opcje] <nazwa_pakietu>
```

Tabela 5.1. Tryby pracy programu RPM i opis ich zastosowania

Tryb	Polecenie	Zastosowanie
Budowanie	rpm -[b t] 0	Tworzy pakiet
Poprawianie praw dostępu	rpm --setperms	Poprawia prawa dostępu plików wchodzących w skład pakietu
Instalacja	rpm -i	Dodaje nowe oprogramowanie
Zapytanie	rpm -q	Pozwala sprawdzić, jakie pliki i pakiety są zainstalowane
Odbudowywanie bazy danych	rpm --rebuilddb	Odbudowuje bazę danych o pakiecie
Ustawianie właścicieli i grup	rpm --setugids	Ustawia właściciela i grupę pakietu
Wyświetlanie RC	rpm --showrc	
Sprawdzanie sygnatury	rpm --checksig	Sprawdza, czy w pakiecie nie występują błędy
Odinstalowywanie	rpm -e	Usuwa pakiet
Weryfikacja	rpm -V -y --verify	Sprawdza, czy pakiet jest prawidłowo zainstalowany (czy wszystkie pliki są na swoich miejscach)

Większość opcji konfiguracyjnych (których jest kilkadziesiąt) tylko nieznacznie zmienia zachowanie programu. Oto bardziej użyteczne opcje.

- allfiles** Instaluje lub uaktualnia wszystkie pliki wchodzące w skład pakietu.
- force** To samo co `replacepkgs`, `replacefiles` i `oldpackage` razem.
- h lub --hash** Wyświetla 50 znaków #, pokazujących postępy podczas instalacji (używany razem z opcją v pozwala uzyskać ładnie sformatowane wyświetlanie statusu).
- includedocs** Instaluje pliki dokumentacji (zazwyczaj opcja załączona domyślnie).
- ignoreos** Wymusza instalację lub uaktualnienie nawet w przypadku, gdy pakiet przeznaczony jest dla innego systemu operacyjnego.
- keep-temp** Zapobiega usuwaniu plików tymczasowych tworzonych podczas instalacji. Opcja przydatna tylko przy wyszukiwaniu błędów występujących podczas instalacji.
- percent** Pokazuje postępy w instalacji, podając procentowo liczbę rozpakowanych plików (opcja ta ma ułatwiać wywoływanie programu RPM ze skryptów).
- quiet** Powoduje wyświetlanie minimalnej ilości komunikatów (nadal wyświetlane są wszystkie komunikaty o błędach).

replacefiles	Powoduje zainstalowanie pakietu nawet wtedy, gdy instalowane pliki muszą zostać zapisane na miejsce plików pochodzących z innych pakietów.
replacepkgs	Powoduje zainstalowanie pakietu nawet wtedy, jeśli był on już zainstalowany.
test	Nie instaluje pakietu, sprawdzając tylko występowanie potencjalnych konfliktów.

Jeśli chcesz uzyskać pełną listę opcji, przejrzyj strony `man`. Nazwa pakietu jest zwykle pełną ścieżką dostępu, choć w większości wersji RPM możliwe jest też podanie adresu URL. Żeby zainstalować np. pakiet `xgames`, można wydać następujące polecenie:

```
rpm -i /mnt/cdrom/col/install/Xgames.rpm
```

W tym przypadku pakiet nazywa się `Xgames.rpm` i podana jest pełna ścieżka dostępu do niego (znajduje się on na płycie CD-ROM w katalogu `col/install`).

Tryb weryfikacji jest przydatny w momencie, gdy chcesz upewnić się, że jakaś część oprogramowania została zainstalowana prawidłowo lub chcesz sprawdzić, czy jest ona wciąż dostępna w systemie. Jest to czasem niezbędne, gdyż zdarza się, że przy instalacji nowego oprogramowania nadpisywane są istniejące pliki. Podczas weryfikacji porównywane są nazwy, rozmiar, sumy kontrolne, prawa dostępu, typ, identyfikator właściciela i grupy każdego z plików. Każda rozbieżność z informacją zawartą w samym pakiecie jest sygnalizowana.

Aby sprawdzić, czy pakiet jest zainstalowany prawidłowo, użyj opcji `-w` i jego nazwy. RPM sygnalizuje błędy używając ośmiu symboli:

- 5** nieprawidłowa suma MD5 (kontrolna),
- S** nieprawidłowy rozmiar pliku,
- L** nieprawidłowe dowiązanie symboliczne,
- T** nieprawidłowy czas,
- D** nieprawidłowe urządzenie,
- U** nieprawidłowy właściciel,
- G** nieprawidłowa grupa,
- M** nieprawidłowy tryb (prawa dostępu i typ pliku).

Jeśli w pakiecie występują jakieś błędy, zazwyczaj najprostszym rozwiązaniem jest przeinstalowanie go.

installpkg

Program `installpkg` został przeniesiony z systemu UNIX, gdzie jest stosowany od lat. Może być używany do instalowania pakietów podobnie jak program RPM. Analogiczne jak w poprzednim przykładzie, polecenie instalujące pakiet `xgames` ma następującą postać:

```
installpkg /usr/col/Xgames
```

Domyślnie program `installpkg` szuka pakietów z rozszerzeniem `.tgz`. Jeśli pakiet, który chcesz zainstalować, ma inne rozszerzenie, a wiesz, że program `installpkg` sobie z nim poradzi, możesz podać pełną nazwę, zawierającą również rozszerzenie. Podane wyżej przykładowe polecenie spowoduje zainstalowanie wszystkich plików z pakietu `/usr/col/Xgames.tgz`.

Program `installpkg` instaluje oprogramowanie do katalogu bieżącego, więc pakiet `Xgames` zostanie zainstalowany w tym katalogu, w którym polecenie zostało wydane.

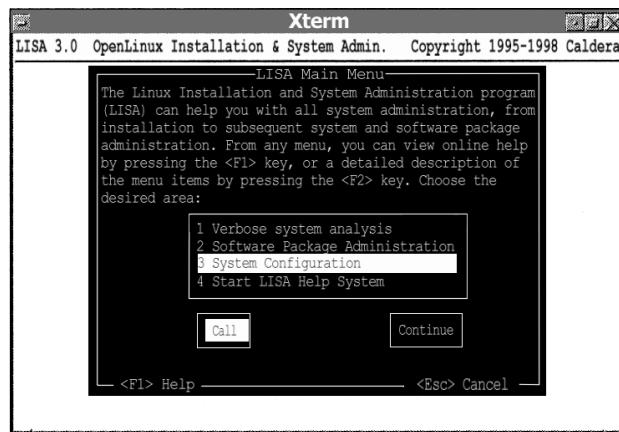
Inne programy instalujące pakiety oprogramowania

Wiele wersji Linuxa wypracowało sobie własne systemy zarządzania pakietami oprogramowania. Mogą one mieć zarówno interfejs tekstowy (tak jak programy `installpkg` i RPM) lub graficzny. Dobrym przykładem narzędzia pracującego w trybie graficznym jest program LISA (Linux Installation and System Administration). Jest to program rozwijany z dystrybucją Caldera OpenLinux (Slackware) i kilkoma innymi wersjami Linuxa. Może być używany zarówno w trybie tekstowym, jak i graficznym. Oferuje system menu ułatwiający posługiwanie się nim – jest to jego główna przewaga nad programami `installpkg` i RPM.

Po uruchomieniu programu LISA możesz wybrać, czy chcesz zajmować się administrowaniem systemem, czy też zarządzaniem pakietami (rys. 5.1). Klawisze kurSORA pozwalają wybrać jedną z pozycji menu.

Rysunek 5.1.

Interfejs oparty na systemie menu (program LISA) jest znacznie wygodniejszy niż obsługa z wiersza poleceń



Z menu głównego wybierz opcję zarządzania oprogramowaniem (ang. *Software Package Administration*). Pojawi się nowe menu (pokazane na rys. 5.2), pozwalające przeglądać zainstalowane pakiety i dodawać nowe.

Rysunek 5.2.

*LISA pozwala
przeglądać
zainstalowane
pakiety
oprogramowania
i dodawać nowe*



Jeśli chcesz sprawdzić, jakie pakiety są aktualnie dostępne, LISA pokaże Ci listę standardowych pakietów o których „wie” Linux. Rysunek 5.3 pokazuje główną listę dostępnych pakietów. Dokładniejsze informacje o każdym z nich można otrzymać wybierając je z menu. Nowe pakiety dodać można wkładając np. płytę CD-ROM i podając pełną ścieżkę dostępu do nich, a następnie wydając polecenie wyszukiwania nowych pakietów. Potem wystarczy tylko wybrać instalację nowego oprogramowania, a LISA automatycznie przeprowadzi pozostałą część procesu instalacji, podobnie jak programy RPM czy `installpkg`.

Rysunek 5.3.
Główne pakiety
oprogramowania
rozprowadzane z taq
wersją Linuksa



Systemy z kilkoma urządzeniami CD-ROM

Teoretycznie nie istnieje górnna granica liczby podłączonych dysków CD-ROM, które mogą być obsługiwane przez Linuksa. Dopóki wystarcza numerów urządzeń (numery główne i poboczne omówione są w rozdziale 33. „Urządzenia”), urządzenie może zostać podłączone. Do czego przydaje się kilka urządzeń CD-ROM? Głównie do obsługi zmieniarzy płyt CD-ROM, nagrywarki i bibliotek przechowywanych na dyskach CD-ROM. Przyjrzymy się bliżej temu problemowi.

Zmieniarze dysków CD-ROM

Najpopularniejszym powodem użycia kilku urządzeń CD-ROM jest zmieniarz płyt kompaktowych. Pozwala ona na załadowanie do odtwarzacza trzech, sześciu, czy jeszcze większej liczby płyt CD-ROM, ale tylko jedna z nich może być odczytywana w danym momencie. Przykładem popularnych zmieniarzy są urządzenia z serii Multispin firmy NEC, które pozwalają na obsługę kilku płyt CD-ROM, zmieniając je na żądanie po otrzymaniu polecień programowych lub po naciśnięciu przycisku na przednim panelu urządzenia.

Zmieniarki skonfigurowane do pracy pod kontrolą Windows, mają zazwyczaj przypisaną osobną literę dysku dla każdej załadowanej płyty. Na przykład, jeśli zmieniarka obsługuje sześć płyt, a ostatnim dyskiem w komputerze jest `c:`, to zostaną jej przypisane litery od `d:` do `i:`. Nie zmienia to faktu, że nie można korzystać z dwóch płyt równocześnie; po prostu przy zmianie litery dysku zmieniarka automatycznie podsuwa czytnikowi odpowiednią płytę.

Linux potrafi obsłużyć zmieniarkę na dwa sposoby. Można skonfigurować ją tak, aby była widziana jako jedno urządzenie CD-ROM i ręcznie (za pomocą przycisków na przednim panelu) zmieniać płyty, można również przydzielić każdej z płyt osobną nazwę urządzenia. Pierwsza metoda jest prostsza, ale druga jest wygodniejsza w użyciu i daje więcej możliwości, jest ona również bardziej elastyczna. Ponieważ Linux nie obsługuje zmiany płyty CD-ROM w czasie, gdy jest ona zamontowana, należy ją koniecznie odmontować przed wymianą, bez względu na to, którą opcję konfiguracji wybierzesz. Aby skonfigurować zmieniarkę jako kilka urządzeń, powtórz instalację dysku CD-ROM, za każdym razem zwiększając numer urządzenia (`/dev/cd0`, `/dev/cd1` itd.). Tylko jedno z tych urządzeń może być dołączone do `/dev/cdrom`.

Nagrywarki CD-ROM

Nagrywarki płyt CD to następny powód, by zainstalować w systemie linuxowym więcej niż jedno urządzenie CD-ROM. Pozwalają one na stworzenie własnego dysku CD-ROM, zazwyczaj w jednym podejściu, choć napędy pozwalające na zapis wielokrotny stają się również coraz popularniejsze. Producenci tych urządzeń nie zapewniają kompatybilności z systemem Linux, ale dla większości z nich dostępne są odpowiednie sterowniki – przeważnie napisane przez samych użytkowników.

Nagrywarki mogą być używane jako zwykłe napędy CD bez żadnych dodatkowych zabiegów. Ponieważ większość z nich działa w oparciu o interfejs SCSI, instalacja przebiega tak samo jak w przypadku napędu CD-ROM tego typu. Jeśli chcesz skonfigurować urządzenie jako nagrywarkę, potrzebowałbyś wspomnianych wcześniej sterowników, dostępnych zazwyczaj w węzłach FTP. Ponieważ kopiowanie płyt CD jest najczęstszym sposobem wykorzystania nagrywarki, możesz ją podłączyć jako drugie urządzenie (`/dev/cd1`), jako pierwsze podłączając zwykły napęd CD-ROM. Oba napędy (obojętnie czy IDE, czy SCSI) mogą być używane jednocześnie.

Biblioteki na płytach CD-ROM

Jeszcze jednym powodem używania wielu urządzeń CD-ROM jest konfiguracja, w której zachowują się one jak biblioteka danych. Przypuśćmy, że posiadasz dużą ilość danych, które muszą być dostępne w sieci, czy też dokumentację, którą chcesz udostępnić w Internecie. Jakiekolwiek byłyby powody, użycie kilku dysków CD-ROM jest w takich przypadkach lepsze, niż stosowanie dysku twardego o bardzo dużej pojemności (na pewno pod względem kosztów).

W takim przypadku należy po prostu skonfigurować kolejno urządzenia (`/dev/cd0`, `/dev/cd1` itd.) i zamontować je w odpowiednich miejscach systemu plików. Użytkownicy korzystający z serwera zostaną automatycznie skierowani do odpowiednich dysków.

Zmienianie płyt CD-ROM

Jeśli dysk CD-ROM ma być dostępny dla systemu, musi zostać zamontowany za pomocą polecenia:

```
mount -t typ urządzenie punkt_zamontowania
```

typ to dowolna instrukcja odnosząca się do dysków CD-ROM (jak określenie systemu plików ISO9660 lub załaczenie trybu read-only), urządzenie to nazwa urządzenia, a punkt_zamontowania to miejsce w głównym systemie plików, w którym dostępne będą dane zapisane na dysku CD-ROM. Przykładowo, aby zamontować CD-ROM zapisany w standardzie ISO9660, znajdujący się w pierwszym napędzie, gdy punktem zamontowania ma być katalog /usr/cdrom, należy wydać następujące polecenie:

```
mount -t iso9660 /dev/cd0 /usr/cdrom
```

Od tej chwili wszystkie żądania odczytu z katalogu /usr/cdrom i jego podkatalogów zostaną skierowane do dysku CD-ROM. Kiedy chcesz zmienić płytę na inną, nie wolno po prostu wcisnąć przycisku Eject (większość CD-ROM-ów zresztą nie zareaguje na to). Należy najpierw odmontować włożoną płytę poleceniem umount, jako parametr podając punkt zamontowania lub nazwę urządzenia:

```
umount /dev/cd0
```

lub

```
umount /usr/cdrom
```

Oba powyższe polecenia dają identyczny efekt: odmontowują system plików zapisany na dysku CD-ROM.

Podsumowanie

W tym rozdziale przyjrzaliśmy się narzędziom służącym do zarządzania pakietami oprogramowania, omówiliśmy dodawanie nowego oprogramowania i konfigurowanie więcej niż jednego urządzenia CD-ROM. Kończy on część poświęconą instalacji systemu Linux. Teraz, gdy system jest już zainstalowany, możesz przejść do innych rozdziałów (niekoniecznie zachowując kolejność).

O instalowaniu kart dźwiękowych dowiesz się więcej z rozdziału 21. „Linux i multimedia”

Jak zainstalować i skonfigurować Xfree86, serwer X, który umożliwi korzystanie z interfejsu graficznego, powie Ci rozdział 22. „Instalacja i konfiguracja XFree86”.

Aby dowiedzieć się więcej o urządzeniach CD-ROM, głównych i pobocznych numerach urządzeń, oraz o tym, jak zainstalować nowe urządzenia, przeczytaj rozdział 33. „Urządzenia”.

Część druga

Poznawanie Linuxa

W tej części:

- υ Od czego zacząć
- υ Podstawowe polecenia
- υ System plików
- υ Prawa dostępu do plików i katalogów
- υ Programy użytkowe projektu GNU
 - υ bash
 - υ pdksh
 - υ tcsh
- υ Programowanie w języku powłoki
- υ FTP i Telnet

Rozdział 6.

Od czego zacząć?

Ed Trevis i Tim Parker

W tym rozdziale:

- υ Uruchamianie i zamykanie systemu
- υ Co to znaczy zalogować się?
- υ Hasła
- υ Zakładanie nowego konta
- υ Wypróbowywanie nowego konta
- υ Polecenie who
- υ Terminale wirtualne
- υ Polecenia i programy

Uruchamianie i zamykanie systemu

Proces uruchamiania Linuxa zależy od konfiguracji programu LILO. W najprostszym przypadku uruchamia się on automatycznie z dysku twardego lub dyskietki. Jeśli w systemie zainstalowane są inne wersje Linuxa lub inne systemy operacyjne, to aby system został załadowany, należy zazwyczaj po uruchomieniu komputera wprowadzić jakąś etykietę.

Podczas inicjalizacji systemu Linux przez ekran przewija się mnóstwo komunikatów. Są to informacje o wykrytym sprzęcie oraz o sterownikach programowych ładowanych przy uruchamianiu. Po zakończeniu sekwencji startowej powinieneś zobaczyć komunikat

darkstar login:

Nazwa na początku tego komunikatu prawdopodobnie będzie inna; zależy ona od wersji instalowanego oprogramowania. Niektóre systemy pokazują również informację o wersji, na przykład tak:

```
Welcome to Linux 1.2.13  
darkstar login:
```

Inna postać tych komunikatów z pewnością nie jest powodem do zmartwień. Różnią się one w zależności od dystrybucji. Po zalogowaniu się zawsze wyświetlany jest jakiś znak zachęty, na przykład symbol dolara.



System linuxowy zawsze musi zostać prawidłowo zamknięty. Niewłaściwe zakończenie działania systemu, np. przez wyłączenie komputera, może spowodować poważne uszkodzenie systemu plików. Procedura zamykania systemu opisana jest w następującym podrozdziale. Jeśli zdarzy Ci się rozpocząć przez przypadek uruchamianie Linuxa, powinieneś pozwolić na dokonanie sekwencji startowej, a następnie zamknąć system. Przerywając sekwencję startową, ryzykujesz utratę danych.

Wiesz już, jak uruchomić Linuxa, ale jeszcze ważniejsze jest, by wiedzieć, jak go prawidłowo zamknąć. Podobnie jak w większości systemów pozwalających na jednoczesną pracę wielu użytkowników (np. UNIX czy Windows NT), nieprawidłowe zamknięcie Linuxa może skończyć się uszkodzeniem pojedynczych plików, lub, co gorsza, całego systemu plików. Spowodowane jest to faktem, że Linux przechowuje najświeższe informacje o strukturze systemu plików (tablica I-node) w pamięci, w celu przyspieszenia operacji dyskowych. Podczas prawidłowego zamknięcia systemu wersja przechowywana w pamięci jest zapisywana na dysku. Jeśli tablica I-node nie zostanie prawidłowo zapisana, wartość dysku twardego może nie zgadzać się z danymi ją opisującymi. Rezultaty takiej sytuacji są nieprzewidywalne. Poza tym Linux używa dysków twardych w sposób ciągły, niezależnie od tego, czy akurat korzystasz z systemu, czy nie. Przerwanie zapisu może również spowodować uszkodzenie systemu plików. Prawidłowe zamknięcie systemu wymusza najpierw zakończenie wszystkich zapisów na dysk.

Najłatwiejszym sposobem na poprawne zamknięcie systemu jest użycie klawiszy *Alt+Control+Delete*. Na ekranie pojawi się seria komunikatów. Po zakończeniu procedury zamykającej system uruchomi się ponownie. Cała operacja zajmuje kilka sekund. Jest szybka i łatwa, a jednocześnie chroni przed uszkodzeniem systemu plików.

Z metodą opisaną powyżej wiąże się jednak pewne niebezpieczeństwo: wszyscy użytkownicy zalogowani do systemu zostaną brutalnie odłączeni, tracąc wszystkie niezapisane dane. Należy więc uprzedzić wszystkich użytkowników przed jej zastosowaniem. Aby sprawdzić, kto aktualnie jest zalogowany, można użyć polecenia *who* (jest ono opisane dokładniej w dalszej części tego rozdziału).

Polecenia zamykające Linuxa

Istnieje również kilka innych metod wyłączenia systemu. Najczęściej spotykane polecenia służące do wykonania tego zadania to *shutdown*, *haltsys* i *fasthalt* (ich nazwy

mogą się różnić w zależności od wersji oprogramowania). Musisz sam sprawdzić, które z nich są obsługiwane w Twoim systemie.

Drugim z najpopularniejszych sposobów zamykania systemu jest użycie polecenia `shutdown`. Jego składnia pozwala na podanie kilku parametrów, na przykład czasu, po jakim ma nastąpić wyłączenie, i czy po wyłączeniu powinno nastąpić ponowne uruchomienie systemu. Parametr określający czas jest w większości wersji wymagany. Przykładowo, polecenie

```
shutdown -t45
```

powoduje zamknięcie systemu po 45 sekundach od jego wydania. Niektóre wersje nie wymagają przełącznika `-t`, interpretując liczbę jako ilość sekund, jaką mają oczekwać. Jeśli chcesz zamknąć system natychmiast, powinieneś wpisać liczbę 0 lub też (w niektórych systemach) użyć opcji `now`:

```
shutdown now
```

Opcja `-r` powoduje restart systemu po zamknięciu go. Może być ona użyteczna podczas testowania nowych konfiguracji, na przykład dołączania nowych wersji sterowników. Listę innych dostępnych opcji można zazwyczaj uzyskać, wpisując polecenie `shutdown` bez parametrów. Również strony systemu pomocy `man` zawierają odpowiednie informacje.

Polecenia `fasthalt` i `haltsys` zostały przeniesione z innych wersji UNIX-a i, jak wskazują ich nazwy, służą do natychmiastowego zamknięcia systemu.

Co to znaczy zalogować się?

Po uruchomieniu systemu Linux pozwala Ci się zalogować. Twój identyfikator, a raczej identyfikator Twojego konta, dzięki któremu system wie, kto aktualnie go używa, nazywany jest po angielsku *login*. Linux pamięta zestaw identyfikatorów osób, których dostęp do systemu jest dozwolony. Bez znajomości jednego z nich i towarzyszącego mu hasła dostęp do systemu nie jest możliwy (a przynajmniej nie powinien być).

W obrębie jednego systemu identyfikatory użytkowników nie mogą się powtarzać, zwykle każdy z nich ma przypisane inne hasło. Hasło funkcjonuje podobnie jak numer identyfikacyjny w banku: pozwala systemowi zweryfikować tożsamość użytkownika. Różni użytkownicy mają różne prawa dostępu. Niektórzy z nich mogą robić z systemem co im się tylko zamarzy, inni zaś mogą mieć dostęp tylko do wyznaczonych danych. Przywileje te są ustalane przez administratora systemu (osobę zakładającą nowe konta).



Identyfikator użytkownika zazwyczaj odpowiada w jakiś sposób jego prawdziwemu nazwisku czy imieniu. W większych systemach może on na przykład składać się z kombinacji pierwszej litery imienia i nazwiska, na przykład `tparker` czy `rmaclean`. Mniejsze systemy zazwyczaj są mniej formalne i identyfikatorami mogą być zdrobnienia imion, iniciały itp. Pamiętaj, że identyfikator typu `szef` nic nie mówi

innej osobie korzystającej z systemu i może być przyczyną nieporozumień.

Choć w obrębie jednego systemu identyfikatory nie mogą się powtarzać, można stworzyć konta dla użytkowników o takich samych imionach i nazwiskach po prostu zmieniając jedną czy dwie litery; będą one traktowane przez system jako zupełnie oddzielne.

Nic natomiast nie stoi na przeszkodzie, by jeden użytkownik posiadał dwa lub więcej kont. W rzeczywistości, będąc administratorem systemu, powinieneś posiadać jedno konto z uprawnieniami administratora i jedno jako zwykły użytkownik.

Nawet jeśli podasz niewłaściwy identyfikator użytkownika, na przykład imię swojego psa (niewłaściwy, czyli nie rozpoznawany przez system), i tak zostaniesz zapytany o hasło. Cokolwiek wprowadzisz, Linux nie pozwoli na dostęp do systemu, wypisując komunikat `Login incorrect`. Taki sam komunikat zostanie wyświetlony, jeśli podasz właściwy identyfikator, ale nieprawidłowe hasło.

Jedyne konto dostępne po instalacji systemu, o identyfikatorze `root`, posiada pełne prawa dostępu, w związku z czym jego używanie wiąże się z różnymi bezpieczeństwami. W podrozdziale „Tworzenie nowego konta” opisana zostanie procedura zakładania konta bezpiecznego w użyciu. Identyfikator takiego konta może być dowolny. Niektóre wersje programu instalującego Linuxa wymuszają utworzenie takiego konta już podczas instalacji.



Dlaczego Linux pyta o hasło nawet wtedy, gdy podany identyfikator nie jest prawidłowy? Powodem takiego zachowania jest fakt, że Linux używa programu `login`, który najpierw żąda wprowadzenia identyfikatora i hasta, a dopiero potem sprawdza ich poprawność. Następnie albo wyświetla komunikat o tym, że dostęp do systemu nie został przyznany, albo przekazuje sterowanie do programu obsługującego dalszą część procesu logowania.

Dlaczego nie powinieneś używać konta root

Od czasu do czasu użycie konta `root` jest niezbędne, ponieważ tylko użytkownik `root` posiada pełne uprawnienia. Nie powinieneś go jednak używać podczas normalnej pracy, a już na pewno nie wtedy, gdy chcesz eksperymentować z nieznanymi poleceniami. Ograniczone prawa dostępu zapewniają wtedy wzgłeńne bezpieczeństwo systemu.

Linux, jak już wiesz, jest systemem wielozadaniowym i wieloużytkownikowym. W tym samym czasie z systemu może korzystać kilka osób (oczywiście za pośrednictwem terminali). Wielozadaniowość oznacza, że Linux może wykonywać jednocześnie więcej niż jeden program. Przykładowo, można sprawdzać pisownię w dokumencie, jednocześnie ładując z Internetu jakąś dokumentację itp. (możliwość korzystania z systemu przez wielu użytkowników wymusza wielozadaniowość, ponieważ każdy użytkownik musi mieć możliwość uruchamiania różnych programów). Linux pod tym względem

działa znakomicie, nie pozwalając na uszkodzenie systemu czy wyników pracy innych użytkowników.



Konto `root` nie nakłada na użytkownika żadnych ograniczeń. Oznacza to, że używając jakiegoś polecenia, celowo czy przypadkowo możesz zniszczyć cały system plików Linuxa. Dlatego używaj tego konta tylko wtedy, gdy jest to naprawdę konieczne. Unikaj eksperymentowania, gdy jesteś zalogowany jako `root`.

Jeśli zalogujesz się jako `root`, masz pełną kontrolę nad systemem. Identyfikator ten nazywa się też czasem – nie bez powodu – `superuser`. Używając porównania, zwykłego użytkownik jest pasażerem samolotu, zaś `root` jego pilotem. Jest odpowiedzialny za załogę, pasażerów i cały sprzęt. Pytanie „Hmm, do czego to służy?” staje się w tym momencie naprawdę niebezpieczne.

W systemach UNIX-owych zdarzają się sytuacje, że nowy administrator po zainstalowaniu systemu i zalogowaniu się jako `root` niszczy go po kilku sekundach. Jeśli jednak będziesz przestrzegał podanych tu wskazówek i nie będziesz wydawał poleceń, których skutków nie jesteś w stanie przewidzieć, żarty w stylu „ilu nowych użytkowników potrzeba, żeby wyłożyć system?” nie będą się odnosiły do Ciebie.



Inny termin, z którym nieraz jeszcze się zetkniesz, to administrator systemu. Jest to osoba, która instaluje i opiekuje się systemem linuxowym. Ilość pracy, jaką musi wykonać, jest bardzo różna, w zależności od systemu. Pełnoetatowy administrator może okazać się niezbędny np. w biurze posiadającym kilka większych komputerów, wielu użytkowników i takie urządzenia, jak drukarki czy napędy taśmowe, połączone w sieć. Twój system linuxowy najprawdopodobniej nie będzie wymagał zbyt intensywnej opieki.

Administracja systemem, włączając takie czynności, jak na przykład zakładanie i usuwanie kont, wymaga przywilejów oferowanych przez konto `superuser` czy `root`. Administrator jest więc jedną osobą, która ma uzasadnione powody do ich używania

Twoje pierwsze logowanie

Przymykając na razie oko na wszystkie powyższe ostrzeżenia, zaloguj się jako `root`. Ponieważ jest to jedyny użytkownik mający uprawnienia do tworzenia nowych kont, jest to nieuniknione. Poza tym trzeba będzie przeprowadzić kilka operacji wymagających specjalnych uprawnień. Zaraz po pierwszym zalogowaniu się do systemu utworzymy jakieś bezpieczne konto.

W odpowiedzi na monit

darkstar login:

wpisz

root

i wciśnij klawisz *Enter*. Jeśli podczas instalacji Linuxa zostałeś zapytany o hasło dla użytkownika *root*, musisz je teraz podać. W przeciwnym przypadku system nie zapyta o hasło. Jeśli zdarzy Ci się zapomnieć hasła administratora, będziesz niestety zmuszony do przeinstalowania całego systemu¹.



W systemie Linux małe i wielkie litery są rozróżniane (podobnie jak we wszystkich wersjach UNIX-a). Wielka litera „R” to zupełnie inny znak niż małe „r”. Podczas wydawania poleceń musisz przestrzegać właściwej wielkości liter, inaczej bowiem nie zostaną one rozpoznane. Większość poleceń powinna być wydawana małymi literami, choć istnieje kilka wyjątków. Dotyczy to również identyfikatorów: *Root* czy *rOot* nie zostanie rozpoznany jako prawidłowy identyfikator (chyba że utworzysz konto o takiej właśnie nazwie). Również hasła takie jak *pies*, *Pies* i *PIES* są w systemie Linux rozpoznawane jako zupełnie różne ciągi znaków. Jest jednak dość ciekawy wyjątek od tej reguły: jeśli identyfikator użytkownika zostanie wpisany tylko wielkimi literami, zostanie on zaakceptowany, ale od tej chwili wszystkie komunikaty będą wypisywane wielkimi literami. Jest to pozostałość z zamierzchłych czasów, gdy niektóre terminale obsługiwały tylko wielkie litery. Dziś już się ich nie spotyka, ale w programie *login* pozostawiono ten fragment kodu jako ciekawostkę historyczną. Nie radzimy jednak używać tego trybu, bo system wygląda wtedy dość dziwnie.

Jeśli pomyliś się, wpisując identyfikator lub hasło, Linux nie pozwoli na dostęp do systemu – musisz spróbować jeszcze raz. Ponieważ używasz konsoli głównej, możesz próbować do skutku. Niektóre systemy, ze względów bezpieczeństwa, blokują konsolę po zadanej liczbie nieudanych prób wejścia do systemu. Konsola główna jednak prawie zawsze pozwala na dowolną ilość prób.

Po tym, jak zalogowałeś się jako *root*, system uruchamia nową sesję. W tym momencie ekran zazwyczaj wygląda mniej więcej tak:

```
darkstar login: root
Last login: Sun Dec 11 17:26:18 on tty1
Linux 1.2.13 (POSIX).
You have mail.
```

```
If it's Tuesday, this must be someone else's fortune.
darkstar:~#
```

¹ Niestety (a może na szczęście?), użycie dyskietki startowej pozwala na zmianę hasła administratora bez jego znajomości (*przyp. tłum.*).

Niektóre systemy po zalogowaniu nie wypisują żadnych komunikatów, wyświetlając tylko znak zachęty (na przykład #) i kursor. W przypadku systemu z powyższego przykładu, Linux podaje informacje o tym, kiedy dane konto było ostatnio używane (ta informacja może się nie pokazać podczas pierwszego logowania) i dostarcza kilku informacji o wersji. Informuje też o tym, że dla danego użytkownika przyszła pocztą elektroniczną wiadomość i oczekuje ona na przeczytanie. Jeśli w systemie zostały zainstalowane gry, Linux przywita Cię powiedzeniem lub cytatem (generowanym przez program `fortune`). Dystrybutorzy Linuxa modyfikują dane wyświetlane podczas logowania, tak że naprawdę trudno przewidzieć, co pojawi się na ekranie akurat w Twoim systemie.

Jeśli system dostarcza takiej informacji, zawsze warto sprawdzić czas ostatniego logowania się. Ma to szczególne znaczenie, gdy z systemu korzystają inni użytkownicy. Nieprawidłowa wartość oznacza zazwyczaj, że ktoś włamał się do systemu i używa Twojego konta i uprawnień bez Twojej wiedzy i zgody. Aby zaradzić takiej sytuacji, powinieneś zmienić hasło (wkrótce pokażemy, jak to zrobić).

Czytaniem oczekującej poczty zajmiemy się nieco później, teraz przyjrzyjmy się ważniejszym tematom (jeśli Cię to nurtuje, większość systemów wysyła pocztę do administratora w czasie instalacji – dotyczy ona spraw związanych z rejestracją oprogramowania).

Powiedzonko pokazujące się po uruchomieniu systemu jest wybierane losowo z długiej listy, nie spodziewaj się więc, że będzie ono takie jak w podanym przykładzie. Nie zobaczysz go wcale, jeśli nie zainstalowałeś pakietu `games`. Jeżeli chcesz, możesz oczywiście zainstalować go w każdej chwili.

Ostatni wiersz wyświetlany na ekranie (zakończony znakiem #) jest tzw. „zachętą systemową”. Informuje ona, że Linux jest gotowy na przyjmowanie poleceń. Zawiera też kilka informacji:

darkstar to nazwa systemu,

znak ~ oznacza położenie w systemie plików (dokładniej zagadnienie to zostanie omówione w rozdziale 8. „System plików”),

znak # zazwyczaj oznacza, że jesteś zalogowany jako `root` (choć w niektórych systemach pojawia się znak \$, co uniemożliwia łatwe odróżnienie, czy jesteś zalogowany jako `root` czy zwykły użytkownik). Zgodnie z konwencją stosowaną w systemie UNIX, zwykły użytkownik zobaczy zamiast niego symbol % lub \$, zależnie od używanej powłoki, a symbol # jest zarezerwowany dla użytkownika `root`.

Hasła

W Linuxie (podobnie jak we wszystkich systemach UNIX-owych) identyfikatorem administratora jest `root`. Nieważne, czy system jest mały czy duży, jeśli możesz zalogować

się jako `root`, masz nad nim pełną kontrolę. Oczywiście zezwolenie komuś obecemu na zalogowanie się jako `root` niesie ze sobą mnóstwo potencjalnych niebezpieczeństw.

Aby uniknąć problemów, konto `root` należy zawsze zabezpieczyć hasłem, które bezwzględnie powinno być tajne. Zauważysz, że Linux mógł nie zapytać o hasło podczas pierwszego logowania. Powodem takiej sytuacji jest fakt, że podczas instalacji jest ono ustawiane jako puste (ang. *null string*). Linux w takim przypadku zazwyczaj nie pyta o hasło. Jeśli podczas instalacji hasło zostało podane, musisz je również podać podczas logowania.

Hasło puste jest najmniej bezpiecznym hasłem, ponieważ każdy, kto zna identyfikator (`root`) ma dostęp do systemu, należy więc je zmienić. Linux pozwoli Ci wybrać nowe hasło i zaakceptuje je bez problemu. Niestety, stwarza to złudne poczucie bezpieczeństwa.

Powszechnie wiadomo, że dawniej użytkownicy wybierali hasła, które łatwo im było zapamiętać: imię psa, datę urodzenia, rodzinne miasto itp. Niestety, hasła tego typu są łatwe do odgadnięcia. Niektórzy administratorzy systemów wymuszali więc użycie przypadkowo wybieranych, trudnych do złamania, ale i do zapamiętania, haseł typu `s8t6WLk`. Użytkownicy zazwyczaj nie potrafili zapamiętać takich haseł, więc je zapisywali, po czym notatki kładli na biurku. Powodowało to powstanie kolejnej dziury w systemie bezpieczeństwa.

Z tych powodów administratorzy systemów zalecają używanie haseł nie związańnych osobistości z użytkownikiem i nie będących pojedynczym słowem, które można znaleźć w słowniku. Ten ostatni wymóg spowodowany jest faktem, że niektóre procedury łamiące hasła opierają swe działanie na dużym słowniku, sprawdzając kolejno wszystkie wyrazy. Dobrze jest również używać w haśle zarówno liter jak i cyfr. Przykładowo, hasło `sie48kiera` jest o wiele trudniejsze do złamania niż `siekiera`. Program łamiący hasła nie jest w stanie w rozsądny czasie sprawdzić wszystkich kombinacji słów ze słownika i cyfr.

Najlepsze są hasła składające się z dużych i małych liter oraz cyfr, ale nadal dające się zapamiętać, np. `3majE2rda` czy `100Krotka`. Tego typu trudne do złamania hasła czasem nazywa się hasłami mocnymi – w odróżnieniu od haseł słabych, łatwych do odgadnięcia.

Z punktu widzenia bezpieczeństwa systemu, hasła powinny być zmieniane możliwie najczęściej. Większość administratorów uważa, że zmiana hasła raz na dwa lub trzy miesiące jest wystarczająca. Pozwala to również zmniejszyć ewentualne szkody, jeśli hasło zostało już złamane, ale nie spowodowano większych spustoszeń.



Nie pozostawiaj swojego terminala bez opieki, kiedy jesteś zalogowany. Ktoś może to wykorzystać i usunąć jakieś ważne dla Ciebie pliki albo wystąpić obraźliwą pocztą do kogoś, kogo nie chciałbyś obrazić. Zawsze, gdy wychodzisz, wyloguj się lub zablokuj terminal. Oczywiście, jeżeli terminal stoi w Twojej sypialni i nikogo nie ma w domu, nie jesteś połączony do sieci bezpośrednio ani przez modem, nie musisz się szczególnie martwić. Wylogowywanie się powinno jednak wejść Ci w krew jak najszybciej – przy pracy w sieci jest ono niezbęd-

ne.

Oczywiście ilość zabezpieczeń używanych w systemie powinna zależeć od jego dostępności i od tego, jak ważne dane są w nim zapisane. Hasło użytkownika `root` powinno być jednak zawsze bardzo dobre.

Jeśli praca z Linuxem jest dla Ciebie tylko eksperimentem, wiele z problemów związanych z bezpieczeństwem nie ma większego znaczenia. Mimo wszystko, utrzymywanie bezpieczeństwa na wysokim poziomie nie jest trudne i nie należy go zaniedbywać, a nabycie przyzwyczajenia można później przenieść do większych systemów UNIX-owych.

Do zmiany lub przypisania hasła służy polecenie `passwd` (skrócona wersja angielskiego słowa *password* – hasło). Zmienić hasło użytkownika `root` można tylko wtedy, gdy jesteś zalogowany jako `root`, ale użytkownik `root` może zmienić hasło każdego innego użytkownika. Każdy z użytkowników może prawie zawsze zmienić swoje hasło (chyba że zabroni tego administrator systemu).

Aby zmienić hasło użytkownika `root`, zaloguj się jako `root` i użyj polecenia `passwd`. Zostaną wyświetlane następujące komunikaty:

```
darkstar:~# passwd  
Changing password for root  
Enter new password:
```

Teraz należy wprowadzić nowe, bezpieczne hasło. To, co wpisujesz, nie jest wyświetlane na ekranie; rozwiążanie takie zabezpiecza przed podglądamieniem w czasie wpisywania hasła.



Upewnij się, że wpisałeś hasło powoli i ostrożnie. Kiedy któryś z użytkowników zapomni hasła, może ono być zmienione przez administratora. Jeśli jednak zapomnisz hasła administratora, pozostaje Ci tylko przeinstalować system.

Ponieważ bardzo ważne jest, by hasło zostało wpisane prawidłowo, system wymaga ponownego podania hasła:

```
Retype new password:
```

Również teraz hasło nie zostanie wyświetcone. Jeśli obie wprowadzone wersje hasła są identyczne, zostanie wyświetlony komunikat:

```
Password changed.  
darkstar:~#
```

Nowe, obowiązujące od tego momentu hasło zostało zapamiętane w odpowiednich plikach konfiguracyjnych, a stare zostało usunięte.

Jeśli dwie wprowadzone wersje hasła nie są identyczne, hasło nie zostanie zmienione i wypisany zostanie następujący komunikat:

You misspelled it. Password not changed.

W takim przypadku należy ponowić procedurę zmiany hasła.



Jeśli chcesz przerwać program i wrócić do interpretera poleceń, wciśnij **Control+C** (czyli wciśnij klawisz **C**, trzymając wciśnięty klawisz **Control** - taką kombinację klawiszy oznacza się też czasem jako **^C**). Przeważnie pozwala to zakończyć każdy uruchomiony program, nie powodując żadnych problemów, i wrócić do powłoki. Przykładowo, jeśli zacząłeś zmieniać hasło, ale żadne nie przychodzi Ci do głowy, wciśnięcie **Control+C** kończy program **passwd**, nie powodując zmiany hasła.

Zakładanie nowego konta

Po przypisaniu hasła dla konta **root** należy założyć nowe konto, za pomocą którego będzie można bezpiecznie eksperymentować i poznawać tajniki Linuxa. Niektóre systemy linuxowe wymuszają utworzenie takiego konta podczas instalacji. Nawet jeśli utworzyłeś takie konto podczas instalacji, prawdopodobnie i tak wkrótce będziesz dodawał nowe konta dla kolegów czy brata.

Do tego celu można wykorzystać program **adduser**, który automatyzuje i upraszcza proces zakładania nowego konta (kiedyś jedyną metodą była ręczna edycja plików konfiguracyjnych). Po uruchomieniu go, ekran powinien wyglądać mniej więcej tak:

```
darkstar:~# adduser
Adding a new user. The username should be not exceed 8 characters in
length, or you may run into problems later.
```

```
Enter login name for new account (^C to quit):
```

Wyświetlane komunikaty mogą być nieco inne, ale sam program działa zawsze tak samo. Niektóre wersje Linuxa zawierają analogiczne programy działające w trybie pełnoekranowym, a nawet graficznym, ale wszystkie one spełniają dokładnie te same funkcje i wymagają podania takich samych informacji.

Każdy użytkownik systemu musi posiadać własne konto (tożsame z identyfikatorem). Możesz utworzyć dla siebie konto, którego będziesz stale używał, możesz też założyć konto „eksperymentalne”, które usuniesz, gdy lepiej poznasz Linuxa. Inaczej niż w niektórych systemach pozwalających na pracę wielu użytkowników (np. Windows NT), możesz zakładać i usuwać konta jak często masz na to ochotę. Można też kilkakrotnie zakładać i usuwać konto o takim samym identyfikatorze.

Identyfikatory użytkowników mogą składać się z dowolnych liter i cyfr, warto jednak przestrzegać zasad, by ich długość nie przekraczała 8 znaków. Pozwoli to łatwo używać ich również przy połączeniach internetowych. Zazwyczaj identyfikator użytkownika pochodzi od prawdziwego imienia lub nazwiska osoby używającej danego konta, na przy-

kład michalk czy mkowalski. Jeśli z Twojego systemu będzie korzystać więcej użytkowników, warto przyjąć jakąś jednolitą konwencję i trzymać się jej. Pomoże to zachować porządek i orientować się, kto jest kim.

W niektórych wersjach Linuxa składnia polecenia `adduser` wymaga podania identyfikatora konta, np.:

```
adduser michalk
```

Jeśli tak jest w przypadku Twojego systemu, po wpisaniu samego polecenia `adduser` wyświetlony zostanie komunikat o błędzie.

W identyfikatorach użytkowników zaleca się używać tylko małych liter, a ich długość nie powinna przekraczać 8 znaków. Oczywiście dłuższy identyfikator zawierający małe i wielkie litery utrudni życie hakerom próbującym włamać się do systemu, ale możesz natknąć się na pewne trudności podczas używania sieci.

Na potrzeby tego rozdziału założymy konto o identyfikatorze `burek`. Można oczywiście wybrać inną nazwę, chyba że ta akurat Ci odpowiada.

```
Enter login name for new account (^C to quit): burek
Editing information for new user [burek]
Full Name:
```

Po wprowadzeniu identyfikatora program `adduser` zada serię pytań dotyczących nowego użytkownika oraz środowiska, w jakim będzie on pracował. Kolejność tych pytań może być nieco inna niż w przykładzie.

W przykładzie powyżej system oczekuje na podanie prawdziwego imienia i nazwiska użytkownika (można tu używać małych i wielkich liter oraz spacji). Choć informacje te nie są wymagane, system wykorzystuje je w niektórych sytuacjach, gdzie, mogą one okazać się pomocne. Również inni użytkownicy sieci mogą zidentyfikować użytkownika dzięki tym informacjom. Idźmy więc dalej:

```
Full Name: Pies Burek
GID [100]:
```

Teraz system czeka na wprowadzenie numeru grupy (ang. *Group ID*; temat ten zostanie omówiony szczegółowo w rozdziale 35. „Użytkownicy i konta”). W podanym przykładzie podpowiadana jest wartość 100 – jest to pierwsza dostępna wartość. Można jej użyć, chyba że komputer jest częścią większego systemu posiadającego własne konwencje dotyczące tworzenia nowych kont. Można również później zmienić odpowiednie przyporządkowania.



W programie `adduser`, podobnie jak w wielu innych programach linuxowych, domyślne wartości podawane są w nawiasach kwadratowych. Można oczywiście wpisać inną wartość, ale jeśli chcesz zaakceptować wartość domyślną, wystarczy wcisnąć klawisz `Enter`.

Czasem również w nawiasie kwadratowym podawane są dwie wartości, rozdzielone znakiem `/`, `|` lub nawet nie rozdzielone, na przy-

kład [Y/n], [Y|n] czy [Yn]. Oznacza to, że dostępne są tylko dwie wartości, a domyślnie wybrana zostanie ta pisana wielkimi literami - jeśli chcesz ją zaakceptować, wystarczy wcisnąć *Enter*, ale jeśli chcesz wybrać drugą z możliwości - musisz ją wpisać.

Po wprowadzeniu informacji o numerze grupy (lub zaakceptowaniu sugerowanej wartości) program `adduser` oczekuje na podanie dalszych informacji:

```
GID [100]:  
Checking for an available UID after 100  
First unused uid is 101  
UID [101]:
```

Jeśli akceptujesz wartość domyślną identyfikatora grupy, zazwyczaj nie jest ona wyświetlana na ekranie. Może to być trochę mylące, szczególnie gdy przeglądasz wydane przed chwilą polecenia i zastanawiasz się, co właśnie zrobiłeś. Jeżeli nie jest wypisana żadna wartość, oznacza to, że użyto wartości domyślnej. Trzeba się do tego przyzwyczaić, ponieważ tak zachowuje się większość programów linuxowych.

Następnie należy podać numer identyfikacyjny użytkownika (ang. User ID, UID). W tym przypadku program `adduser` podpowiada wartość 101. By ją zaakceptować, wciśnij *Enter*. Podobnie jak w przypadku identyfikatora grupy, nie ma jakichś szczególnych powodów, byś miał go zmieniać – chyba że system jest częścią większej organizacji.



Numer identyfikacyjny użytkownika jest używany przez system Linux za każdym razem, gdy zachodzi potrzeba zidentyfikowania użytkownika. System jest zaprojektowany w taki sposób, że używa numeru zamiast identyfikatora tekstowego, ponieważ liczba zajmuje mniej miejsca i łatwiej nią manipulować. Każdy użytkownik musi mieć przypisany inny numer. Zgodnie z przyjętą konwencją, numery poniżej 100 (w niektórych systemach poniżej 500) są zarezerwowane dla użytkowników specjalnych (np. użytkownik `root` ma numer 0). Numery przydzielane zwykłym użytkownikom zaczynają się zwykle od 100 (lub 500).



Niektóre wersje Linuxa, np. Caldera OpenLinux, po wydaniu polecenia `adduser` nie wymagają podania żadnych informacji. Po prostu same wyszukują dostępne numery i przypisują je do nowego konta, w pozostałe pola wpisując wartości domyślne. Jest to rozwiązanie łatwe i przyjemne, dopóki nie trzeba używać jakichś niestandardowych ustawień (np. ze względów organizacyjnych).

Należy podać również lokalizację katalogu domowego (macierzystego) i nazwę interpreta poleceń, który będzie używany przez nowego użytkownika:

```
Home Directory [/home/burek]:  
Shell [/bin/bash]:
```

Dobrze pozostawić tu wartości domyślne. Więcej o katalogach możesz przeczytać w rozdziale 9. „Prawa dostępu do plików i katalogów”, a o różnych rodzajach powłok systemu operacyjnego w rozdziale 11. „bash”, rozdziale 12. „pdksh” i rozdziale 13. „tcsh”.

Ostatnią informacją, jaką należy podać, jest hasło dla nowego użytkownika. Jeśli wciśniesz *Enter*, hasło będzie identyczne jak nazwa konta, albo zostanie ustawione hasło puste – zależnie od wersji Linuxa. Pamiętaj o tym, że nawet proste hasło jest lepsze niż żadne.

```
Password [burek]:
```

```
Information for new user [burek]:  
Home directory: [/home/burek] Shell: [/bin/bash]  
uid: [101] gid: [100]
```

```
Is this correct? [y/N]:
```

Jeśli po weryfikacji podanych informacji stwierdzisz, że wszystko jest w porządku, wpisz *y* i wciśnij *Enter*. Jeśli wciśniesz tylko klawisz *Enter*, system potraktuje to jako odpowiedź *n* (domyślna) i będziesz musiał wprowadzić wszystkie informacje jeszcze raz.

Następnie utworzony zostanie katalog domowy użytkownika, a w plikach konfiguracyjnych zostaną zapisane odpowiednie informacje. Na ekranie mogą pojawiać się odpowiednie komunikaty:

```
Adding login [burek] and making directory [/home/burek]  
  
Adding the files from the /etc/skel directory:  
./.kermrc -> /home/burek/.kermrc  
./.less -> /home/burek/.less  
./.lessrc -> /home/burek/.lessrc  
./.term -> /home/burek/.term  
./.term/.termrc -> /home/burek/.term/.termrc  
darkstar:~#
```

Jak usunąć niepotrzebne już konta z pliku */etc/passwd*, dowiesz się w rozdziale 35. „Użytkownicy i konta”.

Wylogowywanie się

Stworzyłeś już nowe konto, możesz więc go teraz używać czytając kilka kolejnych rozdziałów i poznając Linuxa. By zakończyć sesję jako *root*, wyloguj się z systemu za pomocą polecenia *logout*:

```
darkstar:~#logout  
  
Welcome to Linux 1.2.13  
darkstar login:
```

Teraz można ponownie zalogować się jako *root*, albo jako użytkownik, którego właśnie dodaleś do systemu.

Niektóre systemy pozwalają na wylogowanie się po wciśnięciu kombinacji klawiszy *Control+D*. Jeśli taka kombinacja nie jest obsługiwana, może zostać wyświetlony następujący komunikat:

Use "logout" to leave the shell.

Jeśli znasz systemy UNIX-owe, możesz być przyzwyczajony do używania tej kombinacji klawiszy. Jednak domyślna powłoka (`bash`) systemu Linux nie obsługuje jej (chyba że ustawisz odpowiednie mapowanie klawiatury). Czasem dostępne jest również polecenie `logoff`, spełniające tę samą funkcję co `logout`.

Wypróbowywania nowego konta

Pora teraz wypróbować nowe konto. Przyjrzymy się też kilku interesującym cechom Linuxa.

Zaloguj się, podając identyfikator konta, które założyłeś. Jeśli byłeś zapobiegliwy i przypisałeś mu jakieś niepuste hasło, musisz je wprowadzić.

```
darkstar login: burek
Password:
Last login: Sun Dec 11 19:14:22 on tty1
Linux 1.2.13 (POSIX).

Quiet! I hear a hacker....
darkstar:~$
```

Większość komunikatów jest podobna do tych, które wyświetlane są przy logowaniu się użytkownika `root`. Niektóre systemy nie wyświetlają żadnych komunikatów, poza znakiem \$, ale można to zmienić. Znak dolara informuje, że jesteś zalogowany jako zwyczajny użytkownik i pracujesz pod kontrolą powłoki `bash` (która jest proponowana domyślnie przez program `adduser`). Nie jest również wyświetlana informacja „You have mail” (bo nie otrzymałeś jeszcze żadnej poczty).



Zdarza się, że Linux jest skonfigurowany tak, że wysyła pocztę do wszystkich nowych użytkowników. Może to być jakiś rodzaj powitania, informacje o systemie itp.

Aby przekonać się o różnicach pomiędzy użytkownikiem `root` a zwykłym użytkownikiem, spróbuj użyć polecenia `adduser`:

```
darkstar:~$ adduser  
bash: adduser: command not found
```

Komunikat, który został wyświetlony, wygląda nieco tajemniczo. Jest to jednak typowy komunikat o błędzie, więc przyjrzyjmy się mu bliżej.

Komunikaty o błędach

W komunikacie o błędzie uzyskanym w poprzednim podrozdziale zawartych jest kilka informacji. Program, który podał ten komunikat, to `bash`, i jego nazwa pojawia się na początku komunikatu. Następnie wyświetlana jest nazwa programu, który sprawia kłopoty (`adduser`) i właściwy komunikat: polecenie nie zostało odnalezione (ang. *command not found*).

Gdyby przetłumaczyć ten komunikat na „normalny” język, mógłby on brzmieć następująco: „To ja, `bash`. Wydałeś polecenie `adduser` – szukałem programu o takiej nazwie, ale bezskutecznie. Nie wiem, o co Ci chodzi?”. Z czasem przyzwyczaisz się do takich komunikatów – ich zrozumienie przestanie być dla Ciebie problemem.

Ścieżki przeszukiwania

Dlaczego użytkownik `root` ma dostęp do polecenia `adduser`, a zwykły użytkownik nie? W systemie jest wiele katalogów, zawierających jeszcze więcej plików (jednym z nich jest program `adduser`). Teoretycznie Linux mógłby przeszukać wszystkie katalogi i znaleźć zadany program. Jednak w przypadku, gdyby użytkownik `root` przez pomyłkę zamiast `adduser` wpisał `aduser`, Linux musiałby sprawdzić nazwy wszystkich plików w systemie, zanim stwierdziłby, że jest to pomyłka. W najlepszym przypadku trwałoby to kilka sekund i na pewno nie byłoby wygodne.

Z tego powodu w systemie zdefiniowane są ścieżki przeszukiwania (ang. *search paths*). Zostaną one omówione dokładniej w rozdziale 8. Jeśli w katalogach określonych tymi ścieżkami dany program nie zostanie odnaleziony, wygenerowany zostanie komunikat

o błędzie. Ponieważ `root` używa wielu programów o charakterze administracyjnym, jego ścieżki przeszukiwania zawierają katalog z programami tego typu (`/sbin` czy `/usr/sbin`). Jak łatwo się domyślić, ścieżki przeszukiwania zdefiniowane dla zwykłego użytkownika nie zawierają takiego wpisu.



Ścieżki dostępu w systemie Linux spełniają w zasadzie tę samą funkcję, co w systemach DOS i Windows, choć inną jest składnia polecenia definiującego je i nazwy katalogów wyglądają nieco inaczej.

Można oczywiście uruchomić program nie znajdujący się w żadnym z katalogów uwzględnionych w ścieżce przeszukiwania – należy po prostu dokładnie podać, w którym miejscu program ten jest zapisany, wtedy Linux nie będzie musiał go szukać. Spróbuj uruchomić program `adduser` bezpośrednio z katalogu `/sbin`:

```
darkstar:~$ /sbin/adduser  
bash: /sbin/adduser: Permission denied
```

Tym razem `bash` odnalazł odpowiedni program (bo powiedziałeś mu dokładnie, gdzie powinien go znaleźć), ale stwierdził, że zwykły użytkownik nie posiada uprawnień do jego uruchomienia. Jak widać, Linux ogranicza poczynania użytkowników w zależności od ich uprawnień i tylko użytkownik `root` (przynajmniej na razie) może uruchomić program `adduser` (o prawach dostępu dowiesz się więcej z rozdziału 9.).

Polecenie who

Jednym z prostszych poleceń Linuxa, którego będziesz używał jeszcze wielokrotnie, poznając tajniki programowania w języku powłoki, jest polecenie `who`. Wyświetla ono informację o tym, kto jest aktualnie zalogowany do systemu. Jest ono niezwykłe pod tym względem, że nie pobiera żadnych argumentów. Po jego wykonaniu mogą na przykład zostać wyświetlane następujące informacje:

```
$ who  
michals tty02 May 18 18:29  
root      tty01 May 18 15:18  
root      tty03 May 18 18:29  
burek    ttyp1 May 18 17:31
```

Polecenie `who` wypisuje dane o każdym zalogowanym użytkowniku w trzech kolumnach. Pierwsza z nich zawiera identyfikator zalogowanego użytkownika. Druga kolumna zawiera nazwę terminalu, którego używa (jest to nazwa urządzenia – temat ten jest omówiony szerzej w rozdziale 33. „Urządzenia”). Ostatnia kolumna zawiera dane o tym, kiedy użytkownik się zalogował.

Jeśli użytkownik jest zalogowany na więcej niż jednej konsoli wirtualnej (konsole wirtualne zostaną omówione w następnym podrozdziale) lub terminalu, wyświetlane zostaną dane o wszystkich sesjach. W poprzednim przykładzie użytkownik `root` jest zalogowany na dwóch różnych terminalach.

Informacje podawane przez polecenie `who` są odczytywane za każdym razem, gdy zostanie ono wydane, więc użytkownik zawsze otrzymuje najświeższe dane.

Terminale wirtualne

W systemie Linux może być w danej chwili zalogowany więcej niż jeden użytkownik, a każdy z nich może wykonywać więcej niż jedno zadanie. Poważny system wielozadaniowy posiada zwykle kilka terminali (składających się z klawiatury i monitora), podłączonych poprzez sieć do komputera głównego.

Ponieważ system Linux przeznaczony jest głównie dla komputerów domowych, a mało kto ma w domu kilka terminali, przewidziano możliwość kilkakrotnego zalogowania się za pomocą jednej klawiatury i jednego monitora. Jest to możliwe właśnie dzięki terminalom wirtualnym, nazywanym również konsolami wirtualnymi. Pozwalają one jednej klawiaturze i monitorowi zachowywać się jak kilka (do 12) terminali, identyfikowanych przez klawisze funkcyjne.

Przykładowo, wcisnij kombinację klawiszy `Alt+F2`. Zawartość ekranu powinna zniknąć, a zamiast niej powinien pojawić się znajomy komunikat:

```
Welcome to Linux 1.2.13  
darkstar login:
```

Zaloguj się jako zwykły użytkownik (nie jako `root`), a następnie wydaj polecenie `who`. Powinno ono podać następujące informacje:

```
darkstar:~$ who  
burek  tty2    Dec 14 01:42  
burek  tty1    Dec 14 01:40
```

Polecenie `who` wyświetli dane o wszystkich zalogowanych użytkownikach. Oczywiście zamiast tekstu `burek` pojawi się nazwa Twojego konta.

Zgodnie z konwencją, `tty1` jest główną konsolą systemu. Jest to „normalny” ekran, który jest wyświetlany po uruchomieniu systemu (chyba że od razu uruchamiany jest interfejs graficzny). Jeśli przełączysz się na inną konsolę wirtualną, możesz wrócić do konsoli głównej, wciskając `Alt+F1`.

Jeżeli chcesz sprawdzić, ile konsoli wirtualnych może działać w Twoim systemie, spróbuj kolejno nacisnąć klawisze `Alt+Fx`. W niektórych systemach można również użyć kombinacji `Alt+Strzałka_w_prawo` i `Alt+Strzałka_w_lewo`. Większość systemów linuxowych uruchamia się, oferując sześć, dziesięć lub dwanaście konsoli wirtualnych. Możesz zmienić tę liczbę, ale w praktyce raczej nie korzysta się z więcej niż trzech lub czterech konsoli, więc ustawienia domyślne są wystarczające.

Po co w ogóle używać konsoli wirtualnych? Często zdarza się, że właśnie jesteś w trakcie robienia czegoś bardzo skomplikowanego, gdy okazuje się, że najpierw musisz zrobić coś innego. Wtedy najłatwiej jest przełączyć się na inną konsolę i tam wykonać niezbędne polecenia.

Inne zastosowanie to odblokowanie terminalu w przypadku, gdy jakiś program odmówi posłuszeństwa lub terminal zamiast liter zacznie wyświetlać dziwne znaczki. Można wówczas spróbować rozwiązać problem z innego terminalu lub zamknąć system, jeśli jest to konieczne.

Linux udostępnia również bardzo potężne środowisko wielozadaniowe, jakim jest system X. Instalacja i konfiguracja tego systemu opisana jest w rozdziale 22. „Instalowanie i konfigurowanie XFree86”.

Polecenia i programy

Często zamiast mówić „w wierszu poleceń wpisz `who` i wcisnij *Enter*” używa się krótkich wyrażeń, takich jak „uruchom program `who`”, „wydaj polecenie `who`” czy nawet „uruchom `who`”. Oczywiście z kontekstu wynika, że ktoś mówiący, że uruchomił `who`, ma na myśli, że wpisał w wierszu poleceń `who` i wcisnął *Enter*. Istnieje jednak pewna różnica pomiędzy poleceniem a programem. Polecenie to coś, co wpisujesz w odpowiedzi na zachętę generowaną przez interpreter polecień. Po naciśnięciu klawisza *Enter* Linux próbuje zanalizować wprowadzony tekst. Niektóre polecenia obsługiwane są przez sam interpreter, inne to po prostu zewnętrzne programy przez niego uruchamiane.

Ujmując rzecz zupełnie scieśle, polecenie jest tym, co wpisujesz, a program to coś, co przetwarza to polecenie. Mimo to proste programy dające od razu wyniki swojego działania, takie jak np. `who`, często również są nazywane poleceniami, choć tak naprawdę na dysku twardym zapisany jest również program `who`. Nazwa „program” używana jest raczej w stosunku do aplikacji nieco bardziej rozbudowanych, wymagających współpracy z użytkownikiem, jak choćby program `adduser` czy edytory tekstów.

Podsumowanie

W tym rozdziale pokazaliśmy, w jaki sposób przypisać hasło użytkownikowi `root` i założyć nowe konto, którego będziemy używać w kilku następnych rozdziałach. Omówiliśmy również kilka kwestii związanych z terminologią.

Aby dowiedzieć się czegoś więcej o podstawowych poleceniach dostępnych w systemie Linux, przejdź do rozdziału 7. „Podstawowe polecenia Linuxa”.

Interpreter poleceń Bourne Again Shell (`bash`) opisany jest dokładniej w rozdziale 11.

Rozdział 9. omawia problemy związane z prawami dostępu do plików i sposobami ograniczania dostępu do danych.

Rozdział 7.

Podstawowe polecenia

Linuxa

Ed Trejis i Tim Parker

W tym rozdziale:

- υ Jak działają polecenia Linuxa
- υ Konwencje używane do opisu polecień
- υ Pomoc dostępna podczas pracy
- υ Strony `man`
- υ Symbole wieloznaczne: * oraz ?
- υ Zmienne środowiskowe
- υ Procesy i ich wyłączanie
- υ Polecenie `su`, czyli jak stać się kimś innym
- υ Program `grep`

Jak działają polecenia Linuxa

Polecenia dostępne w systemie Linux są zwykle bardzo elastyczne. Przeważnie istnieje kilka metod, dzięki którym możesz dopasować działanie polecenia do swoich potrzeb. Przyjrzymy się dwóm głównym sposobom modyfikacji zachowania się programów: podawaniu lub przekierowaniu (czasem nazywanym również przeadresowaniem; ang. *redirecting*) wejścia i wyjścia oraz używaniu opcji.

Można wyobrazić sobie program jako czarną skrzynkę – część linii produkcyjnej – do której z jednej strony wędrują dane, z drugiej zaś odbierane są wyniki ich przetwarzania. Opcje pozwalają na dopasowanie do naszych potrzeb procesów odbywających się w takiej skrzynce. Przekierowanie pozwala podać efekty działania jednej skrzynki bezpośrednio na wejście drugiej. Analogia jest dość prosta, ale dobrze pomaga zrozumieć niektóre aspekty pracy z Linuxem i UNIX-em.

Gdy już zrozumiesz, jak działają opcje i przekierowanie, będziesz potrafił (przynajmniej z grubsza) użyć każdego polecenia linuxowego i UNIX-owego, ponieważ UNIX został zaprojektowany przy przestrzeganiu kilku głównych zasad. Podstawową z nich jest to, że polecenia są proste i spełniają jedno zadanie, a jednocześnie są elastyczne jeśli chodzi o miejsce, z którego pobierają i w którym zapisują dane. Oczywiście przez 35 lat istnienia UNIX-a powstały programy nie spełniające tych warunków, ale mimo wszystko systemy UNIX-owe, w tym Linux, są pod tym względem zadziwiająco spójne.



Wciśnięcie klawiszy *Control+U* w wierszu poleceń powoduje usunięcie wszystkiego, co się w nim znajduje. Jest to przydatne w momencie, gdy zdarzy Ci się pomyłka na początku długiego polecenia, albo rozmyślisz się w ostatnim momencie przed jego zatwierdzeniem. Do usunięcia wpisanego polecenia można również użyć klawisza Backspace, ale pierwszy sposób zazwyczaj jest szybszy. Innym rozwiązaniem jest wciśnięcie kombinacji klawiszy *Control+C*, która anuluje wszystko, co napisałś i ustawi kursor w nowym wierszu.

Klawisze kurSORA są bardzo użyteczne podczas wpisywania poleceń; niestety, nie wszystkie wersje powłoki je obsługują. Działają one podobnie jak w programie DOSKEY (lewo i prawo - poruszają kursorem, góra i dół - pozwalają wybrać polecenie z listy ostatnio wydanych poleceń).

Opcje poleceń

Często zdarza się, że jakieś polecenie Linuxa pozwala zrobić prawie dokładnie (ale nie dokładnie) to, czego potrzebujesz. Zamiast zmuszać Cię do użycia innego, działającego dokładnie zgodnie z Twoimi intencjami polecenia, Linux pozwala zmodyfikować działanie istniejącego. Prosty przykład: chcesz posortować listę nazwisk. Możesz w tym celu użyć polecenia *sort*, które pozwala wykonać taką operację, ale Tobie potrzebna jest lista uporządkowana „od końca”. Zamiast szukać innego polecenia, wystarczy, że podasz jednoliterową opcję, która odwróci kolejność sortowania. Niektóre polecenia udostępniają bardzo dużo opcji, inne nieco mniej.

Świetnym przykładem prostego, użytecznego a jednocześnie mającego mnóstwo opcji polecenia jest *ls*. Wypisuje ono nazwy wszystkich plików znajdujących się w katalogu. Proste, nieprawda? Spróbuj je wykonać:

```
darkstar:~$ ls  
darkstar:~$
```

Nic nie zostało wyświetlone – jest to normalne, jeśli używasz nowo założonego konta.

Spróbuj teraz wydać polecenie `ls -a` (ważne jest, by po `ls` była spacja, natomiast nie może jej być między znakiem `-` oraz literą `a`).

```
darkstar:~$ ls -a  
./                      .bash_history          .kermrc           .lessrc
```

Działanie polecenia zostało zmodyfikowane przez dodanie do niego opcji `-a`. Domyślnie `ls` wyświetla tylko te pliki, których nazwa nie zaczyna się od kropki (tak oznaczane są w Linuxie pliki systemowe). Nazwy wszystkich plików w bieżącym katalogu zaczynają się od kropki, więc nie zostały one wypisane. Opcja `-a` powoduje wypisanie wszystkich nazw plików, również tych zaczynających się od kropki.

Polecenie `ls` pozwala również na użycie innych opcji. Można nawet używać kilku opcji równocześnie, na przykład tak:

```
darkstar:~$ ls -al  
total 10  
drwxr-xr-x    3      burek   users    1024   Dec 21 22:11   ./  
drwxr-xr-x    6      root     root    1024   Dec 14 01:39   ../  
-rw-r--r--    1      burek   users    333    Dec 21 22:11   .bash_history  
-rw-r--r--    1      burek   users    163    Dec  7 14:25   .kermrc  
-rw-r--r--    1      burek   users     34    Jun  6 1995   .less  
-rw-r--r--    1      burek   users    116    Nov 23 1995   .lessrc  
drwxr-xr-x    1      burek   users    1024   Dec  7 13:36   .term/
```

Wyświetlony został spis plików w katalogu wraz z dotyczącymi ich szczegółami (które zostaną wyjaśnione w rozdziale 8. „System plików”). Opcja `-l` może być użyta samodzielnie; powoduje ona właśnie wyświetlenie szczegółowych informacji o plikach. Czasem nazwy plików są tak długie, że nie mieścią się w jednym wierszu – w takiej sytuacji Linux po prostu przeniesie dalszą część do wiersza następnego.



Ujmując rzecz ściśle, myślnik (`-`) nie jest częścią opcji. Jest to po prostu znak, który powiadamia Linuxa, że następny należy traktować jako właściwe oznaczenie opcji. Przed myślnikiem musi wystąpić spacja, nie powinno jej być natomiast między myślnikiem a nazwą opcji. Jeśli jakiś tekst ma znaleźć się w wierszu polecień po nazwie opcji, również powinien być oddzielony spacją¹.

Po myślniku można podać więcej niż jedną nazwę opcji, jak w przykładzie `ls -al`. Ich porządek zazwyczaj nie gra roli, czyli polecenie `ls -la` działa dokładnie tak samo jak `ls -al`. Jednoczesne podawanie kilku opcji ma zastosowanie tylko w programach, w których ich nazwy są jednoliterowe.

¹ Informacja ta jest nieścisła: spacja nie jest wymagana przed symbolami `>` czy `|`, również niektóre programy (na przykład `tar`) nie wymagają, a wręcz zabraniają oddzielania spacją oznaczenia opcji i dalszej części polecenia (na przykład nazwy pliku; przyp. tłum.).

Można też podać opcje oddzielnie, np. `ls -a -l`. Takie rozwiązanie jest przeważnie stosowane, gdy po jakiejś opcji wymagane są dodatkowe dane.

Domyślnie polecenie `ls` porządkuje wyświetlane nazwy plików w kolejności alfabetycznej. Czasem bardziej interesująca jest informacja o tym, kiedy dany plik został utworzony czy zmodyfikowany. W takim przypadku pomocna może być opcja `t`, która powoduje, że nazwy plików sortowane są według daty – najpierw wyświetlane są pliki najnowsze. Po wydaniu polecenia `ls -alt` możesz więctrzymać następujący wynik:

```
darkstar:~$ ls -alt
total 10
drwxr-xr-x    3      burek   users   1024   Dec 21 22:11   .
-rw-r--r--    1      burek   users    333   Dec 21 22:11   .bash_history
drwxr-xr-x    6      root     root    1024   Dec 14 01:39   ../
-rw-r--r--    1      burek   users    163   Dec  7 14:25   .kermrc
drwxr-xr-x    1      burek   users   1024   Dec  7 13:36   .term/
-rw-r--r--    1      burek   users    116   Nov 23 1995   .lessrc
-rw-r--r--    1      burek   users     34   Jun  6 1995   .less
```

Opcja `-r` powoduje, że polecenie `ls` wyświetla nazwy plików posortowane w odwrotnej kolejności. Dość często używa się jej w połączeniu z opcją `t`. Wydając polecenie `ls -altr` możesz otrzymać następujące informacje:

```
darkstar:~$ ls -altr
total 10
-rw-r--r--    1      burek   users     34   Jun  6 1995   .less
-rw-r--r--    1      burek   users    116   Nov 23 1995   .lessrc
drwxr-xr-x    1      burek   users   1024   Dec  7 13:36   .term/
-rw-r--r--    1      burek   users    163   Dec  7 14:25   .kermrc
drwxr-xr-x    6      root     root    1024   Dec 14 01:39   ../
-rw-r--r--    1      burek   users    333   Dec 21 22:11   .bash_history
drwxr-xr-x    3      burek   users   1024   Dec 21 22:11   .
```

Istnieje jeszcze wiele innych opcji modyfikujących działanie polecenia `ls` – pokazaliśmy tylko te najczęściej używane. Ważne jest, byś zdawał sobie sprawę z możliwości dostosowywania poleceń do swoich potrzeb za pomocą opcji.

Inne parametry

Polecenia często wymagają użycia parametrów nie będących nazwami opcji. Takie parametry (na przykład nazwy plików czy katalogów) nie są poprzedzane myślnikiem.

Polecenie `ls` wyświetla zawartość bieżącego katalogu. Można jednak łatwo uzyskać informacje o dowolnym innym katalogu, podając jego nazwę jako parametr, na przykład polecenie `ls /bin` powoduje wyświetlenie nazw plików zapisanych w katalogu `/bin`. I w tym przypadku można używać opcji: polecenie `ls -l /bin` spowoduje wyświetlenie szczegółowych informacji o tych plikach (a jest ich naprawdę dużo!).

Można również zażądać informacji o jednym konkretnym pliku, podając jako parametr polecenia `ls` jego nazwę: polecenie `ls -la .lessrc` wyświetli dane o pliku `.lessrc`. Jeśli nie zostanie on znaleziony, nie zostaną wyświetcone żadne informacje.



W przypadku podstawowych poleceń Linuxa istotne jest, czy nazwa opcji pisana jest małą, czy wielką literą. Polecenie `ls` pozwala na przykład na użycie opcji `R`, która powoduje wyświetlenie nazw plików znajdujących się w podkatalogach danego katalogu, czyli oznacza zupełnie co innego niż `r`.



Nazwy opcji pochodzą zazwyczaj od słów angielskich, np. `-a` od *all files*, `-l` od *long list* itp. Łatwiej je dzięki temu zapamiętać. Niektóre oznaczenia opcji są wspólne dla większości programów, na przykład opcja `v` (ang. *verbose*) oznacza, że mają być wyświetlane komunikaty o podejmowanych działaniach.

Nie można jednak oczekwać, że w innym poleceniu dana opcja będzie znaczyła to samo. Przykładowo, w większości poleceń opcja `-r` oznacza *recursive*, czyli z uwzględnieniem podkatalogów, ale akurat w przypadku polecenia `ls` oznacza ona *reverse*, czyli kolejność odwrotną, a podkatalogi uwzględnia się opcją `-R`. Akurat w przypadku polecenia `ls` opcja powodująca wyświetlanie nazw plików w odwrotnej kolejności jest używana częściej, niż wyświetlanie nazw plików znajdujących się w podkatalogach, dlatego też została ona oznaczona łatwiejszą do wpisania małą literą `r`. Może się wydawać, że wcisnięcie klawisza *Shift* nie wymaga szczególnego wysiłku, ma to jednak dość duże znaczenie, gdy trzeba wpisać kilka liter, z których tylko jedna jest wielką literą.

Przekierowanie wejścia i wyjścia

Wiele poleceń Linuxa pozwala podać nazwę pliku lub katalogu, z którego należy pobrać dane wejściowe (np. `ls -s /bin`). Można również podłączyć strumień danych z wyjścia innego programu (mechanizm obsługujący takie połączenie nazywa się po angielsku *pipe* – rurka, a po polsku – potok) przez wpisanie dwóch polecień oddzielonych znakiem `|` (znajdującym się na klawiaturze razem ze znakiem `\`). Oznacza on: „użyj wyjścia pierwszego polecenia jako wejścia drugiego”. Wydanie polecenia `prog_1 | prog_2` spowoduje wykonanie obu programów, jednego po drugim, a następnie wyprowadzenie wyników działania ostatniego z nich.

Używając przytoczonego wcześniej porównania, zamiast jednej uruchamiane są dwie czarne skrzynki. Wyjście pierwszej zostaje podłączone do wejścia drugiej. Dopiero wyjście drugiej jest wyprowadzane na ekran.



Choć Linux nie dba o to, czy znak | jest otoczony spacjami, czy też nie, warto postawić spacje z obu jego stron, ponieważ znacznie poprawia to czytelność polecenia.

Zauważysz pewnie, że wynik wykonania polecenia `ls -l /bin` jest dość długi i większość informacji tylko mignie na ekranie. Ty zaś nie zdążysz ich przeczytać. Możesz skierować wyjście tego polecenia na wejście programu formatującego tekst o nazwie `more`, który posiada zdolność wyświetlania informacji porcjami mieszczącymi się jednorazowo na ekranie. Kiedy wpiszesz `ls -l /bin | more`, zobaczysz mniej więcej coś takiego:

```
darkstar:~$ ls -l /bin | more
total 1611
-rwxr-xr-x  1      root    bin     1248   Sep 17  04:25  arch
-rwxr-xr-x  1      root    bin   295940  Sep  5  01:45  bash
-rwxr-xr-x  1      root    bin     4840   Nov 24  1993  cat
-rwxr-xr-x  1      root    bin    9220   Jul 20  12:06  chgrp
-rwxr-xr-x  1      root    bin   13316   Jul 20  12:06  chmod
-rwxr-xr-x  1      root    bin   13316   Jul 20  12:06  chown
lrwxrwxrwx  1      root    root     18    Dec  7  13:37  compress
Σ->/usr/bin/compress
-rwxr-xr-x  1      root    bin   21508   Jul 20  12:06  cp
-rwxr-xr-x  1      root    bin   41988   May  1  1994  cpio
lrwxrwxrwx  1      root    root     4    Dec  7  13:40  csh ->tcsh
-rwxr-xr-x  1      root    bin    5192   Nov 24  1993  cut
-rwxr-xr-x  1      root    bin   19872   Mar 23  1994  date
-rwxr-xr-x  1      root    bin   17412   Jul 20  12:06  dd
-rwxr-xr-x  1      root    bin   13316   Jul 20  12:06  df
-rwxr-xr-x  1      root    bin   66564   Jun  9  1994  dialog
-rwxr-xr-x  1      root    bin    1752   Sep 17  04:25  dmesg
lrwxrwxrwx  1      root    root     8    Dec  7  13:37  dnsdomainname
Σ->hostname
-rwxr-xr-x  1      root    bin   13316   Jul 20  12:06  du
-rwxr-xr-x  1      root    bin   3312    Mar 23  1994  echo
-rwxr-xr-x  1      root    bin   36684   May  4  1994  ed
-rwxr-xr-x  1      root    bin    326    Mar 23  1994  false
-- More --
```

Komunikat `-- More --` na dole ekranu wskazuje, że pozostał jeszcze jakiś tekst do wyświetlenia. Aby zobaczyć następną porcję, wciśnij spację. Po wypisaniu całego tekstu program `more` kończy działanie.



Polecenie `more` potrafi jeszcze wiele innych rzeczy. Można na przykład (choć nie we wszystkich wersjach) cofnąć się do poprzedniego ekranu wciskając klawisz `b` (ang. *back*). Można też wyjść z programu naciskając klawisz `q` (ang. *quit*), dzięki czemu nie trzeba przechodzić przez wszystkie strony do końca tekstu, jeśli odpowiednie informacje zostały już znalezione. Kombinacja `Control+C` spełnia tę samą funkcję.

Gdy uruchomiony jest program `more`, można wcisnąć klawisz `h` (ang. *help*), co spowoduje wyświetlenie listy wszystkich dostępnych poleceń.



Czasem w Linuxie zamiast programu `more` dostępny jest `less`. Jedną z różnic pomiędzy tymi programami jest to, że `less` wymaga wcisnięcia klawisza `q` dla opuszczenia programu, nawet jeśli wyświetlony został cały tekst. Może to wydawać się nieporęczne, ale zapewnia przed przypadkowym opuszczeniem programu, gdy nacisniesz spację o jeden raz za dużo.

Program `less` powstał jako lepsza wersja programu `more`, ale nowsze wersje tych programów prawie niczym się nie różnią.

Program `man`, omówiony później, używa programu `less` do wyświetlania sformatowanego tekstu. Większość innych systemów UNIXowych używa do tego celu programu `more`. Nie przejmuj się tym, pamiętaj tylko, że aby wyjść z programu `less`, trzeba wcisnąć klawisz `q`.

Bardzo przydatna bywa również możliwość wysłania wyników działania programu do pliku, zamiast na ekran. Czasem potrzebujesz na przykład zachować wyniki działania programu na później, albo chcesz przesyłać je do pliku z powodu długiego czasu wykonywania programu.

Aby skierować wyniki działania programu do pliku, należy użyć symbolu `>`. Przykładowo, można przesyłać dane wyjściowe polecenia `ls -l /bin` do pliku `output`, wydając polecenie `ls -l /bin > output`. Również w tym przypadku spacje otaczające znak `>` są opcjonalne, ale pozwalają na zwiększenie czytelności wydanego polecenia.

Jeśli teraz wydasz polecenie `ls` lub `ls -l`, to stwierdzisz, że w katalogu bieżącym został utworzony nowy plik o nazwie `output`.

Jego zawartość obejrzeć można za pomocą polecenia `more`. Jako parametr należy podać nazwę pliku, którego zawartość chcesz wyświetlić: `more output`.



Bądź ostrożny! Używając symbolu `>` zamazujesz istniejącą zawartość pliku, do którego wysydasz informacje (o ile taki plik już istniał). Jeśli na przykład w katalogu bieżącym znajdowałby się plik o nazwie `output`, wydanie polecenia `ls -l /bin > output` spowoduje nieodwracalną utratę danych zapisanych pierwotnie w tym pliku. Linux nie ostrzeże Cię o zaistniałej sytuacji! Aby zabezpieczyć się przed takimi pomyłkami, warto przed wydaniem polecenia upewnić się, że plik, do którego zamierzasz skierować dane, nie istnieje.

Szczególną ostrożność zachować należy w przypadku, gdy nie znajdujesz się w swoim katalogu domowym i jesteś zalogowany jako root. Może na przykład zdarzyć się pomyłka tego rodzaju: `ls -l /bin > more`. Takie polecenie, wydane w katalogu `/bin`, spowoduje zniszczenie kodu programu `more` i konieczność jego przeinstalowania.

Jeśli chcesz dopisać informacje na koniec istniejącego pliku, użyj symbolu `>>`. Polecenie `who >> output` powoduje dopisanie informacji o tym, kto jest zalogowany, na koniec utworzonego wcześniej pliku.

Wyniki ostatniej operacji możesz obejrzeć, używając programu `more` lub `less`, ale ponieważ interesująca jest tylko końcówka pliku, spróbuj użyć polecenia `tail`. Polecenie `tail output` spowoduje wyświetlenie kilku ostatnich wierszy pliku `output`.

Konwencje używane do opisu poleceń

Istnieje zestaw powszechnie akceptowanych reguł, używanych do opisywania w sposób scisły i konsekwentny poleceń Linuxa. Dotyczą one sposobu oznaczania parametrów obowiązkowych czy opcjonalnych itp. Czasem konwencje te są używane, by dać pełny opis danego polecenia, czasem w przykładach jego użycia lub dla wyjaśnienia zasady działania. Na takie opisy możesz natknąć się w dokumentacji, na stronach `man` i w innych źródłach informacji.

Jeśli przyswoisz sobie poniższe zasady, łatwo będzie Ci orientować się w składni polecień Linuxa i UNIX-a.

Sześć podstawowych reguł opisywania poleceń

1. Tekst nie zawarty pomiędzy znakami `[]`, `< >` lub `{ }` musi być wprowadzony dokładnie tak, jak jest podany.
2. Tekst zawarty w nawiasach kwadratowych `[]` jest opcjonalny. Można go wpisać lub nie. Przykładowo, składnia `ls [-l]` oznacza, że trzeba wpisać `ls` (zasada 1.), ale tekst `-l` jest opcjonalny i nie musi zostać użyty. Nie wpisuj przypadkiem nawiasów kwadratowych! W naszym przykładzie możesz wpisać `ls` lub `ls -l`, nie powinieneś natomiast pisać `ls [-l]`.
3. Nawiasy trójkątne (`< >`) oznaczają, że to, co zawarte pomiędzy nimi, musi być zastąpione przez odpowiedni tekst (zazwyczaj nazwę lub wartość). Składnia `more <nazwa_pliku>` oznacza, że powinieneś zamienić oznaczenie `<nazwa_pliku>` na rzeczywistą nazwę pliku, którego zawartość chcesz obejrzeć. Jeśli chcesz obejrzeć zawartość pliku `output`, powinieneś wydać polecenie `more output`. Pamiętaj, że nie powinieneś wpisywać samych nawiasów.

4. Nawiasy klamrowe ({}) oznaczają, że musisz wybrać jedną z kilku wartości podanych w ich wnętrzu. Poszczególne wartości są zazwyczaj oddzielone od siebie znakiem |, używanym tu znaczeniu „lub”. Na przykład składnia polecenie -{a|b} oznacza, że należy wpisać albo polecenie -a, albo polecenie -b.
5. Wielokropek (...) oznacza „i tak dalej”. Zazwyczaj używany jest z takimi parametrami, jak nazwy plików; dokładniej zostanie on opisany później.
6. Szósta zasada mówi, że nawiasy mogą być używane razem, według potrzeb. Przykładowo, uruchamiając program more nie trzeba podawać nazwy pliku. Jego składnia może więc wyglądać następująco: more [<nazwa_pliku>]. Zewnętrzne nawiasy klamrowe wyjaśniają, że parametr jest opcjonalny. Jeśli zdecydujesz się go użyć, powinieneś w miejsce oznaczenia <nazwa_pliku> podstawić rzeczywistą nazwę pliku. Ponieważ jako parametry tego programu można podać więcej niż jedną nazwę pliku, pełniejszy opis składni wygląda tak: more [<nazwa_pliku> ...]. Oznacza to, że możesz podać dowolną ilość parametrów będących nazwami plików.

Pomoc dostępna podczas pracy

Linux posiada system pomocy dostępnej podczas pracy. Jeśli zapomnisz, jak należy użyć danego polecenia, bądź też nie wiesz, jakie znaczenie ma dana opcja, informacje te może podać system. Dwie metody uzyskiwania pomocy, które wypróbowujemy, to polecenie help – dostępne w interpreterze poleceń bash, i użycie programu man, obsługiwany nie tylko w Linuxie, ale i w większości systemów UNIX-owych.

Strony man

Nazwa man pochodzi od angielskiego słowa *manual*, czyli instrukcja obsługi (jak zwykle, twórcy UNIX-a skrócili nazwę, by można ją było szybciej wpisać). Wpisanie polecenia man <nazwa_programu> pozwala obejrzeć dokumentację dotyczącą danego programu.

Na przykład, po wydaniu polecenia man passwd wyświetlane zostaną informacje na temat polecenia passwd.

Ogólnie rzecz biorąc, strony man mają następującą strukturę:

POLECENIE(1)	Linux Programmer's Manual	POLECENIE(1)
NAME	Polecenie – krótkie omówienie działania.	
SYNOPSIS	<pełna składnia polecenia>	
DESCRIPTION	Opisowe wyjaśnienie działania polecenia.	
OPTIONS	Lista wszystkich dostępnych opcji wraz z ich opisem.	
FILES		

Lista plików używanych przez polecenie lub w jakiś sposób
z nim powiązanych.

SEE ALSO	POLECENIE_POKREWNE(1), POLECENIE_PODOBNE(1) itd.
BUGS	Czy to możliwe, żeby w Linuxie były jakieś błędy?
AUTHOR	Informacje o autorze
Linux 1.0	22 June 1994

Strona `man` zawierająca informacje o poleceniu `passwd` jest dość zrozumiała. Powinieneś jednak mieć świadomość, że czasem strony te pisane są językiem bardzo formalnym i mało zrozumiałym. Nie jest to spowodowane złośliwością ich autorów – po prostu w możliwie krótkim tekście zawarta jest duża ilość informacji, co nie ułatwia jej odbioru. Zajrzyj na przykład na stronę poświęconą poleceniu `ls` (wydaj polecenie `man ls`). Zaauważ, jak wiele jest dostępnych opcji i ile miejsca zajmuje ich opisanie.

Choć zrozumienie stron `man` wymaga nieco praktyki (i uważnego czytania), to kiedy już się do nich przyzwyczaisz, pierwszą rzeczą, jaką będziesz robił, gdy natkniesz się na nieznane Ci polecenie, będzie przejrzenie odpowiednich stron `man`.

Słowa kluczowe na stronach man

Czasem wiesz dokładnie, co chcesz zrobić, ale nie wiesz, jakiego polecenia powinieneś w tym celu użyć. Spróbuj w takiej sytuacji wydać polecenie `man -k <słowo_kluczowe>`. Program `man` w odpowiedzi poda listę nazw poleceń, w których pole `NAME` (patrz struktura powyżej), wliczając w nie krótki opis, zawiera zadane słowo kluczowe.

Oto co otrzymasz, gdy słowem kluczowym będzie wyraz `manual`:

```
darkstar:~$ man -k manual
man(1) - Format and display the online manual pages
whereis(1) - Locate binary, manual and or source of program
xman(1) - Manual page display program for the X Window System
```

Należy jednak rozważnie dobierać słowa kluczowe. Słowo `directory` (katalog) nie jest jeszcze takie złe, ale np. użycie słowa `file` (plik) spowoduje wyświetlenie bardzo dużej liczby nazw poleceń.



Zauważysz na pewno, że po nazwie polecenia występuje numer w nawiasach okrągłych, przeważnie (1). Odnosi się on do sekcji dokumentacji. Dawno temu, kiedy instrukcje obsługi systemu UNIX były drukowane w opasłych tomach, „normalne” polecenia znajdowały się w sekcji 1, pliki i programy używane przez administratatorów w sekcji 5, narzędzia programistyczne i procedury opisane były w sekcji 3, i tak dalej. Niektóre strony `man` nie opisują więc poleceń, tylko pliki czy odwołania systemowe.

Jeśli dany wpis pojawia się w więcej niż jednej sekcji, `man` pokaże

domyślnie stronę pochodząą z sekcji o najniższym numerze. Pozostałe strony można obejrzeć, podając numer sekcji bezpośrednio. Przykładowo, w sekcji 5 również znajduje się strona na temat polecenia `passwd`. Aby ją obejrzeć, musisz wydać polecenie `man 5 passwd`.

Ogólnie rzecz biorąc, `man <n> <polecenie>` wyświetli stronę `man` opisującą polecenie, znajdująca się w sekcji o numerze `n`.

help - polecenie interpretera poleceń bash

Po wydaniu jakiegoś polecenia w wierszu poleceń powłoka interpretuje je, a wyniki tej interpretacji przekazuje systemowi operacyjnemu. Linux następnie podejmuje odpowiednie kroki. Wiele z polecen Linuxa wymaga odnalezienia i uruchomienia jakiegoś programu. Jednak niektóre proste, często używane polecenia, interpreter potrafi obsłużyć samodzielnie, dzięki czemu eliminowana jest konieczność uruchamiania odrębnych programów. Inna grupa polecen obsługiwanych przez powłokę to udogodnienia, które powodują, że praca z nią jest przyjemniejsza i łatwiejsza. Jednym z nich jest polecenie `help`, dostarczające informacji o wbudowanych w interpreter polecen funkcjach.

W wierszu polecień wpisz `help`. Powinien zostać wyświetlony następujący komunikat:

```
GNU bash, version 1.14.15(1)
Shell commands that are defined internally. Type 'help' to see this list.
Type 'help name' to find out more about the function 'name'.
Use 'info bash' to find out more about the shell in general.

A star (*) next to a name means that the command is disabled.

%[DIGITS | WORD] [&] . [filename]
: [arg ...]
alias [name[=value] ...] bg [job_spec]
bind [-lvd] [-m keymap] [-f filename] break [n]
builtin [shell-builtin [arg ...]] case WORD in [PATTERN [| PATTERN])
cd [dir] command [-pVv] [command [arg ...]]
continue [n] declare [-frxi] name [=value] ...
dirs [-l] echo [-neE] [arg ...]
enable [-n] [name ...] eval [arg ...]
exec [ [-] file [redirection ...]] exit [n]
export [-n] [-f] [name ...] fc [-e name] [-nlr] [first] [last]
fg [job_spec] for NAME [in WORDS ...;] do COMMANDS
function NAME {COMMANDS;}
hash [-r] [name ...] getopts optstring name [arg]
history [n] [[-awrn] [filename]] help [pattern ...]
jobs [-lnp] [jobspec ...] | jobs if COMMANDS; then COMMANDS; [elif
let arg [arg ...] kill [-s sigspec | -sigspec] [pid]
logout local name[=value]
pushd [dir | +n |-n] popd [+n | -n]
read [-r] [name ...] pwd
return [n] readonly [-n] [-f] [name ...]
set [--abefhknutuvxldHCP] [-o optio select NAME [in WORDS ...;] do COMMAND
shift [n]
```

```

source filename
test [expr]
trap [arg] [signal_spec]
typeset [-[frxi]] name[=value]
umask [-S] [mode]
unset [-f] [-v] [name ...]
variables - Some variable names await [n]
while COMMANDS; do COMMANDS; done {COMMANDS}
suspend [-f]
times
type [-all] [-type | path] [name ...]
ulimit [-Shacdmcftpnuv [limit]]
unalias [-a] [name ...]
until COMMANDS; do COMMANDS; done

```

Aby obejrzeć cały wyświetlany tekst, będziesz musiał użyć programu `more` (wydaj polecenie `help | more`).

Symbole wieloznaczne - „*” oraz „?”

Symbole `*` oraz `?` służą do zastępowania dowolnych znaków w poleceniach powłoki, podobnie jak joker służy do zastępowania dowolnych innych kart w talii. Nie ma chyba wątpliwości, że są one bardzo przydatne.

W systemie Linux dostępnych jest kilka metod zastępowania znaków. Są one używane do skracania zapisu nazw plików (lub katalogów), na których dane polecenie ma operować. W tym podrozdziale omówimy dwa najczęściej używane symbole wieloznaczne: `*` oraz `?`. Dostępnych jest jeszcze kilka symboli o podobnej funkcji, ale ich użycie jest dość skomplikowane – omawianie ich w tym momencie tylko zaciemniłoby problem.

Najczęściej używa się symbolu `*`. Zastępuje on dowolną kombinację jednego lub więcej znaków. Przykładowo, zapis `c*` oznacza wszystkie nazwy plików zaczynające się od litery `c`. Można łatwo się o tym przekonać wydając polecenie `ls /bin/c*`. Wyświetlony zostanie spis wszystkich plików w katalogu `/bin`, których nazwy zaczynają się na `c`. Zgodnie z oczekiwaniemi działają również takie kombinacje jak `ls /bin/c*t` czy `ls /bin /*t`.

Symbol `?` ma nieco mniejsze możliwości: zastępuje on jeden dowolny znak. Różnicę pomiędzy tymi symbolami można łatwo zauważycy wydając polecenia `ls /bin/d*` i `ls /bin/d?`.



Symbole wieloznaczne mogą być używane tylko przy podawaniu nazw plików i katalogów. Nie należy się więc spodziewać, że po wpisaniu `pass*` i wcisnięciu klawisza *Enter* Linux wykona polecenie `passwd`.



Bądź bardzo ostrożny podczas używania symboli wieloznacznych w połączaniu z niebezpiecznymi poleceniami, np. usuwającymi pliki. Dobrym pomysłem jest wydanie najpierw polecenia `ls` z parametrem (zawierającym symbole wieloznaczne), którego zamierzasz użyć. Przekonasz się wtedy, czy chodzi Ci o te właśnie pliki. Nie zaszkodzi również sprawdzić dwa razy polecenia przed zatwierdzeniem go

klawiszem *Enter*.

Zmienne środowiskowe

Kiedy użytkownik loguje się do systemu, Linux zapisuje w tle sporo przydatnych danych, gotowych w każdej chwili do użycia. Są one przechowywane w tzw. zmiennych środowiskowych (ang. *environment variables*), posiadających nazwy powiązane w jakiś sposób z ich zawartością. W pewnym sensie my również używamy tego typu zmiennych: znamy pewną informację, którą można określić jako dzień tygodnia (nazwa zmiennej środowiskowej), ale co dzień takiej zmiennej przypisujemy inną wartość.

Aby obejrzeć listę eksportowanych (dostępnych dla każdego programu, który uruchamiasz) zmiennych środowiskowych, można użyć polecenia `env`:

```
darkstar:~$ env
FMHOME=/frame
HOME=/user/burek
HUSHLOGIN=FALSE
HZ=100
LOGNAME=burek
MAIL=/usr/spool/mail/burek
PATH=:./frame/bin:/bin:/usr/bin:/usr/burek/bin
SHELL=/bin/sh
TERM=ansi
TZ=EST5EDT
```

Nazwa zmiennej środowiskowej znajduje się po lewej, a jej wartość po prawej stronie znaku równości. Zmienne eksportowane są dostępne dla każdego uruchamianego programu, aż do momentu wylogowania się użytkownika. Można również definiować zmienne, które dostępne będą tylko w jednym programie.

Najważniejszą chyba ze zmiennych środowiskowych jest zmienność `PATH`, której wartość określa ścieżki przeszukiwania. Jak przekonasz się w następnym rozdziale, kiedy wpisujesz polecenie, Linux przeszukuje każdą z lokalizacji podanych w zmiennej `PATH`.

Dłuższa lista zmiennych środowiskowych wyświetlana jest po wydaniu polecenia `set`. Oto wynik jej działania w tym samym systemie co poprzednio:

```
darkstar:~$ set
FMHOME=/frame
HOME=/user/burek
HUSHLOGIN=FALSE
HZ=100
IFS=

LOGAME=burek
MAIL=/usr/spool/mail/burek
MAILCHECK=600
OPIND=1
PATH=:./frame/bin:/bin:/usr/bin:/usr/burek/bin:
PS1=$
```

```
PS2=>
SHELL=/bin/sh
TERM=ansi
TZ=EST5EDT
```

Zmienne, które nie pojawiły się na poprzedniej liście, to zmienne lokalne; nie są one eksportowane. Więcej informacji o eksportowaniu zmiennych znaleźć można w rozdziale 10. Można myśleć o zmiennych lokalnych jako o informacjach potrzebnych tylko w danym czasie i miejscu (na przykład nie ma sensu przechowywanie informacji o tym, na którym jesteś piętrze, gdy opuścisz już budynek). Choć czasem dane wyświetlane przez polecenia `set` i `env` pokrywają się, podają one nieco inne informacje.

Procesy i ich wyłączanie

W poprzednim rozdziale dowiedzieliśmy się o istnieniu polecenia `who`. Pobiera ono od jądra systemu informacje o tym, kto jest zalogowany, i wyświetla je na ekranie.

Linux gromadzi znacznie więcej informacji o systemie, niż tylko wiadomości o tym, kto jest zalogowany. Ponieważ jest to system wielozadaniowy, jednocześnie może wykonywać wiele programów. Każdy z nich jest osobnym procesem i o każdym z nich przechowywana jest spora ilość danych. Każde polecenie, które wydajesz, zwykle powoduje uruchomienie co najmniej jednego procesu.

Polecenie ps

Aby dowiedzieć się, jakie procesy są aktualnie wykonywane, wydaj polecenie `ps` (ang. *process status*, status procesów). Na ekranie wyświetlona zostanie krótka lista uruchomionych procesów (a dokładniej procesów, których jesteś właścicielem):

```
darkstar:~$ ps
PID   TTY      STAT      TIME      COMMAND
41    v01      S          0:00     -bash
134   v01      R          0:00     ps
```

Informacja podana w pierwszej kolumnie (`PID`) to identyfikator (numer) procesu. Linux używa go do rozróżniania procesów. Musisz znać ten numer, jeśli chcesz wyłączyć (ang. *kill*, zabić) proces. Każdy uruchomiony proces posiada własny numer z przedziału od 0 do 65536 (gdy numery procesów dojdą do wartości maksymalnej, numerowanie zaczyna się od zera, z zachowaniem unikalności każdego numeru). Identyfikatory numeryczne używane są dlatego, że ich stosowanie jest szybsze i łatwiejsze niż stosowanie nazw procesów.

Kolumna `TTY` zawiera informacje o tym, z którego terminala proces został uruchomiony.

Kolumna `STAT` podaje aktualny status procesu. Najczęściej spotyka się tu wartości `S` (ang. *sleeping*, uśpiony) oraz `R` (ang. *running*, działający). Proces uśpiony nie jest aktualnie aktywny. Nie daj się jednak zwieść pozorom, możliwe, że zrobił sobie tylko krótką przerwę – jego stan może się zmienić na `R` i z powrotem na `S` wielokrotnie w ciągu sekundy.

Kolumna `TIME` zawiera ilość czasu systemu przeznaczoną dla danego procesu. Żaden z uruchomionych w przykładzie procesów nie zajmuje znaczącej ilości czasu. Często zdarza się tak w przypadku szybkich komputerów – proces wykona się tak szybko, że nie zasłuży sobie nawet na jedną cyfrę w tej kolumnie.

Ostatnia kolumna – `NAME` – zawiera nazwę uruchomionego programu. Jest to zazwyczaj polecenie, które zostało wydane, aby uruchomić program. Często zdarza się również, że jeden program wywołuje inne, tworząc procesy potomne (ang. *child processes*). Te procesy również zostaną wypisane, mimo że nie zostały uruchomione bezpośrednio przez użytkownika. Interpreter poleceń, który uruchamia się przy logowaniu, ma przed nazwą myślnik (w przykładzie: `-bash`). Można również później uruchomić jakiś interpreter, ale do jego nazwy myślnik nie zostanie dodany.



Jeśli jesteś zalogowany jako `root`, po wydaniu polecenia `ps` otrzymasz pełną listę wszystkich procesów w systemie, ponieważ `root` jako administrator systemu jest właścicielem wszystkich procesów.

Jeśli jesteś „zwykłym” użytkownikiem, ale zalogowałeś się na więcej niż jednym terminalu (również wirtualnym, który można wybrać kombinacją klawiszy `Alt+Fn`, temat ten omówiony jest dokładniej w rozdziale 6. „Od czego zacząć?”), zobaczyz również swoje procesy uruchomione z innych terminali.

Nie przejmuj się, jeśli wyświetcone zostaną inne niż w przykładzie kolumny z informacjami. Różnią się one nieco od siebie w różnych wersjach programu `ps`, zależą także od użytych opcji. Podstawowe informacje pozostają jednak takie same.

Jedną z użytecznych opcji polecenia `ps` jest opcja `u`. Jest ona skrótem od angielskiego słowa `user` (użytkownik), ale oprócz identyfikatora użytkownika dodaje kilka innych kolumn z informacjami:

```
darkstar:~$ ps -u
USER      PID %CPU %MEM   SIZE  RSS TTY STAT START TIME COMMAND
burek     41  0.1  6.8   364   472 v01 S    23:19  0:01 -bash
burek    134  0.0  3.3    72   228 v01 R    23:34  0:00 ps -u
```

Oprócz nazwy użytkownika w kolumnie `USER`, pojawiły się też interesujące informacje o użyciu procesora (`%CPU`) i pamięci (`%MEM`) oraz czasie rozpoczęcia procesu.

Jeśli chcesz zobaczyć listę wszystkich procesów, nie tylko tych, których jesteś właścicielem, użyj opcji `a` (w przypadku użytkownika `root` nawet bez tej opcji wyświetlane są wszystkie procesy):

```
darkstar:~$ ps -a
PID  TTY  STAT   TIME  COMMAND
62   v03  S      0:00   /sbin/agetty 38400 tty3
63   v04  S      0:00   /sbin/agetty 38400 tty4
64   v05  S      0:00   /sbin/agetty 38400 tty5
65   v06  S      0:00   /sbin/agetty 38400 tty6
330  v02  S      0:00   -bash
217  v01  S      0:00   -bash
218  v01  S      0:00   ps -a
```

Jak widać, w systemie działają również procesy nie uruchamiane przez użytkowników. Większość z procesów, które widać w przykładzie, działa niezależnie od tego, czy ktokolwiek jest zalogowany. Te o nazwie podobnej do `/sbin/agetty 38400 tty3` to procesy logowania, działające na każdej konsoli wirtualnej.

Użycie łącznie opcji `a` i `u` daje następujące efekty:

```
darkstar:~$ ps -au
USER      PID %CPU %MEM   SIZE  RSS TTY STAT START TIME COMMAND
burek     121  0.1  6.8   364   472 v02 S    17:56 0:01 -bash
burek     134  0.0  3.3    72   228 v02 R    17:56 0:00 ps -u
root      72   0.0  3.6   390   532 v01 S    17:55 0:01 -bash
root      74   0.0  1.5    41   224 v03 S    17:55 0:00 /sbin/agetty
Σ38400 tty3
root      75   0.0  1.5    41   224 v04 S    17:55 0:00 /sbin/agetty
Σ38400 tty4
root      76   0.0  1.5    41   224 v05 S    17:55 0:00 /sbin/agetty
Σ38400 tty5
root      77   0.0  1.5    41   224 v06 S    17:55 0:00 /sbin/agetty
Σ38400 tty6
root      78   0.0  1.5    56   228 s00 S    17:55 0:02 gpm -t mman
```

Więcej informacji technicznych możesz uzyskać, używając opcji `l`:

```
darkstar:~$ ps -l
UID      PID  PPID PRI NI   SIZE  RSS WCHAN STAT TTY TIME COMMAND
0      501     41     1  15  0   364   472 114d9c S    v01 0:00 -bash
0      501    121    41   29  0    64   208    0 R    v01 0:00 ps -l
```

Interesujące informacje zawiera kolumna `PPID` (ang. *Process Parent ID*, numer procesu wywołującego) – dzięki danym w niej zawartym można stwierdzić, przez jaki inny proces dany proces został uruchomiony. Zauważ, że proces `ps -l` został uruchomiony przez proces `-bash`. Numer `PPID` dla interpretera polecen wynosi 1; jeśli porównasz go z wynikami działania polecenia `ps -au`, wówczas zauważysz, że jest to numer procesu `init`. Proces `init` jest pierwszym procesem uruchamianym w systemie, wywołującym wszystkie inne. Jeśli coś złego stanie się z nim, system po prostu się zawiesi.



Składnia polecenia `ps` nie jest zbyt spójna, jeśli chodzi o użycie opcji. Myślnik przed nazwą opcji nie we wszystkich wersjach programu jest wymagany (`ps -l` oznacza to samo, co `ps`). Ponieważ jednak większość poleceń linuxowych (jak również niektóre UNIX-owe wersje programu `ps`) wymaga myślnika przy podawaniu opcji, zalecamy stosowanie go.

Ważna jest również kolejność opcji, szczególnie jeśli łączysz opcje `l` oraz `u`. Spróbuj wpisać `ps -lu`, a następnie `ps -ul`. To zachowanie nie jest opisane na stronach `man`. Wnioski są dwa: po pierwsze, staraj się nie używać zbyt wielu opcji polecenia `man` jednocześnie; po drugie, strony `man` nie zawsze zawierają wszystkie po-

trzebne informacje.

Wyłączanie procesów: polecenie kill

Choć zazwyczaj procesy działają poprawnie i nie ma z nimi problemów, zdarza im się wymknąć spod kontroli. Przeważnie przyczyną takiego zachowania są błędy w programie, może to zdarzyć się również wtedy, gdy wprowadzisz błędne polecenie lub niewłaściwą opcję.

Umiejętność znalezienia i usunięcia takiego procesu jest niezbędna dla wszystkich użytkowników Linuxa. Kiedy jesteś administratorem systemu, staje się ona szczególnie ważna w sytuacji, gdy jakiś proces „poszedł w krzaki” albo się zapętlił.

Polecenie `kill` jest używane do zakończenia pracy procesu, gdy nie da się tego zrobić w innym sposob².



Zanim podejmiesz kroki opisane poniżej, upewnij się, że nie możesz zatrzymać procesu, wciskając *Control+C* lub w jakiś inny sposób (np. klawiszem `q`).

1. Przełącz się na inną konsolę wirtualną i zaloguj się jako `root`.
2. Uruchom polecenie `ps -u` i znajdź numer `PID` nieposłusznego procesu. Użyjesz go w następnym kroku.
3. Użyj polecenia `kill <PID>`. Upewnij się, że wpisałeś prawidłowy numer! Jako `root` możesz wyłączyć każdy proces, kiedy pomylisz się, wpisując numer `PID`.
4. Sprawdź, czy proces został usunięty, wydając polecenie `ps -u`. Możesz również wpisać `ps -u <PID>`; polecenie to wyświetli status tylko interesującego Cię procesu. Jeśli nie zobaczysz żadnych informacji, znaczy to, że proces jest już martwy. Przejdz wtedy do punktu 8. Lepiej jest jednak obejrzeć pełną listę procesów poleceniem `ps -u` (jeśli nie jest ona zbyt dłużna), ponieważ czasem (choć rzadko) zdarza się, że proces „odżywa” z innym numerem `PID`! Jeśli tak jest, przejdź do punktu 6.
5. Jeśli proces wciąż działa i ma ten sam `PID`, użyj opcji `-9` polecenia `kill` (`kill -9 <PID>`). Sprawdź, czy teraz został wyłączony, tak jak w punkcie 4. Jeśli nie, przejdź do punktu 7. Jeśli tak, przejdź do punktu 8.
6. Jeśli proces pojawił się ponownie z nowym numerem `PID`, oznacza to, że jest on uruchamiany automatycznie przez jakiś inny proces. Jedyną rzeczą, jaką możesz zrobić w takim wypadku, to zakończyć działanie procesu wywołującego.
7. Użyj polecenia `ps -1` i odczytaj numer `PPID` procesu, który sprawia kłopoty. Powinieneś dokładniej sprawdzić, jaki proces go uruchamia, wpisując `ps -u`

² Choć nie tylko: służy ono również do przesyłania do procesu innych sygnałów (przyp. tłum.).

<PPID> przed wyłączeniem procesu wywołującego. Jeśli jest to proces, który możesz wyłączyć, zrób to i przejdź do punktu 4. i, jeśli to konieczne, 5.

8. Proces został wyłączony. Pamiętaj, by się wylogować. Nie pozostawaj zalogowany jako `root` na konsoli wirtualnej, bo możesz wpaść w kłopoty, gdy zapomnisz, jakie posiada on uprawnienia.



Zdarza się, że procesu nie da się wyłączyć w żaden sposób. W takim przypadku najlepszą (i chyba jedyną) metodą jest ponowne uruchomienie systemu.

Linux nie pozwala zwykłym użytkownikom usuwać procesów innych użytkowników. Przykładowo, jeśli jesteś normalnym użytkownikiem i spróbowajesz zakończyć proces `init`, otrzymasz następujący komunikat:

```
darkstar:~$ kill 1  
kill: (1) - Not owner
```

Prawdę mówiąc, nawet użytkownik `root` nie jest w stanie wyłączyć tego procesu. Jest to właśnie jeden z tych „niezabijalnych” procesów, ponieważ jest on niezbędny do działania systemu.

Polecenie su, czyli jak stać się kimś innym

Zazwyczaj kiedy chcesz na chwilę zalogować się jako inny użytkownik, po prostu przełączasz się na inną konsolę wirtualną. Czasem jest to jednak niewygodne (np. gdy zajęłeś już wszystkie konsole) lub niemożliwe (gdy zalogowałeś się przez modem i nie masz dostępu do konsoli wirtualnych).

W takich przypadkach z pomocą przychodzi polecenie `su` (ang. *super user*). Jeśli nie podasz żadnych parametrów, zostaniesz zapytany o hasło użytkownika `root`. Jeśli wprowadzisz je prawidłowo, pojawi się znak zachęty `#` i będziesz posiadał wszelkie przywileje użytkownika `root`.

Można również stać się dowolnym innym użytkownikiem – trzeba tylko wydać polecenie `su <nazwa_użytkownika>`. Jeśli wydasz to polecenie gdy jesteś zalogowany jako `root`, wówczas nie zostaniesz zapytany o hasło, ponieważ i tak mógłbyś je zmienić, a poza tym masz dostęp do plików każdego użytkownika. W przeciwnym przypadku musisz podać hasło odpowiedniego użytkownika.



Choć polecenie `su` zapewnia wszystkie przywileje, jakie miałbyś logując się jako dany użytkownik, nie jest jednak wykonywana pełna procedura logowania, tzn. nie są przetwarzane pliki konfiguracyjne. Nie jest to więc dobra metoda, jeśli zamierzasz pracować nad czymś dłużej. Nie powinieneś też próbować rozwiązywać problemów,

korzystając z tego polecenia.

Program grep

Program `grep` potrafi znaleźć w pliku i wyświetlić wiersz zawierający tekst pasujący do zadanego wzorca.

Jego niewiele mówiąca nazwa jest skrótem pełnej (choć wcale nie bardziej zrozumiałej) nazwy angielskiej: *Global Regular Expression Parser*.

Przedstawimy dwie najprostsze metody używania programu `grep`. Jedną z nich jest filtrowanie danych wyjściowych pochodzących z innych programów, jego składnia jest wówczas następująca:

```
polecenie | grep < wzorzec >.
```

Przykładowo, jeśli chcesz obejrzeć listę procesów, ale tylko tych działających, możesz użyć polecenia `ps -a | grep R`. Program `grep` wyświetli tylko te wiersze danych generowanych przez program `ps`, w których znajdzie zadany wzorzec – wielką literę `R`, oznaczającą właśnie proces aktywny. Zauważ jednak, że gdyby ktoś uruchomił program `Robak`, to wiersz zawierający informacje o nim również zostałby wypisany (bez względu na jego status, ponieważ zawiera literę `R`). Można łatwo obejść tę niedogodność, każąc programowi `grep` szukać wielkiej litery `R` otoczonej spacjami: `ps -a | grep " R "`. Jeśli szukany tekst zawiera spacje lub inne znaki specjalne, należy użyć cudzysłowu.

Drugim sposobem użycia programu `grep` jest wyszukiwanie wierszy zawierających zadany tekst w pliku. Jego składnia w takim przypadku jest następująca: `grep < wzorzec > < nazwa_pliku >`. Bądź ostrożny, bo bardzo łatwo pomylić się wpisując parametry w złej kolejności. Powinieneś jak najprecyzyjniej dobierać wzorzec wyszukiwanego tekstu, by uniknąć znajdowania wierszy nie zawierających interesujących informacji.

Podsumowanie

Jak dotychczas, poznałeś już na tyle dużo poleceń Linuxa, by czuć się pewnie, jeśli chodzi o typowe ich zastosowania.

Ważne jest, byś umiał korzystać ze stron `man` dostarczanych wraz z Linuxem. Dobrym ćwiczeniem będzie teraz przejrzenie stron dotyczących polecień omawianych w ostatnich dwóch rozdziałach, czyli np. `login`, `passwd`, `who` oraz `adduser`. Jeśli zainteresuje Cię któryś z polecień wypisanych w sekcji `See also:` (Patrz również:), przejdź do odpowiedniej strony i sprawdź, do czego ono służy.

W rozdziale 8. wyjdziemy w końcu z katalogu domowego i „powieszmy” trochę w systemie plików. Jako administrator powinieneś wiedzieć przecież, co zawiera dysk twardego.

dy! Istnieją na przykład specjalne katalogi, przeznaczone tylko dla administratora, zawierające wiele ciekawych programów.

Przedstawimy też kilka innych ważnych poleceń. Poznasz większość poleceń niezbędnych zwykłemu użytkownikowi i posmakujesz kilku przeznaczonych tylko dla administratora. Jeśli jesteś zainteresowany tematami pokrewnymi, możesz przejść do innych rozdziałów.

Praca z edytormi tekstu dostarczonymi z Linuxem jest omówiona w rozdziale 16. „Edytory tekstu vi i emacs”.

O konfiguracji systemu X możesz dowiedzieć się więcej z rozdziału 22. „Instalacja i konfiguracja Xfree86”.

Programowanie w systemie Linux omówione jest w części piątej, począwszy od rozdziału 25. „gawk”.

Rozdział 8.

System plików

Ed Trejis i Tim Parker

W tym rozdziale:

- υ Pliki – informacje ogólne
- υ Katalogi – informacje ogólne
- υ Poruszanie się po systemie plików
- υ Tworzenie i usuwanie plików
- υ Usuwanie plików i katalogów
- υ Ważne katalogi systemu Linux

Aby zrozumieć działanie Linuxa i używać go w stopniu więcej niż podstawowym, musisz wiedzieć, czym w Linuxie są pliki i jak są one zorganizowane. Jeśli pracowałeś wcześniej z innymi systemami operacyjnymi, takimi jak DOS czy Windows, znasz już tę problematykę, ponieważ zastosowana w nich koncepcja plików i katalogów bazuje na rozwiązaniu przyjętym w systemie UNIX. Jak się jednak przekonasz, UNIX-owy (i Linuxowy) system plików jest jednak o wiele bardziej elastyczny, niż systemy wykorzystywane w DOS-ie i Windows.

Pliki – informacje ogólne

Najprostsza koncepcja pliku – z którą pewnie spotkałeś się w innych systemach operacyjnych – to odrębny zbiór danych znajdujący się na dysku twardym. Słowo „odrębny” oznacza, że w systemie istnieć może wiele plików, ale ich zawartości nie mogą się pokrywać. Wynika stąd, że niezbędna jest metoda ich identyfikacji. W Linuxie, pliki identyfikowane są poprzez nazwę i położenie (miejscie) w systemie plików. W każdym miejscu (lub *katalogu*) może istnieć co najwyżej jeden plik o danej nazwie. Przykładowo, je-

śli stworzysz plik o nazwie `opowiadanie`, a następnie do góry wpadnie Ci inny świetny pomysł, musisz następny plik nazwać jakoś inaczej, np. `opowiadanie2`, albo umieścić go w innym katalogu (jeżeli nie chcesz zapisać go na miejsce starego, tracąc jego zawartość bezpowrotnie).

Popularne typy plików

Pliki mogą zawierać bardzo różne rodzaje informacji. Najczęściej w systemie Linux będziesz miał do czynienia z trzema typami plików.

- v Pliki użytkowników. Są to pliki, które tworzysz i modyfikujesz; w najprostszym przypadku mogą one zawierać tekst lub liczby – tego typu pliki nauczysz się tworzyć w dalszej części tego rozdziału. Bardziej skomplikowane pliki użytkowników muszą być interpretowane przez inne programy, by można było zrozumieć ich zawartość. Pliki pochodzące z arkuszy kalkulacyjnych nie zawierają na pierwszy rzut oka nic oprócz dziwnych znaczków, jeśli oglądać je bezpośrednio. By z nimi pracować, musisz uruchomić arkusz kalkulacyjny, który będzie umiał je zinterpretować.
- v Pliki systemowe. Zawierają one informacje (najczęściej w postaci tekstu ASCII), które są czytane i używane przez Linuxa, na przykład po to, by stwierdzić, kto ma prawo zalogować się do systemu. Jako administrator jesteś odpowiedzialny za modyfikowanie ich zawartości. Przykładowo, kiedy dodajesz do systemu nowego użytkownika, modyfikujesz plik `/etc/passwd`, zawierający właśnie informacje o użytkownikach. Zwykli użytkownicy raczej nie mają do czynienia z plikami tego typu, za wyjątkiem ich własnych plików startowych znajdujących się w katalogach domowych (macierzystych).
- v Pliki wykonywalne. Zawierają one kod, który może być wykonany przez komputer i przeważnie nazywane są po prostu programami. Polecenie wykonania takiego pliku nakazuje komputerowi zrealizowanie zawartych w nim instrukcji. Mimo że zawartość pliku wykonywalnego jest zupełnie niezrozumiała dla człowieka, jest jednak całkowicie jasna (przynajmniej powinna być) dla komputera. Tworzenie lub modyfikowanie plików tego typu wymaga specjalnych narzędzi. Dowiesz się o nich więcej w części piątej, „Linux dla programistów”.

Choć podzieliliśmy pliki na trzy kategorie, powinieneś zdawać sobie sprawę, że z punktu widzenia systemu plików nie różnią się one niczym. Każdy plik zawiera po prostu pewne dane zapisane na dysku twardym. To, co w jest w nim faktycznie zapisane, nie ma dla systemu znaczenia aż do momentu, gdy zechcesz użyć takiego pliku. Na przykład, interpreter poleceń `bash` potrafi uruchomić dowolny plik wykonywalny, ale może nie być w stanie zinterpretować pliku systemowego czy pliku z danymi użytkownika. Pliki danych zasadniczo zrozumiałe są tylko dla programów, które je utworzyły, nie są natomiast zrozumiałe dla samego systemu operacyjnego. Jedyne odstępstwo od tej reguli to pliki z danymi używane podczas uruchamiania i pracy systemu, których zawartość jest zrozumiała dla Linuxa.

Nazwy plików

W systemie Linux nazwy plików mogą mieć do 256 znaków długości. Mogą zawierać małe i wielkie litery, cyfry oraz kilka innych znaków (na przykład myślnik, podkreślenie, kropka czy spacja) Choć możesz używać nazw plików o długości do 256 znaków, musisz zdawać sobie sprawę z dwóch ograniczeń.

Po pierwsze, nie wszystkie znaki w nazwie są znaczące. Jeśli masz dwie nazwy plików o długości 250 znaków, różniące się tylko ostatnim znakiem, Linux potraktuje je jako jednakowe. Dzieje się tak, ponieważ rozpoznaje on pliki na podstawie pierwszych 32 (w niektórych wersjach 64) znaków nazwy. Reszta nazwy pliku jest po prostu dla wygody użytkownika; Linux zapisuje ją, ale nie wykorzystuje podczas identyfikacji pliku.

Po drugie, musisz pamiętać, że nazwy czasem trzeba wpisywać. Zdrowy rozsądek podpowiada, by nie używać nadmiernie długich nazw. Jeśli posiadasz plik zawierający dane statystyczne z lutego, lepiej nazwać go `luty_stat` (lub `dane_luty`), niż `dane_statystyczne_z_lutego`. Oczywiście z punktu widzenia systemu druga nazwa jest równie dobra jak pierwsza.

Nazwy plików nie zawierają zwykle zarezerwowanych znaków specjalnych, takich jak gwiazdka, znak zapytania, lewy ukośnik (/) i spacja, ponieważ mają one szczególne znaczenie dla interpretera poleceń. Dwa z nich zostały omówione dokładniej w poprzednim rozdziale, przy okazji dyskutowania symboli wieloznacznych; inne zostaną przedstawione w rozdziałach poświęconych powłokom systemu. Choć możliwe jest tworzenie plików, których nazwy zawierają te znaki, nie jest to zalecane, gdyż są one potencjalną przyczyną problemów dla systemu operacyjnego i aplikacji.

Katalogi - informacje ogólne

Linux, podobnie jak wiele systemów komputerowych, organizuje pliki w *katalogi*. O katalogach można myśleć jak o szufladach w kartotece, zawierających teczki – pliki z danymi. Pomiędzy biurową kartoteką a linuxowym systemem plików jest jednak zasadnicza różnica: szuflada raczej nie zawiera w sobie innej szuflady, natomiast katalog może zawierać inne katalogi. W systemie Linux istnieje *de facto* jeden główny katalog, który zawiera wszystkie inne katalogi wchodzące w skład systemu plików – bezpośrednio lub poprzez inne katalogi.

Katalogi nadzędne i podkatalogi

Jeśli w katalogu A znajduje się katalog B, mówimy, że B jest podkatalogiem A, oraz że A jest katalogiem nadzędnym katalogu B. Z tymi określeniami spotkasz się jeszcze wiele razy.

W Linuxie nie ma formalnego ograniczenia głębokości zagnieżdżenia katalogów, podobnie jak nie ma ograniczenia dla liczby plików w katalogu. Limitem jest tylko ilość wolnego miejsca na dysku twardym. Katalog może zawierać dowolną liczbę podkatalogów,

każdy z nich znów nieograniczoną liczbę własnych podkatalogów i tak dalej. W każdym katalogu czy podkatalogu może znajdować się dowolna ilość plików. Kluczem do zrozumienia systemu plików w Linuxie jest fakt, że istnieje jeden katalog na najwyższym poziomie, który w swoich podkatalogach i podkatalogach podkatalogów itd. zawiera cały system plików.

Katalog główny

W systemie Linux katalog zawierający wszystkie inne katalogi nazywany jest katalogiem głównym (ang. *root directory*). Jest on nadzędny dla wszystkich innych katalogów; każdy z nich znajduje się na którymś z poziomów jego podkatalogów. Katalog główny oznaczany jest symbolem /.

Struktura katalogów bywa też nazywana *drzewem katalogów* (ang. *directory tree*), gdyż rozgałęzia się ona na kształt drzewa, począwszy od katalogu głównego (ang. *root – korzeń*). Czasem jest ona również nazywana strukturą hierarchiczną, ponieważ istnieje w niej hierarchia poziomów, z katalogiem głównym (/) stojącym na samej górze.

Na koniec jeszcze krótka uwaga. Otóż w Linuxie (a zatem i w tej książce) często używa się określenia „system plików” (ang. *file system*) w miejsce „struktury katalogów” (*directory structure*). Wynika to z pewnych uwarunkowań historycznych (okreście *file system* używane jest w UNIX-ie).

Jak nazywane są katalogi

Nazwy katalogów podlegają takim samym regułom jak nazwy plików. Mogą zawierać małe i wielkie litery, cyfry oraz takie znaki, jak myślnik, kropka czy podkreślenie. Linux w zasadzie nie odróżnia nazw katalogów od nazw plików. Podobnie jak w przypadku plików, w systemie może znajdować się dowolna ilość katalogów o takich samych nazwach, pod warunkiem, że nie znajdują się one w tym samym katalogu nadzędnym.

Znak / (ang. *slash*, ukośnik) jest używany do wskazywania na pliki i katalogi znajdujące się w innych katalogach. Przykładowo, `usr/bin` oznacza, że `bin` to coś, czego należy szukać w katalogu `usr`. Nie jest wcale powiedziane, czy jest to plik, czy katalog, ale wiadomo na pewno, że katalogiem jest `usr` (bo w pliku nie może być zawarty inny plik ani katalog). Jeśli dana jest ścieżka dostępu `usr/bin/grep`, wiadomo, że `usr` i `bin` to nazwy katalogów, ale niczego nie można powiedzieć na pewno o identyfikatorze `grep`. Polecenie `ls`, wyświetlając zawartość katalogu, zaznacza podkatalogi, pokazując przy ich nazwach znak / (czasem wymaga to podania opcji `-F`); na przykład jeśli `bin` jest podkatalogiem katalogu bieżącego, zostanie wyświetlony zostanie tekst `bin/`, co sugeruje, że może istnieć plik lub katalog o nazwie `bin/bash`, czyli że `bin/` jest katalogiem.

Jak wspomniano wcześniej, katalog główny oznaczany jest symbolem / i nie ma innej nazwy. Łatwo jest określić, czy znak / jest użyty do rozdzielenia nazw katalogów, czy oznacza katalog główny. Jeśli nie znajdują się przed nim żadne inne identyfikatory, musi być to katalog główny. Na przykład `/usr` oznacza podkatalog `usr` w katalogu głównym, zaś `/usr/bin` oznacza coś o nazwie `bin`, znajdujące się w katalogu `usr`, który z kolei jest podkatalogiem katalogu głównego. Katalog główny, z definicji, nie może być podkatalogiem żadnego innego katalogu.

Katalog domowy

W systemie Linux każdy użytkownik posiada swój własny katalog, nazywany katalogiem domowym (macierzystym). W tym katalogu można przechowywać swoje pliki i zakładać podkatalogi. Użytkownik ma zazwyczaj pełną kontrolę nad tym, co znajduje się w jego katalogu domowym. Zwykle nie są tam przechowywane żadne pliki należące do innych użytkowników ani pliki systemowe, więc można tworzyć i usuwać pliki i katalogi według uznania.



Katalog domowy nie zapewnia prywatności. Normalnie każdy użytkownik może wejść do katalogu innego użytkownika i czytać (a także kopować) jego pliki (choć nie może ich usuwać ani zmieniać). Kiedy Linux udostępnia katalog domowy, daje jakby własne biuro, które nie jest zamknięte, podobnie jak wszystkie szafki i szuflady.

Jeśli chcesz zachować prywatność, musisz zablokować dostęp do katalogu domowego. Temat ten jest omówiony w rozdziale 9. „Prawa dostępu do plików i katalogów”. Zaglądanie do czyjegoś katalogu domowego jest uważane za niegrzeczne i wściekskie, tak jak zaglądanie do czyjegoś biurka, ale świat jest pełny wściekskich ludzi.

Należy również wziąć pod uwagę to, że każdy użytkownik zalogowany jako `root` może czytać i manipulować wszystkimi plikami w systemie. Jeśli nie możesz zaufać administratorowi, to po prostu nie używaj systemu.

Położenie katalogu domowego w systemie plików nie może być zmienione przez użytkownika. Dzięki temu zapewniony jest względny porządek; nie bez znaczenia są również kwestie związane z bezpieczeństwem. Położenie katalogów domowych zależy od wersji Linuxa oraz upodobań administratora. Zwykle są one podkatalogami katalogu `/home` lub `/usr`. Po zalogowaniu się do systemu bieżącym katalogiem jest właśnie katalog domowy.

Poruszanie się po systemie plików

Poruszanie się po systemie plików w Linuxie jest bardzo proste. Wymaga opanowania jedynie dwóch poleceń, z których jedno nie wymaga żadnych parametrów.

Polecenie `pwd` - gdzie to ja jestem

Wpisz w wierszu poleceń polecenie `pwd`. Zostaną wyświetlane komunikaty:

```
darkstar:~$ pwd  
/home/burek  
darkstar:~$
```

Informacja, którą otrzymałeś, oznacza, że katalogiem bieżącym jest `/home/burek` (jeśli logujesz się jako użytkownik o identyfikatorze innym niż `burek`, zamiast `burek` wyświetlony zostanie tenże identyfikator). Jest to Twój katalog domowy. Znajdujesz się w nim zawsze po zalogowaniu się do systemu.

Nazwa `pwd` pochodzi od angielskich słów *print working directory* (wyświetl bieżący katalog) – twórcy UNIX-a jak zwykle skrócili ją, by była łatwiejsza do wpisania. Zamiast terminu *katalog bieżący* czasem używa się określenia *katalog aktualny* – mają one to samo znaczenie.

Możesz się zastanawiać, co to właściwie jest katalog bieżący. Określenie to oznacza, że wszystkie wydane polecenia będą domyślnie działać w tym właśnie katalogu. Po wydaniu polecenia `ls` wyświetcone zostaną informacje o zawartości katalogu bieżącego. Bieżący katalog zmienić można poleciением `cd`, ale o tym za chwilę.

Absolutne i relatywne ścieżki dostępu

Jeśli podana zostanie tylko nazwa pliku, Linux szuka go w bieżącym katalogu. Polecenie `more mojplik` wyświetli na ekranie zawartość pliku `mojplik`, ale pod warunkiem, że znajduje się on w bieżącym katalogu – w przeciwnym przypadku program `more` nie będzie mógł go odnaleźć.

Czasem jednak chodzi o plik zapisany w jakimś innym miejscu. Założmy, że założyłeś katalog `ksiazka`, w którym znajduje się plik `rozdzial1`. Aby obejrzeć jego zawartość, powinieneś wydać polecenie `more ksiazka/rozdzial1`. Tak podana ścieżka nazywana jest relatywną (względną) ścieżką dostępu, ponieważ jest ona zależna od tego, w którym katalogu aktualnie się znajdujesz. Położenie pliku zostało podane względem katalogu bieżącego. Gdy zmienisz katalog bieżący, polecenie nie zadziała.

Istnieją dwie specjalne, zarezerwowane nazwy katalogów: „..” oraz „...”. Jedna kropka oznacza zawsze katalog bieżący, natomiast dwie kropki to katalog w stosunku do niego nadzędny (jak używać tych nazw pokażemy w dalszej części tego rozdziału). Ścieżki dostępu zawierające te nazwy są z definicji ścieżkami względnymi¹.

Ścieżka dostępu, która jest prawidłowa bez względu na to, w jakim katalogu aktualnie się znajdujesz, nazywana jest ścieżką absolutną (czasem też pełną lub bezwzględną). Ścieżka taka zawsze rozpoczyna się od symbolu `/`, który – jak wiesz – oznacza katalog główny. Jeśli więc podasz ścieżkę `/home/burek/ksiazka/rozdzial1`, żaden program nie będzie miał wątpliwości, o jaki plik chodzi. Każdy plik w systemie posiada inną pełną ścieżką dostępu (pełną nazwę). W podanym wyżej przykładzie Linux rozpocznie szukanie w katalogu głównym. Następnie znajdzie katalog `home` i na chwilę ustawi go jako bieżący. Potem poszuka katalogu `burek`, wejdzie do niego, znajdzie katalog `ksiazka` i dopiero po wejściu do tego katalogu będzie szukał pliku lub katalogu o nazwie `rozdzial1`. Pełna ścieżka dostępu jest swego rodzaju mapą, pozwalającą na odnalezienie pliku czy katalogu w systemie plików, bez względu na to, jak jest on rozbudowany.

¹ Można wymyślić ścieżkę bezwzględną zawierającą te symbole, na przykład `/usr/bin/.../lib`, ale takie konstrukcje nie mają zastosowania w praktyce (*przyp. tłum.*).

Ktoś inny może mieć w swoim katalogu domowym katalog `ksiazka`, który również może zawierać plik `rozdzial1`. Używając ścieżki względnej, `ksiazka/rozdzial1`, nie można określić, o który z tych plików chodzi – zależy to od katalogu aktualnego. Jednak absolutne ścieżki dostępu do tych plików są różne, dajmy na to `/home/burek/ksiazka/rozdzial1` i `/home/reksio/ksiazka/rozdzial1`, nie pozostawiając żadnych wątpliwości. Pliki te zapisane są w dwóch różnych katalogach, a zbieżność nazw może być czysto przypadkowa. Mogą one oczywiście mieć zupełnie różne zawartości.

Idziemy na spacer: polecenie cd

Polecenie `cd` (ang. *change directory*, zmień katalog) pozwala na zmianę bieżącego katalogu. Jeśli pracowałeś wcześniej z systemem DOS, spotkałeś się już z tym poleceniem (tak, zostało ono przeniesione z UNIX-a!).

Jego składnia jest następująca:

```
cd <katalog>
```

Pomiędzy poleceniem `cd` a nazwą katalogu musi wystąpić spacja. Nazwa katalogu może być ścieżką absolutną lub względną. Oto przykład jego użycia.

```
darkstar:~$ pwd  
/home/burek  
darkstar:~$ cd ..  
darkstar:/home$ pwd  
/home  
darkstar:~$ cd ..  
darkstar:/$ pwd  
/  
darkstar:/$ cd ..  
darkstar:/$ pwd  
/
```

Jak widać w powyższym przykładzie, rozpoczęliśmy nasz „spacer” w katalogu `/home/burek`. Potem przeszliśmy poziom wyżej (używając symbolu „..”) – do katalogu `/home`. Następnie jeszcze poziom wyżej – do katalogu głównego. Wyżej już się nie da – próba zmiany katalogu na nadrzędny nie daje żadnych rezultatów.

Zauważ, że Linux zwykle podaje nazwę aktualnego katalogu w wierszu poleceń, więc nie trzeba co chwilę wpisywać polecenia `pwd` (będziemy to jednak robić w naszych przykładach, że by nie pozostawić żadnych wątpliwości). Nie jest to jednak bezwzględna reguła, ponieważ może być zmienione przez administratora².

Załóżmy, że chcesz przejść do jednego z podkatalogów katalogu domowego. Wróćmy więc do niego, a następnie przejdźmy do podkatalogu `ksiazka`:

```
darkstar:/$ cd /home/burek  
darkstar:~$ pwd  
/home/burek  
darkstar:~$ cd ksiazka  
darkstar:~/ksiazka$ pwd  
/home/burek/ksiazka
```

² Użytkownik może również dokonać takiej zmiany (*przyp. tłum.*).

W tym przykładzie, najpierw – używając pełnej ścieżki dostępu – zmieniliśmy katalog na `/home/burek`, a następnie weszliśmy do podkatalogu `ksiazka`. Ponieważ wiedzieliśmy, że katalog `ksiazka` jest podkatalogiem katalogu domowego, mogliśmy użyć krótszej ścieżki relatywnej. Gdy jednak chcesz uniknąć pomyłek, używaj ścieżek absolutnych, np.:

```
darkstar:~$ cd /usr/bin  
darkstar:/usr/bin$ pwd  
/usr/bin
```

Gdy podasz ścieżkę absolutną, polecenie zadziała prawidłowo bez względu na to, w jakim katalogu znajdujesz się, gdy je wydajesz. Katalog, w którym znajdujesz się po wydaniu polecenia `cd ..` zależy od tego, który katalog jest katalogiem bieżącym.

By przekonać się o skutkach zmiany katalogu bieżącego, wydaj polecenie `ls`. Lista plików jest bardzo dłużna. Polecenie `ls` wyświetla dane o zawartości katalogu bieżącego, czyli `/usr/bin`, w którym znajduje się o wiele więcej plików niż w Twoim katalogu domowym.

Wszędzie dobrze, ale w domu najlepiej

Istnieje pewna poręczna sztuczka, która – nie wiedzieć czemu – jest dość rzadko wykorzystywana przez użytkowników. Wydaj polecenie `cd` bez żadnych parametrów:

```
darkstar:/usr/bin$ cd  
darkstar:~$ pwd  
/home/burek
```

Polecenie `cd` bez parametrów zawsze przenosi Cię do katalogu domowego. Kiedy ząglabisz się bardzo w podkatalogi jakiegoś katalogu, możesz szybko wrócić do katalogu domowego, wpisując `cd`, albo do katalogu głównego, wydając polecenie `cd /`.

Znak `~` w wierszu poleceń ma szczególne znaczenie. Zastępuje on nazwę katalogu domowego. Mimo to nie ma powodu, by zamiast polecenia `cd` używać `cd ~`. Symbol ten bywa jednak przydatny w innych sytuacjach.

Linux symbolem `~` oznacza również katalog nadrzędny w stosunku do katalogów domowych użytkowników. Dzięki temu można użyć polecenia `cd ~<użytkownik>`, by przenieść się do katalogu domowego użytkownika o danym identyfikatorze. Jest to szczególnie przydatne w dużych systemach, w których katalogi domowe rozmieszczone są w katalogach innych niż `/home` (a nawet w kilku różnych katalogach).

Kiedy trzeba przejść do jakiegoś odległego katalogu, dobrze jest zrobić to w kilku krokach. Ułatwi to uniknięcie pomyłek podczas wpisywania nazw (czasem nawet trudno zorientować się, w którym miejscu wystąpił błąd). Zaoszczędzi również pisania w przypadku, gdy popełnisz jakiś błąd. Spójrzmy na następujący przykład:

```
darkstar:~$ cd /usr/docs/faq/unix  
bash: /usr/docs/faq/unix: No such file or directory
```

A przecież dałbyś sobie głowę uciąć, że taki katalog istnieje. Spróbujmy inaczej:

```
darkstar:~$ cd /usr  
darkstar:/usr$ cd docs  
bash: docs: No such file or directory
```

A więc tu jest błąd. Spróbujmy go zidentyfikować:

```
darkstar:/usr$ ls  
bin/ doc/ games/ info/ man/ sbin/ spool/
```

No tak. Katalog nie nazywa się `docs`, tylko `doc`.

```
darkstar:/usr$ cd doc  
darkstar:/usr/doc$ cd faq/unix  
darkstar:/usr/doc/faq/unix$ pwd  
/usr/doc/faq/unix
```

Tworzenie i usuwanie plików

W Linuxie istnieje wiele sposobów tworzenia i usuwania plików. Niektóre z nich są tak łatwe, że trzeba bardzo uważać, by nie pozbyć się jakichś ważnych danych!



Przez następne podrozdziały przejdź bardzo uważnie, zalogowany jako zwykły użytkownik. Dopiero kiedy dokładnie zrozumiesz działanie podanych tu poleceń, będziesz mógł bezpiecznie używać ich zalogowany jako `root`.

W systemie Linux nie ma sposobu na przywrócenie usuniętych plików! Uważaj więc na wydawane polecenia.

Wróć do swojego katalogu domowego za pomocą polecenia `cd`. Upewnij się, że jesteś tam, gdzie powinieneś, wyając polecenie `pwd`.

W poprzednim rozdziale utworzyliśmy plik, wyając polecenie `ls -l /bin >output`. Spowodowało ono powstanie nowego pliku o nazwie `output`. Przekierowanie jest więc jedną z metod tworzenia plików.

A co w przypadku, jeśli chciałbyś do pliku zapisać tekst inny niż dane wyjściowe jakiegoś polecenia? Łatwym, choć niezbyt elastycznym sposobem jest użycie programu `cat`.

Polecenie cat

Polecenie `cat` jest jednym z najprostszych – a co za tym idzie – najużyteczniejszych polecen Linuxa. Program ten pobiera znaki ze swojego wejścia (domyślnie jest to klawatura) i przekazuje je na wyjście (domyślnie na ekran terminala). Do czego może służyć taki dziwadło? Spróbujmy. Uruchom program `cat`.

```
darkstar:~$ cat
```

Kursor przeniósł się do następnego wiersza, ale nic innego się nie stało. Polecenie `cat` oczekuje teraz na jakieś dane wejściowe, więc mu je podajmy:

```
hello  
hello  
co  
co  
asdf  
asdf
```

Cokolwiek byśmy wpisali, zostanie jeszcze raz wyświetcone na ekranie po wciśnięciu klawisza *Enter*. Jak się z tego wydostać? Wciśnij ^D (*Control+D*). Jeśli nie jesteś na początku wiersza, będziesz to musiał zrobić dwa razy. ^D to w systemie Linux symbol końca pliku. Kiedy program `cat` natknie się w danych wejściowych na taki znak, wówczas uzna, że plik wejściowy się skończył i pora przejść do następnego. Jeśli nie ma następnego pliku, należy zakończyć działanie.

W naszym przykładzie `cat` pobierał dane z klawiatury i wypisywał je na ekran. Nie jest to zbyt użyteczne. Na szczęście jest to program o wiele bardziej elastyczny, niż mogłoby się wydawać.



Kiedy mówimy, że program zakończył działanie, mamy na myśli, że skończył on wykonywać wszelkie operacje i oddał sterowanie do interpretera polecen. Może wydawać się dziwne, że mówimy „program zakończył działanie”, jeśli z punktu widzenia użytkownika to on zakończył działanie programu. Jest to pozostałość z wczesnych lat istnienia UNIX-a - programiści tworzący ten system mieli po prostu zwyczaj patrzeć na świat z perspektywy pisanych przez siebie programów.

Jak więc użyć programu `cat` do utworzenia pliku? Wystarczy przekierować jego wyjście:

```
darkstar:~$ cat >nowyplik  
Ahoj, przygodo  
jakis tekst
```

Mogna wpisać dowolną ilość tekstu. Kiedy skończysz, wciśnij w nowym wierszu klawisz ^D . Program `cat` zamiast na ekranie wyświetlać to, co pisaleś, posłał dane do pliku.

Chcialbyś teraz pewnie obejrzeć zawartość pliku `nowyplik`. Mogesz oczywiście użyć w tym celu programów `more` lub `less`, ale dlaczego nie miałby zrobić tego program `cat`? W takim przypadku jego składnia jest następująca:

```
cat <nazwa_pliku_do_wyświetlenia>.
```

Spróbujmy:

```
darkstar:~$ cat nowyplik  
Ahoj, przygodo  
jakis tekst  
darkstar:~$
```

Możliwe jest również dołaczanie tekstu na koniec istniejącego pliku za pomocą symbolu >> (uwaga: znak ^D nie pojawi się na ekranie; został tu dodany dla poprawienia czytelności).

```
darkstar:~$ cat >>nowyplik
Inne wiersze tekstu
^D
darkstar:~$ cat nowyplik
Ahoj, przygodo
jakis tekst
Inne wiersze tekstu
darkstar:~$
```

Zobaczmy, co jeszcze potrafi program `cat` (jest to w zasadzie jego podstawowe przeznaczenie):

```
darkstar:~$ cat >innyplik
Calkiem inny tekst
^D
darkstar:~$
```

Spróbuj teraz wydać polecenia:

```
darkstar:~$ cat nowyplik innyplik > trzeciplik
darkstar:~$ cat trzeciplik
Ahoj, przygodo
jakis tekst
Inne wiersze tekstu
Calkiem inny tekst
darkstar:~$
```

Nazwa programu `cat` jest skrótem od angielskiego słowa *concatenate*, „czyli połącz, klej”. Pobiera on wszystkie pliki z wejścia i kleje w jeden plik, przekazując go na wyjście.

Czasem jednak trzeba zmienić jeden wiersz tekstu w istniejącym pliku albo utworzyć większy, bardziej skomplikowany plik tekstowy. Trudno w takiej sytuacji korzystać z programu `cat`, który nie pozwala na poprawianie błędów powstałych w trakcie pisania. Powinieneś raczej użyć któregoś z edytorów tekstu dostarczanych z Linuxem. Są one omówione w rozdziale 16. „Edytory tekstu: vi i emacs”.

Tworzenie katalogów

Aby utworzyć nowy katalog, należy użyć polecenia `mkdir`. Jego składnia ma postać

```
mkdir <nazwa_nowego_katalogu>.
```

Utwórzmy w katalogu domowym podkatalog `nowykat`:

```
darkstar:~$ ls
innyplik          nowyplik        trzeciplik
darkstar:~$ mkdir nowykat
darkstar:~$ ls
innyplik          nowykat/       nowyplik        trzeciplik
```



Polecenie `mkdir` jest Ci pewnie znane z systemu MS-DOS. Mogło być tam skrócone do formy `md`. Mogłoby się wydawać, że w Linuxie skrót taki powinien również być dostępny, ponieważ jak dotąd wszystkie polecenia miały nazwy skrócone do granic możliwości. Tak jednak nie jest. Linux upiera się przy pełnej formie `mkdir`.

Jeśli często przełączasz się pomiędzy systemami Linux i DOS, warto używać w obu tych systemach polecenia `mkdir` - dzięki temu unikniesz pomyłek. Często również zdarzają się pomyłki polegające na tym, że w systemie MS-DOS użytkownik próbuje wydawać polecenia linuxowe, na przykład `ls`, zamiast ich DOS-owych odpowiedników.

Polecenie `mkdir` tworzy wpis dla podkatalogu w linuxowej tablicy zawierającej informacje o plikach i katalogach, nazywanej tablicą I-node. Żadne inne dane nie są na razie zapisywane na dysku, ponieważ katalog nie zawiera jeszcze żadnych fizycznych plików. Katalogi są używane w Linuxie jako udogodnienie dla użytkownika.

Parametrami polecenia `mkdir` mogą być ścieżki absolutne i relatywne, na przykład:

```
darkstar:~$ pwd
/home/burek
darkstar:~$ ls
darkstar:~$ mkdir ksiazka1
darkstar:~$ ls
ksiazka1/
darkstar:~$ mkdir /home/burek/ksiazka2
darkstar:~$ ls
ksiazka1/      ksiazka2/
```

W pierwszym przypadku użyliśmy ścieżki względnej, tworząc podkatalog `ksiazka1` w katalogu bieżącym; w drugim – absolutnej, tworząc podkatalog `ksiazka2` w tym samym miejscu. Obie metody dają taki sam wynik.

Przenoszenie i kopiowanie plików

Przenoszenie i kopiowanie plików to jedne z najczęściej wykonywanych przez administratora i użytkowników czynności. Polecenie `mv` służy do przenoszenia (co sprowadza się zazwyczaj do zmiany nazwy), a `cp` do kopирования plików. Składnia obu polecień jest podobna:

```
mv <źródło> <cel>
cp <źródło> <cel>
```

Jak widać, są to polecenia bardzo proste w użyciu. Oto kilka przykładów:

```
darkstar:~$ ls
innyplik      nowykat/      nowyplik      trzeciplik
darkstar:~$ mv innyplik przeniesionyplik
darkstar:~$ ls
nowykat/      nowyplik      przeniesionyplik      trzeciplik
darkstar:~$ cp trzeciplik xyz
darkstar:~$ ls
nowykat/      nowyplik      przeniesionyplik      trzeciplik      xyz
```

Można teraz użyć polecenia `cat` (albo `more` czy `less`), by przekonać się, że zawartość pliku `innyplik` jest teraz w pliku `przeniesionyplik`, oraz że plik `xyz` ma taką samą zawartość jak plik `trzeciplik`.

Przenoszenie i kopiowanie plików może być nieco trudniejsze w momencie, gdy pliki znajdują się w różnych katalogach, ponieważ rzeczywiste nazwy plików zawierają pełną ścieżkę dostępu do nich. Mimo tego Linux pozwala na opuszczenie części pełnej nazwy pliku, dzięki czemu polecenia stają się nieco krótsze.

Przypuśćmy, że chcesz przenieść plik `nowyplik` do katalogu `nowykat`. Jeśli chcesz, aby zachował on swą nazwę, możesz wydać polecenie:

```
darkstar:~$ mv nowyplik nowykat/nowyplik
```

Ale łatwiej jest wpisać

```
darkstar:~$ mv nowyplik nowykat
```

ponieważ jest to polecenie krótsze. Ponieważ jako cel podana została nazwa katalogu, Linux domyśli się, że plik ma zostać do niego przeniesiony bez zmiany nazwy.

Można również użyć polecenia `cd`, zmieniając katalog na katalog docelowy, i z niego właśnie rozpocząć kopianie:

```
darkstar:~$ cd nowykat  
darkstar:~/nowykat$ cp ../nowyplik .
```

Ten przykład nie jest już tak intuicyjny jak dwa poprzednie. Jako źródło podano `../nowyplik`, czyli plik o nazwie `nowyplik` znajdujący się w katalogu nadrzędnym w stosunku do bieżącego. Jako cel został podany katalog bieżący (kropka na końcu polecenia). Innymi słowy, polecenie `cp` przechodzi poziom wyżej w strukturze katalogów, znajduje tam plik `nowyplik`, wraca i kopiuje go do bieżącego katalogu. Ponieważ taka procedura jest mniej intuicyjna, zwykle kopianie przeprowadza się, będąc w katalogu zawierającym plik źródłowy.

Podczas przenoszenia i kopowania pliku można również zmienić jego nazwę w następujący sposób:

```
darkstar:~$ cp trzeciplik nowykat/innanazwa
```

Polecenie to utworzy kopię pliku `trzeciplik`, o nazwie `innanazwa` w katalogu `nowykat`.



Podczas kopiowania lub przenoszenia plików pomiędzy katalogami należy zawsze sprawdzić, czy katalog docelowy istnieje. Założymy, że tak nie jest; co się wtedy stanie? Spójrzmy na dwa przykłady.

Przypuśćmy, że do polecenia `mv nowyplik nowykat` wkradnie się literówka, i w efekcie zostanie wydane polecenie `mv nowyplik mo-`

wykat. Plik nie zostanie wówczas przeniesiony do katalogu nowykat, zamiast tego jego nazwa zostanie zmieniona na mowykat.

Może zdarzyć się odwrotna sytuacja, kiedy nie zdajesz sobie sprawy, że istnieje katalog o danej nazwie. Założmy, że chcesz utworzyć w bieżącym katalogu kopię pliku tekstowego o nazwie plik.txt, nazywając ją kopial. Należy w tym celu wydać polecenie cp plik.txt kopial. Nie zadziała ono jednak zgodnie z oczekiwaniemi, jeśli w katalogu bieżącym znajduje się podkatalog o nazwie kopial - wówczas w tym podkatalogu znajdzie się kopia pliku plik.txt, o takiej samej nazwie.

Polecenie mv jest o wiele bardziej wydajne niż polecenie cp, ponieważ tak naprawdę nie dotyczy ono zawartości pliku, a jedynie zmienia jego nazwę³ (wliczając w nią pełną ścieżkę dostępu), informując system, że dane zapisane już na dysku mają być dostępne pod inną etykietą.

Kiedy używasz polecenia cp, tworzysz na dysku drugą fizyczną kopię danych. Jest to procedura trwająca o wiele dłużej, niż zmiana nazwy (choć w przypadku małych plików różnica jest niezauważalna). Nie warto kopiować plików, jeśli trzeba je tylko przenieść.

Przenoszenie i kopiowanie plików za pomocą symboli wieloznacznych

Co zrobić, jeśli w katalogu jest 20 plików i trzeba skopiować je wszystkie do innego katalogu? Można w takim przypadku użyć symboli wieloznacznych: * oraz ?.

Jeśli chcesz skopiować wszystkie pliki znajdujące się w danym katalogu, użyj znaku *:

```
darkstar:~$ cp * /tmp
```

Powyższe polecenie kopiuje wszystkie pliki z bieżącego katalogu do podkatalogu tmp w katalogu głównym.

Można używać symbolu * w połączeniu z innymi literami po to, by wzorzec pasował tylko do niektórych plików. Założmy, że w katalogu znajdują się pliki ksiazka1, ksiazkapomysl, ksiazka-rozdzial-1 i wiersz.ksiazka. Jeśli trzeba skopiować pierwsze trzy pliki, powinieneś wydać polecenie cp ksiazka* /tmp. W miejscu tekstu ksiazka* Linux podstawi wszystkie nazwy plików znajdujących się w katalogu zaczynające się od liter ksiazka. Plik wiersz.ksiazka nie spełnia tego kryterium, nie zostanie więc skopiowany (ale zostałby skopiowany plik o nazwie ksiazka.wiersz).

Jak się okazało, polecenia cp i mv nie są skomplikowane, za to dość skomplikowane jest podawanie, jakich plików mają one dotyczyć. Jeśli wydaje Ci się to bardzo zawiłe, nie

³ Chyba że plik przenoszony jest do podkatalogu zapisanego w innym fizycznym systemie plików, czyli np. na innym dysku twardym czy na dyskietce (przyp. tłum.).

przejmuj się zbytnio. Nawet eksperci czasem mylą się przy wydawaniu tych „prostych” poleceń. Przećwicz podane przykłady i przemyśl je. Używanie symboli wieloznacznych rządzi się bardzo ścisłymi regułami, ale potrzeba chwili na ich zrozumienie i treningu, by stały się oczywiste.

Przenoszenie katalogów

Aby przenieść katalog, można użyć polecenia `mvdir`. Jego składnia jest następująca:

```
mvdir <katalog> <cel>.
```

Przenieśmy na przykład katalog `nowykat` z katalogu domowego do katalogu `/tmp`:

```
darkstar:~$ mvdir nowykat /tmp  
darkstar:~$ cd /tmp  
darkstar:/tmp$ ls  
nowykat/
```

`nowykat` jest teraz podkatalogiem katalogu `/tmp`.

Podczas przenoszenia katalogu przenoszone są również wszystkie jego podkatalogi.

Aby zmienić nazwę katalogu nie przenosząc go, można użyć polecenia `mv`. Przykładowo, aby zmienić nazwę katalogu `nowykat` na `starykat`, należy wydać polecenie:

```
mv nowykat starykat
```

Wszystkie pliki będące wcześniej w katalogu `nowykat` znajdują się teraz w katalogu `starykat`.

Usuwanie plików i katalogów

Teraz, kiedy potrafisz już tworzyć pliki i katalogi, przyszedł czas by dowiedzieć się, jak usunąć wyniki Twoich zabaw.

Do usuwania plików służy polecenie `rm <nazwa_pliku>`. Na przykład polecenie

```
darkstar:~$ rm zdechla_kaczka
```

usuwa plik `zdechla_kaczka` z katalogu domowego. Polecenie

```
darkstar:~$ rm /tmp/zdechla_kaczka
```

usuwa plik o tej samej nazwie z katalogu `/tmp`.

Z tym poleceniem również można używać symboli wieloznacznych, ale pamiętaj, że może to spowodować mnóstwo kłopotów, jeśli polecenie zostanie wydane w złym miejscu. Na przykład polecenie

```
darkstar:~$ rm *
```

usuwa bezpowrotnie wszystkie pliki z katalogu domowego. Nie ma żadnej możliwości odzyskania ich, jeśli więc polecenie takie wydałeś przez pomyłkę, masz pecha. Polecenie `rm` należy wydawać bardzo ostrożnie.

Można używać łącznie ścieżek dostępu i symboli wieloznacznych, np. polecenie

```
darkstar:~$ rm /tmp/*kaczka
```

usuwa z katalogu `/tmp` wszystkie pliki, które na końcu nazwy mają słowo `kaczka`.

Kiedy plik zostanie usunięty, przepada na zawsze! Powinieneś więc pomyśleć dwa razy przed usunięciem każdego pliku. Jeśli chcesz uniknąć problemów, możesz stosować się do którejś z poniższych wskazówek.

- v Uruchom najpierw polecenie `ls` z symbolem wieloznaczny, jakiego zamierzasz użyć do usunięcia plików:

```
darkstar:~$ ls *nic
drugie_nic           picnic           pierwsze_nic
```

Prawdopodobnie nie chodziło Ci o usunięcie pliku `picnic`, a stało by się tak, gdybyś nie sprawdził, co zostanie usunięte.

- v Zawsze używaj opcji `-i` polecenia `rm`. Powoduje ona, że zostaniesz poproszony o potwierdzenie usunięcia każdego pliku:

```
darkstar:~$ rm -i *nic
rm: remove `drugie_nic'? y
rm: remove `picnic'? n
rm: remove `pierwsze_nic'? y
```

Wpisanie `y` lub `Y` potwierdza usunięcie pliku, każdy inny znak traktowany jest jako odpowiedź przecząca. Niestety, metoda ta może być dość irytująca podczas usuwania większej ilości plików.

Usuwanie katalogów

Polecenie `rm` służy do usuwania plików. Jeśli spróbujesz za jego pomocą usunąć katalog, otrzymasz komunikat o błędzie. Do usuwania katalogów służy polecenie `rmdir <katalog>`. Zanim jednak będzie można usunąć dany katalog, należy usunąć z niego wszystkie pliki i podkatalogi. W przeciwnym przypadku otrzymasz komunikat:

```
rmdir: <katalog>: Directory not empty
```

Sytuacje takie jak ta mogą Cię nieco zdziwić:

```
darkstar:/home$ ls
burek/    reksio/    lopez/
darkstar:/home$ ls reksio
core      proba     obwl
```

```
darkstar:/home$ rm reksio/*
darkstar:/home$ ls reksio
darkstar:/home$ rmdir reksio
rmdir: reksio: Directory not empty
darkstar:/home$
```

Powodem wyświetlenia takiego komunikatu są pliki w katalogu `reksio`, których nazwa zaczyna się od kropki – czyli ukryte pliki systemowe. Nie są one normalnie wyświetlane przez polecenie `ls`, nie pasują też do wzorca `*`. Aby je zobaczyć, wydaj polecenie `ls -a`; polecenie `rm .*` pozwala je usunąć.

```
darkstar:/home$ ls -a reksio
./ ../.bashrc .profile
darkstar:/home$ rm reksio/.*
rm: cannot remove '.' or '..'
darkstar:/home$ ls -a reksio
./ ..
darkstar:/home$ rmdir reksio
darkstar:/home$ ls
burek/ lopez/
darkstar:/home$
```

Problemy tego typu mogą zdarzać się przy pracach administracyjnych.

Czasem zachodzi potrzeba usunięcia katalogu mającego wiele poziomów podkatalogów. Z pomocą przyjdzie wtedy opcja `r` (ang. *recursive*) polecenia `rm: rm -r <katalog>`. Dany katalog zostanie usunięty wraz ze wszystkimi podkatalogami i plikami w nim zapisanymi.



Opcji `-r` polecenia `rm` należy używać tylko w wyjątkowych przypadkach. Pomyłkowo użyta może ona mieć katastrofalne konsekwencje. Przykładowo, jeśli jesteś zalogowany jako `root` i wydasz polecenie `rm -r /`, to będziesz miał okazję do powtórzenia procedury instalacyjnej. Wierz lub nie, ale takie sytuacje zdarzają się zaskakująco często.

Kompresja

Większość plików na dyskach CD-ROM z dystrybucją Linuxa przechowywanych jest w postaci skompresowanej, co umożliwia zapisanie większej ilości informacji. Jeśli pracowałeś w systemie DOS lub Windows, widziałeś pewno programy pozwalające na spakowanie wielu plików do jednego archiwum, takie jak PKZIP czy WINZIP. Te same techniki stosowane są w programach UNIX-owych od lat, nieco tylko różnią się nazwami i metodami kompresji.

Podczas instalacji Linuxa ogromna ilość plików zostaje rozpakowana na dysku twardym. Mimo tego nadal można na nim znaleźć pliki skompresowane.

Każdy plik z rozszerzeniem `.gz`, na przykład `pak.gz`, jest plikiem skompresowanym. By rozpakować ten typ pliku, wydaj polecenie `gunzip <nazwa_pliku>`, na przykład

`gunzip pak.gz`. Polecenie to utworzy plik rozpakowany, nie posiadający już rozszerzenia `.gz`, czyli o nazwie `pak`. Aby ponownie skompresować plik, użyj polecenia `gzip`: `gzip pak`.

Innym rodzajem plików spakowanych są pliki z rozszerzeniem `.zip`. Można je rozpakować poleceniem `unzip <plik>`, a spakować poleceniem `zip <plik>`.

W systemie Linux dostępnych jest jeszcze kilka innych mechanizmów kompresji. Pliki kompresowane z zastosowaniem tych mechanizmów mają rozszerzenia `.z` oraz `.Z` (te dwa rozszerzenia nie są generowane przez ten sam program).

Ważne katalogi systemu Linux

Większość katalogów systemu Linux posiada standardowe nazwy. Inne systemy UNIX-owe zawierają takie same katalogi, a w nich podobne programy. Ten podrozdział omawia krótko kilka najważniejszych katalogów.

/

Jest to katalog główny. W nim i w jego podkatalogach znajduje się cały system plików. Nie zaśmiecaj go swoimi plikami!

/home

W tym katalogu znajdują się katalogi domowe użytkowników. W niektórych systemach UNIX-owych nazywa się on `/usr` lub `/u`.

/bin

Ten katalog zawiera wiele podstawowych programów linuxowych. Jego nazwa pochodzi od angielskiego słowa *binaries*, czyli pliki binarne, które mogą być wykonywane przez komputer.

/usr

W tym katalogu przechowywane są podkatalogi związane z użytkownikami; niektóre z nich opisane są w następnych podrozdziałach, z pozostałych najważniejsze są następujące katalogi:

- docs** zawiera dokumentację programów i inne przydatne informacje,
- man** strony `man`,

games różne zabawne programy.

/usr/bin

Katalog ten zawiera programy dostępne dla użytkowników.

/usr/spool

Ten katalog zawiera kilka podkatalogów: `mail` przechowuje pliki poczty, `spool` – pliki, które mają zostać wydrukowane, a `uucp` pliki, które mają zostać skopiowane do innego systemu UNIX-owego.

/dev

Linux traktuje wszystko, z czego można odczytywać lub zapisywać dane, jak pliki. Katalog `/dev` zawiera pliki urządzeń. Są to specjalne pliki, które obsługują fizyczne części komputera. Na przykład, kopując zawartość jakiegoś pliku do pliku `/dev/fd0`, faktycznie wysyłasz go do stacji dysków. Twój terminal to jeden z plików `/dev/tty`. Partyce dysku twardego to `/dev/hda0` itp. Nawet pamięć RAM jest urządzeniem i posiada odpowiedni plik.

Ciekawym urządzeniem jest `/dev/null`. Dane wysyłane do tego urządzenia wędrują do nikąd, są po prostu tracone.

/usr/sbin

W tym katalogu przechowywane są programy przeznaczone dla administratora.

/sbin

Tu przechowywane są programy uruchamiane automatycznie przez system Linux.

/etc

Ten katalog i jego podkatalogi zawierają systemowe pliki konfiguracyjne. Są one zazwyczaj plikami tekstowymi, mogą więc być edytowane przez administratora w celu zmiany konfiguracji systemu.

Podsumowanie

Teraz powinieneś już czuć się pewnie pracując z Linuxem. Zrozumienie i umiejętność poruszania się po systemie plików jest bardzo ważna; na szczęście prawie wszystkie systemy Linuxowe zorganizowane są w standardowy sposób.

Jeśli kiedyś natkniesz się na problem związany z plikami i katalogami, pamiętaj, że wszyscy masz pod ręką strony `man`. Linux jest bardzo elastyczny, jeśli chodzi o tworzenie plików, podawanie ścieżek dostępu do plików i katalogów oraz ustawianie praw dostępu. Nie bój się eksperymentować (jeśli jesteś zalogowany jako normalny użytkownik). Każda operacja może zostać wykonana na wiele sposobów – nie poprzestawaj na podanych na papierze receptach.

Powinieneś również przejrzeć rozdział 9. „Prawa dostępu do plików i katalogów”, rozdział 10. „Programy użytkowe projektu GNU”, a także rozdział 13. „`tcsh`”, szczególnie jeśli zamierzasz tworzyć programy czy makropolecenia na bazie poleceń systemowych, albo jeśli chcesz dowiedzieć się czegoś więcej o pomocnych w obsłudze Linuxa mechanizmach wbudowanych w interpreter poleceń. Rozdział 16. zawiera również wiele użytecznych informacji o edytowaniu plików tekstowych.

Kiedy już zapoznasz się z powłoką i nabierzesz wprawy w manipulowaniu plikami i katalogami, możesz przejść do bardziej zaawansowanych tematów w części trzeciej, „Edycja i skład” i części ósmej, „Programowanie dla zaawansowanych”.

Praca z edytorami tekstów dostarczonymi z Linusem omówiona jest w rozdziale 16. „Edytory tekstu: `vi` i `emacs`”.

O konfigurowaniu systemu X mówi rozdział 22. „Instalacja i konfiguracja `XFree86`”.

Jeśli chcesz dowiedzieć się czegoś więcej o administrowaniu systemem, zajrzyj do części szóstej, rozpoczynając od rozdziału 32. „Podstawy administracji systemem”.

Rozdział 9.

Prawa dostępu

do plików i katalogów

Tim Parker

W tym rozdziale:

- υ Posiadanie plików i katalogów
- υ Ustawienia zmiennej UMASK
- υ Modyfikowanie praw dostępu do plików
- υ Modyfikowanie praw dostępu do katalogów

Dla kogoś, kto dopiero zaczyna pracę z systemem Linux, prawa dostępu do plików i katalogów mogą wydawać się co najmniej niezrozumiałe. Wynika to głównie z faktu, że jest wokół nich mnóstwo niedopowiedzeń. Warto jednak poświęcić chwilę czasu i prześledzić dokładniej ten rozdział, ponieważ temat nie jest szczególnie trudny do zrozumienia.

Widziałeś już, że po wydaniu polecenia `ls -l` na ekranie pojawia się oprócz nazw plików spora ilość informacji. Spójrzmy na przykład:

```
-rwxr-xr-x    2      reksio  users   4512  May  9  09:20  ksiazka1  
-rwxr-xr-x    2      reksio  users   5727  May  9  09:22  ksiazka
```

Można w nim wyróżnić siedem kolumn zawierających różne dane. Są to, od lewej do prawej:

- υ prawa dostępu (wyjaśnimy je za chwilę),
- υ ilość dowiązań,
- υ identyfikator użytkownika posiadającego dany plik (w tym przypadku `reksio`),
- υ identyfikator grupy posiadającej dany plik (w tym przypadku `users`),

- υ rozmiar pliku w bajtach,
- υ data utworzenia pliku,
- υ nazwa pliku.

W tym rozdziale skoncentrujemy się na prawach dostępu, właściwemu oraz grupie posiadającej plik.

Każdy plik i katalog w systemie Linux ma swojego właściciela i grupę, która go posiada, oraz zestaw praw dostępu. Modyfikowanie tych atrybutów powoduje, że do plików ma dostęp więcej lub mniej osób. Prawa dostępu określają też, czy plik może zostać wykonany jako polecenie.

Posiadanie plików i katalogów

Domyślnie właścicielem pliku jest użytkownik, który go utworzył – jego identyfikator wyświetlanym jest w trzeciej kolumnie wyprowadzonej zawartości katalogu. Grupa, do której należał on w momencie tworzenia pliku, jest wyświetlana w czwartej kolumnie (o grupach jeszcze nie mówiliśmy; grupa to po prostu pewna liczba identyfikatorów użytkowników, którym nadano jedną „etykietkę”, by łatwiej można było kontrolować dostęp do plików i katalogów). Bycie właścicielem pliku pozwala zmieniać prawa dostępu do niego i identyfikatora właściciela. Jeśli zmienisz identyfikator właściciela na inny niż swój, najprawdopodobniej nie będziesz mógł już więcej zmieniać atrybutów pliku.

Użytkownicy i posiadanie

Identyfikatory właścicieli plików ustalane są podczas ich tworzenia. Pliki systemowe należą do użytkowników takich jak, `root`, `uucp` czy `bin`. Nie powinieneś tego zmieniać, nawet jeśli jesteś zalogowany jako `root`. Ustawienia te określane są w chwili instalacji systemu, a ich zmiana może uniemożliwić dostęp do plików.



Choć zmiana właściciela plików systemowych jest czasem kusząca, może to spowodować poważne problemy z programami, które uruchamiane są przez zwykłych użytkowników i potrzebują czytać z plików o zmienionych atrybutach. W większości przypadków programy takie blokują się lub kończą działanie. Zmieniaj identyfikator właściciela pliku tylko wtedy, gdy jesteś pewien, że są one zwykłymi plikami użytkowników.

Aby zmienić właściciela pliku, możesz użyć polecenia `chown` (ang. *change ownership*), którego składnia jest następująca:

```
chown <właściciel> <nazwa_pliku>
```

gdzie właściciel jest identyfikatorem nowego właściciela pliku. Możesz również używać symboli wieloznacznych, jak * czy ?, by zmieniać identyfikatory właścicieli całych grup plików.

W poniższym przykładzie zmieniony (na reksio) został właściciel pliku mojplik:

```
darkstar:~$ ls -l mojplik
-rw-r--r--    1      burek   users   114     Dec  8  14:31  mojplik
darkstar:~$ chown reksio mojplik
darkstar:~$ ls -l mojplik
-rw-r--r--    1      reksio   users   114     Dec  8  14:31  mojplik
```

Zanim nastąpi zmiana identyfikatora właściciela pliku, Linux sprawdzi, czy w systemie istnieje odpowiedni użytkownik. Po zmianie może się okazać, że nie masz odpowiednich praw dostępu by zrobić z plikiem cokolwiek innego, bądź więc ostrożny. Aby zmodyfikować plik z poprzedniego przykładu lub zmienić jego właściciela z powrotem na burek, musisz zalogować się jako reksio lub root (albo użyć polecenia su).



Choć Linux pokazuje identyfikator właściciela pliku, w rzeczywistości zapamiętuje on jego numer identyfikacyjny. Wynika to z faktu, iż z punktu widzenia systemu łatwiej jest operować na identyfikatorach liczbowych, niż nazwach użytkowników.



Czasem w systemie można znaleźć pliki, których właścicielami są nie istniejący użytkownicy. Ma to miejsce zwykle wtedy, gdy administrator usuwa konto użytkownika, a jego pliki pozostają w systemie (w chwili usunięcia konta Linux nie przegląda automatycznie struktury katalogów, nie jest zatem w stanie zmienić właściciela takich plików). W takim przypadku tylko użytkownik root może zmienić właściciela pliku.

Grupy

Pliki (oraz użytkownicy) należą do grup. Grupy są użyteczne głównie w większych systemach; może się zdarzyć, że nigdy nie będziesz potrzebował wykorzystywać możliwości wynikających z ich stosowania. Jest to jednak bardzo wygodny sposób zapewniania odpowiednich praw dostępu więcej niż jednemu użytkownikowi, ale nie wszystkim. Przykładowo, użytkownicy pracujący nad określonym projektem mogą należeć do grupy projekt. Pliki używane przez całą grupę będą również należeć do tej grupy, co daje tym użytkownikom określone prawa dostępu.

W skład grup wchodzą na ogół użytkownicy powiązani ze sobą określonymi relacjami. W powyższym przykładzie relację taką tworzy projekt opracowywany przez członków grupy; do jednej grupy mogą też należeć pracownicy danego wydziału, osoby wykonujące określone zadania (administratorzy, programiści itp.), lub też zespoły użytkowników w dużej sieci.

Przez cały czas jesteś członkiem jednej grupy. Gdy się logujesz, zostajesz przypisany do swojej grupy domyślnej, która jest ustalana przez administratora w chwili tworzenia Twojego konta. Możesz należeć do wielu różnych grup, ale w danym momencie możesz być zalogowany tylko do jednej z nich. Jeśli chcesz zmienić grupę, do której aktualnie jesteś przypisany, użyj polecenia `newgrp`. Na przykład, jeśli jesteś członkiem grup `users` oraz `programmers` i potrzebujesz zmienić grupę, do której jesteś aktualnie przypisany, na `programmers`, ponieważ daje Ci to uprawnienia do uruchamiania kompilatora, powinieneś wydać polecenie:

```
newgrp programmers
```

Linux nie umożliwia ustalenia wprost, do której grupy jesteś aktualnie przypisany. Możesz to zrobić, zapisując jakiś plik, a potem sprawdzając, do jakiej grupy on należy. Jeśli spróbujesz zmienić grupę na taką, do której nie należysz, Linux powie Ci tylko:

```
darkstar:~$ newgrp programmers
newgrp: Sorry
```

Zmiana przynależności do grupy

Wiesz już, jak zmienić właściciela pliku. Możesz też zmienić grupę, do której plik należy; aby to zrobić, nie musisz do niej należeć, ale musisz być właścicielem pliku. Przed wykonaniem tej operacji Linux sprawdza, czy grupa docelowa istnieje w systemie.

Polecenie służące do zmiany przynależności pliku do grupy to `chgrp`, a jego składnia jest podobna do składni polecenia `chown`:

```
chgrp <grupa> <nazwa_pliku>
```

Przykładowo, jeśli chcesz zmienić identyfikator grupy posiadającej plik `ksiazka` na `autorzy`, powinieneś wpisać:

```
$ ls -l ksiazka
-rwxr-xr-x    2      burek   users   4512   May  9  09:20  ksiazka
$ chgrp autorzy ksiazka
$ ls -l ksiazka
-rwxr-xr-x    2      burek   autorzy 4512   May  9  09:20  ksiazka
```

Można również, podobnie jak w przypadku polecenia `chown`, używać symboli wieloznacznych:

```
$ ls -l ksiazka*
-rwxr-xr-x    2      burek   users   4512   May  9  09:20  ksiazka1
-rwxr-xr-x    2      burek   users   4618   May  9  09:21  ksiazka2
-rwxr-xr-x    2      burek   users   4512   May 10  11:45  ksiazka3
$ chown reksio ksiazka*
$ ls -l ksiazka*
-rwxr-xr-x    2      reksio   users   4512   May  9  09:20  ksiazka1
-rwxr-xr-x    2      reksio   users   4618   May  9  09:21  ksiazka2
-rwxr-xr-x    2      reksio   users   4512   May 10  11:45  ksiazka3
$ chgrp autorzy ksiazka*
$ ls -l ksiazka*
-rwxr-xr-x    2      reksio   autorzy 4512   May  9  09:20  ksiazka1
-rwxr-xr-x    2      reksio   autorzy 4618   May  9  09:21  ksiazka2
-rwxr-xr-x    2      reksio   autorzy 4512   May 10  11:45  ksiazka3
```

Zauważ, że polecenia chgrp oraz chown nie zmieniają żadnych innych parametrów plików, takich jak data, czas ich utworzenia czy prawa dostępu.

Prawa dostępu

Od początku istnienia UNIX-a starano się, by prawa dostępu były wyrażone w miarę prosto, a jednocześnie zapewniały dużą elastyczność i bezpieczeństwo. Model zastosowany w systemie UNIX (a zatem i w Linuxie) jest stosunkowo prosty i zakłada istnienie trzech typów praw dostępu do plików i katalogów: praw do zapisu (*write*), do odczytu (*read*) i do wykonywania (*execute*).

Posiadanie prawa do odczytu pozwala oglądać zawartość pliku. W przypadku katalogu pozwala ono na wyświetlenie jego zawartości, np. polecienniem `ls`.

Prawo do zapisu pozwala modyfikować (i usuwać) pliki, nawet jeśli nie jesteś ich właścicielem. W przypadku katalogu musisz posiadać to prawo, by tworzyć, przenosić lub usuwać pliki w nim przechowywane.

Prawo do wykonywania pozwala na wykonanie programu znajdującego się w pliku (np. przez wpisanie jego nazwy). Nie ma ono zastosowania do plików, które nie mogą zostać zinterpretowane przez system operacyjny. Przykładowo, nawet jeśli plikowi z danymi statystycznymi nadasz atrybuty pozwalające Ci go wykonać, i tak nie będziesz mógł tego zrobić. Jeżeli natomiast plik zawierałby kod programu, uruchomienie tego ostatniego wymagałoby prawa do wykonywania. Jeśli posiadasz prawa wykonywania dla katalogu, oznacza to, że możesz do niego wejść polecienniem `cd`.

UNIX dzieli wszystkich użytkowników na trzy kategorie, w zależności od tego, czy są oni właścicielami pliku, czy nie, oraz czy znajdują się w grupie posiadającej plik. Każdemu plikowi i katalogowi przypisany jest osobny zestaw praw dostępu dla właściciela (ang. *owner*), osobny dla grupy, do której on należy (ang. *group*), i trzeci dla wszystkich pozostałych użytkowników systemu (ang. *other* lub *world*). Zestawy praw zapisywane są jedne po drugich w spójnej postaci; najpierw prawo do odczytu, potem zapisu, na końcu wykonywania, kolejno dla właściciela, grupy i innych użytkowników.

W sumie daje to dziewięć bitów informacji. W pierwszej kolumnie katalogu wyświetlonego polecienniem `ls -l` wyświetlanych jest jednak dziesięć wartości. Pierwsza z nich wskazuje, czy dana pozycja reprezentuje katalog czy plik (istnieje jeszcze kilka innych możliwych wartości, ale nie będziemy się tu nimi zajmować). Spójrzmy na konkretny przykład:

```
-rw-r--r--    1      burek   users  163  Dec 7 14:31  mojplik
```

Pierwszym znakiem w grupie praw dostępu jest `-`, co oznacza, że jest to zwykły plik. Dla katalogu w tym miejscu znalazłaby się litera `d`. Następne dziewięć znaków dzieli się na trzy grupy po trzy znaki i oznacza kolejno prawa dostępu dla właściciela, grupy i innych użytkowników. Każda trójka pokazuje uprawnienia do czytania, zapisu i wykonywania, dokładnie w takim porządku. Prawo do odczytu sygnalizowane jest literą `r`, do zapisu `w`, a do wykonywania `x`. Brak określonych uprawnień oznaczany jest symbolem `-`.

W naszym przykładzie właściciel pliku `mojplik` posiada prawo do jego odczytywania i zapisywania, co sygnalizowane jest znakami `rw-`. Plik nie może zostać natomiast wykonany przez wpisanie jego nazwy w wierszu poleceń, co jest konsekwencją braku prawa do wykonywania. Prawa grupy do pliku opisane są znakami `r--`, co oznacza, że członkowie grupy mogą odczytywać zawartość pliku, nie mogą natomiast go modyfikować ani wykonywać. Dokładnie takie same prawa (czyli zezwolenie wyłącznie na odczyt) mają pozostały użytkownicy.

Ustawienia zmiennej UMASK

Skąd Linux wie, jakie prawa dostępu powinien zastosować do nowo tworzonego (na przykład w wyniku użycia operatora przekierowania standardowego wyjścia) pliku? Odpowiedź zawiera się w zmiennej `UMASK` (ang. *user file creation mask*), która definiuje prawa dostępu dla każdego pliku, który tworzysz. Administrator systemu może ustawić tę zmienną indywidualnie dla każdego użytkownika, bądź globalnie, dla wszystkich użytkowników. Możesz zmieniać wartość własnej zmiennej `UMASK`, ale nie możesz zmieniać jej wartości dla innych użytkowników (chyba że jesteś zalogowany jako `root`).

Wartość zmiennej `UMASK` można poznać w każdej chwili, wydając polecenie `umask` (pisane małymi literami, w odróżnieniu od nazwy samej zmiennej):

```
$ umask
022
```

W odpowiedzi możesz też otrzymać cztery cyfry zamiast trzech, ale pierwsza w takim przypadku nic nie oznacza i należy ją zignorować. Co oznaczają pozostałe? Jest to zestaw cyfr ósemkowych zawierających informacje o domyślnych prawach dostępu do pliku. Zestaw dopuszczalnych wartości wraz z ich objaśnieniami przedstawia tabela 9.1.

Tabela 9.1. Wartości ósemkowe używane w zmiennej UMASK i ich znaczenie

Liczba ósemkowa	Prawa dostępu
0	Odczyt i zapis (oraz wykonywanie dla katalogów)
1	Odczyt i zapis
2	Odczyt (oraz wykonywanie dla katalogów)
3	Odczyt
4	Zapis (oraz wykonywanie dla katalogów)
5	Zapis
6	Wykonywanie (tylko dla katalogów, dla plików – brak praw dostępu)
7	Brak praw dostępu

Dla podanego wcześniej przykładu (022), właściciel pliku ma więc prawo do odczytu i zapisu (oraz wykonywania dla katalogu), natomiast grupa i inni użytkownicy – tylko

prawo do odczytu (i wykonywania dla katalogu). Odpowiada to następującym danym wyświetlanym przez polecenie `ls -l`:

```
-rw-r--r--
```

Jeśli zamiast pliku utworzony zostanie katalog, system dodatkowo przypisałby mu prawo do wykonywania (umożliwiające wejście do niego) dla właściciela, grupy i innych użytkowników, co daje następujący zestaw:

```
drwxr-xr-x
```

Zauważ, że nie ma sposobu, by automatycznie przypisywać plikowi prawa do wykonywania dla kogokolwiek. Rozwiążanie takie, przyjęte celowo, zmusza administratora systemu (jak również zwykłego użytkownika – *przyp. tłum.*) do „ręcznego” ustalania prawa do wykonywania.

Aby zmienić wartość zmiennej `UMASK`, podaj po poleceniu `umask` nowe wartości, których chcesz używać. Przykładowo, zestaw 077 odbiera wszystkie prawa dostępu dla grupy i innych użytkowników:

```
$ umask
0022
$ who > plik1
$ ls -l
total 2
-rw-r--r--    1      burek   users  37      May  9  11:18  plik1
$ umask 077
$ who > plik2
$ ls -l
total 4
-rw-r--r--    1      burek   users  37      May  9  11:18  plik1
-rw-----    1      burek   users  37      May  9  11:18  plik2
```

Ani grupie, ani innym użytkownikom systemu nie przyznano żadnych praw do pliku `plik2`. Dostęp do niego ma tylko właściciel. Zmienna `UMASK` zachowuje swoją wartość do momentu wylogowania się z systemu.

Modyfikowanie praw dostępu do plików

Wcześniej czy później nadchodzi chwila, w której trzeba zmienić prawa dostępu do plików, bądź to dodając prawo do wykonywania (co pozwoli uruchomić program), bądź poszerzając lub ograniczając prawa dostępu dla innych użytkowników systemu. Do zmiany praw dostępu przypisanych do pliku służy w systemie UNIX (i Linux) polecenie `chmod` (ang. *change mode*).

Składnia polecenia `chmod` jest następująca:

```
chmod <specyfikacja_praw> plik
```

Istnieją dwa sposoby podawania specyfikacji praw dostępu: bezpośredni, za pomocą wartości numerycznych (ang. *absolute setting*), oraz symboliczny, wykorzystujący

oznaczenia literowe (ang. *symbolic setting*). Drugi sposób jest łatwiejszy do zrozumienia, zaczniemy więc od niego.

Gdy używasz postaci symbolicznej, musisz najpierw podać, które z praw dostępu mają zostać zmienione (dostępne są cztery wartości: `u` – użytkownik, `g` – grupa, `o` – inni, `a` – wszyscy, a także ich kombinacje). Następnie należy wpisać znak `+` lub `-`, w zależności od tego, czy prawa dostępu mają zostać przyznane, czy odebrane. Jako ostatnie podajemy prawa, których polecenie ma dotyczyć (`r` – czytanie, `w` – zapis, `x` – wykonywanie). Ogólnie rzecz biorąc, składnia polecenia `chmod` w tej wersji jest następująca:

```
chmod [u|g|o|a][+|-][r|w|x] <nazwa_pliku> [<nazwa_pliku> ...]
```

Pomiędzy znakami symbolicznego opisu praw dostępu nie powinno być znaku spacji, natomiast musi on zostać wpisany po nazwie polecenia oraz przed nazwą pliku. Podajmy może kilka przykładów. Aby dodać prawo do wykonywania dla grupy i innych użytkowników, wpisz

```
chmod go+r plik
```

Żeby odebrać prawa zapisu i odczytu dla właściciela, grupy i innych, wydaj jedno z poniższych poleceń:

```
chmod ugo-rw plik
chmod a-rw plik
```

Pozostało nam kilka istotnych uwag. Po pierwsze, nie wszystkie systemy obsługują symbol `a` w znaczeniu „wszyscy użytkownicy”. Jeśli symbol `a` nie byłby rozpoznawany, w poprzednim przykładzie musiałbyś użyć pierwszego z podanych poleceń. Możesz jednocześnie ustalać prawa dostępu do wielu plików, albo przez podanie kolejno ich nazw (rozdzielonych spacjami) po poleceniu `chmod`, albo za pomocą symboli wieloznacznych (ang. *wildcards*). Warto także wiedzieć, że pierwotne ustawienia zmienianych praw dostępu nie mają znaczenia – polecenie `chmod` zastępuje stare prawa nowymi, ignorując poprzednie wartości. Jeśli jednak w poleceniu nie wyspecyfikowano którejś z klas użytkowników (tj. pominięto literę `u`, `g` lub `o`), dla pominiętych użytkowników zachowane zostaną przez domniemanie prawa obowiązujące wcześniej. Oto jeszcze jeden przykład:

```
$ ls -l
total 4
-rwxrwxrwx    1      burek    users   37      May  9  11:18  plik1
-rw-----    1      burek    users   37      May  9  11:18  plik2
$ chmod go-rw plik*
$ ls -l
total 4
-rwx--x--x    1      burek    users   37      May  9  11:18  plik1
-rw-----    1      burek    users   37      May  9  11:18  plik2
```

Efektem wykonania polecenia `chmod` dla pliku `plik1` było oczywiście odebranie praw do zapisu i odczytu dla grupy i pozostałych użytkowników. Ponieważ do pliku `plik2` nie były przypisane prawa odczytu ani zapisu dla grupy ani innych użytkowników, polecenie `chmod` nie zmieniło w ogóle jego atrybutów (jest to jak najbardziej prawidłowe).



Każdy, kto ma prawo odczytu z pliku, może również go skopiować. Właścicielem kopii jest osoba, która kopiowała plik. Może ona później modyfikować kopię pliku, zmieniać jej atrbuty itp.



Odebranie wszystkim prawa do zapisu nie zabezpiecza pliku przed jego usunięciem. Co prawda nie da się w takiej sytuacji usunąć pliku przez pomyłkę, ponieważ użytkownik zawsze zostanie zapytany o potwierdzenie, jednak wystarczy odpowiedzieć twierdząco, a plik zostanie usunięty.

Drugim sposobem modyfikacji praw dostępu jest podanie ich w formie bezpośredniej (numerycznej). W tej metodzie należy za pomocą liczb ósemkowych podać wprost, jakie uprawnienia mają właściciel, grupa oraz inni użytkownicy. Nie obejdzie się bez odrobiny perwersji – liczby te są dopełnieniem do ośmiu wartości używanych w zmiennej `UMASK`. Dostępne wartości to:

0 lub --- brak praw dostępu

1 lub **--x** wykonywanie

2 lub **-w-**zapis

3 lub **-wx** zapis i wykonywanie

4 lub **r--** odczyt

5 lub **r-x** odczyt i wykonywanie

6 lub **rw-** odczyt i zapis

7 lub **rwx** odczyt, zapis i wykonywanie

Musisz zdecydować, która z tych ośmiu liczb ma stosować się do właściciela, grupy i innych użytkowników. Aby np. ustalić dla pliku standardowy zestaw praw dostępu, pozwalający владельcy na zapis i odczyt, zaś grupie i pozostałym użytkownikom tylko na odczyt, należy użyć wartości 644. Oto kilka innych przykładów:

```
$ ls -l plik
-rw-r--r--    1      burek   users   114   Dec  7  14:31   plik
$ chmod 345 plik
$ ls -l plik
--wxr--r-x    1      burek   users   114   Dec  7  14:31   plik
$ chmod 701 plik
$ ls -l plik
-rwx-----x    1      burek   users   114   Dec  7  14:31   plik
```

Zaletą tej metody jest bezpośrednie (w przeciwieństwie do wzelnego) podawanie praw dostępu. Łatwiej jest też powiedzieć komuś „zmień prawa dostępu na 755” niż „zmień prawa dostępu na odczyt-zapis-wykonywanie, odczyt-wykonywanie, odczyt-wykonywanie”.

Główną wadą tej metody jest trudność w zapamiętaniu odpowiednich kombinacji liczb (chyba że często się ich używa). Ponadto, nawet jeśli zamierzasz zmienić tylko jedno z praw dostępu (np. dodać prawo do wykonywania dla właściciela czy prawo do zapisu dla grupy), trzeba ustalić i podać pełną wartość numeryczną opisującą wszystkie prawa. W takim przypadku metoda symboliczna jest o wiele prostsza.

Sposób użycia polecenia `chmod` zależy od doświadczenia. Jeśli będziesz używał Linuxa przez dłuższy czas, zauważysz, że częściej wykorzystujesz bezpośrednią metodę podawania praw dostępu, wracając do symbolicznej tylko przy wprowadzaniu prostych zmian.

Modyfikowanie praw dostępu do katalogów

Prawa dostępu do katalogów można modyfikować, używając polecenia `chmod`, dokładnie w ten sam sposób, jak w przypadku plików, ponieważ Linux zasadniczo nie rozróżnia katalogów i plików. Pamiętaj, że jeśli nie masz prawa wykonywania dla katalogu, nie możesz do niego wejść polecienniem `cd`, tak więc manipulacja prawem do wykonywania może mieć dla użytkowników poważne konsekwencje.



Każdy, kto ma prawo zapisu w katalogu, może usuwać z niego pliki, nawet jeśli nie jest ich właścicielem i nie ma do nich prawa zapisu. W związku z tym katalogi mają przeważnie atrybuty `drwxr-xr-x`. Oznacza to, że tylko właściciel katalogu ma prawo tworzyć i usuwać pliki. Przyznanie prawa zapisu w katalogu wszystkim użytkownikom jest bardzo niebezpieczne!

Atrybuty katalogów również można zmieniać z wykorzystaniem zapisu ósemkowego lub symbolicznego, co ilustruje następujący przykład:

```
$ mkdir katalog
$ ls -l
total 2
drwxr-xr-x    2      reksio  users   512      May  9  12:10  katalog
$ chmod go+w katalog
$ ls -l
total 2
drwxrwxrwx    2      reksio  users   512      May  9  12:10  katalog
$ chmod 755 katalog
$ ls -l
total 2
drwxr-xr-x    2      reksio  users   512      May  9  12:10  katalog
```



Jeśli znasz dobrze system dwójkowy, wygodnie będzie Ci myśleć o zestawie `rwx` jako o trzycyfrowej liczbie dwójkowej. Jeśli dane prawo nie jest przyznane, odpowiedni bit ma wartość 0 jeśli zostało nadane - bit ma wartość 1. Przykładowo interpretowany „dwójkowo” zapis `r-x` to 101, czyli 5 w systemie ósemkowym, podobnie zapis `--`

✗ to 001, czyli ósemkowo 1.

Podsumowanie

Teraz nie powinieneś mieć już kłopotów z modyfikacją praw dostępu do plików i katalogów. Jak wspomnieliśmy na początku rozdziału, jest to temat frustrujący większości użytkowników Linuxa. Mamy nadzieję, że czytając uważnie ten rozdział, zrozumiałeś idee stojące za opisanymi w nim poleceniami. Obecnie przejdziemy do kolejnych rozdziałów, w których omówimy bardziej szczegółowo zagadnienia związane z interpreterami poleceń.

Interpreter poleceń Bourne Again Shell omówiony jest w rozdziale 11. „bash”.

Polecenia powłoki umożliwiające rozszerzanie możliwości funkcjonalnych interfejsu użytkownika przedstawia rozdział 14. „Programowanie w języku powłoki”.

Edytory tekstu (pozwalające na pisanie programów) opisane są w rozdziale 16. „Edytory tekstu: vi i emacs”.

Rozdział 22. „Instalacja i konfiguracja XFree86” omawia interfejs graficzny X.

Rozdział 10.

Programy użytkowe projektu GNU

Peter MacKinnon

W tym rozdziale:

- υ Aktualnie dostępne oprogramowanie podlegające licencji GNU

Projekt GNU, założony i prowadzony przez Free Software Foundation (FSF), ma na celu zapewnienie darmowego oprogramowania (w formie kodu źródłowego) każdemu zainteresowanemu. Istnieje długi manifest, który wyjaśnia motyw takiej filantropii (za którą powinniśmy być naprawdę wdzięczni, ponieważ niektóre z tych programów są najlepsze w swojej klasie!). Jedną z głównych tez tego manifestu jest to, że wysokiej jakości oprogramowanie jest niezbędne do życia, podobnie jak jedzenie czy powietrze. Fakt, że oprogramowanie GNU jest rozprowadzane bezpłatnie, nie oznacza, że nie jest ono objęte prawami autorskimi. Głównym celem licencji GNU jest to, aby oprogramowanie pozostawało bezpłatne.

Jeśli chcesz dowiedzieć się czegoś więcej o FSF, możesz do nich napisać pod adres:

Free Software Foundation
675 Massachusetts Avenue
Cambridge, MA 02139.

Możesz również zamówić kopię ich biuletynu, wysyłając pocztą elektroniczną list do gnu@prep.ai.mit.edu.

Dystrybucja Linuxa dołączona do tej książki zawiera praktycznie wszystkie programy GNU dostępne w tej chwili. Są one spakowane za pomocą programu archiwizującego `tar`, a następnie kompresora `gzip` (również objętego licencją GNU). Kompresor ten jest nieco wydajniejszy od standardowo używanego w UNIX-ie programu `compress`. Pliki skompresowane programem `gzip` mają rozszerzenie `.gz`, pochodzące z programu `compress -z`. Program `gzip` radzi sobie z rozpakowywaniem obu typów plików.

W nazwie każdego ze skompresowanych plików zawarty jest numer wersji pakietu, by łatwo było zorientować się, która wersja jest „najświeższa”. Po rozpakowaniu program może zostać skompilowany i zainstalowany. Większość z nich posiada napisane w tym celu pliki `makefile`, obsługiwane przez program `make`.

Aktualnie dostępne oprogramowanie podlegające licencji GNU

Fundacja Free Software Foundation udostępniła do tej pory tak wiele oprogramowania, że nie sposób opisać każdą z aplikacji szczegółowo. Następne podrozdziały skrótnie opisują programy dołączone do tej wersji Linuxa. Są to krótkie podsumowania, oparte na informacjach pochodzących z GNU.

acm

`acm` jest sieciowym symulatorem walki powietrznej, przeznaczonym dla środowiska LAN. Pozwala rozprawić się z przeciwnikiem za pomocą pocisków powietrze-powietrze i kabini. Działa pod kontrolą systemu X.

Autoconf

`Autoconf` generuje skrypty powłoki, które pozwalają automatycznie konfigurować pakiet y z kodem źródłowym (takie jak dostarczane na licencji GNU) na podstawie pliku zawierającego dane o wymaganiach stawianych systemowi. Aby można było go używać, musi również być zainstalowany pakiet `m4`.

bash

Interpreter poleceń o nazwie `bash` jest rozszerzoną wersją interpretera Bourne Shell (stąd jego pełna nazwa – Bourne Again Shell). Oferuje on wiele rozszerzeń dostępnych w powłokach `csh` i `ksh`. Obsługuje także zarządzanie zadaniami, historię poleceń, oraz edycję wiersza poleceń w trybie edytorów `vi` i `emacs`. Dokładniejszy opis tego interpretera znajduje się w rozdziale 11. „bash”.

bc

`bc` to język programowania zorientowany na obliczenia algebraiczne. Może być używany w sposób interaktywny lub przy użyciu pliku wejściowego. Wersja GNU tego programu posiada składnię zbliżoną do języka C oraz kilka rozszerzeń w stosunku do pierwotnej wersji (nazwy zmiennych składające się z wielu znaków, słowo kluczowe `else` i

pełna obsługa algebra Boole'a). W przeciwnieństwie do wersji pierwotnej, nie wymaga programu `dc`, spełniającego funkcję kalkulatora.

BFD

Biblioteka BFD (ang. *Binary File Description*) pozwala programom operującym na plikach pośrednich `.o` (jak `ld` czy `gdb`) na obsługę różnych ich formatów. Zapewnia ona przenośny interfejs, tak że tylko BFD zna szczegóły danego formatu. Wszystkie programy używające BFD obsługują zarówno format `a.out` (domyślny format plików wykonywalnych generowanych przez kompilator C), jak i `COFF`.

Binutils

Binutils to kolekcja narzędzi dla programistów, takich jak `ar`, `c++filt`, `demangle`, `gprof`, `ld`, `objcopy`, `objdump`, `ranlib`, `size`, `strings` i `strip`. Wersja 2 jest napisana zupełnie od nowa i używa biblioteki BFD. Linker `ld` przy komunikatach o błędach dotyczących wielokrotnych definicji i niezdefiniowanych odniesień podaje również numery wierszy kodu źródłowego. Program `nmconv` pozwala na konwersję plików pośrednich `.o` do formatu NLM (Novell NetWare Loadable Modules). Program `objdump` potrafi wyświetlić dane takie jak nazwy użytych symboli, z pliku w dowolnym formacie rozpoznawanym przez BFD.

Bison

Bison jest kompatybilny ze swym pierwowzorem – programem `yacc`. Potrafi on na podstawie zdefiniowanej gramatyki stworzyć interpreter języka formalnego w postaci programu w języku C.

Kompilator języka C

Wersja 2 kompilatora GNU C (`gcc`) obsługuje trzy języki: C, C++ oraz C z obiektami. Wybór języka zależy od rozszerzenia nazwy pliku zawierającego kod źródłowy lub opcji kompilatora. Biblioteki wymagane do uruchamiania programów w języku C z obiektami są rozprowadzane razem z `gcc`. Kompilator ten obsługuje w pełni standard ANSI C, tradycyjny język C oraz kilka rozszerzeń, jak np. funkcje zagnieżdżone czy nielokalne wywołania `goto`. Potrafi generować pliki pośrednie i informacje dla debugera w wielu różnych formatach. Więcej na jego temat możesz dowiedzieć się z rozdziału 27. „Programowanie w języku C”.

Biblioteki dla języka C

Biblioteki GNU C obsługują standard ANSI C i dodają kilka własnych rozszerzeń. Przykładowo, biblioteka `stdio` pozwala zdefiniować nowe rodzaje strumieni używane z poleceniem `printf`.

Biblioteki dla języka C++

Biblioteki GNU C++ (`libg++`) to zbiór klas dla języka C++, nowa biblioteka `iostream` i narzędzia ułatwiające pracę z kompilatorem `g++`. Wśród klas można znaleźć na przykład implementacje liczb całkowitych o zwiększonej precyzyji, ułamków, liczb zespolonych, a takżełańcuchów znaków o zadanej długości. Istnieją też pliki zawierające prototypy, dzięki którym można generować często używane klasy kontenerów.

Calc

`Calc` to narzędzie matematyczne używane przez niektóre programy (jak np. `Emacs`). Może również służyć jako kalkulator, który ma takie zalety, jak wybór pomiędzy notacją algebraiczną lub odwrotną notacją polską (Reverse Polish Notation, RPN), funkcje logarytmiczne, trygonometryczne i finansowe, liczby zespolone, wektory, macierze, daty, czas, nieskończoności, zbiory, upraszczanie wyrażeń algebraicznych, różniczkowanie i całkowanie.

Chess

`Chess` to komputerowa wersja gry w szachy. Może być uruchomiona zarówno z interfejsem tekstowym (dzięki wykorzystaniu biblioteki `curses`), jak i graficznym (w systemie X). Zaimplementowano w niej wiele specjalistycznych algorytmów, również heurystycznych, co nie pozwoli wygrać nowicjuszowi.

CLISP

`CLISP` to implementacja języka Common Lisp, używanego powszechnie do programowania sztucznej inteligencji. Zawiera interpreter oraz kompilator, posiada również interfejs w dwóch wersjach językowych: angielskiej i niemieckiej. Wersję językową można wybrać w trakcie komplikacji programu.

Common Lisp

GNU Common Lisp (`gcl`) to kompilator i interpreter języka Common Lisp. Jest on w dużym stopniu przenośny, zadziwiająco wydajny i posiada debugger pracujący z in-

interpretowanym kodem. `gcl` zawiera również narzędzia ułatwiające pisanie programów dla X, takie jak profiler i interfejs `xlib`.

cpio

`cpio` to program, który kopiuje archiwa na dyski lub taśmy i z powrotem. Można go również użyć do kopiowania plików do większego archiwum lub do innych katalogów.

CVS

Concurrent Version System (`cvs`) służy do zarządzania wersjami kodu źródłowego po-wstającej aplikacji, gdy nad jej napisaniem pracuje wiele osób. Działa w połączeniu z programem RCS, który również służy do zarządzania kodem źródłowym.

dc

`dc` to kalkulator liczący w odwrotnej notacji polskiej. Może być używany interaktywnie lub za pomocą pliku wejściowego.

DejaGnu

`DejaGnu` to narzędzie służące do pisania skryptów testujących dowolny program. Zawiera język skryptowy `Tcl` i pochodny język `expect`, służący do uruchamiania skryptów sy-mulujących działania użytkownika.

Diffutils

Pakiet `Diffutils` zawiera programy służące do porównywania plików: `diff`, `diff3`, `sdiff` i `cmp`. Program `diff` rozprowadzany na licencji GNU porównuje pliki wiersz po wierszu, wyświetlając znalezione różnice w kilku formatach. Jest bardziej wydajny niż jego tradycyjna wersja.

ecc

`ecc` to program do korekcji błędów, oparty na algorytmie Reeda-Solomona. Potrafi skorygować trzy błędne bajty w bloku 255 – bajtowym, a wykryć znacznie więcej.

ed

`ed` to standardowy edytor tekstu, posiadający interfejs oparty na wierszu poleceń.

Elib

Jest to niewielka biblioteka funkcji w języku Lisp przeznaczona dla edytora `Emacs`, która potrafi między innymi obsługiwać dane o strukturze podwójnej listy.

GNU Emacs

GNU `Emacs` to druga implementacja bardzo popularnego edytora napisanego przez Richarda Stallmana. `Emacs` jest zintegrowany z językiem Lisp, mogącym służyć do rozszerzania możliwości tego programu. Posiada interfejs oparty na systemie X. Oprócz potężnego zestawu własnych poleceń, `Emacs` potrafi emulować inne edytory, takie jak `vi` czy `EDT`. Jeśli chcesz dowiedzieć się więcej o tym programie, zajrzyj do rozdziału 16. „Edytory tekstu: `vi` i `emacs`”.

GNU Emacs 19

`Emacs 19` to wzbogacona wersja edytora `Emacs`, zawierająca poprawioną obsługę interfejsu X. Zapewnia dostęp do menedżera zasobów, obsługę RCS, zawiera także wiele nowszych wersji bibliotek. `Emacs 19` pracuje zarówno pod kontrolą systemu X, jak i w trybie tekstowym.

es

`es` jest to interpreter poleceń oparty na `rc`. Obsługuje wyjątki oraz funkcje zwracające wartości inne niż liczby. Używa się go wygodnie zarówno w trybie interaktywnym, jak i w skryptach, głównie dlatego, że jego reguły są prostsze niż reguły języków wchodzących w skład powłok C Shell i Bourne Shell.

Fileutils

`Fileutils` to kolekcja standardowych (i nieco mniej standardowych) programów linuksowych służących do obsługi systemu plików. Zawiera m.in. programy `chgrp`, `chmod`, `chown`, `cp`, `dd`, `df`, `dir`, `du`, `install`, `ln`, `ls`, `mkdir`, `mkfifo`, `mknod`, `mv`, `mvdir`, `rm`, `rmdir`, `touch` i `vdir`.

find

Program `find` służy do wyszukiwania plików spełniających zadane kryteria i wykonywania na nich jakichś operacji; może być używany interaktywnie i za pomocą skryptów. Rozprowadzany jest wraz z programem `xargs`, który potrafi wykonać polecenie dla większej liczby plików.

finger

Program `finger` służy do wyświetlania informacji o jednym lub więcej użytkownikach Linuxa. Potrafi obsłużyć wiele komputerów połączonych w sieć, potrafi też przyjąć, że jeden z nich zawiera informacje o wszystkich użytkownikach. Dzięki temu zapytanie skierowane do dowolnego komputera zwraca pełną informację o użytkowniku. Oto przykładowy efekt wykonania polecenia `finger`:

```
# finger tparker@tpci.com
Login: tparker                               Name: Tim Parker
Directory: /usr/tparker                         Shell: /bin/sh
On since Sat Jun 6 11:33 on ttv02, idle 51 days 21:22 (messages off)
On since Sun Jun 7 15:42 on ttv00, idle 0:02
On since Sat Jun 6 11:33 on ttv01, idle 21 days 10:011
Mail last read Tue Jun 16 18:38:48 1998
```

Jak widać, ta wersja programu `finger` (wyniki jego działania różnią się w zależności od systemu operacyjnego i wersji) pokazuje informacje zawarte w pliku `/etc/passwd` (identyfikator, katalog domowy itd.) oraz informacje dotyczące ostatnich sesji.

flex

`flex` to zamiennik programu `lex`, który jest generatorem interpreterów języków formalnych. Wygenerowane przez program `flex` interpretery są bardziej wydajne, niż pochodzące z programu `lex`. `flex` potrafi również stworzyć odpowiedni kod w języku C.

Fontutils

`Fontutils` to pakiet używany do tworzenia czcionek dla programów `Ghostscript` i `TeX`. Zawiera również narzędzia do konwersji i kilka innych programów użytkowych. W jego skład wchodzą m.in. `bpltoobzr`, `bzrto`, `charspace`, `fontconvert`, `gsrenderfont`, `imageto`, `imgrotate`, `limn` oraz `xbfe`.

gas

`gas` to asembler, tłumaczający kod źródłowy do postaci pliku pośredniego `.o`. Programy napisane w całości w asemblerze działają w różnych systemach operacyjnych, nie tylko w Linuxie.

gawk

Jest to program kompatybilny z `awk`, modyfikujący pliki zgodnie z zadanym wzorcem. Zawiera również kilka rozszerzeń nie spotykanych w innych implementacjach, na przykład `awk` czy `nawk`, funkcje do zmiany wielkości liter w znalezionym tekście. Dokładniejsze informacje znajdziesz w rozdziale 25. „`gawk`”.

gdb

`gdb` to debugger obsługiwany z wiersza poleceń. Pliki pośrednie i tablice symboli są wczytywane za pośrednictwem biblioteki BFD, co umożliwia wyszukiwanie błędów w programach zapisanych w różnych formatach. Inne zalety `gdb` to praca zdalna poprzez połączenie szeregowe lub TCP/IP oraz obsługa pułapek (przerwanie programu po spełnieniu zadanego warunku). Dostępna jest również wersja z interfejsem X, o nazwie `xxgdb`.

gdbm

Jest to biblioteka GNU zastępująca tradycyjne biblioteki służące do obsługi baz danych – `dbm` oraz `ndbm`. Implementuje bazy danych z wyszukiwaniem za pomocą tablic hashujących.

Ghostscript

`Ghostscript` to język opisu grafiki kompatybilny z językiem PostScript. Pobiera on polecenia postscriptowe i wykonuje je, kierując wyniki na drukarkę, do okna w systemie X lub zapisując do pliku, który można wydrukować później (można też wyniki zapisać w postaci mapy bitowej, nadającej się do dalszej edycji, za pomocą dowolnego programu graficznego).

`Ghostscript` zawiera również bibliotekę graficzną, która może być używana z poziomu języka C. Pozwala to tworzyć programy dziedziczące wszystkie zalety PostScriptu bez konieczności znajomości tego języka. Więcej informacji o języku `Ghostscript` znajdziesz w rozdziale 24. „`Ghostscript`”.

Ghostview

`Ghostview` to przeglądarka wielostronnicowych plików postscriptowych, interpretowanych przez program `Ghostscript`. Pracuje w środowisku X.

gmp

`gmp` to biblioteka pozwalająca na wykonywanie obliczeń na liczbach całkowitych ze znakiem o zadanej precyzji oraz na użycie ułamków.

GNats

`GNats` (ang. *GNU's A Tracking System*) to system śledzenia i zgłaszania błędów. Oparty jest na modelu organizacji posiadającej węzeł centralny, do którego trafiają wszystkie zgłoszenia o problemach, a ich rozwiązania rozsyłane są pocztą elektroniczną. Pierwotnie

przewidziany do wyłapywania usterek w powstających programach, nadaje się również do administrowania siecią, zarządzania projektami i do innych zastosowań.

GNU Graphics

GNU Graphics to zestaw programów, które generują rysunki na podstawie plików binarnych lub ASCII. Dane wyjściowe mogą być przekazane do pliku postscriptowego lub okna X. Zawiera również pewną liczbę przykładowych skryptów w języku powłoki używających poleceń `graph` oraz `plot` i narzędzi do prezentacji danych statystycznych.

GNU Shogi

Shogi to japońska gra zbliżona do szachów, z tym że tracone figury mogą wrócić do gry. Jego implementacja oparta jest na **GNU Chess** i wykorzystuje te same algorytmy i techniki. Program umożliwia również wprowadzanie dodatkowych informacji określających sekwencje ruchów wykonywanych podczas otwarcia. Obsługuje zarówno interfejs tekstowy, jak i graficzny.

gnuplot

gnuplot to interaktywny program służący do rysowania wykresów funkcji matematycznych i danych statystycznych. Pozwala tworzyć grafikę dwu – i trójwymiarową.

GnuGo

Implementacja japońskiej gry Go.

gperf

gperf to program użytkowy służący do generowania „idealnych” tablic haszujących. Możliwe jest tworzenie funkcji haszujących zarówno w języku C, jak i C++.

grep

Pakiet ten zawiera programy `grep`, `egrep` i `fgrep`. Wersje GNU tych programów, służących do szukania w plikach tekstów spełniających zadane warunki, działają o wiele szybciej niż ich tradycyjne odpowiedniki.

Groff

Groff to system formatowania dokumentów. Zawiera sterowniki dla formatu PostScript i dvi (TeX). W jego skład wchodzą programy `eqn`, `nroff`, `pic`, `refer`, `tbl`, `troff` oraz makra `man`, `ms` i `mm`. Do komplikacji niezbędny jest kompilator GNU C++ w wersji 2.5 lub późniejszej.

gzip

Program do kompresji i dekompresji plików. Używa innych (wydajniejszych) algorytmów niż standardowy program `compress`, ale potrafi radzić sobie z plikami spakowanymi tym programem, jak również programem `pack`.

hp2xx

`hp2xx` to program, który wczytuje pliki w formacie HPGL, rozkłada wszystkie polecenia rysowania na elementarne wektory, po czym zapisuje je w różnych formatach wektorowych (PostScript, Metafont, formaty TeX i uproszczony HPGL) oraz rastrowych (PBM, PCX oraz HP-PCL).

indent

Program `indent` służy do formatowania kodu źródłowego napisanego w języku C zgodnie ze standardem GNU, ale potrafi też dostosować się do standardu K&R, BSD i innych. Można również zdefiniować własny standard. Rozpoznaje komentarze w stylu C++.

Ispell

Interaktywny korektor ortograficzny. Potrafi korzystać ze słowników systemowych oraz podanych przez użytkownika. Dostępny jest interfejs niezależny oraz zintegrowany z edytorem `Emacs`.

m4

Implementacja tradycyjnego procesora makr dla języka C. Zawiera kilka rozszerzeń, takich jak obsługa więcej niż dziewięciu parametrów makr, dodawanie plików, wykonywanie poleceń powłoki i obsługa arytmetyki.

make

`make` to wzbogacona wersja tradycyjnego programu do zarządzania powiązanymi ze sobą plikami. Wersja GNU obsługuje długie nazwy opcji, komplikację równoległą, elastyczne wzorce reguł, wykonanie warunkowe i potężny zestaw funkcji do manipulowania tekstem. W najnowszych wersjach udoskonalono raportowanie błędów i dodano obsługę popularnej składni `+=` dla dołączania tekstu do wartości zmiennej. Jeśli chcesz dowiedzieć się więcej na temat tego programu, przejdź do rozdziału 56. „Zarządzanie kodem źródłowym”.

mtools

`mtools` to zestaw programów (na licencji *public domain*), które pozwalają Linuxowi czytać, zapisywać i manipulować zbiorami w DOS-owym systemie plików (zazwyczaj na dysku).

MULE

Rozszerzenie `Emacs 18` (ang. *MULtilingual Enhancement*) służące do obsługi różnych języków. Potrafi wyświetlić wiele zestawów znaków równocześnie, włączając w to znaki japońskie, chińskie, koreańskie, wietnamskie, greckie, zestawy ISO Latin 1-5, ukraińskie i rosyjskie. Bufor tekstowy `MULE` może zawierać tekst składający się ze znaków we wszystkich tych językach. Do ich wprowadzania można używać różnych metod offeredanych przez `MULE`.

NetFax

`NetFax` to darmowy program do obsługi faksu dla sieciowych systemów linuxowych. Wymaga modemu obsługującego polecenia faksu w standardzie Class 2.

NetHack

Gra przygodowa, działająca w trybie ASCII lub graficznym.

NIH Class Library

Biblioteka przenośnych klas dla języka C++, podobnych do zawartych w `Smalltalk-80`, a opracowanych przez Keitha Gorlena pracującego w National Institutes of Health (NIH).

nvi

`nvi` to darmowa implementacja edytora `vi`. Zawiera kilka udoskonaleń, takich jak dzierzenie ekranu, użycie wielu buforów, możliwość obsługi znaków ośmiobitowych, plików o dowolnym rozmiarze i długości wiersza, cofanie polecień o dowolna liczbę poziomów oraz ulepszoną obsługę wyrażeń wieloznacznych.

Octave

`Octave` to język wysokiego poziomu przeznaczony głównie do obliczeń numerycznych. Udostępnia wygodny interfejs oparty na wierszu polecen, umożliwiający rozwiązywanie numeryczne problemów liniowych i nieliniowych.

Obsługuje arytmetykę liczb rzeczywistych, zespolonych, macierzy i wektorów, rozwiązuje układy równań nieliniowych, całkuje funkcje w przedziałach skończonych i nieskończonych, potrafi również całkować układy równań różniczkowych.

Oleo

`Oleo` to arkusz kalkulacyjny obsługujący interfejs X oraz interfejs tekstowy. Potrafi eksportować pliki do formatu PostScript i konfigurować klawiaturę podobnie jak system `Emacs`. Pracując pod kontrolą X i w PostScrpicie, obsługuje czcionki o zmiennej szerokości.

p2c

`p2c` to translator kodu napisanego w języku Pascal na C. Rozpoznaje wiele rodzajów Pascala, takich jak Turbo, HP, VAX i ISO i generuje działający (przeważnie) kod w języku C. Może zaoszczędzić mnóstwo pracy, gdy trzeba przetłumaczyć programy z Pascala na C.

patch

`patch` to program, który na podstawie pliku wygenerowanego przez program `diff` i starej wersji pliku potrafi wygenerować jego nowszą wersję. Używany głównie do prowadzenia drobnych poprawek w kodzie źródłowym.

PCL

Darmowa wersja dużego podzbioru systemu `CLOS` (Common Lisp Object System). Wymaga opisanego wcześniej programu `CLISP`.

perl

`perl` to język programowania opracowany przez Larryego Walla, który łączy w sobie zalety języków `sed`, `awk`, skryptów powłoki i języka C, posiada również możliwość wywoływanego funkcji systemowych i bibliotecznych języka C. Jest bardzo często używany do tworzenia rozbudowanych aplikacji nie obsługujących skomplikowanych struktur danych. `Emacs 19` udostępnia tryb edycji zorientowany na pracę z kodem źródłowym w języku `perl`.

ptx

Jest to wersja GNU programu do generowania indeksów. Obsługuje wiele plików wejściowych jednocześnie, potrafi produkować pliki w formacie kompatybilnym z TeX lub w formacie KWIC (ang. *KeyWords In Context*, słowa kluczowe w kontekście), które mogą być odczytane bez użycia żadnego programu formatującego.

rc

`rc` to interpreter poleceń, którego cechą charakterystyczną jest składnia bardzo przypominająca język C (nawet bardziej niż w przypadku interpretera `csh`). Może być używany interaktywnie i w skryptach.

RCS

System zarządzania wersjami kodu źródłowego (ang. *Revision Control System*). Używany łącznie z programem `diff` może obsługiwać pliki binarne, takie jak programy czy pliki pośrednie. Więcej informacji o nim znaleźć można w rozdziale 56.

recode

Program `recode` pozwala zmienić używany w dokumencie tekstowym zestaw znaków. Kiedy tłumaczenie dokładne jest niemożliwe, usuwa nieprawidłowe znaki lub zastępuje je podobnymi. Rozpoznaje ponad 150 różnych zestawów znaków i pozwala na konwersję pomiędzy prawie każdymi dwoma.

regex

Biblioteka służąca do obsługi symboli wieloznacznych, wykorzystywana w wielu programach. Teraz jest dostępna również w wersji samodzielnej. Jej szybsza wersja dołączana jest do edytora `sed`.

Scheme

Język programowania związany z LISP-em. Główną różnicą jest fakt, że Scheme potrafi przekazywać funkcje jako argumenty innych funkcji i zwracać funkcje jako rezultat wywołania funkcji. Funkcje mogą również być wartością wyrażenia, bez konieczności definiowania ich pod konkretną nazwą.

screen

screen to program, który potrafi uruchomić kilka terminali (`tty`) na jednym fizycznym terminalu tekstowym. Każdy z terminali wirtualnych emuluje VT100 z kilkoma dodatkowymi funkcjami; może zostać również uśpiony, a następnie przywrócony jako inny typ terminalu.

sed

sed to nieinteraktywna, zorientowana na pracę ze strumieniami danych wersja edytora ed. Jest często używana w skryptach powłoki i niezastąpiona w przypadku, gdy zachodzi potrzeba dokonania takich samych zmian w wielu plikach lub napisania programu konwertującego. Dołączona jest do niego biblioteka rx, będąca szybszą wersją biblioteki regex.

Shellutils

Shellutils to zestaw programów, które mogą być używane interaktywnie oraz w skryptach, takich jak: basename, date, dirname, echo, env, expr, false, groups, id, nice, nohup, printenv, printf, sleep, stty, su, tee, test, true, tty, uname, who, whoami oraz yes.

Smalltalk

Interpreter języka programowania zorientowanego obiektowo, napisany w C. Jest coraz popularniejszy wśród programistów, ponieważ uważany jest za język „czysto obiektowy”.

Zaletami wersji GNU są: możliwość zapisywania binarnego pliku obrazu, wywołania kodu napisanego w C i przekazania do niego parametrów, tryb edycji Emacs, wersja protokołu X, która może być wywołana z poziomu języka Smalltalk, oraz automatycznie ładowane osobiste pliki konfiguracyjne. Zaimplementowane są wszystkie klasy i protokoły Smalltalk-80, za wyjątkiem obsługi grafiki.

Superopt

`Superopt` to program do optymalizacji funkcji, który stosuje wielokrotne generowanie i testowanie, by znaleźć sekwencję instrukcji realizującą daną funkcję w najkrótszym czasie. Jego interfejs jest prosty: trzeba tylko podać programowi `gso` (GNU Super Optimizer) żądaną funkcję, procesor docelowy oraz maksymalną liczbę instrukcji, którą możesz zaakceptować.

tar

`tar` to program do archiwizacji plików, obsługujący zapis na różnych nośnikach, automatyczną kompresję i dekompresję archiwów, archiwizację w systemach zdalnych oraz mający inne zalety pozwalające używać go do robienia pełnych i różnicowych kopii zapasowych.

Biblioteka Termcap

Odpowiednik standardowej biblioteki `libtermcap.a`. Nie nakłada, w przeciwieństwie do innych implementacji, ograniczeń na rozmiar wpisów `Termcap`.

TeX

System formatowania dokumentów, który doskonale radzi sobie ze skomplikowanym składem, nie wyłączając równań matematycznych. Jest to standardowy system formatowania tekstu dla programów objętych licencją GNU.Więcej informacji na jego temat zawiera rozdział 19. „TeX i LaTeX”.

Texinfo

`Texinfo` to zestaw programów generujących podręczniki w wersji nadającej się do druku oraz w formacie z odnośnikami hipertekstowymi (nazywanym *Info*). Zawiera również programy do przeglądania dokumentacji zapisanej w formacie Info. Wersja 3 zawiera zarówno programy samodzielne, jak i zintegrowane z edytorem `Emacs`. Edytor `Emacs` posiada specjalny tryb służący do tworzenia plików w tym formacie. W skład zestawu wchodzą programy: `makeinfo`, `info`, `texi2dvi`, `texindex`, `tex2patch` i `fixfonts`.

Textutils

Tradycyjne programy do manipulowania plikami tekstowymi, takie jak: `cat`, `csum`, `comm`, `csplit`, `cut`, `expand`, `fold`, `head`, `join`, `nl`, `od`, `paste`, `pr`, `sort`, `split`, `sum`, `tac`, `tail`, `tr`, `unexpand`, `uniq`, oraz `wc`.

Tile Forth

Tile Forth to 32-bitowa implementacja standardu Forth-83, napisana w języku C. Tradycyjne wersje pisane są w asemblerze, co powoduje, że wykorzystują sprzęt w najwyższym stopniu, ale niestety zmniejsza ich przenośność.

time

Program `time` używany jest do zbierania informacji statystycznych o liczbie użytkowników, systemie oraz rzeczywistym czasie procesora zajmowanym przez procesy.

tput

`tput` to przenośny sposób na wykorzystanie specjalnych poleceń terminali w skryptach powłoki. Wersja GNU używa bazy danych `Termcap`, zamiast – jak robią to inne wersje – `Terminfo`.

UUCP

`UUCP` służy do kopiowania plików pomiędzy komputerami UNIX-owymi. Obsługuje wiele protokołów: `f`, `g`, `v` (wszystkie okna i wielkości pakietów), `G`, `t`, `e`, `Zmodem`, oraz nowe protokole dwukierunkowe (`i` oraz `j`). Jeśli posiadasz bibliotekę Berkley Sockets, pozwala nawiązywać połączenia TCP. Za pomocą biblioteki TLI może również obsługiwać połączenia TLI.

uuencode/uudecode

Programy `uuencode` i `uudecode` używane są do kodowania danych binarnych, które mają zostać przesłane poprzez sieci obsługujące tylko podstawowe znaki ASCII (najczęściej w grupach dyskusyjnych).

wdiff

`wdiff` to jeszcze jeden interfejs dla programu `diff`. Porównuje dwa pliki, wskazując, które słowa zostały usunięte lub dodane do pierwszego tak, by powstał drugi. Obsługuje wiele formatów wyjściowych i współpracuje z takimi programami, jak `more`. `wdiff` jest szczególnie przydatny, gdy teksty różnią się tylko kilkoma słowami i układem akapitów.

Podsumowanie

Projekt GNU dostarcza oprogramowania UNIX-owego każdemu, kto jest nim zainteresowany, pod warunkiem że pozostanie ono darmowe. Oprogramowanie to może być skompilowane w wielu systemach operacyjnych, także w Linuxie. Wiele z programów to ulepszone wersje programów standardowych, takich jak kompilatory, interpretery poleceń czy debuggery. Istnieją również gry, edytory tekstu, programy kalkulacyjne i komunikacyjne objęte licencją GNU. Każdy z nich może zostać osobno zdekompresowany i skompilowany.

Edytory tekstów, które mogą służyć do tworzenia własnych dokumentów, opisane są szczegółowo w rozdziale 16. „Edytory tekstu: vi i emacs”.

O drukowaniu dokumentów możesz przeczytać w rozdziale 20. „Drukowanie”.

Programowanie w języku `awk`, przeznaczonym głównie do zarządzania plikami, opisane jest w rozdziale 25. „gawk”.

Rozdział 11.

bash

Rick McMullin i Tim Parker

W tym rozdziale:

- υ Po co komu interpreter poleceń
- υ Powłoka Bourne Again Shell
- υ Dostosowywanie interpretera `bash`
- υ Polecenia powłoki `bash` – podsumowanie

Ten i dwa następne rozdziały pozwolą Ci poznać różne typy interpreterów poleceń dostępnych w systemie Linux: `bash`, `pdksh` i `tcsh`. Zaczniemy od omówienia powłoki `bash` (Bourne Again Shell), ponieważ jest ona w systemie Linux powłoką domyślną, w związku z czym korzysta z niej większość nowych użytkowników. Przyjrzymy się najczęściej wykorzystywanym poleceniom i zmiennym środowiskowym powłoki `bash`. Po przeczytaniu tego rozdziału będziesz potrafił pracować z tym interpreterem szybciej i wydajniej.

Po co komu interpreter poleceń

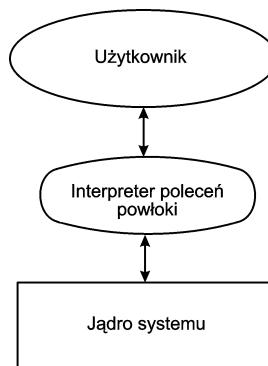
Czym tak właściwie jest interpreter poleceń powłoki? Jest to termin używany w systemie Linux bardzo często, ale jego znaczenie jest często mgliste dla początkujących (a czasem również bardziej zaawansowanych) użytkowników. Ten podrozdział wyjaśni Ci, co to takiego i dlaczego jest taki ważny.

Co to jest interpreter poleceń powłoki?

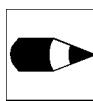
Interpreter poleceń powłoki to interfejs pomiędzy Tobą (czyli użytkownikiem) a Linuxem (albo bardziej precyzyjnie: jądrem systemu operacyjnego). Rysunek 11.1 obrazuje połączenia pomiędzy użytkownikiem, interpreterem poleceń powłoki (zwany również

Rysunek 11.1.

Powiązania pomiędzy użytkownikiem, interpreterem poleceń i systemem operacyjnym



krócej powłoką lub interpreterem poleceń) i systemem operacyjnym. Każde polecenie, jakie wydajesz, używając Linuxa, jest interpretowane przez powłokę, a następnie przekazywane do jądra systemu.



Jeśli znasz system MS-DOS, dostrzegasz na pewno podobieństwo tych relacji do relacji pomiędzy użytkownikiem systemu DOS i programem COMMAND.COM. Jedyną różnicą jest fakt, że w systemie DOS interpreter poleceń jest scalony z jądrem systemu.

Interpreter poleceń obsługuje pewną liczbę poleceń wewnętrznych (ang. *built-in*). Umożliwia również uruchomienie wszystkich aplikacji i programów użytkowych dostępnych w systemie.

Wprowadzone przez użytkownika polecenie zawsze jest interpretowane przez powłokę. Na przykład w poprzednich rozdziałach, gdy wydawałeś polecenia, ucząc się zarządzać plikami i katalogami, powłoka interpretowała je, a następnie wykonywała albo uruchamiała odpowiednie programy.

Niektóre z poleceń (np. `pwd`) są wewnętrznymi poleceniami interpretera, inne to programy, które znajdują w katalogach gdzieś w systemie plików (na przykład `rm` lub `cp`). Z punktu widzenia użytkownika rozróżnienie takie nie jest istotne. Rysunek 11.2 pokazuje, w jaki sposób powłoka interpretuje polecenia.

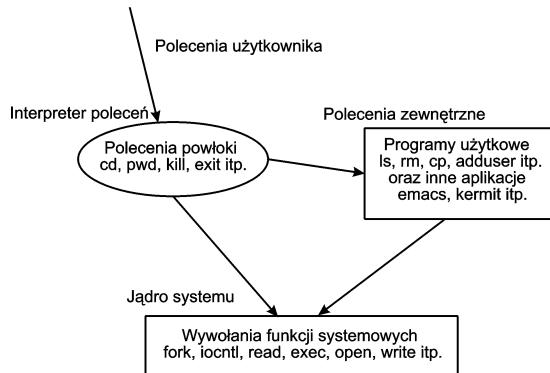
Rysunek 11.2 ilustruje kroki podejmowane przez powłokę w celu zinterpretowania polecień wydawanych przez użytkownika. Najpierw sprawdza ona, czy polecenie nie jest jednym z poleceń wewnętrznych (jak np. `pwd`). Jeśli nie, próbuje znaleźć i uruchomić odpowiedni program. Może to być zarówno standardowe polecenie Linuxa, jak `ls` czy `rm`, lub też jakaś aplikacja komercyjna, na przykład `xv`.

Interpreter poleceń szuka programów w katalogach zawartych w ścieżce przeszukiwania. Jak wspomnieliśmy w jednym z wcześniejszych rozdziałów, ścieżka przeszukiwania zawiera listę katalogów, w których można znaleźć pliki wykonywalne. Jeśli jednak przeszukiwanie nie zakończy się sukcesem (i oczywiście polecenie nie jest jednym z wewnętrznych poleceń powłoki), wyświetlony zostanie komunikat:

```
$ zrobto
zrobto: not found
```

Rysunek 11.2.

Interpretacja poleceń przez powłokę



Ostatnim etapem wykonania polecenia (jeśli może ono być zinterpretowane) jest rozbicie go na wywołania funkcji systemowych, przekazywane do jądra systemu.

Powłoka zawiera również potężny, interpretowany język programowania, zbliżony nieco do języka znanego z DOS-owych plików wsadowych, ale o wiele bardziej elastyczny. Obsługuje on większość konstrukcji, które dostępne są w językach wysokiego poziomu, takich jak pętle, funkcje, zmienne czy tablice.

Jak uruchamia się interpreter poleceń

Wiesz już, że interpreter poleceń jest podstawową metodą porozumiewania się użytkownika i jądra systemu. Jak jednak uruchamiany jest ten program? W systemie Linux jest on uruchamiany w momencie, gdy użytkownik zaloguje się do systemu. Zazwyczaj interpreter poleceń pozostaje uruchomiony aż do wylogowania.

Każdemu użytkownikowi przypisany jest domyślny interpreter poleceń. Informacja o nim znajduje się w pliku zawierającym hasła, czyli `/etc/passwd`. Zawiera on (miedzy innymi) identyfikator użytkownika, zakodowaną wersję hasła i nazwę programu, który należy uruchomić po zalogowaniu. Nie musi to być (choć prawie zawsze jest) interpreter poleceń Linuxa.

Najczęściej używane powłoki

W systemach linuxowych i UNIX-owych dostępnych jest kilka rodzajów powłok. Najpopularniejsze z nich to Bourne Shell (`sh`), C Shell (`csh`) i Korn Shell (`ksh`). Każda ma swoje wady i zalety.

Powłoka `sh`, której autorem jest Steven Bourne, jest podstawowym interpreterem poleceń w systemach UNIX-owych i jest dostępna w każdym z nich. Ma dość duże możliwości jeśli chodzi o obsługę skryptów, ale praca interaktywna nie jest jej najmocniejszą stroną.

Powłoka C Shell (`csh`), której autorem jest Billy Joy, jest o wiele wygodniejsza pod względem pracy interaktywnej (posiada na przykład takie ułatwienia jak automatyczne

dokańczanie poleceń). Jeśli chodzi o programowanie, nie jest tak dobra jak `sh`, ale wielu programistów używa jej ze względu na fakt, że składnia poleceń jest zbliżona do składni języka C (stąd zresztą pochodzi jej nazwa).

Interpreter `ksh`, którego autorem jest Dave Korn, łączy w sobie zalety obu poprzednich powłok – jest wygodny zarówno jeśli chodzi o pracę interaktywną, jak i programowanie. Ponadto jest całkowicie kompatybilny z powłoką `sh`.



Na rynku dostępnych jest wiele dobrych książek o trzech wymienionych wyżej interpreterach poleceń. Jeśli chcesz ich używać, powinieneś zaopatrzyć się w któryś z nich. Ponieważ jednak większość użytkowników poprzestaje na powłokach rozprowadzanych z Linusem, w dalszej części skupimy się właśnie na nich.

Poza wyżej wymienionymi, powstało wiele innych powłok, które dziedziczą podstawowe własności po jednym lub więcej interpreterze, tworząc jego nową wersję. Z większością dystrybucji Linuxa rozprowadzane są powłoki `csh` (rozszerzona wersja `sh`), `bash` (Bourne Again Shell, rozszerzona wersja powłoki `sh`) i `pdksh` (Public Domain Korn Shell, wersja powłoki `ksh`), które zostaną omówione bliżej w tym i dwóch następnych rozdziałach.

Powłoka Bourne Again Shell

Jak sama nazwa wskazuje, Bourne Again Shell to wersja powłoki Bourne Shell (`sh`). `bash` jest z nią całkowicie kompatybilny, ale zawiera wiele rozszerzeń i nowych rozwiązań. Przejął również niektóre zalety powłok `csh` i `ksh`. Jest to interpreter potężny i elastyczny jeśli chodzi o programowanie, a także przyjazny dla użytkownika pod względem pracy interaktywnej.

Dlaczego warto używać powłoki `bash` zamiast `sh`? Głównym powodem jest poprawiona obsługa interfejsu użytkownika. Wydawanie poleceń w interpreterze `sh` nie jest zbyt wygodne, jeśli więc regularnie i często wydajesz dużą liczbę poleceń, `bash` ma kilka cech mogących znacznie ułatwić Ci pracę.

Dokańczanie poleceń

Podczas wydawania polecenia jego pełne brzmienie często nie jest konieczne do zrozumienia i prawidłowego zinterpretowania go. Dla przykładu założmy, że katalog bieżący zawiera następujące podkatalogi:

```
News/ bin/ games/ mail/ test/
```

Jeśli chcesz przejść do podkatalogu `test`, możesz wydać polecenie

```
cd test
```

Oczywiście działa ono prawidłowo. `bash` pozwala jednak zrobić to nieco szybciej i wygodniej. Ponieważ `test` to jedyny podkatalog zaczynający się na literę `t`, `bash` powinien domyślić się, że właśnie o ten katalog chodzi, gdy wpiszesz:

```
cd t
```

Jednym podkatalogiem, do którego może odnosić się litera `t`, jest `test`. Jeśli chcesz, by `bash` dokończył polecenie za Ciebie, wciśnij klawisz *Tab*:

```
cd t<Tab>
```

Gdy to zrobisz, dokończone polecenie pojawi się na ekranie. Nie zostanie ono wykonane, dopóki nie zatwierdzisz go klawiszem *Enter*, dzięki czemu można sprawdzić, czy chodziło właściwie o to, co dopisał interpreter.

Przy wydawaniu krótkich poleceń (tak jak w powyższym przykładzie) korzystanie z tej możliwości nie ma większego sensu – może nawet wydłużyć czas ich wydawania. Dopiero gdy nabierzesz wprawy, a wydawane przez Ciebie polecenia staną się dłuższe, będziesz się zastanawiał, jak mogłeś wcześniej obejść się bez dokończania poleceń.

Co się jednak stanie, gdy od litery `t` zaczynać się będzie nazwa więcej niż jednego podkatalogu? Stworzy to pewien problem, gdy będziesz chciał użyć dokończania poleceń. Zobaczmy, jak potoczą się wypadki, gdy w katalogu bieżącym znajdują się następujące podkatalogi:

```
News/ bin/ mail/ test/ tools/ working/
```

W katalogu bieżącym znajdują się teraz dwa podkatalogi o nazwach zaczynających się na literę `t`. Jak używając dokończania poleceń przejść do podkatalogu `test`? Jeśli wpiszesz jak poprzednio `cd t<Tab>`, `bash` nie będzie potrafił rozstrzygnąć, o który katalog chodzi.

W takim przypadku interpreter wygeneruje krótki dźwięk, który ma oznajmić, że potrzebne są dokładniejsze informacje. Wiersz poleceń nie zostanie zmieniony, dzięki czemu nie musisz ponownie wpisywać polecenia. Powinieneś teraz dopisać literę `e` i ponownie wcisnąć klawisz *Tab*. Powłoka potrafi już określić, który z katalogów miałeś na myśli. Jeśli chcesz wejść do podkatalogu `tools`, zamiast litery `e` powinieneś oczywiście dopisać `o`:

```
cd to<tab>
```

Gdy podczas wpisywania polecenia wciśnięty zostanie klawisz *Tab*, `bash` próbuje dokończyć polecenie. Jeśli nie jest to możliwe, dopisywane są znaki co do których nie ma wątpliwości, a następnie generowany jest dźwięk oznaczający konieczność podania dodatkowych informacji. Można następnie wpisać dodatkowe litery i ponownie wcisnąć klawisz *Tab*, powtarzając ten proces aż do chwili, gdy powłoka będzie potrafiła dokończyć polecenie.

Symbole wieloznaczne

Innym ułatwieniem dla użytkownika jest możliwość stosowania w wierszu poleceń symboli wieloznacznych. `bash` obsługuje trzy ich rodzaje:

- * zastępuje dowolny znak lub ciąg znaków,
- ? zastępuje dowolny pojedynczy znak,
- [...] zastępuje dowolny znak zawarty pomiędzy nawiasami.

Znak * może być używany w sposób podobny do dokańczania poleceń. Założmy na przykład, że w katalogu bieżącym znajdują się następujące podkatalogi:

```
News/ bin/ mail/ test/ working/
```

Jeśli chcesz przejść do katalogu `test`, możesz wydać polecenie `cd test`, użyć mechanizmu dokańczania poleceń:

```
cd t<tab>
```

lub użyć symbolu *:

```
cd t*
```

Symbol * zastępuje dowolny znak lub ciąg znaków, więc interpreter w miejscu `t*` podstawi nazwę `test` (jest to jedyna nazwa podkatalogu w katalogu bieżącym, która pasuje do wzorca).

Metoda ta działa prawidłowo wtedy, gdy tylko jedna nazwa podkatalogu zaczyna się na literę `t`. Jeśli jest ich więcej, wzorzec `t*` zostanie zastąpiony przez listę plików i podkatalogów rozpoczynających się na literę `t`. Polecenie `cd` spowoduje przejście do pierwszego katalogu z tej listy, niekoniecznie do tego, o który chodziło.

Bardziej praktyczne zastosowanie znaku * to wykonywanie poleceń na kilku plikach jednocześnie. Przykładowo, niech bieżący katalog zawiera następujące pliki:

```
ch1.doc ch2.doc ch3.doc chimp config mail/ test/ tools/
```

Jeśli chcesz wydrukować wszystkie pliki z rozszerzeniem `.doc`, możesz to łatwo zrobić wydając polecenie

```
lpr *.doc
```

W tym przypadku `bash` zastąpi wzorzec `*.doc` listą wszystkich plików z rozszerzeniem `.doc` znajdujących się w katalogu bieżącym. Ostatecznie polecenie to jest więc równoważne następującemu:

```
lpr ch1.doc ch2.doc ch3.doc
```



Zakładając, że zawartość bieżącego katalogu jest taka, jak w poprzednim przykładzie, pliki z rozszerzeniem .doc można również wydrukować każdym z poniższych poleceń:

```
lpr *doc  
lpr *oc  
lpr *c
```

Symbol ? działa podobnie jak symbol *, z tym że zastępuje dokładnie jeden znak. Jeśli zawartość katalogu bieżącego jest taka sama jak w poprzednim przykładzie, pliki ch1.doc ch2.doc ch3.doc można wydrukować poprzez wydawanie polecenia

```
lpr ch?.doc
```

Symbol [...] pozwala bardziej szczegółowo określić o jakie znaki lub zakresy znaków chodzi. Aby – tak jak w poprzednim przykładzie – wydrukować pliki ch1.doc ch2.doc ch3.doc, można wydać polecenie

```
lpr ch[123].doc
```

Inna możliwość to podanie zakresu znaków, np.:

```
lpr ch[1-3].doc
```

Przywoływanie wydanych poleceń

Interpreter bash zapamiętuje pewną liczbę wydanych wcześniej poleceń. Ich ilość określona jest wartością zmiennej środowiskowej HISTSIZE. Jeśli chcesz dowiedzieć się więcej o zmiennych tego typu, zajrzyj do podrozdziału „Zmienne powłoki bash”.

Kiedy logujesz się do systemu, lista wydanych poleceń ładowana jest z pliku, nazywanego plikiem historii – domyślnie jest to plik o nazwie .bash_history, znajdujący się w katalogu domowym. Jego nazwa przechowywana jest w zmiennej HISTFILE i może zostać zmieniona. Zauważ, że rozpoczyna się ona od kropki, czyli jest nazwą pliku ukrytego, wyświetlana podczas przeglądania zawartości katalogu tylko na wyraźne żądanie, np. po wydaniu polecenia ls -a.

Samo zapamiętywanie wydanych poprzednio poleceń nie jest zbyt przydatne, więc bash udostępnia kilka sposobów przywoływania ich. Najłatwiej użyć w tym celu klawiszy kurSORA – strzałka w góRĘ przywołuje wcześniejsze polecenie, natomiast do późniejszego wrócić można, naciskając klawisz ze strzałką w dól (podobnie jak w DOS-owym programie DOSKEY).

Jeśli zachodzi taka potrzeba, można edytować przywołane z listy polecenie. bash posiada dość złożony zestaw funkcji służących do tego celu, jednak omawianie ich wykracza poza zakres tej książki. Przy wprowadzaniu niewielkich poprawek można z powodzeniem stosować najprostsze metody, na przykład użyć klawiszy kurSORA – strzałki w lewo i w prawo pozwalają poruszać się w obrębie jednego polecenia. Tekst można wstawiać

w dowolnym miejscu polecenia i usuwać klawiszami *Delete* czy *Backspace*. Większość użytkowników uważa takie możliwości za wystarczające.



Zestaw poleceń edycyjnych dostępnych w powłoce `bash` jest bardzo podobny do używanego w edytorech `vi` i `emacs`.

Używając poleceń `history` i `fc`, można również przeglądać i edytować plik historii. Polecenie `history` może zostać wywołane na dwa sposoby. Składnia pierwszego z nich jest następująca:

```
history [n]
```

Jeśli nie zostały podane żadne dodatkowe opcje, wyświetlana jest cała zawartość pliku historii, np.:

```
1 mkdir /usr/games/pool
2 cp XpoolTable-1.2.linux.tar.z /usr/games/pool
3 cd /usr/games/pool
4 ls
5 gunzip XpoolTable-1.2.linux.tar.z
6 tar -xf XpoolTable-1.2.linux.tar
7 ls
8 cd Xpool
9 ls
10 xinit
11 exit
12 which zip
13 zip
14 more readme
15 vi readme
16 exit
```

Użycie parametru `n` powoduje wyświetlenie tylko `n` ostatnich wierszy tego pliku, np. wydanie polecenia `history 5` pozwala obejrzeć pięć ostatnio wydanych poleceń.

Drugi sposób wywołania tego polecenia używany jest do modyfikowania zawartości pliku historii. Polecenie `history` ma wtedy składnię:

```
history [-r|w|a|n] [nazwa_pliku]
```

Opcja `-r` powoduje, że plik o danej nazwie zostanie odczytany i będzie traktowany jako bieżący plik historii. Opcja `-w` powoduje zapisanie listy wydanych poleceń do pliku historii (usuwając jego dotychczasową zawartość). Opcja `-a` dopisuje wydane ostatnio polecenia do istniejącego pliku. Opcja `-n` powoduje, że wiersze pliku historii są wczytywane do listy ostatnio wydanych poleceń.

Wraz ze wszystkimi powyższymi opcjami można również podać nazwę pliku, który będzie używany jako plik historii. Jeśli nazwa ta nie jest podana, przyjmowana jest nazwa przechowywana w zmiennej `HISTFILE`.

Polecenie `fc` może również być używane do edycji listy wydanych poleceń na dwa sposoby. Pierwszy z nich ma następującą składnię:

```
fc [-e nazwa_edytora] [-n] [-l] [-r] [początek] [koniec]
```

Opcja `-e nazwa_edytora` pozwala podać, jaki edytor ma zostać użyty do edytowania poleceń. początek i koniec pozwalają wybrać zakres poleceń, które mają zostać pobrane z pliku historii. Mogą to być numery poleceń, bądź łańcuchy znaków, które polecenie `fc` będzie próbowało odnaleźć w pliku historii poleceń.

Opcja `-n` wyłącza numerowanie wierszy z poleceniami. Opcja `-r` wyświetla polecenia w odwrotnym porządku. Opcja `-l` przesyła znalezione polecenia na ekran. We wszystkich przypadkach oprócz opcji `-l`, wybrane polecenia ładowane są do edytora tekstu.



Edytor tekstu uruchamiany jest na podstawie nazwy podanej w opcji `-e nazwa_edytora`. Jeśli nie została ona podana, używany jest edytor wskazywany przez zmienną `FCEDIT`. Jeśli taka zmienna nie istnieje, używana jest wartość zmiennej `EDITOR`, a w końcu, jeśli i taka zmienna nie została zdefiniowana, uruchamiany jest program `vi`.

Aliases

Aliases to skróty dłuższych, często używanych poleceń, służące wygodzie użytkownika.

Jeśli na przykład regularnie wydajesz następujące polecenie:

```
cd /usr/X11/lib/X11/fvwm/sample-configs
```

wygodnie byłoby zastąpić je poleciem krótszym, łatwiejszym do wpisania. Zamiast wpisywać za każdym razem wersję pierwotną, można stworzyć alias o nazwie np. `goconfig`, po wywołaniu którego nastąpi wykonanie pierwotnego, dłuższego polecenia. Aby utworzyć taki alias, trzeba skorzystać z polecenia `alias` powłoki `bash`:

```
alias goconfig='cd /usr/X11/lib/X11/fvwm/sample-configs'
```

Od tego momentu aż do zamknięcia interpretera `bash` polecenie `goconfig` spowoduje wykonanie pierwotnego, o wiele dłuższego polecenia.

Jeśli uznasz, że dany alias nie jest już potrzebny, można usunąć go za pomocą polecenia `unalias`:

```
unalias goconfig
```

Jest kilka aliasów, które przydają się wielu użytkownikom. Warto zapisać je w pliku, który jest uruchamiany za każdym razem, gdy logujesz się do systemu. Oto przykłady takich aliasów:

alias ll='ls -l' pozwala uzyskać pełny opis zawartości katalogu za pomocą polecenia `ll`,

alias lo='logout' pozwala nieco szybciej się wylogować,

alias ls='ls -F' powoduje, że po wpisaniu polecenia `ls` wykona się `ls -F`, czyli przy nazwach katalogów wyświetlane będą symbole `/`.

Jeśli jesteś użytkownikiem systemu DOS, aliasy emulujące niektóre polecenia tego systemu mogą ułatwić Ci pracę:

- ⦿ alias dir='ls'
- ⦿ alias copy='cp'
- ⦿ alias rename='mv'
- ⦿ alias md='mkdir'
- ⦿ alias rd='rmdir'



W poleceniu definiującym aliasy po żadnej ze stron znaku równości nie powinno być spacji, ponieważ powłoka nie zinterpretuje wtedy polecenia prawidłowo. Cudzysłów konieczny jest tylko w przypadku, gdy polecenie zastępowane aliasem zawiera spacje lub inne znaki specjalne.

Polecenie `alias` wydane bez żadnych parametrów powoduje wyświetlenie informacji o wszystkich zdefiniowanych aktualnie aliasach. Oto przykładowy wynik działania takiego polecenia:

```
alias dir='ls'
alias copy='cp'
alias rename='mv'
alias md='mkdir'
alias net='term < /dev/modem > /dev/modem 2 > /dev/null &
alias rd='rmdir'
```

Przekierowanie wejścia

Przekierowanie (lub przeadresowanie, ang. *redirection*) wejścia pozwala na zmianę źródła danych wykorzystywanych przez polecenie. Polecenia wydawane w interpreterze bash zwykle potrzebują jakichś informacji wejściowych, na przykład polecenie `rm` wymaga podania nazwy pliku czy plików, które mają zostać usunięte. Jeśli informacje te nie zostaną podane, polecenie `rm` wygeneruje komunikat o błędzie albo wskazówkę o sposobie uzyskania dodatkowych informacji o jego składni.

Inne polecenia wymagają o wiele bardziej złożonych danych wejściowych. Dane takie zwykle umieszcza się w pliku. Przykładowo, polecenie `wc` (ang. *word count*, licz słowa) zlicza znaki, słowa i wiersze tekstu, który jest podany na jego wejście. Jeśli polecenie `wc` wydane zostanie bez żadnych argumentów, program będzie czekał na wprowadzenie tekstu, w którym zostaną zliczone słowa. Wygląda to tak, jakby interpreter przestał

działać: co prawda wszystko, co wpisujesz, pojawia się na ekranie, ale polecenia nie są wykonywane. Możesz przerwać wpisywanie danych wciskając *Control+D*. Wtedy wyniki działania programu `wc` zostaną wyświetcone na ekranie. Jeśli jako parametr programu `wc` podana zostanie nazwa pliku, zostanie on potraktowany jako plik wejściowy, np.:

```
wc test  
11 2 1
```

Inną metodą podania zawartości pliku `test` na wejście programu `wc` jest przekierowanie wejścia, czyli poinformowanie interpretera polecen, że dane mają zostać odczytane nie z konsoli, ale z pliku `test`. Służy do tego operator `<`. W naszym przypadku operacja ta wyglądałaby tak:

```
wc < test  
11 2 1
```

Przekierowanie wejścia wykorzystywane jest dość rzadko, ponieważ programy pobierające dane z pliku wejściowego zazwyczaj posiadają opcję pozwalającą podać jego nazwę w wierszu polecen. Możesz jednak znaleźć się w sytuacji, gdy będziesz musiał przekaźać zawarte w pliku dane do polecenia nie posiadającego takiej możliwości. W takiej sytuacji przeadresowanie wejścia wybawi Cię z kłopotu.

Przekierowanie wyjścia

Przekierowanie wyjścia używane jest o wiele częściej niż przekierowanie wejścia. Pozwala ono przesyłać wyniki działania programu do pliku (lub urządzenia – *przyp. tłum.*), zamiast wyświetlać je na ekranie terminalu.

Możliwość przeadresowania wyjścia przydaje się w wielu sytuacjach. Może się np. zdarzyć, że wyniki wykonania polecenia nie mieszczą się na ekranie, wygodnie jest wówczas zapisać je do pliku i przeglądać później za pomocą edytora tekstu. Czasem również zachodzi potrzeba zachowania wyników działania programu, by je komuś pokazać lub wydrukować. Mechanizm ten bywa przydatny również wtedy, gdy wyniki działania jednego polecenia mają być później wykorzystane jako dane wejściowe innego programu (wyjście jednego polecenia na wejście drugiego można podać również używając mechanizmu potoków – ang. *pipes* – omówionego w następnym podrozdziale).

Przekierowania wyjścia można dokonać bardzo podobnie jak wejścia, tyle że zamiast symbolu `<` należy użyć znaku `>`.



Najłatwiej zapamiętać, który symbol służy do przekierowania wejścia, a który wyjścia, wyobrażając sobie lejek (symbole `<` i `>` przypominają go nieco kształtem), którego wąska końcówka wprowadza dane do polecenia lub pliku. Szersza końcówka lejka – obojętnie czy w symbolu `< czy >` – zawsze wskazuje na miejsce, skąd dane mają być pobierane.

Aby np. zapisać zawartość katalogu wyświetlana poleciem `ls` w pliku `katalog`, należy więc wydać polecenie:

```
ls > katalog
```

Ciągi poleceń

Ciągi poleceń (nazywane też potokami, ang. *pipes*) umożliwiają połączenie kilku poleceń tak, by wyjście poprzedniego stanowiło wejście następnego. Wyniki wykonania ostatniego z poleceń wyświetlane są na ekranie (lub przesyłane do pliku, jeśli zostały przekierowane).

Ciągi poleceń tworzy się oddzielając poszczególne polecenia symbolem `|`, np.:

```
cat przyklad | grep "Koniecznie" | wc -l
```

Wyjście polecenia `cat` (wyświetlającego zawartość pliku) zostanie skierowane na wejście programu `grep` (który wyświetli wszystkie wiersze zawierające tekst `Koniecznie`), a jego wyjście na wejście programu `wc` (opcja `-l` powoduje, że zliczane są tylko wiersze tekstu). Wyjście tego ostatniego zostanie wyświetcone na ekranie.

Pokażmy może wynik działania takiego ciągu poleceń na rzeczywistym pliku. Niech w pliku `plan_dnia` znajduje się następujący tekst:

```
Plan na dzisiaj:  
Kupić kasety  
Odebrać zdjęcia  
Koniecznie oddać film  
Koniecznie zapłacić za prąd
```

Wynikiem działania powyższego ciągu poleceń jest liczba 2, która oznacza, że są dwie rzeczy, które musisz dziś koniecznie zrobić:

```
cat plan_dnia | grep "Koniecznie" | wc -l  
2
```

Znak zachęty

Znak zachęty to tekst, który jest wyświetlany przez powłokę, by zakomunikować użytkownikowi, że oczekuje ona na wprowadzenie danych. W powłoce `bash` istnieją dwa poziomy znaków zachęty. Znak zachęty pierwszego poziomu sygnalizuje, że interpreter oczekuje na wydanie polecenia; zazwyczaj jest to znak `$` lub `%`. Jest on przechowywany jako wartość zmiennej `PS1`. Zmieniając wartość tej zmiennej możesz sprawić, że wyświetlany będzie dowolny tekst, na przykład

```
PS1="Wprowadz polecenie"
```

Znak zachęty drugiego poziomu wyświetlany jest, gdy `bash` oczekuje na podanie dodatkowych informacji niezbędnych do wykonania polecenia. Domyślnie wyświetlany jest

w takiej sytuacji symbol >. Postać tego znaku zachęty określona jest wartością zmiennej PS2 i można ją zmienić na przykład w taki sposób:

```
PS2="Podaj więcej danych"
```

Poza wyświetlaniem tekstów statycznych, istnieje również możliwość użycia kilku symboli specjalnych, które powodują wyświetlenie informacji takich jak bieżący katalog czy aktualny czas. Tabela 11.1 zawiera najczęściej używane spośród tych symboli.

Tabela 11.2. Kody specjalne używane przy wyświetlaniu znaku zachęty

Symbol	Znaczenie
\!	Wyświetla kolejny numer polecenia w historii poleceń
\#	Wyświetla numer aktualnego polecenia
\\$	Wyświetla znak \$, jeśli użytkownik nie jest zalogowany jako root, w przeciwnym wypadku wyświetla #
\ \	Wyświetla znak \
\d	Wyświetla aktualną datę
\h	Wyświetla nazwę komputera
\n	Przechodzi do następnego wiersza
\nnn	Wyświetla znak, który odpowiada ósemkowej wartości nnn.
\s	Wyświetla nazwę powłoki
\t	Wyświetla aktualny czas
\u	Wyświetla identyfikator użytkownika
\w	Wyświetla podstawową nazwę bieżącego katalogu
\w	Wyświetla nazwę bieżącego katalogu

Powysze symbole mogą być łączone, co pozwala na tworzenie znaków zachęty zawierających różne przydatne informacje. Poniżej podajemy przykładowe wartości zmiennej PS1.

```
PS1="\t"  
Tej wartości zmiennej PS1 odpowiada znak zachęty o następującej postaci  
(nie ma po nim spacji):  
02:16:15
```

Wpisanie

```
PS1=\t  
powoduje, że jako znak zachęty wyświetlany będzie tekst  
t
```

Widać więc, jak ważne jest stosowanie cudzysłówów. Natomiast zmienna PS1 o wartości

```
PS1="\t\\ \"  
daje następujący znak zachęty:  
02:16:30\
```

W tym przypadku po ostatnim znaku jest spacja, ponieważ znalazła się ona na końcu tekstu będącego wartością zmiennej `PS1`.

Zarządzanie zadaniami

Zarządzanie zadaniami to zdolność kontrolowania zachowań aktualnie działających procesów. Mówiąc konkretniej, jest to możliwość uśpienia (wstrzymania, ang. *suspend*) działającego procesu i późniejszego uruchomienia go. Interpreta bash śledzi wszystkie uruchamiane procesy (uruchamiane przez użytkownika, nie przez inne procesy) i pozwala je wstrzymywać oraz przywracać w dowolnej chwili (więcej informacji na ten temat znajdziesz w rozdziale 34. „Procesy”).

Wciśnięcie klawiszy *Control+Z* powoduje wstrzymanie działającego procesu. Polecenie `bg` uruchamia go ponownie w tle, natomiast polecenie `fg` – na pierwszym planie. Polecenia te są najczęściej używane w przypadku, gdy użytkownik przez pomyłkę uruchomi na pierwszym planie program, który miał zostać uruchomiony w tle. Program uruchomiony na pierwszym planie blokuje dostęp do interpretera poleceń i nie pozwala użytkownikowi na zrobienie czegokolwiek aż do momentu zakończenia swojego działania. Zazwyczaj nie jest to problemem, ponieważ większość poleceń potrzebuje na zakończenie działania co najwyżej kilku sekund, ale w przypadku programów, które działają dłucho, wygodniej jest uruchomić je w tle, by móc nadal używać interpretera.

Jeśli na przykład uruchomisz polecenie `find / -name "test" >find.out` (które przeszukuje cały system plików w poszukiwaniu pliku o nazwie `test`, wysyłając wyniki do pliku `find.out`) na pierwszym planie, na wyniki będziesz musiał poczekać od kilkunastu sekund do kilku minut, zależnie od wielkości systemu plików. W takim przypadku można wcisnąć klawisze *Control+Z*, co spowoduje wstrzymanie programu `find`, a następnie wydać polecenie `bg`, które uruchomi go ponownie w tle, dzięki czemu można będzie znów używać interpretera poleceń.

Dostosowywanie interpretera bash

W tym rozdziale opisano już kilka sposobów dostosowywania powłoki do własnych potrzeb i przyzwyczajeń. Jednak wprowadzone usprawnienia działały tylko do momentu wylogowania się z systemu. Można uczynić je bardziej trwałymi poprzez zapisanie odpowiednich poleceń w pliku inicjalizacyjnym.

Do pliku inicjalizacyjnego powłoki `bash` można wpisać dowolne polecenia, które mają być wykonywane przy każdym uruchomieniu powłoki, na przykład zestaw poleceń `alias` czy polecenia nadające odpowiednie wartości zmiennym środowiskowym.

Plik inicjalizacyjny nazywa się `.profile` lub `.bash_profile`, zależnie od wersji Linuxa. Dla uproszczenia przyjmijmy, że nazywa się on `.profile`. Każdy, kto używa interpretera `bash`, posiada taki plik w swoim katalogu domowym. Jest on odczytywany podczas uruchamiania interpretera i wykonywane są wówczas wszystkie zawarte w nim polecenia.

Wiele systemów linuxowych używa domyślnego pliku inicjalizacyjnego `profile` (zauważ, że przed jego nazwą nie ma kropki). Jest on zapisany w katalogu `/etc`. Jeśli chcesz dostosować interpreter `bash` do swoich potrzeb, powinieneś skopiować ten plik do swojego katalogu domowego, zmienić jego nazwę na `.profile` i wprowadzić odpowiednie modyfikacje.



Niektóre wersje programu instalacyjnego robią automatycznie kopię pliku `.profile` w katalogu domowym podczas zakładania konta dla nowego użytkownika. Nie jest to jednak reguła, więc powinieneś to sprawdzić, pamiętając o tym, że pliki o nazwach zaczynających się od kropki domyślnie nie są wyświetlane. Aby je zobaczyć musisz użyć polecenia `ls -a` lub `ls -A`.

Polecenia powłoki bash - podsumowanie

Niektóre z bardziej przydatnych polecen powłoki `bash` zabrano w tabeli 11.2

Tabela 11.2. *Niektóre polecenia powłoki bash*

Polecenie	Funkcja
<code>alias</code>	Umożliwia tworzenie aliasów (skróconych nazw polecień)
<code>bg</code>	Powoduje, że wstrzymany proces kontynuuje działanie w tle
<code>cd</code>	Przechodzi do katalogu, którego nazwa podana jest jako argument
<code>exit</code>	Kończy działanie powłoki
<code>export</code>	Powoduje, że wartość zmiennej będzie dostępna dla wszystkich procesów potomnych
<code>fc</code>	Pozwala na edycję historii polecień
<code>fg</code>	Powoduje, że wstrzymany proces kontynuuje wykonywanie na pierwszym planie
<code>help</code>	Wyświetla informacje o wbudowanych w interpreter poleceniach
<code>history</code>	Wypisuje listę ostatnio używanych polecień
<code>kill</code>	Służy do zatrzymywania procesów
<code>pwd</code>	Wyświetla nazwę katalogu bieżącego
<code>unalias</code>	Usuwa alias

Interpreter `bash` posiada jeszcze wiele innych polecień wewnętrznych, ale w tabeli znajdują się tylko te najczęściej używane. Informacje o pozostałych poleceniach znajdziesz na stronach `man` (które możesz obejrzeć po wydaniu polecenia `man bash`).

Zmienne powłoki bash

Konfiguracja powłoki `bash` określona jest przez wartości wielu zmiennych konfiguracyjnych. Najbardziej przydatne z nich wraz z krótkim opisem zebrane zostały w tabeli 11.3.

Tabela 11.3. Najbardziej przydatne zmienne powłoki bash

Nazwa zmiennej	Znaczenie
EDITOR, FCEDIT	Domyślny edytor polecenia <code>fc</code>
HISTFILE	Nazwa pliku historii poleceń
HISTSIZE	Rozmiar historii poleceń
HOME	Nazwa katalogu domowego bieżącego użytkownika
OLDPWD	Poprzedni katalog bieżący
PATH	Ścieżka przeszukiwania
PS1	Znak zachęty pierwszego poziomu
PS2	Znak zachęty drugiego poziomu (wyświetlany, gdy konieczne jest podanie dodatkowych informacji)
PWD	Nazwa bieżącego katalogu
SECONDS	Liczba sekund, które upłyнуły od uruchomienia powłoki <code>bash</code>

Zmiennych używanych przez powłokę `bash` jest znacznie więcej, niż tu podaliśmy. Jak zwykle, dokładniejsze informacje znaleźć możesz na stronach `man`.

Podsumowanie

W tym rozdziale przedstawiliśmy kilka użytecznych cech interpretera poleceń `bash`. Wiesz już, jak działa dokończanie poleceń, aliasy i zarządzanie zadaniami, co pozwoli Ci usprawnić pracę z tą powłoką.

W kolejnym rozdziale przyjrzymy się innemu interpreterowi poleceń dostępnemu w systemach linuxowych. Będzie to Public Domain Korn Shell (`pksh`). On również posiada wiele udogodnień, dzięki czemu masz w zasadzie wolną rękę przy wyborze powłoki, z którą chcesz pracować. Jeśli chcesz, możesz ominąć następny rozdział i przejść do innych tematów.

Więcej o programowaniu w języku powłoki możesz dowiedzieć się z rozdziału 14. „Programowanie w języku powłoki”.

Edytory tekstu, których użyć można do tworzenia i edytowania plików, omówione są w rozdziale 16. „Edytory tekstu: vi i emacs”.

Rozdział 12.

pdksh

Rick McMullin i Tim Parker

W tym rozdziale:

- υ Powłoka `pdksh` (Public Domain Korn Shell)
- υ Znaki zachęty
- υ Zarządzanie zadaniami
- υ Skróty klawiaturowe
- υ Dostosowywanie powłoki `pdksh`
- υ Polecenia wewnętrzne powłoki `pdksh`
- υ Zmienne powłoki `pdksh`

W poprzednim rozdziale omówiliśmy szczegóły pracy z powłoką `bash`. Ponieważ jednak nie każdy musi chcieć używać akurat tego interpretera, z Linuxem rozprowadzanych jest jeszcze kilka innych. Jednym z nich jest `pdksh`, odmiana powłoki Korn Shell (`ksh`). Omówimy tu w skrócie metody dostosowywania go do swoich potrzeb, najważniejsze polecenia i zmienne.

Powłoka pdksh (Public Domain Korn Shell)

Interpreter Korn Shell, którego autorem jest David Korn, jest trzecią z powłok używanych najczęściej w systemach UNIX-owych. Został napisany później niż powłoki `sh` i `csh`, dzięki czemu łączy w sobie ich zalety. Nie jest jednak rozprowadzany razem z systemami UNIX-owymi, więc wiele osób po prostu nie miało okazji go poznać. Implementuje udogodnienia wprowadzone przez powłokę `csh`, a jednocześnie pozostaje całkowicie

kompatybilny z powłoką `sh`. Wersja interpretera Korn Shell przeznaczona dla Linuxa jest dostępna za darmo i nazywa się właśnie `pdksh`.

Dokańczanie poleceń

Podczas wydawania polecenia jego pełne brzmienie często nie jest konieczne do zrozumienia i prawidłowego zinterpretowania go. Dokańczanie poleceń pozwala na wpisanie tylko części polecenia, a następnie, po wciśnięciu odpowiedniej kombinacji klawiszy, interpreter próbuje dokończyć polecenie za Ciebie.

Domyślnie `pdksh` nie pozwala na używanie tego udogodnienia. Trzeba zażądać tego *explicitamente*, wydając jedno z polecień:

```
set -o emacs  
set -o vi
```

Powodują one, że `pdksh` będzie reagował na kombinacje klawiszy tak, jak robią to edytory `emacs` lub `vi`. Wybierz ten edytor, z którym jesteś lepiej obeznany.



Większość ludzi uważa, że edytor `emacs` jest łatwiejszy w obsłudze niż `vi`.

W trybie `vi`, zanim użyjesz jakiegokolwiek polecenia edycyjnego, musisz wejść w tryb wydawania poleceń. Można to zrobić poprzez naciśnięcie klawisza *Escape*. Nie można wtedy jednak dopisywać do wiersza poleczeń żadnego tekstu, najpierw należy przejść do trybu edycji (jest na to kilka sposobów, można na przykład wcisnąć klawisz `i` – ang. *insert* – wstaw).

Kiedy dokańczanie poleceń jest uaktywnione, można go użyć wciskając dwukrotnie klawisz *Escape* (w trybie `emacs`) lub klawisz `\` (w trybie `vi`). Przykładowo, jeśli w bieżącym katalogu znajdują się następujące podkatalogi i pliki:

```
News/ bin/ games/ mail/ sample.text test/
```

a chcesz edytować plik `sample.text` za pomocą edytora `vi`, możesz wydać polecenie
`vi sample.text`

Jednak już po wpisaniu pierwszej litery nazwy pliku nie ma wątpliwości, o który plik chodzi. Aby `pdksh` dokończył tę nazwę za Ciebie, wciśnij dwa razy klawisz *Escape*, jeśli używasz trybu `emacs`:

```
vi s<escape><escape>
```

Jeżeli używasz trybu `vi`, wciśnij klawisz `\`:

```
v s\
```

Rezultat dokończenia pojawi się na ekranie. Polecenie nie zostanie wykonane, dopóki nie zatwierdzisz go klawiszem *Enter*, dzięki czemu można sprawdzić, czy rzeczywiście chodziło Ci o postać polecenia podpowiadanej przez `pdksh`.

W przypadku, gdy plik `sample.text` nie jest jedynym plikiem o nazwie zaczynającej się na literę `s`, `pdksh` dopisuje znaki, co do których nie ma wątpliwości, po czym generuje krótki dźwięk, oznaczający konieczność wprowadzenia dodatkowych informacji.



Odpowiednikiem klawisza *Escape* jest kombinacja *Control+[,* oznaczana zazwyczaj jako `^[,`. Dwukrotne wcisnięcie klawisza *Escape* może być więc zapisane jako `^[[^[,`. Taki zapis używany jest w wielu książkach i na stronach `man`.

Symbole wieloznaczne

Powłoka `pdksh` daje również możliwość używania w wierszu poleceń symboli wieloznacznych i, podobnie jak `bash`, obsługuje trzy ich rodzaje:

- * zastępuje dowolny znak lub ciąg znaków,
- ? zastępuje dowolny pojedynczy znak,
- [...] zastępuje dowolny znak zawarty pomiędzy nawiasami.

Znak * może być stosowany podobnie jak dokańczanie poleceń. Jeśli w bieżącym katalogu znajdują się następujące podkatalogi i pliki:

```
News/ bin/ mail/ test/ sample.text working/
```

a chcesz edytować plik `sample.text` edytorem `vi`, możesz wydać polecenie:

```
vi s*
```

Symbol * zastępuje dowolny znak lub ciąg znaków, więc interpreter poleceń zamienia `s*` na `sample.text` (jedyna nazwa pliku w tym katalogu, która pasuje do wzorca).

Metoda ta działa prawidłowo wtedy, gdy tylko jedna nazwa pliku zaczyna się na literę `s`. Jeśli jest ich więcej, wzorzec `s*` zostanie zastąpiony przez listę plików i podkatalogów rozpoczynających się na literę `s`. Podane wyżej polecenie spowoduje uruchomienie edytora `vi` z pierwszym plikiem z tej listy. Po zakończeniu jego edycji do edytora załadowany zostanie drugi plik itd. Metoda ta jest dobra, jeśli trzeba edytować więcej niż jeden plik. Jeśli natomiast zamierzales edytować tylko plik `sample.text`, trzeba to zrobić w inny sposób.

Bardziej praktyczne zastosowanie znaku * to wykonywanie poleceń na kilku plikach jednocześnie. Przykładowo, niech bieżący katalog zawiera następujące pliki i podkatalogi:

```
News/ bin/ games/ mail/ sample.text temp1.out
      temp2.out      temp3.out      test/
```

Jeśli chcesz usunąć wszystkie pliki z rozszerzeniem `.out`, możesz to zrobić wydając polecenie

```
rm *.out
```

W tym przypadku `pdksh` zastąpi wzorzec `*.out` listą wszystkich plików z rozszerzeniem `.out`. Po wykonaniu tego podstawienia polecenie przyjmuje następującą postać:

```
rm temp1.out temp2.out temp3.out
```

Polecenie `rm` zostanie więc wywołane z argumentami `temp1.out temp2.out temp3.out`

Symbol `?` działa na podobnych zasadach, z tym że zastępuje dokładnie jeden znak. Jeśli zawartość katalogu bieżącego jest taka sama jak w poprzednim przykładzie, pliki `temp1.out temp2.out temp3.out` można usunąć poleceniem

```
rm temp?.out
```

Symbol `[...]` pozwala bardziej szczegółowo określić, o jakie znaki lub zakresy znaków chodzi. By usunąć pliki jak wyżej, można wydać jedno z polecień:

```
rm temp[123].out
rm temp[1-3].out
```

Przywoływanie wydanych poleceń

`pdksh` obsługuje przywoływanie wydanych wcześniej poleceń prawie w taki sam sposób, jak robi to interpreter `bash`. Pamięta on pewną liczbę ostatnio wydanych poleceń (określona wartością zmiennej `HISTSIZE`).

Lista wydanych poleceń ładowana jest podczas logowania z pliku historii, którego nazwa przechowywana jest w zmiennej `HISTFILE` – domyślnie jest to plik `.ksh_history` znajdujący się w katalogu domowym. Nazwa ta, podobnie jak miało to miejsce w przypadku pliku `.bash_history`, rozpoczyna się od kropki. Oznacza to, że jest to plik ukryty i podczas przeglądania zawartości katalogu wyświetlony zostanie tylko w przypadku wyraźnego żądania, np. po wydaniu polecenia `ls -a`.

Interpreter daje możliwość poruszania się po liście wcześniej wydanych poleceń. Klawisze służące do tego celu różnią się w zależności od tego, czy używasz trybu edycji `emacs`, czy `vi`.

W przypadku trybu `emacs`, polecenie wydane wcześniej zostanie wyświetcone po naciśnięciu klawiszy *Control+p*, polecenie następne – *Control+o*. W trybie `vi` do poprzedniego polecenia wrócić można, wciskając klawisze `K` lub `-`, natomiast do następnego `J` lub `+`.



Jeśli używasz poleceń edytora `vi`, to zanim będziesz mógł przeglądać polecenia wydane wcześniej, musisz przejść do trybu wydawania poleceń (naciskając dwukrotnie klawisz `Escape`).

Jeśli zachodzi taka potrzeba, można edytować przywołane z listy polecenie. Interpreter `pdksh` udostępnia dość złożony zestaw funkcji służących do tego celu, jednak omawianie ich wykracza poza zakres tej książki. Przy wprowadzaniu niewielkich poprawek można z powodzeniem stosować najprostsze metody, na przykład użyć klawiszy kursora – strzałki w lewo i w prawo pozwalają poruszać się w obrębie jednego polecenia. Tekst można wstawiać w dowolnym miejscu polecenia i usuwać klawiszami `Delete` czy `Backspace`. Większość użytkowników uważa takie możliwości za wystarczające.



O wiele bardziej zaawansowane możliwości edycji dostępne są za pomocą kombinacji klawiszy takich, jak w edytورach `vi` oraz `emacs` (zależnie od wybranego trybu). Jeśli znasz dobrze któryś z tych edytorów, może to być pewnym udogodnieniem.

Plik historii poleceń można również edytować za pomocą polecenia `fc` (ang. *fix command*). Jeśli przeczytałeś rozdział 11. „bash”, pamiętasz na pewno, że interpreter `bash` obsługuje jeszcze jedno polecenie: `history`. Nie jest ono dostępne w powłoce `pdksh`, ponieważ wszystkie jego funkcje spełnia polecenie `fc`.



Choć polecenie `history` nie jest obsługiwane przez powłokę `pdksh`, działa ono poprawnie, ponieważ zazwyczaj w pliku konfiguracyjnym `.kshrc` zdefiniowany jest alias o następującej postaci:

`alias history='fc -l'`

Pozwala on na normalne używanie polecenia `history`, mimo że nie jest ono wewnętrznym poleceniem interpretera.

Składnia polecenia `fc` jest taka sama jak w powłoce `bash`:

`fc [-e nazwa_edytora] [-nlr] [początek] [koniec]`

Opcja `-e nazwa_edytora` pozwala podać, jaki edytor ma być użyty do edytowania poleceń. `początek` i `koniec` pozwalają określić zakres polecen, które mają zostać pobrane z pliku historii. Mogą to być numery polecień bądź łańcuchy znaków, które polecenie `fc` będzie próbowało odnaleźć w pliku historii polecen.

Opcja `-n` wyłącza numerowanie wierszy z poleceniami. Opcja `-r` wyświetla polecenia w odwrotnym porządku. Opcja `-l` przesyła znalezione polecenia na ekran. We wszystkich przypadkach oprócz opcji `-l`, wybrane polecenia ładowane są do edytora tekstu.

Edytor tekstu uruchamiany jest na podstawie nazwy podanej w opcji `-e nazwa_edytora`. Jeśli nie została ona podana, używany jest edytor wskazywany przez zmienną `FCEDIT`.

Jeśli taka zmienna nie istnieje, używana jest wartość zmiennej `EDITOR`, a w końcu, jeśli taka zmienna również nie jest zdefiniowana, uruchamiany jest edytor `vi`.

Polecenie `fc` wywołane bez argumentów powoduje załadowanie do edytora tekstu ostatnio wydanego polecenia. Po wyjściu z edytora polecenie jest wykonywane.

Oto kilka przykładów użycia polecenia `fc`:

- fc** ładuje do edytora tekstu ostatnio wydane polecenie,
- fc-l 5 10** wyświetla polecenia o numerach od 5 do 10 (włącznie),
- fc 6** ładuje do edytora tekstu polecenie o numerze 6,
- fc mo** ładuje do edytora tekstu ostatnie polecenie, które zaczyna się od tekstu `mo`.

Aliases

Kolejnym ułatwieniem oferowanym przez interpreter `pdksh` są aliasy. Są to zwykle skróty dłuższych poleceń, wykonywanych przez `pdksh` w chwili napotkania aliasu. Założmy np., że posiadasz w swoim katalogu listę rzeczy do zrobienia zapisaną w pliku `rzeczy-do-zrobienia.txt` i codziennie ją modyfikujesz i przeglądzasz, często więc musisz wydawać polecenie

```
vi rzeczy-do-zrobienia.txt
```

które do najkrótszych nie należy. Warto stworzyć dla niego alias o nazwie np. `zrob`, który od tej chwili będzie można wpisywać w zastępstwie podanego wcześniej polecenia. Dokonuje się tego za pomocą polecenia `alias` interpretera `pdksh`:

```
alias zrob='vi rzeczy-do-zrobienia.txt'
```

Od tej chwili, aż do zakończenia działania `pdksh`, wpisanie polecenia `zrob` równoważne będzie wydaniu dłuższego polecenia `vi rzeczy-do-zrobienia.txt`. Jeśli uznasz, że alias stał się już niepotrzebny, możesz go usunąć poleceniem `unalias`:

```
unalias zrob
```

Alias przestanie istnieć i próba jego wykonania spowoduje wyświetlenie komunikatu `Command not found` (nie odnaleziono polecenia).

Użytkownicy przywiązani do poleceń DOS-u mogą zdefiniować dla własnej wygody następujące aliasy:

```
alias dir='ls'  
alias copy='cp'  
alias rename='mv'  
alias md='mkdir'  
alias rd='rmdir'
```

Inne przydatne aliasy to:

```
alias ll='ls -l'  
alias lo='logout'  
alias ls='ls -F'
```



Podobnie jak w przypadku powłoki `bash`, przy definiowaniu aliasów wokół znaku równości nie powinno być spacji. Cudzysłowy są wymagane tylko wtedy, gdy tekst polecenia zastępowanego aliasem zawiera spacje lub inne znaki specjalne.

Polecenie `alias` wydane bez żadnych argumentów powoduje wypisanie wszystkich aktywnych w danej chwili aliasów. Jeśli chcesz, aby aliasy tworzone były automatycznie podczas uruchamiania interpretera, powinieneś wpisać odpowiednie polecenia do pliku konfiguracyjnego, omówionego dokładniej w podrozdziale „Dostosowywanie powłoki `pdksh`”.

Przekierowanie wejścia

Przekierowanie (lub przeadresowanie, ang. *redirection*) wejścia pozwala na zmianę źródła danych wykorzystywanych przez polecenie. Większość wydawanych w środowisku `pdksh` (i innych powłok – *przyp. tłum.*) poleceń wymaga do wykonania odpowiednich danych wejściowych; w przypadku prostszych poleceń całość informacji przekazywana jest za pomocą argumentów ich wywołania. Przykładowo, używając polecenia `rm` podaje się jako argument nazwę usuwanego pliku; jej pominięcie wyświetla wskazówkę o sposobie uzyskania dodatkowych informacji o składni tego polecenia.

Istnieją również polecenia wymagające nieco bardziej złożonych danych wejściowych. Dane takie zwykle umieszczane są w pliku. Przykładem może tu być polecenie `wc` (*word count*), zliczające znaki, słowa i wiersze w przekazanych mu danych wejściowych. Aby zliczyć znaki, słowa i wiersze w pliku, należy podać jego nazwę jako argument wywołania polecenia `wc`:

```
wc test  
11 2 1
```

Innym sposobem przekazania poleceniu `wc` danych zawartych w pliku `test` jest przeadresowanie wejścia polecenia z terminala do pliku. Służy do tego operator `<`, nakazujący poleceniu pobranie danych z pliku o podanej nazwie. W naszym przypadku operacja ta wyglądałaby tak:

```
wc < test  
11 2 1
```

i, jak widać, przyniosłaby identyczny wynik.

Przeadresowanie wejścia wykorzystywane jest raczej rzadko, gdyż polecenia pobierające dane z pliku wejściowego posiadają zwykle opcję pozwalającą podać jego nazwę w wierszu polecień. Możesz jednak znaleźć się w sytuacji, gdy będziesz musiał przekazać zawarte

w pliku dane do polecenia nie posiadającego takiej możliwości. W takiej sytuacji przeadresowanie wejścia wybawi Cię z kłopotu.

Przekierowanie wyjścia

Jest ono używane o wiele częściej niż przekierowanie wejścia. Pozwala przesyłać wyniki wykonania polecenia do pliku (lub urządzenia – *przyp. tłum.*), zamiast wyświetlać je na ekranie terminalu.

Możliwość przeadresowania wyjścia przydaje się w wielu sytuacjach. Może się np. zdażyć, że wyniki wykonania polecenia nie mieszczą się na ekranie; w takiej sytuacji można skierować je do pliku, który następnie można przejrzeć za pomocą edytora tekstowego. Przeadresowanie wyjścia polecenia dokonywane jest za pomocą operatora > (dla wejścia, jak pamiętamy, był to operator <).

Aby np. zapisać zawartość katalogu wyświetlana poleciением `ls` w pliku `katalog`, należy więc wydać polecenie:

```
ls > katalog
```

Ciągi poleceń

Podobnie jak powłoka `bash`, również `pdksh` umożliwia „szeregowe łączenie” poleceń. Wyjście polecenia zostaje skierowane na wejście następnego, jego wyjście z kolejnego wejście następnego itd.; wynik wykonania ostatniego polecenia kierowany jest na ekran. Aby utworzyć taki ciąg, należy wpisać nazwy kolejnych poleceń rozdzielone znakami |; przykładem zastosowania tego mechanizmu może być ciąg poleceń

```
cat test.txt | sort | uniq
```

Zawartość pliku `test.txt` (utworzonego poleciением `cat`) podawana jest na wejście programu `sort`, który porządkuje wiersze alfabetycznie. Jego wyjście podawane jest na wejście programu `uniq`, usuwającego z tekstu powtarzające się wiersze. Jeśli w pliku `test.txt` znajduje się następujący tekst:

```
Przykładowy dialog
Czesc
Jak sie masz
Czesc
Nie najgorzej, a ty?
Swietnie
```

na wyjściu ciągu poleceń pojawi się tekst:

```
Czesc
Jak sie masz
Nie najgorzej, a ty?
Przykładowy dialog
Swietnie
```

Wiersze zostały tu posortowane według pierwszego wyrazu i usunięte zostało powtórzanie wiersza Czesc.

Znak zachęty

Inaczej niż w przypadku powłoki `bash`, w `pdksh` istnieją trzy poziomy znaków zachęty. Znak zachęty pierwszego poziomu wyświetlany jest, gdy interpreter oczekuje na wydanie polecenia przez użytkownika. Jego postać określona jest przez wartość zmiennej `PS1` – zwykle jest to znak `$`. Aby go zmienić, na przykład na tekst „`Powiedz mi, co mam robić`”, poprzedzony liczbą wydanych do tej pory poleceń, należy wydać polecenie

```
PS1="! Powiedz mi, co mam robić"
```

Jak widać, nadanie zmiennej wartości uzyskuje się, wpisując jej nazwę, znak równości i wartość, jaką zmienna ma przyjąć. Po obu stronach znaku równości nie powinno być spacji.

Symbol wykrzyknika to nazwa jednej ze zmiennych, która oznacza w powłoce `pdksh` ilość wydanych do chwili bieżącej poleceń. Znak zachęty będzie więc miał postać tekstu `Powiedz mi, co mam robić` poprzedzonego numerem polecenia.

Znak zachęty drugiego poziomu wyświetlany jest, gdy `pdksh` oczekuje na podanie dodatkowych informacji niezbędnych do wykonania polecenia. Domyslnie jest to symbol `>`. Postać tego znaku zachęty określona jest wartością zmiennej `PS2` i można ją zmienić na przykład w taki sposób:

```
PS2="Potrzebuje więcej informacji"
```

Tekst `Potrzebuje więcej informacji` zostanie wyświetlony za każdym razem, gdy interpreter będzie potrzebował jakichś dodatkowych danych do wykonania polecenia.

`pdksh` nie posiada tak zaawansowanych możliwości konfigurowania znaku zachęty jak `bash`. Dozwolone jest jednak używanie zmiennych powłoki, co pozwala na wyświetlenie takich informacji, jak nazwa bieżącego katalogu, na przykład w taki sposób:

```
PS1="($LOGNAME) "
PS1='($PWD) '
```

Pierwsze z powyższych poleceń spowoduje, że jako znak zachęty wyświetlany będzie identyfikator użytkownika. Drugie sprawi, że wyświetlana będzie nazwa bieżącego katalogu. Niezbędny jest w tym przypadku pojedynczy cudzysłów, który powoduje, że wartość zmiennej nie jest przypisywana raz (tak byłoby, gdyby użyty został cudzysłów podwójny), ale odświeżana przed każdym wyświetleniem. Jeśli chcesz dowiedzieć się czegoś więcej o używaniu cudzysłów, zajrzyj do rozdziału 14. „Programowanie w języku powłoki”.

Zarządzanie zadaniami

Zarządzanie zadaniami to zdolność kontrolowania zachowań aktualnie działających procesów. Mówiąc konkretnie, jest to możliwość uśpienia (wstrzymania, ang. *suspend*) działającego procesu i późniejszego uruchomienia go. Interpreta bash śledzi wszystkie uruchamiane procesy (uruchamiane przez użytkownika, nie przez inne procesy) i pozwala je wstrzymywać oraz przywracać w dowolnej chwili.

Wciśnięcie klawiszy *Control+Z* powoduje wstrzymanie działającego procesu. Polecenie `bg` uruchamia go ponownie w tle, natomiast polecenie `fg` – na pierwszym planie.

Polecenia te są najczęściej używane w przypadku, gdy użytkownik myłkowo uruchomi na pierwszym planie program, który miał zostać uruchomiony w tle. Program uruchomiony na pierwszym planie blokuje dostęp do interpretera poleceń i nie pozwala użytkownikowi na zrobienie czegokolwiek aż do momentu zakończenia swojego działania. Zazwyczaj nie jest to problemem, ponieważ większość poleceń potrzebuje na zakończenie działania co najwyżej kilku sekund, ale w przypadku programów, które działają dłucho, wygodniej jest uruchomić je w tle, by móc nadal używać interpretera.

Jeśli na pierwszym planie uruchomisz polecenie, na którego wyniki trzeba czekać dość długo, interpreter poleceń zostanie zablokowany na czas działania tego polecenia. Warto wtedy wcisnąć klawisze *Control+Z*, co spowoduje wstrzymanie programu, a następnie wydać polecenie `bg`, które uruchomii go ponownie w tle, dzięki czemu można będzie znów używać interpretera poleceń.

Skróty klawiaturowe

Bardzo przydatną cechą pdksh, której brakuje w powłoce bash, jest możliwość definiowania własnych skrótów klawiaturowych (ang. *key bindings*). Jeśli na przykład nie podoba Ci się, że poprzednie polecenie przywołuje się kombinacją *Control+p* możesz ją zmienić na dowolną inną. Składnia polecenia, które pozwala to zrobić, jest następująca:

```
bind <kombinacja_klawiszy> <polecenie>
```

Dzięki temu poleceniu możliwe jest dokładne dostosowanie interpretera do swoich wymagań. Najczęściej spotyka się właśnie zmianę klawiszy używanych do poruszania się po historii i wierszu poleceń na klawisze kurSORA, tak jak w powłoce bash. Odpowiednie polecenia przeważnie zapisywane są w pliku .kshrc, odczytywanym i wykonywanym przy każdym uruchomieniu interpretera pdksh. Mają one postać:

```
bind '^['=prefix-2
bind '^XA'=up-history
bind '^XB'=down-history
bind '^XC'=forward-char
bind '^XD'=backward-char
```

W tabeli 12.1 zebrane zostały najczęściej wykorzystywane polecenia edycyjne, których można używać przy definiowaniu skrótów klawiaturowych. Listę wszystkich dostępnych poleceń edycyjnych uzyskać można, wydając polecenie `bind` bez żadnych argumentów.

Tabela 12.1. Najczęściej wykorzystywane polecenia edycyjne powłoki `pdksh`

Polecenie	Znaczenie
abort (^G)	Przerywa działanie innego polecenia edycyjnego, najczęściej przeszukiwania historii poleceń.
backward-char (^B)	Przesuwa kursor o jeden znak do tyłu. Często przypisane do klawisza ze strzałką w lewo.
backward-word (^[b)	Przesuwa kursor na początek wyrazu.
beginning-of-line (^A)	Przesuwa kursor na początek wiersza poleceń.
complete (^[^])	Nakazuje <code>pdksh</code> dokończyć polecenie.
copy-last-arg (^[_)	Powoduje wstawienie do polecenia ostatniego wyrazu poprzedniego polecenia.
delete-char-backward (ERASE)	Usuwa znak na lewo od kursora.
delete-char-forward	Usuwa znak na prawo od kursora.
delete-word-backward (^[ERASE)	Usuwa znaki na lewo od kursora aż do napotkania spacji (lub innego znaku białego).
delete-word-forward (^[D)	Usuwa znaki na prawo od kursora aż do napotkania spacji (lub innego znaku białego).
down-history (^N)	Przesuwa listę historii poleceń o jeden wiersz w dół. Często przypisane do klawisza ze strzałką w dół.
end-of-line (^E)	Przesuwa kursor na koniec bieżącego wiersza.
forward-char (^F)	Przesuwa kursor o jeden znak do przodu. Często przypisane do klawisza ze strzałką w prawo.
forward-word (^[F)	Przesuwa kursor na koniec wyrazu.
kill-line (KILL)	Usuwa bieżący wiersz.
kill-to-eol (^K)	Usuwa wszystkie znaki od kursora do końca wiersza.
list (^[?)	Powoduje, że <code>pdksh</code> wypisuje wszystkie polecenia lub nazwy plików mogące stanowić dokończenie słowa, w którym aktualnie znajduje się kursor.
search-history (^R)	Szuka w liście historii polecień pierwszego polecenia zawierającego podane znaki.
transpose-chars (^T)	Zamienia miejscami znaki z prawej i lewej strony kursora. Jeśli kursor jest na końcu wiersza, zamienia dwa ostatnie znaki.
up-history (^P)	Przesuwa listę historii poleceń o jeden wiersz w góre. Często przypisane do klawisza ze strzałką w góre.

Dostosowywanie powłoki pdksh

W tym rozdziale opisano już kilka sposobów dostosowywania powłoki `pdksh` do własnych potrzeb. Jednak wprowadzone usprawnienia działały tylko do momentu zakończenia sesji programu `pdksh`. Można uczynić je bardziej trwałymi poprzez zapisanie odpowiednich poleceń w pliku konfiguracyjnym.

W pliku konfiguracyjnym można zapisać dowolne polecenia (zwykle są to definicje aliasów i inicjalizacja zmiennych, na przykład określających znak zachęty), które zostaną wykonane przy każdym uruchomieniu powłoki `pdksh`.

Inaczej niż miało to miejsce w interpreterze `bash` (który wiedział, gdzie ma szukać pliku konfiguracyjnego), należy jawnie określić, gdzie znajduje się taki plik. W tym celu trzeba stworzyć (o ile nie istnieje) w katalogu domowym plik `.profile`, który wykonywany jest przy każdym logowaniu do systemu, bez względu na rodzaj używanej powłoki, i dodać do niego następujące wiersze:

```
export ENV=$HOME/.kshrc  
EDITOR=emacs
```

Pierwsze z powyższych poleceń przypisuje wartość zmiennej `ENV`. Właśnie tę wartość sprawdza `pdksh` podczas uruchomienia, szukając nazwy pliku zawierającego dane konfiguracyjne. Jeśli zamierzasz modyfikować zachowanie tej powłoki, powinieneś podać tu nazwę pliku konfiguracyjnego zapisanego w Twoim katalogu domowym. Można oczywiście podać dowolną nazwę pliku, ale najczęściej używana jest nazwa `.kshrc`.

Jeśli nie planujesz dostosowywania `pdksh`, powinieneś ustawić wartość zmiennej `ENV` tak, aby wskazywała na domyślny systemowy plik konfiguracyjny. Plik ten nazywa się `ksh.kshrc` i zapisany jest w katalogu `/etc`.

Drugi wiersz pliku `.profile` określa wartość zmiennej `EDITOR`. Jest ona używana do tego, by zdecydować, jakie kombinacje klawiszy mają być aktywne podczas sesji interpretera. Jeśli wolisz używać trybu `vi`, powinieneś zamiast `emacs` wpisać `vi`.



Zamiast kopiować domyślny plik konfiguracyjny do katalogu domowego, można stworzyć własny plik konfiguracyjny wywołujący najpierw plik domyślny, a następnie wprowadzający odpowiednie zmiany.

Polecenia wewnętrzne powłoki pdksh

W interpreterze `pdksh` dostępnych jest wiele polecen wewnętrznych, ale większości z nich używa się bardzo rzadko. Te najbardziej przydatne zostały zebrane w tabeli 12.2.

Tabela 12.2. Najczęściej wykorzystywane polecenia powłoki pdksh

Polecenie	Znaczenie
.	Odczytuje i wykonuje zawartość pliku (polecenie to jest omówione bliżej w rozdziale 14.)
alias	Tworzy alias – czyli skróconą wersję polecenia
bg	Powoduje, że wstrzymany proces uruchamia się ponownie w tle
cd	Przechodzi do katalogu, którego nazwa podana jest jako argument
exit	Kończy działanie powłoki
export	Powoduje, że wartość zmiennej będzie dostępna dla wszystkich procesów potomnych
fc	Pozwala na edycję historii poleceń
fg	Powoduje, że wstrzymany proces uruchamia się ponownie na pierwszym planie
kill	Służy do zatrzymywania procesów
pwd	Wyświetla nazwę katalogu bieżącego
unalias	Usuwa alias

Zmienne powłoki pdksh

Najczęściej używane zmienne interpretera poleceń `pdksh` przedstawione są, wraz z krótkimi objaśnieniami, w tabeli 12.3.

Tabela 12.3. Najczęściej wykorzystywane zmienne powłoki pdksh

Nazwa zmiennej	Znaczenie
EDITOR, FCEDIT	Domyślny edytor dla polecenia <code>fc</code>
HISTFILE	Nazwa pliku, w którym przechowywana będzie historia poleceń
HISTSIZE	Rozmiar historii poleceń
HOME	Katalog domowy bieżącego użytkownika
OLDPWD	Poprzedni katalog bieżący
PATH	Ścieżka przeszukiwania
PS1	Znak zachęty pierwszego poziomu
PS2	Znak zachęty drugiego poziomu
PWD	Nazwa bieżącego katalogu
SECONDS	Liczba sekund, które upłynęły od uruchomienia powłoki.

Podsumowanie

Przyjrzałeśi się podstawowym cechom powłoki `pdksh` (Public Domain Korn Shell). Pod wieloma względami jest ona podobna do powłoki `bash`, ale różni się kilkoma drobiazgami.

W następnym rozdziale zajmiemy się interpreterem poleceń `tcsh`, który jest linuxowym odpowiednikiem powłoki C. Kiedy już zapoznasz się z właściwościami wszystkich trzech powłok, nie powinieneś mieć problemu z wybraniem tej z nich, która najbardziej odpowiada Twoim potrzebom. Możesz oczywiście użyć każdej z nich w dowolnej chwili, po prostu wpisując jej nazwę (są to w końcu tylko zwykłe programy).

Więcej o programowaniu w języku powłoki możesz dowiedzieć się z rozdziału 14. „Programowanie w języku powłoki”

Edytory tekstu, których można używać do tworzenia i edytowania plików, omówione są w rozdziale 16. „Edytory tekstu: vi i emacs”.

Jeśli chcesz dowiedzieć się, jak skonfigurować interfejs graficzny X, przejdź do rozdziału 22. „Instalacja i konfiguracja XFree86”.

Rozdział 13.

tcsh

Rick McMullin

W tym rozdziale:

- υ Powłoka `tcsh`
- υ Inne przydatne możliwości
- υ Dostosowywanie powłoki `tcsh`
- υ Polecenia wewnętrzne interpretera `tcsh` – podsumowanie
- υ Zmienne powłoki `tcsh`

W dwóch poprzednich rozdziałach przedstawione zostały powłoki `bash` (Bourne Again Shell) oraz `pdksh` (Public Domain Korn Shell). W tym rozdziale omówimy trzecią powłokę dostępną w systemach linuxowych, czyli `tcsh`. Pokażemy niektóre sposoby dostosowywania tego interpretera do własnych potrzeb oraz najważniejsze wbudowane polecenia i zmienne.

Końcową część tego rozdziału poświęcimy na omówienie kilku interesujących możliwości interpretera `tcsh`, niedostępnych ani w powłoce `bash`, ani `pdksh`.

Powłoka tcsh

`tcsh` to zmodyfikowana wersja powłoki C (`csh`). Jest z nią całkowicie kompatybilna, a przy okazji dodaje kilka własnych rozszerzeń, ułatwiających i usprawniających pracę interaktywną, między innymi obsługę historii poleceń i poprawioną edycję wiersza poleceń.

Dokańczanie poleceń

Podobnie jak w powłokach `pdksh` i `bash`, w powłoce `tcsh` również możliwe jest automatyczne dokańczanie poleceń. Wywołuje się je, wciskając klawisz `Tab`, czyli tak samo jak w powłoce `bash`.

Po wcisnięciu klawisza `Tab` `tcsh` próbuje dokończyć polecenie, szukając pliku, którego nazwa pasuje do rozpoczętej, w katalogu, do którego polecenie się odnosi. Przyjmijmy przykładowo, że wpisałeś następujące polecenie:

```
emacs ksi<tab>
```

`tcsh` spróbuje dokończyć polecenie, szukając pliku (lub podkatalogu) w katalogu bieżącym, którego nazwa zaczyna się od liter `ksi` (np. `ksiazka`). Natomiast po wydaniu polecenia

```
emacs /usr/bin/ksi<tab>
```

`tcsh` będzie szukał odpowiedniego pliku lub podkatalogu w katalogu `/usr/bin`.

Oto inne przykłady użycia mechanizmu dokańczania poleceń. Założymy, że w katalogu bieżącym znajdują się następujące pliki i podkatalogi:

```
News/ bin/ mail/ test.txt plik1 std.txt
```

Jeśli chcesz wydrukować plik `test.txt`, możesz wydać polecenie:

```
lpr test.txt
```

Można również zaoszczędzić sobie wpisywania kilku liter i użyć mechanizmu dokańczania poleceń, wciskając klawisz `Tab`:

```
lpr t<Tab>
```

`tcsh` spróbuje dokończyć wydawane polecenie. Ponieważ jedynym plikiem w bieżącym katalogu, którego nazwa zaczyna się na literę `t`, jest plik `test.txt`, interpreter dokończy polecenie w następujący sposób:

```
lpr test.txt
```

Można teraz zatwierdzić polecenie klawiszem `Enter` lub edytować je, jeśli nie chodziło Ci o to, co dopisał interpreter. Powinieneś bardzo uważać przy używaniu dokańczania poleceń z poleceniami takimi jak `rm`, ponieważ bardzo łatwo skasować jakieś dane przez pomyłkę.

Symboli wieloznaczne

Podobnie jak powłoki `bash` i `pdksh`, również `tcsh` pozwala używać symboli wieloznacznych. Obsługiwane są trzy ich rodzaje:

- * zastępuje dowolny znak lub ciąg znaków,
- ? zastępuje dowolny pojedynczy znak,
- [...] zastępuje dowolny znak zawarty pomiędzy nawiasami.

Znak * może być używany w sposób podobny do dokańczania poleceń. Jeśli wydasz polecenie

```
cd t*
```

a w katalogu bieżącym znajduje się tylko jeden podkatalog, którego nazwa zaczyna się na literę t, polecenie to zachowa się dokładnie tak samo, jak gdyby użyto dokańczania poleceń, wciskając klawisz Tab.

Symbol * zastępuje dowolny znak lub ciąg znaków, więc interpreter w miejscu t* podstawi nazwę pliku lub podkatalogu, która pasuje do wzorca – czyli rozpoczyna się na t.

Metoda ta działa prawidłowo wtedy, gdy tylko jedna nazwa podkatalogu zaczyna się na literę t. Jeśli jest ich więcej, wzorzec t* zostanie zastąpiony przez listę plików i podkatalogów rozpoczynających się na literę t. Polecenie cd spowoduje przejście do pierwszego katalogu z tej listy, który nie musi być katalogiem, o który chodziło.

Bardziej praktyczne zastosowanie znaku * to wykonywanie poleceń na kilku plikach jednocześnie. Przykładowo, niech bieżący katalog zawiera następujące pliki:

```
Mail/    atc1.stk      atc2.stk      bin/    borl.stk      cdrom.txt
Σ       lfi.stk temp/
```

Jeśli chcesz wydrukować dwa pliki z rozszerzeniem .stk, których nazwy zaczynają się od liter atc, możesz wydać polecenie

```
lpr a*.stk
```

Zadziała ono prawidłowo, ponieważ w katalogu nie ma innych plików o nazwach zaczynających się na a i z rozszerzeniem .stk.

Mogą również wydrukować te dwa pliki, używając symbolu ? w następujący sposób:

```
lpr atc?.stk
```

Trzecim sposobem jest zastosowanie symbolu [], który pozwala określić zakres znaków:

```
lpr atc[12].stk
```

Przywoływanie wydanych poleceń

Mechanizm przywoływania wydanych poleceń jest podobny jak w obu opisanych w poprzednich rozdziałach powłokach. tcsh przechowuje informacje o pewnej liczbie wprowadzonych wcześniej poleceń. Liczba ta określona jest przez wartość zmiennej history.

Kiedy logujesz się do systemu, lista poleceń ładowana jest z pliku historii – domyślnie jest to plik o nazwie `.history`, znajdujący się w katalogu domowym. Jego nazwa przechowywana jest w zmiennej `histfile` i może zostać zmieniona. Zauważ, że rozpoczyna się ona od kropki, czyli jest nazwą pliku ukrytego, wyświetlaną podczas przeglądania zawartości katalogu tylko na wyraźne żądanie, np. po wydaniu polecenia `ls -a`.



Jeśli chcesz, by historia poleceń była zapisywana do pliku przy wylogowywaniu się, powinieneś się upewnić, czy wartość zmiennej `savehist` jest ustawiona na taką liczbę poleceń, jaką chcesz przechowywać. Przykładowe ustawienia możesz znaleźć na wydruku zawartości pliku `.login` w podrozdziale „Dostosowywanie tcsh”.

Najprostszą metodą dostępu do wydanych wcześniej poleceń jest użycie klawiszy kurSORA. Strzałka w góre powoduje przywołanie polecenia poprzedniego, w dół – następnego. Jeśli wciskając klawisz ze strzałką w góre przez przypadek ominiesz polecenie, o które Ci chodziło, możesz do niego wrócić, naciskając klawisz ze strzałką w dół.

Polecenie znajdujące się w wierszu poleceń może również być edytowane. Służą do tego celu strzałki w lewo i prawo oraz klawisze Backspace i Delete. Większość użytkowników uważa takie możliwości za wystarczające. Bardziej zaawansowane kombinacje klawiszy (podobne do kombinacji używanych w edytorach `emacs` i `vi`) opisane są w podrozdziale „Skróty klawiaturowe”.

Innym sposobem używania historii poleceń jest jej przeglądanie i edytowanie za pomocą polecenia `history`. Polecenie to można wywołać na dwa sposoby. Pierwszy z nich ma składnię

```
history [-hr] [n]
```

Polecenie o takiej składni służy do wyświetlania listy wydanych wcześniej poleceń na ekranie. Użycie parametru `n` pozwala określić liczbę ostatnio wydanych poleceń, które należy wyświetlić (jeśli jej nie podano, wyświetcone zostaną wszystkie zapamiętane polecenia). Opcja `-h` powoduje, że nie zostaną wyświetlane numery wierszy i czas wydania polecenia. Opcja `-r` wyświetla listę w odwrotnym porządku, rozpoczynając od polecień wydanych najpóźniej. Ostatnie pięć poleceń można na przykład obejrzeć wydając polecenie

```
history 5
```

Drugi sposób służy do modyfikowania zawartości pliku historii. Ma on składnię:

```
history -S|-L|-M [nazwa_pliku]
```

Opcja `-s` zapisuje aktualną listę poleceń do pliku, `-L` powoduje dołączenie zawartości pliku historii do aktualnej listy poleceń, natomiast `-M` dołącza zawartość pliku historii do aktualnej listy, sortując ją jednocześnie według czasu wydania polecenia.



Jeśli nazwa pliku nie zostanie jawnie podana, użyta zostanie nazwa przechowywana w zmiennej `histfile`. Jeśli zmienna taka nie jest zdefiniowana, użyta zostanie nazwa `~/.history`.

Można również uruchomić polecenie `history` z opcją `-c`, co powoduje usunięcie bieżącej historii poleceń.

Oprócz polecenia `history` interpreter `tcsh` oferuje również inne metody dostępu do poprzednio wydanych poleceń. Oto przykłady:

- !n** wykonuje ponownie polecenie o numerze n w liście historii poleceń,
- !-n** wykonuje ponownie polecenie, które jest n-te od końca w liście historii poleceń,
- !!** wykonuje ponownie ostatnio wydane polecenie,
- !c** wykonuje ponownie ostatnie polecenie z historii poleceń, które zaczyna się na literę c,
- ?c?** wykonuje ponownie ostatnie polecenie, które zawierało literę c.

Polecenia te pozwalają na zmianę wyrazów bądź liter w poprzednio wydanych poleceniach, umożliwiają również dodanie jakiegoś parametru do wydanego wcześniej polecenia.

Więcej informacji na ten temat możesz znaleźć na stronach `man` poświęconych powyżej `tcsh`, które można obejrzeć po wydaniu polecenia:

```
man tcsh
```

Aliases

Aliases to skróty dłuższych, często używanych poleceń, służące wygodzie użytkownika. Przykładowo, jeśli wykonasz następujące polecenie:

```
alias ls 'ls -F'
```

to za każdym razem, gdy wydasz polecenie `ls`, wykona się zamiast niego polecenie `ls -F`.

Jeśli zdecydujesz, że dany alias nie będzie już więcej potrzebny, możesz go usunąć polecienniem

```
unalias:  
unalias ls
```

Po wydaniu tego polecenia alias przestaje istnieć. Jeśli jego nazwa nie była równocześnie nazwą jakiegoś polecenia, próba wywołania go spowoduje błąd.

Najczęściej używane aliasy to:

```
alias ll 'ls -l'  
alias ls 'ls -F'
```

Jeśli jesteś użytkownikiem systemu DOS, aliasy emulujące niektóre polecenia tego systemu mogą ułatwić Ci pracę:

```
alias dir 'ls'  
alias copy 'cp'  
alias rename 'mv'  
alias md 'mkdir'  
alias rd 'rmdir'
```



Podczas definiowania aliasów cudzysłowy niezbędne są tylko wtedy, gdy polecenie zastępowane aliasem zawiera spacje lub inne znaki specjalne.

Polecenie `alias` bez żadnych argumentów powoduje wypisanie wszystkich zdefiniowanych aliasów, na przykład tak:

```
alias ls 'ls -F'  
alias dir 'ls'  
alias ll 'ls -l'  
alias md 'mkdir'  
alias rd 'rmdir'
```

Przekierowanie wejścia i wyjścia

Standardowe wejście i wyjście polecenia może być przekierowane w taki sam sposób, jak przy użyciu powłok omawianych poprzednio. Do przekierowania wejścia używa się operatora <, natomiast wyjścia – operatora >. Poniższe polecenie spowoduje przeadresowanie wejścia polecenia `cat` do pliku `.cshrc`, czyli wyświetlenie na ekranie zawartości tego pliku:

```
cat <.cshrc
```

Można oczywiście wyświetlić zawartość tego pliku, wydając polecenie `cat .cshrc`. Dlatego przekierowanie wejścia jest używane dość rzadko.

Znacznie częściej w praktyce używa się przekierowania wyjścia. Poniższe polecenie powoduje przeadresowanie wyjścia polecenia `cat` do pliku `cshenv`, czyli przesłanie do tego pliku połączonej zawartości plików `.cshrc` oraz `.login`:

```
cat .cshrc .login > cshenv
```



Plik, do którego przekierowane jest wyjście polecenia, jest tworzony, jeśli nie istnieje, natomiast jeśli istnieje, jego zawartość jest

usuwana bez żadnego ostrzeżenia.

Ciągi poleceń

Implementacja ciągów poleceń (ang. *pipelines*), z jaką mamy do czynienia w powloce `tcsh`, nie różni się niczym z punktu widzenia użytkownika od implementacji w poprzednio omówionych powłokach. Pozwala ona na „szeregową łączenie” poleceń. Wyjście polecenia zostaje skierowane na wejście następnego, jego wyjście z kolejnym na wejście następnego itd.; wynik wykonania ostatniego polecenia wyświetlany jest na ekranie (chyba że wyjście ostatniego polecenia zostanie skierowane do pliku).

Aby utworzyć taki串, należy wpisać nazwy kolejnych poleceń rozdzielone znakami `|`; przykładem zastosowania tego mechanizmu może być串 poleceń

```
cat plik1 plik2 | wc -l
```

Polecenie `cat` połączy zawartości plików `plik1` i `plik2`, po czym prześle je na wyjście. Ponieważ wyjście tego polecenia jest połączone z wejściem programu `wc` – potraktuje on połączone pliki jako dane wejściowe, zliczając występujące w nich wiersze. Wynik działania polecenia `wc` – łączna liczba wierszy w obu plikach – zostanie wyświetlony na ekranie.

Znak zachęty

`tcsh` posiada trzy poziomy znaków zachęty. Znak zachęty pierwszego poziomu wyświetlany jest, gdy interpreter oczekuje na wprowadzenie polecenia. Domyślnie jest to znak `%`. Można go zmodyfikować poprzez nadanie nowej wartości zmiennej `prompt`, na przykład tak:

```
set prompt="%t$"
```

Powyższe polecenie spowoduje, że jako znak zachęty wyświetlany będzie aktualny czas oraz znak dolara.

Znak zachęty drugiego poziomu pojawia się, gdy `tcsh` oczekuje na wprowadzenie danych, działając w pętli `while` albo `for` (pętli tych używa się przy programowaniu w języku powłoki, temat ten omówiony jest dokładniej w rozdziale 14. „Programowanie w języku powłoki”). Domyślnie znakiem zachęty drugiego poziomu jest `%R?`, gdzie `%R` jest specjalnym kodem powodującym wyświetlenie statusu interpretera. Postać znaku zachęty drugiego poziomu przechowywany jest jako wartość zmiennej `prompt2`, można ją zmienić na przykład tak:

```
set prompt2="?"
```

Znak zachęty trzeciego poziomu wyświetlany jest, gdy działa automatyczna korekta błędów. Jego postać jest określona wartością zmiennej `prompt3`, domyślnie jest to wartość `CORRECT>%R (y|n|e)?`. Jeśli chcesz dowiedzieć się więcej na ten temat, zajrzyj do podrozdziału „Poprawianie pomyłek powstających podczas pisania”.

`tcsh` obsługuje symbole specjalne, które mają określone znaczenie, gdy znajdą się w jednej ze zmiennych `prompt`. Najczęściej używane wraz z opisem działania zebrano w tabeli 13.1.

Tabela 13.1. Symbole specjalne używane w definicjach znaków zachęty powłoki `tcsh`

Symbol	Znaczenie
<code>%/</code>	Wyświetla nazwę bieżącego katalogu
<code>%h, %!, !</code>	Wyświetlają bieżący numer polecenia w historii poleceń
<code>%t, %@</code>	Wyświetlają aktualny czas
<code>%n</code>	Wyświetla identyfikator użytkownika
<code>%d</code>	Wyświetla dzień tygodnia
<code>%w</code>	Wyświetla nazwę bieżącego miesiąca
<code>%y</code>	Wyświetla bieżący rok

Przykładowe polecenie:

```
set prompt="%h %/"
```

powoduje, że jako znak zachęty pierwszego poziomu wyświetlany będzie aktualny numer polecenia w historii poleceń oraz nazwa bieżącego katalogu.

Zarządzanie zadaniami

Zarządzanie zadaniami to zdolność kontrolowania zachowań aktualnie działających procesów. Mówiąc konkretniej, jest to możliwość uśpienia (wstrzymania, ang. *suspend*) działającego procesu i późniejszego uruchomienia go. Interpreter `bash` śledzi wszystkie uruchamiane procesy (uruchamiane przez użytkownika, nie przez inne procesy) i pozwala je wstrzymywać oraz przywracać w dowolnej chwili.

Wciśnięcie klawiszy `Control+Z` powoduje wstrzymanie działającego procesu. Polecenie `bg` uruchamia go ponownie w tle, natomiast polecenie `fg` – na pierwszym planie.

Polecenia te są najczęściej używane w przypadku, gdy użytkownik pomyłkowo uruchomi na pierwszym planie program, który miał zostać uruchomiony w tle. Program uruchomiony na pierwszym planie blokuje dostęp do interpretera polecen i nie pozwala użytkownikowi na zrobienie czegokolwiek aż do momentu zakończenia swojego działania. Zazwyczaj nie jest to problemem, ponieważ większość polecień potrzebuje na zakończenie

działania co najwyżej kilku sekund, ale w przypadku programów, które działają dłujo, wygodniej jest uruchomić je w tle, by móc nadal używać interpretera `tcsh`.

Jeśli na przykład na pierwszym planie uruchomisz polecenie

```
find / -name "test" >find.out
```

(które przeszukuje cały system plików w poszukiwaniu pliku o nazwie `test`, wysyłając wyniki do pliku `find.out`), na wyniki będziesz musiał poczekać od kilkunastu sekund do kilku minut, zależnie od wielkości systemu plików. W takim przypadku można wcisnąć klawisze `Control+Z`, co spowoduje wstrzymanie programu `find`, a następnie wydać polecenie `bg`, które uruchomi go ponownie w tle, dzięki czemu można będzie znów używać interpretera poleceń.

Skróty klawiaturowe

Podobnie jak `pksh`, `tcsh` umożliwia zmianę istniejących i dodawanie nowych skrótów klawiaturowych (ang. *key bindings*). Rozwiązania zastosowane w interpreterze `tcsh` są jednak bardziej elastyczne i wygodniejsze.

Możliwe jest zdefiniowanie klawisza skrótu wywołującego dowolny program, nie tylko polecenie wewnętrzne. Dzięki temu można na przykład przypisać do poszczególnych klawiszy polecenia edycyjne edytora `vi`, podczas gdy `pksh` umożliwiał przypisywanie wyłącznie poleceń edytora `emacs`.

Możliwość tworzenia własnych skrótów klawiaturowych może okazać się również bardzo przydatna, gdy przyzwyczajony jesteś do edytora innego niż `vi` lub `emacs`. Składnia polecenia definiującego skrót klawiaturowy ma następującą postać:

```
bindkey [opcja] <tekst_pierwotny lub nazwa_klawisza> <tekst_wyjsciowy  
lub polecenie>
```

Opcje polecenia `bindkey` nie są omawiane w tej książce. Jeśli chcesz się czegoś o nich dowiedzieć, zajrzyj na strony `man` dotyczące interpretera `tcsh`. Podstawową funkcją tego polecenia jest wiązanie kombinacji klawiszy podanej jako pierwszy argument z poleceniem, podanym jako drugi argument.

Tabela 13.2 zawiera listę najbardziej przydatnych poleceń, które warto powiązać z jakiś klawiszami, oraz klawisze, do których są one przyporządkowane domyślnie. Listę wszystkich poleceń powiązanych z kombinacjami klawiszy możesz uzyskać wpisując polecenie `bindkey` bez żadnych parametrów.

Polecenia te pozostają bez zmian bez względu na to, czy pracujesz w trybie `emacs` czy w trybie wstawiania `vi`. Powłoka `tcsh` obsługuje również wiele innych poleceń edycyjnych. Informacje o nich znajdziesz na stronach `man`.

Oto przykłady polecień definiujących skróty klawiaturowe:

```
bindkey ^W kill-whole-line
bindkey ^S beginning-of-line
```

Tabela 13.2. Najczęściej wykorzystywane polecenia edycyjne powłoki tcsh

Polecenie	Znaczenie
beginning-of-line (^A)	Przesuwa kursor na początek wiersza poleceń
backward-char (^B)	Przesuwa kursor o jeden znak do tyłu
end-of-line (^E)	Przesuwa kursor na koniec bieżącego wiersza
forward-char (^F)	Przesuwa kursor o jeden znak do przodu
backward-delete-char (^H)	Usuwa znak na lewo od kursora
kill-line (^K)	Usuwa bieżący wiersz
clear-screen (^L)	Czyści ekran tekstowy
down-history (^N)	Przesuwa o jeden wiersz w dół listę historii poleceń
up-history (^P)	Przesuwa o jeden wiersz w górę listę historii poleceń
kill-whole-line (^U)	Usuwa wszystkie znaki z bieżącego wiersza

Inne przydatne możliwości

tcsh posiada kilka cech, których nie ma żaden z omawianych wcześniej interpreterów. Poniższe podrozdziały omawiają najbardziej przydatne z nich.

Poprawianie pomyłek powstałych w trakcie pisania

Ta cecha, niedostępna w żadnym z pozostałych interpreterów omawianych w tej książce, jest dla niektórych użytkowników (włączając mnie) bez wątpienia spełnieniem marzeń. Jeśli prześladują Cię przypadkowo wciśnięte klawisze, możliwe, że tylko ta cecha sprawi, że będziesz używał tcsh. Można zażądać poprawiania błędów popełnionych w trakcie wydawania polecenia (przed jego zatwierdzeniem), można również zażyczyć sobie, by były one poprawiane w przypadku, gdy polecenie nie zostanie rozpoznane prawidłowo.

Pierwsza możliwość nie jest zbyt przydatna, ponieważ musisz zorientować się, że popełniłeś błąd, zanim skończysz wpisywać polecenie. Można ją wywołać, wciskając klawisze *Esc+S* w wierszu poleceń przed wciśnięciem klawisza *Enter*.

Załóżmy na przykład, że chcesz zmienić katalog na */usr/X386/X11/bin*, ale zamiast tego udało Ci się wpisać

```
cd /usr/X386/X11/bun
```

Jeśli zorientowałeś się, że popełniłeś błąd, zanim wcisnąłłeś *Enter*, możesz wcisnąć klawisze *Esc+S*. `tcsh` poprawi polecenie na:

```
cd /usr/X386/X11/bin
```

Teraz można zatwierdzić je klawiszem *Enter*. Oczywiście poprawianie błędów ma swoje ograniczenia, ponieważ powłoka nie potrafi czytać w myślach użytkownika, mimo to z większością prostych literówek radzi sobie zaskakująco dobrze.

Drugą metodą jest poinstruowanie `tcsh`, by uruchamiał poprawianie błędów w sytuacji, gdy nie potrafi rozpoznać polecenia. Można zażądać, by poprawiane były nazwy poleceń lub całe polecenia (wraz z argumentami), ustawiając odpowiednio wartość zmiennej `correct`, np.:

```
set correct=cmd lub  
set correct=all
```

Przy takich ustawieniach, gdy interpreter natknie się na polecenie, którego nie rozumie, spróbuje je poprawić. Jeśli uda mu się ustalić, w którym miejscu popełniłeś błąd, wyświetli poprawioną wersję polecenia, umożliwiając zatwierdzenie lub odrzucenie jej. Jeśli na przykład zmienna `correct` ma wartość `all` i wydane zostanie polecenie:

```
cd /usr/games
```

`tcsh` wyświetli następujące zapytanie:

```
CORRECT>cd /usr/games (y|n|e) ?
```

Jeśli po takiej reakcji powłoki wciśniesz klawisz `y` (*yes*), wykonane zostanie polecenie w wersji poprawionej. Jeśli wciśniesz `n` (*no*), zostanie wykonane polecenie wpisane pierwotnie, co najprawdopodobniej spowoduje wyświetlenie komunikatu o błędzie. Wciśnięcie klawisza `e` (*edit*) powoduje powrót do edycji wiersza poleceń, umożliwiając ręczne poprawienie popełnionych błędów.

Prepolecenia

Prepolecenia są sposobem na to, by interpreter wywołał zadane polecenie przed każdym wyświetlaniem znaku zachęty. Polecenie, które ma być wykonywane, określone jest przez wartość zmiennej `precmd`. Jeśli zmiennej `precmd` przypisano jakąś wartość, jest ona interpretowana jako polecenie, które wykonywane jest każdorazowo przed wyświetleniem na ekranie znaku zachęty. Przykładowo, po wydaniu polecenia

```
alias precmd time
```

przed każdym pojawieniem się znaku zachęty wywołane zostanie polecenie `time`.

Polecenia wykonywane przy zmianie katalogu

`tcsh` umożliwia również zdefiniowanie polecenia, które będzie wykonywane za każdym razem, gdy zmieniany jest katalog bieżący (zwykle w wyniku wydania polecenia `cd`). Ten typ polecenia jest wykorzystywany częściej niż prepolecenia, ponieważ pozwala na uzyskanie jakiegoś typu informacji o katalogu, do którego wchodzisz.

Polecenie wykonywane przy zmianie katalogu określone jest przez wartość zmiennej `cwdcmd`. Często na przykład używa się tu polecenia `pwd`, które powoduje, że po każdej zmianie katalogu wyświetlana jest pełna ścieżka dostępu do niego. Aby uzyskać taki efekt, należy wydać polecenie:

```
alias cwdcmd 'pwd'
```



Wartość zmiennej `cwdcmd` nie powinna wskazywać na polecenie `cd`, ponieważ może to spowodować, że interpreter `tcsh` wpadnie w nieskończoną pętlę i stracisz nad nim kontrolę.

Monitorowanie sesji

`tcsh` udostępnia mechanizm, który pozwala obserwować każde logowanie się i wylogowywanie dowolnego użytkownika. Jest to możliwe dzięki istnieniu zmiennej `watch`.

Wartość tej zmiennej składa się z par identyfikator użytkownika – numer terminalu. Mogą one zawierać symbole wieloznaczne oraz słówko `any` (ang. *dowolny*). Składnia polecenia definiującego wartość zmiennej `watch` jest następująca:

```
set watch=(<id_użytkownika> <terminal>)
```

`terminal` jest w tym przypadku nazwą odpowiedniego urządzenia w systemie Linux, natomiast `id_użytkownika` to identyfikator użytkownika, którego poczynania chcesz obserwować.

Większość użytkowników korzysta z tej możliwości, by obserwować, kiedy logują się ich przyjaciele. Przykładowo, jeśli czekasz na osobę o identyfikatorze `jas`, możesz wydać polecenie

```
set watch=(jas any)
```

dzięki któremu `tcsh` poinformuje Cię, gdy `jas` zaloguje się do systemu. Domyslnie sprawdzanie następuje co 10 minut, można zmienić tę wartość przez podanie liczby minut pomiędzy sprawdzaniami przed parą identyfikator użytkownika – numer terminalu, w następujący sposób:

```
set watch=(5 jas any)
```

Bądź ostrożny przy wydawaniu tego typu poleceń w wolnych systemach lub w systemach o dużej liczbie użytkowników, ponieważ sam przyczyniasz się wtedy do przeciążania

systemu. Używanie tego polecenia przez krótki okres czasu dla jakiegoś określonego celu nikomu nie zaszkodzi, ale używane bez przerwy powoduje niepotrzebne obciążanie systemu.

Dostosowywanie tcsh

W tym rozdziale przedstawiliśmy już wiele sposobów na to, by dostosować powłokę `tcsh` do swoich potrzeb. Aby wprowadzone zmiany były permanentne, musisz zapisać je do pliku, który jest przetwarzany podczas każdego uruchomienia interpretera `tcsh`.



Nawet drobna pomyłka w pliku konfiguracyjnym `tcsh` może spowodować problemy podczas logowania się. Z tego względu powinieneś robić kopie zapasowe edytowanych plików i uważnie sprawdzać każdą wprowadzoną modyfikację.

Powłoka `tcsh` używa podczas inicjalizacji dwóch plików. Pierwszy z nich to `csh.login`, który znajduje się w katalogu `/etc`. Jeśli chcesz zmieniać zawarte w nim ustawienia, powinieneś najpierw skopiować go do katalogu domowego. Oto przykładowa zawartość tego pliku:

```
if ($?prompt) then
#this means that the shell is interactive
    umask 022
    set cdpath = ( /usr/spool )
    set notify
    set history = 100
    set histfile = .history
#The savehist variable is set to tell tcsh to
#save the history list to the history file on
#logout. The value of 25 means that tcsh will
#save the last 25 commands to the history file.
    set savehist=25
    setenv OPENWINHOME /usr/openwin
    setenv MANPATH /usr/local/man:/usr/man/preformat:/usr/man:/usr/X11/man:
Σ/usr/openwin/man
setenv MINICOM "-c on"
    setenv HOSTNAME "`cat /etc/HOSTNAME`"
    set path = ( $path /usr/X11/bin /usr/andrew/bin $OPENWINHOME/bin Σ
/usr/games . )
endif
#I had problems with the Backspace key installed by 'tset'
#but you might want to try it anyway, instead of the
#'setenv term .....' below it.
#eval `tset -sQ "$term"`
#setenv term console
if ! $?TERM setenv TERM console
set prompt = "%m:%~%# "
alias ls 'ls -F'
if ( { tty -silent } ) then >& /dev/null
    echo ""; fortune; echo ""
endif
```

Drugim z plików inicjalizacyjnych jest plik `csh.cshrc`, który również znajduje się w katalogu `/etc`. Podobnie jak w poprzednim przypadku, by dokonywać w nim jakichś zmian, powinieneś najpierw skopiować go do katalogu domowego.

Podczas logowania interpreter `tcsh` wykonuje polecenia zapisane w pliku `/etc/csh.cshrc`, a następnie `/etc/csh.login`. Po czym sprawdza, czy posiadasz w katalogu domowym własną kopię pliku `csh.cshrc`. Musi się ona nazywać `.cshrc` lub `.tcshrc`. Jeśli któryś z tych plików istnieje, wykonywane są polecenia w nim zapisane.

Następnie `tcsh` sprawdza, czy w katalogu domowym znajduje się kopia pliku `csh.login`. Musi się ona nazywać `.login`. Jeśli plik istnieje, zostaje wykonywany.

Kiedy uruchamiana jest następna kopia interpretera `tcsh`, odbywa się podobna procedura, z tym że nie są przetwarzane pliki `csh.login` ani `.login`.

Polecenia wewnętrzne interpretera `tcsh` - podsumowanie

W tabeli 13.3 zebrano najczęściej używane polecenia powłoki `tcsh` wraz z ich krótkim opisem.

Tabela 13.3. Najczęściej wykorzystywane polecenia powłoki `tcsh`

Polecenie	Znaczenie
<code>alias</code>	Pozwala definiować i wyświetlać zdefiniowane aliasy
<code>bg</code>	Powoduje, że wstrzymany proces uruchamia się ponownie w tle
<code>bindkey</code>	Pozwala na zmianę reakcji powłoki na kombinacje klawiszy
<code>cd</code>	Przechodzi do katalogu określonego jako argument
<code>exit</code>	Kończy pracę powłoki
<code>fg</code>	Powoduje, że wstrzymany proces uruchamia się ponownie na pierwszym planie
<code>history</code>	Pozwala użytkownikowi na przeglądanie i edycję historii poleceń
<code>kill</code>	Używane do zatrzymywania procesów
<code>logout</code>	Kończy pracę powłoki uruchomionej podczas logowania
<code>set</code>	Służy do ustawiania wartości zmiennych <code>tcsh</code>
<code>source</code>	Odczytuje i wykonuje zawartość pliku – polecenie omówione szerzej w rozdziale 14.
<code>unalias</code>	Używane do usuwania istniejącego aliasu

Zmienne powłoki tcsh

W tabeli 13.4 zebrane zostały najważniejsze zmienne wraz z krótkim opisem ich znaczenia.

Tabela 13.4. Najczęściej wykorzystywane zmienne powłoki tcsh

Nazwa zmiennej	Znaczenie
autocorrect	Jeśli zmienna ta jest zdefiniowana, tcsh będzie automatycznie poprawiał literówki w wierszu poleceń
histfile	Nazwa pliku, w którym przechowywana będzie historia poleceń
history	Rozmiar historii poleceń
home	Katalog domowy bieżącego użytkownika
path	Ścieżka przeszukiwania
prompt	Znak zachęty pierwszego poziomu
prompt2	Znak zachęty drugiego poziomu (wyświetlany, gdy interpreter działa w pętli)
prompt3	Znak zachęty trzeciego poziomu (wyświetlany, gdy interpreter próbuje poprawić literówkę)
savehist	Liczba poleceń, które mają być zapisywane w pliku historii
watch	Zbiór par użytkownik-terminal pozwalających obserwować logowanie i wylogowywanie się użytkowników

Podsumowanie

W ostatnich trzech rozdziałach przedstawiliśmy podstawowe zagadnienia związane z używaniem trzech najpopularniejszych linuxowych interpreterów poleceń powłoki. Naj bogatszym i najbardziej elastycznym z nich jest tcsh, co wcale nie oznacza, że właśnie jego powinieneś używać.

Następny rozdział podejmuje zagadnienia związane z językami programowania wbudowanymi w powłoki. Możesz również przejść do innych rozdziałów.

Używanie edytorów tekstów omówione jest w rozdziale 16. „Edytory tekstu: vi i emacs”.

Drukowanie w systemie Linux jest tematem rozdziału 20. „Drukowanie”.

Rozdział 14.

Programowanie

w języku powłoki

Rick McMullin

W tym rozdziale:

- υ Tworzenie i uruchamianie programów powłoki
- υ Używanie zmiennych
- υ Cudzysłowy
- υ Polecenie `test`
- υ Instrukcje warunkowe
- υ Instrukcje iteracji
- υ Funkcje

Trzy poprzednie rozdziały mówiąły o używaniu najpopularniejszych powłok linuxowych. Teraz przyjrzymy się wbudowanym w nie interpretowanym językom programowania.

Poniższy rozdział omawia podstawy pisania skryptów w języku powłoki, ze zwróceniem uwagi na różnice pomiędzy językami wbudowanymi w poszczególne interpretery. Poruszane zostaną następujące tematy:

- υ tworzenie i uruchamianie programów powłoki,
- υ używanie zmiennych,
- υ cudzysłowy,
- υ polecenie `test`,
- υ instrukcje warunkowe,

- υ instrukcje iteracji,
- υ funkcje.

Każde nowe polecenie przedstawione w tym rozdziale poparte jest krótkim przykładem, który ma na celu ułatwienie zrozumienia zasad jego stosowania.

Tworzenie i uruchamianie programów powłoki

W najprostszym przypadku program powłoki (nazywany dalej również skryptem) to plik, który zawiera jedno lub więcej poleceń powłoki. Takich programów można używać w sposób podobny do aliasów, by oszczędzić sobie częstego wpisywania identycznych sekwencji poleceń. Możliwe jest również pisanie prostych programów wymagających interakcji użytkownika, na przykład instalujących inne programy.

Aby stworzyć skrypt powłoki, powinieneś utworzyć nowy plik (na przykład za pomocą edytora tekstów) i umieścić w nim polecenia, które mają zostać wykonane. Założmy, że w Twoim systemie plików zamontowany jest CD-ROM. Jest on montowany podczas uruchamiania systemu. Jeśli zmienisz dysk CD-ROM podczas pracy, musisz nakazać systemowi odczytanie nowego drzewa katalogów. Można to uzyskać przez odmontowanie, a następnie ponowne zamontowanie CD-ROM-u. Odpowiednia sekwencja poleceń ma postać:

```
umount /dev/cdrom  
mount -t iso9660 /dev/cdrom /cdrom
```

Zamiast wpisywać oba te polecenia, można stworzyć krótki program w języku powłoki, nazwać go np. `nowycd`, i posługiwać się nim.

Istnieje kilka metod wykonywania programu zapisanego w pliku. Jedną z nich jest nadanie plikowi prawa do wykonywania, na przykład następującym poleceniem:

```
chmod +x nowycd
```

Możesz teraz wywołać skrypt `nowycd` poprzez wpisanie jego nazwy w wierszu poleceń.



Plik będący programem powłoki musi znajdować się w którymś z katalogów wchodzących w skład ścieżki przeszukiwania, w przeciwnym przypadku interpreter nie będzie potrafił go odnaleźć. Ponadto jeśli używasz interpretera `tcsh`, pierwszy wiersz musi zaczynać się od znaku `#`.

Innym sposobem wykonania programu zapisanego w pliku jest uruchomienie powłoki, dla której program został napisany, z nazwą pliku podaną jako parametr wywołania, np.

```
tcsh nowycd
```

Trzecią metodą jest użycie polecenia . (kropka) w powłokach `pdksh` i `bash` lub `source` w `tcsch`. Polecenie to powoduje wykonanie poleceń zawartych w pliku podanym jako parametr, np.

```
. nowycd
```

w przypadku `bash` i `pdksh` czy też

```
source nowycd
```

w przypadku `tcsch`.

Oto inna sytuacja, w której prosty skrypt może zaoszczędzić nieco czasu. Założmy, że pracujesz nad trzema plikami w danym katalogu i codziennie robisz ich kopię zapasową na dyskietce, wydając polecenia:

```
mount -t msdos /dev/fd0 /a  
cp plik1 /a  
cp plik2 /a  
cp plik3 /a
```

Latwiejszym sposobem będzie umieszczenie tych poleceń w pliku (o nazwie np. `backup`) i wywoływanie go jako programu powłoki jednym z podanych wcześniej sposobów.



Jeśli chcesz używać pliku `backup` jako programu powłoki, powinieneś przypisać mu prawo do wykonywania i umieścić go w jednym z katalogów wchodzących w skład ścieżki przeszukiwania.

Używanie zmiennych

Tak jak w każdym języku programowania, użycie zmiennych w programach powłoki jest sprawą bardzo ważną. Z niektórymi ze zmiennych powłoki (na przykład `PATH` czy `PS1`) miałeś już okazję się zapoznać, wiesz również, jak można nadać im wartość. W tym podrozdziale dowiesz się, jak można tworzyć własne zmienne i posługiwać się nimi w programach powłoki.

Nadawanie wartości zmiennej

W powłokach `bash` i `pdksh` przypisanie odbywa się poprzez podanie nazwy zmiennej, znaku równości i wartości zmiennej, np.:

```
licznik=5
```

W `tcsch` należy do tego celu użyć polecenia `set`, np.:

```
set licznik=5
```



Jeśli używasz interpretera `bash` lub `pdksh`, upewnij się, że przed i po znaku równości nie występują spacje, ponieważ spowodują one błędную interpretację polecenia. Nie ma to znaczenia w przypadku powłoki `csh`.

Zauważ, że nie zachodzi potrzeba deklarowania zmiennych przed ich użyciem, jak ma to miejsce w takich językach, jak C czy Pascal, ponieważ w języku powłoki nie występuje pojęcie typów. Oznacza to, że tej samej zmiennej można używać zarówno do przechowywania tekstów, jak i liczb całkowitych. Tekst możesz przypisać zmiennej w taki sam sposób, jak liczbę całkowitą, np.:

```
imie=Piotr  
w przypadku powłok bash i pdksh lub  
set imie=Piotr  
w przypadku csh.
```

Odczytywanie wartości zmiennej

Potrafisz już przypisać zmiennej wartość; teraz trzeba ją odczytać. Można to zrobić przez wpisanie nazwy zmiennej poprzedzonej znakiem dolara; aby na przykład wypisać wartość zmiennej `licznik` na ekranie, można wydać polecenie:

```
echo $licznik
```



Jeśli znak dolara zostałby pominięty, na ekranie zostałoby wyświetlane słowo `licznik`.

Parametry pozycyjne i inne zmienne wewnętrzne powłoki

Parametry pozycyjne to specjalne zmienne powłoki, w których przechowywane są parametry wywołania przekazywane do programu. Pierwszy z parametrów programu zapisany jest w zmiennej o nazwie `1`, drugi – `2` itd. Są to nazwy zarezerwowane, nie można więc tworzyć własnych zmiennych o takich identyfikatorach. Dostęp do wartości przechowywanych w tych zmiennych można uzyskać poprzez dodać ich nazwy znakiem `$`, podobnie jak ma to miejsce w przypadku pozostałych zmiennych.

Poniższy przykładowy program wymaga podania dwóch parametrów, które następnie wyświetlane są w odwrotnym porządku:

```
#program odwroc - wyświetla parametry w odwrotnej kolejności  
echo "$2" "$1"
```

Jeśli wywołasz taki program w następujący sposób:

```
odwroc kota ogonem
```

wynik będzie następujący:

```
ogonem kota
```

Istnieje jeszcze kilka zmiennych wewnętrznych powłoki, bez znajomości których trudno się obejść. Tabela 14.1 zawiera najważniejsze z nich wraz z krótkimi opisami.

Tabela 14.1. Wewnętrzne zmienne powłoki

Zmienna	Znaczenie
\$#	Zawiera ilość argumentów przekazanych z wiersza poleceń do programu powłoki
\$?	Zawiera wartość zwróconą przez ostatnio wykonane polecenie lub program
\$0	Zawiera pierwszy wyraz wpisanego polecenia (czyli zazwyczaj nazwę skryptu)
\$*	Zawiera wszystkie argumenty wywołania (\$1 \$2 \$3 ...)
"\$@"	Zawiera wszystkie argumenty wywołania, każdy z osobna ujęty w cudzysłów ("\$1" "\$2" "\$3" ...)

Cudzysłowy

Użycie różnego rodzaju cudzysłów oraz znaku \ (backslash) ma zasadnicze znaczenie dla programowania w języku powłoki. Służą one do ukrywania niektórych symboli przed interpreterem polecień.

Podwójny cudzysłów jest „najsłabszy”. Jeśli tekst jest otoczony pojedynczym cudzysłowem, ignorowane są wszystkie znaki białe w nim zawarte, ale nadal interpretowane są wszystkie pozostałe symbole specjalne. Jeśli na przykład chcesz przypisać tekst `Witam Cie` do zmiennej `pozdrowienie`, powinieneś wydać polecenie:

```
pozdrowienie="Witam Cie" lub  
set pozdrowienie="Witam Cie",
```

zależnie od tego, jakiego interpretera używasz.

Dzięki cudzysłowowi cały tekst `Witam Cie` traktowany jest jako jeden wyraz. Jeśli nie zostanie on otoczony cudzysłowem, `bash` i `pksh` wygenerują komunikat o błędzie, natomiast `tcsh` podstawi tekst `Witam`, ignorując dalszą część wiersza.

„Najsielniejszy” jest cudzysłów pojedynczy ('). Ukrywa on przed interpreterem wszystkie symbole specjalne. Stosowany jest, gdy dany tekst nie jest przeznaczony do interpretacji przez powłokę, ale przez inny program.

Jeśli chciałbyś w zmiennej pozdrowienie zamieścić również identyfikator użytkownika, powinieneś wydać polecenie:

```
pozdrowienie="Witam Cie $LOGNAME"

dla interpreterów bash i pdksh lub

set pozdrowienie="Witam Cie $LOGNAME"

dla tcsh.
```

Spowoduje ono podstawienie do zmiennej pozdrowienie tekstu Witam Cie root (o ile jesteś zalogowany jako root). W przypadku otoczenia tekstu pojedynczym cudzysłowem, nie uzyskałbyś pożądanego efektu, ponieważ wartość zmiennej LOGNAME nie zostałaby podstawiona. Zmienna pozdrowienie zawierałaby tekst Witam Cie \$LOGNAME.



Zmienna LOGNAME to zmienna wewnętrzna powłoki, zawierająca identyfikator użytkownika, który uruchomił daną powłokę.

Trzecią metodą ukrywania symboli specjalnych przed interpreterem poleceń jest użycie symbolu \. Ukrywa on przed interpreterem jeden dowolny znak (ten, który następuje zaraz po symbolu \). Polecenie z pierwszego przykładu może więc być również zapisane jako

```
pozdrowienie=Witam\ Cie (dla powłok bash i pdksh) lub
set pozdrowienie=Witam\ Cie dla powłoki tcsh.
```

Oto inny przykład użycia tego symbolu:

```
cena_dysku=\$5.00 (dla powłok bash i pdksh) lub
set cena_dysku=\$5.00 dla powłoki tcsh.
```

Teraz ukryty przed interpreterem został znak dolara, co zapobiega podstawieniu w miejscu \$5 piątego parametru pozycyjnego (ponieważ jest to nazwa jednej ze zmiennych wewnętrznych). Gdyby nie zastosowano symbolu \, interpreter poleceń w miejsce sekwencji znaków \$5 próbowałby podstawić wartość piątego parametru pozycyjnego, a jeśli parametr taki nie byłby zdefiniowany, wstawiony zostałby ciąg pusty, czyli zmienna cena_dysku otrzymałaaby wartość .00.



W powyższym przykładzie można również użyć pojedynczego cudzysłów.

Odwrócony cudzysłów (`) pojedynczy spełnia nieco inną funkcję. Pozwala on na użycie rezultatu działania jakiegoś polecenia w innym poleceniu. Przykładowo, jeśli chcesz przypisać zmiennej katalog listę plików w bieżącym katalogu, powinieneś wydać polecenie

```
katalog=`ls`
```

```
dla powłoki bash i pdksh lub  
set katalog=`ls`  
dla tcsh.
```

Polecenie to spowoduje wykonanie polecenia `ls`, zapisując wyniki jego działania do zmiennej `katalog`. Jak przekonasz się w podrozdziale dotyczącym instrukcji iteracji, możliwość taka bywa bardzo użyteczna.

Polecenie test

Polecenie `test` używane jest w powłokach `bash` i `pdksh` do warunkowego wykonywania części programu. Stosuje się je zazwyczaj do sprawdzania warunków będących częścią instrukcji warunkowej lub warunków wejścia do albo wyjścia z pętli. Jego składnia jest następująca:

```
test wyrażenie  
lub  
[ wyrażenie ]
```

W poleceniu tym użyć można kilku operatorów. Dzielą się one na cztery grupy: operatory całkowite (działające na liczbach całkowitych), tekstowe, plikowe i logiczne. Operatory całkowite i tekstowe pełnią w zasadzie te same funkcje, różnią się tylko typem danych, na których operują. Zebrane one zostały w tabeli 14.2.

Tabela 14.2. Operatory całkowite polecenia `test`

Operator	Znaczenie
<code>int1 -eq int2</code>	Zwraca wartość logiczną True (prawda), jeśli argument <code>int1</code> jest równy <code>int2</code>
<code>int1 -ge int2</code>	Zwraca wartość logiczną True (prawda), jeśli argument <code>int1</code> jest większy lub równy <code>int2</code>
<code>int1 -gt int2</code>	Zwraca wartość logiczną True (prawda), jeśli argument <code>int1</code> jest większy niż <code>int2</code>
<code>int1 -le int2</code>	Zwraca wartość logiczną True (prawda), jeśli argument <code>int1</code> jest mniejszy lub równy <code>int2</code>
<code>int1 -lt int2</code>	Zwraca wartość logiczną True (prawda), jeśli argument <code>int1</code> jest mniejszy niż <code>int2</code>
<code>int1 -ne int2</code>	Zwraca wartość logiczną True (prawda), jeśli argument <code>int1</code> nie jest równy <code>int2</code>

Tabela 14.3 przedstawia operatory tekstowe (słujące do manipulowania łańcuchami znaków) obsługiwane we wszystkich trzech powłokach.

Tabela 14.3. Operatory tekstowe polecenia test

Operator	Znaczenie
<code>tekst1 = tekst2</code>	Zwraca wartość logiczną True (prawda), jeśli <code>tekst1</code> jest identyczny z <code>tekst2</code>
<code>tekst1 != tekst2</code>	Zwraca wartość logiczną True (prawda), jeśli <code>tekst1</code> nie jest identyczny z <code>tekst2</code>
<code>tekst</code>	Zwraca wartość logiczną True (prawda), jeśli wartość zmiennej <code>tekst</code> jest zdefiniowana
<code>-n tekst</code>	Zwraca wartość logiczną True (prawda), jeśli <code>tekst</code> ma długość większą od 0
<code>-z tekst</code>	Zwraca wartość logiczną True (prawda), jeśli <code>tekst</code> ma długość równą 0

Operatory plikowe używane są np. do sprawdzania, czy plik istnieje, bądź też do sprawdzania, jaki rodzaj pliku został podany jako parametr. Zebrano je w tabeli 14.4.

Tabela 14.4. Operatory plikowe polecenia test

Operator	Znaczenie
<code>-d nazwapliku</code>	Zwraca wartość logiczną True (prawda), jeśli <code>nazwapliku</code> odnosi się do jest katalogu
<code>-f nazwapliku</code>	Zwraca wartość logiczną True (prawda), jeśli plik <code>nazwapliku</code> jest zwykłym plikiem
<code>-r nazwapliku</code>	Zwraca wartość logiczną True (prawda), jeśli plik <code>nazwapliku</code> może być czytany przez proces
<code>-s nazwapliku</code>	Zwraca wartość logiczną True (prawda), jeśli plik <code>nazwapliku</code> ma długość większą od 0
<code>-w nazwapliku</code>	Zwraca wartość logiczną True (prawda), jeśli plik <code>nazwapliku</code> może być zapisywany przez proces
<code>-x nazwapliku</code>	Zwraca wartość logiczną True (prawda), jeśli plik <code>nazwapliku</code> jest wykonywalny

Operatory logiczne używane są do łączenia dwóch lub więcej operatorów różnego typu lub do zaprzeczenia pojedynczego operatora. Tabela 14.5 zawiera dostępne operatory logiczne.

Tabela 14.5. Operatory logiczne polecenia test

Operator	Znaczenie
<code>! wyrażenie</code>	Zwraca wartość logiczną True (prawda), jeśli wyrażenie nie jest prawdziwe
<code>wyr1 -a wyr2</code>	Zwraca wartość logiczną True (prawda), jeśli oba wyrażenia <code>wyr1</code> i <code>wyr2</code> są prawdziwe
<code>wyr1 -o wyr2</code>	Zwraca wartość logiczną True (prawda), jeśli któryś z wyrażeń <code>wyr1</code> lub <code>wyr2</code> jest prawdziwe

Odpowiedniki polecenia test w powłoce tcsh

W powłoce `tcsh` nie jest dostępne polecenie `test`. Wszystkie wymienione wyżej funkcje obsługiwane są za pomocą wyrażeń, w których można stosować operatory prawie identyczne jak te występujące w języku C. Wyrażenia są najczęściej używane z instrukcjami `if` oraz `while`, o których będzie mowa w podrozdziałach „Instrukcje warunkowe” oraz „Instrukcje iteracyjne”.

`tcsh` obsługuje takie same typy operatorów, jak powłoki `bash` i `pdksh`: operatory całkowite, tekstowe, plikowe i logiczne. Wraz z krótkimi wyjaśnieniami zostały one przedstawione w poniższych tabelach.

Tabela 14.6. Operatory całkowite powłoki `tcsh`

Operator	Znaczenie
<code>int1 >= int2</code>	Zwraca wartość logiczną True (prawda), jeśli liczba <code>int1</code> jest większa lub równa <code>int2</code>
<code>int1 > int2</code>	Zwraca wartość logiczną True (prawda), jeśli liczba <code>int1</code> jest większa niż <code>int2</code>
<code>int1 <= int2</code>	Zwraca wartość logiczną True (prawda), jeśli liczba <code>int1</code> jest mniejsza lub równa <code>int2</code>
<code>int1 < int2</code>	Zwraca wartość logiczną True (prawda), jeśli liczba <code>int1</code> jest mniejsza niż <code>int2</code>

Tabela 14.7. Operatory tekstowe obsługiwane w powłoce `tcsh`

Operator	Znaczenie
<code>tekst1 == tekst2</code>	Zwraca wartość logiczną True (prawda), jeśli <code>tekst1</code> jest taki sam jak <code>tekst2</code>
<code>tekst1 != tekst2</code>	Zwraca wartość logiczną True (prawda) jeśli <code>tekst1</code> nie jest identyczny z <code>tekst2</code>

Tabela 14.8. Operatory plikowe obsługiwane w powłoce `tcsh`

Operator	Znaczenie
<code>-r nazwapliku</code>	Zwraca wartość logiczną True (prawda), jeśli plik <code>nazwapliku</code> można odczytywać
<code>-w nazwapliku</code>	Zwraca wartość logiczną True (prawda), jeśli możliwy jest zapis do pliku <code>nazwapliku</code>
<code>-x nazwapliku</code>	Zwraca wartość logiczną True (prawda), jeśli plik <code>nazwapliku</code> może być wykonywany
<code>-e nazwapliku</code>	Zwraca wartość logiczną True (prawda), jeśli plik <code>nazwapliku</code> istnieje

cd. na następnej stronie

Tabela 14.8. cd. Operatory plikowe obsługiwane w powłoce tcsh

Operator	Znaczenie
-o nazwapliku	Zwraca wartość logiczną True (prawda), jeśli właścicielem pliku nazwapliku jest bieżący użytkownik
-z nazwapliku	Zwraca wartość logiczną True (prawda), jeśli plik nazwapliku ma długość większą od 0
-f nazwapliku	Zwraca wartość logiczną True (prawda), jeśli plik nazwapliku jest zwykłym plikiem
-d nazwapliku	Zwraca wartość logiczną True (prawda), jeśli nazwa nazwapliku odnosi się do katalogu

Tabela 14.9. Operatory logiczne obsługiwane w powłoce tcsh

Operator	Znaczenie
! wyrażenie	Zwraca wartość logiczną True (prawda), jeśli wyrażenie nie jest prawdziwe
wyr1 && wyr2	Zwraca wartość logiczną True (prawda), jeśli oba wyrażenia wyr1 i wyr2 są prawdziwe
wyr1 wyr2	Zwraca wartość logiczną True (prawda), jeśli któryś z wyrażeń wyr1 lub wyr2 jest prawdziwe

Instrukcje warunkowe

Każdy z omawianych interpreterów obsługuje dwa rodzaje instrukcji warunkowych: `if` oraz `case`. Są one używane do wykonywania części programu w zależności od tego, czy pewne warunki są spełnione, czy też nie. Ich składnia różni się w zależności od powłoki.

Polecenie if

Wyrażenie typu `if-then-else` obsługiwane jest we wszystkich trzech powłokach. Umożliwia ono sprawdzanie w programach powłoki nawet bardzo skomplikowanych warunków. Jego składnia jest taka sama w powłokach `bash` i `tcsh`:

```
if [ warunek ]
then
    polecenia
elif [ warunek2 ]
then
    polecenia
else
    polecenia
fi
```



Wyrażenia `elif` i `else` są opcjonalne. Powłoki `bash` i `pdksh` w większości wyrażeń złożonych do zasygnalizowania końca wyrażenia używają jego odwróconej nazwy. Koniec wyrażenia `if` sygnalizowany jest więc słowem `fi`.

Instrukcja `elif` to skrót od `else if` (w przeciwnym przypadku, jeżeli...). Jest ona wykonywana tylko wtedy, jeśli żaden z warunków w poprzedzających ją poleceńach `if` i `elif` nie był spełniony. Polecenia zawarte w bloku po instrukcji `else` zostaną wykonane w przypadku, gdy żaden z warunków w poprzedzającym ją poleceniu `if` i poleceńach `elif` nie był spełniony.

W powłoce `tcsh` polecenie `if` może mieć dwie różne formy. Pierwsza z nich jest odpowiednikiem składni używanej w pozostałych powłokach:

```
if (warunek1) then
    polecenia
else if (warunek2) then
    polecenia
else
    polecenia
endif
```

Druga forma jest formą uproszczoną, pozwalającą na sprawdzenie tylko jednego warunku. Jeśli jest on prawdziwy, wykonywane jest pojedyncze polecenie, w przeciwnym przypadku nic się nie dzieje. Składnia tej formy polecenia `if` jest następująca:

```
if (warunek) polecenie
```

Zaoszczędza ona programistę nieco pisania w przypadku prostych instrukcji warunkowych.

Oto przykład użycia instrukcji warunkowej `if` w powłokach `bash` i `pdksh`. Sprawdzona, czy w bieżącym katalogu znajduje się plik o nazwie `.profile`.

```
if [ -f .profile ]
then
    echo "Plik .profile znaleziony w bieżącym katalogu."
else
    echo "Nie znaleziono pliku .profile."
fi
```

Ten sam przykład napisany dla interpretera `tcsh` wygląda następująco:

```
# if ( { -f profile } ) then
#     echo "Plik .profile znaleziony w bieżącym katalogu."
else
    echo "Nie znaleziono pliku .profile."
endif
```



Zauważ, że w przypadku skryptu `tcsh` pierwszy wiersz zaczyna się od znaku `#`. Znak ten jest wymagany, by interpreter `tcsh` rozpoznał plik jako program powłoki.

Polecenie case

Polecenie `case` pozwala na sprawdzenie, czy wartość zmiennej pokrywa się z jednym z kilku wzorców, i wykonanie jednego z bloków kodu w zależności od tego, z którym z nich się zgadza. Jest ono nieco bardziej zaawansowane w porównaniu ze swoimi odpowiednikami w językach C i Pascal, ponieważ pozwala na używanie w charakterze wzorca tekstów, włącznie z symbolami specjalnymi, podczas gdy w wymienionych wcześniej językach można używać tylko typów całkowitych i wyliczeniowych.

Również w przypadku tego polecenia składnia dla powłok `bash` i `pdksh` jest jednakowa, a inna dla `tcsh`. Dla interpreterów `bash` i `pdksh` jest ona następująca:

```
case tekst in
    tekst1)
        polecenia;;
    tekst2)
        polecenia;;
    *)
        polecenia;;
esac
```

Wartość zmiennej `tekst` jest porównywana z wzorcami `tekst1` i `tekst2`. Jeśli porównanie w którymś przypadku wypadnie pomyślnie, wykonywane są polecenia aż do podwójnego średnika. Jeśli `tekst` nie zgadza się ani z `tekst1`, ani z `tekst2`, wykonane zostaną polecenia w bloku za symbolem `*`. Jest to przypadek domyślny, gwiazdka symbolizuje dowolny tekst.

Odpowiednikiem polecenia `case` w interpreterze `tcsh` jest polecenie `switch`. Jego składnia jest podobna, jak w języku C:

```
switch (tekst)
    case tekst1:
        polecenia
    breaksw
    case tekst2:
        polecenia
    breaksw
    default:
        polecenia
    breaksw
endsw
```

Mechanizm działania tego polecenia jest taki sam, jak polecenia `case` w powłokach `bash` i `pdksh`. Każdy tekst zapisany po słowie kluczowym `case` jest porównywany z wartością zmiennej `tekst`. Jeśli któreś porównanie wypadnie pomyślnie, wykonywany jest odpowiedni blok kodu, aż do napotkania słowa kluczowego `breaksw`. W przeciwnym przypadku wykonany zostanie blok instrukcji po słowie kluczowym `default`.

Poniżej zamieszczamy przykład użycia instrukcji `case` w interpreterach `bash` i `pdksh`. Sprawdza on, czy pierwszym argumentem przekazanym wierszu poleceń jest `-i` lub `-e`. W pierwszym przypadku zlicza ilość wierszy pliku o nazwie podanej jako drugi parametr, zaczynających się na literę `i`. Jeśli pierwszym argumentem jest `-e`, program zlicza wiersze pliku podanego jako drugi argument, które zaczynają się na literę `e`. Jeśli pierwszym argumentem nie jest ani `-i`, ani `-e`, wówczas wyświetlany jest krótki komunikat o błędzie.

```

case $1 in
  -i)
    licz=`grep ^i $2 | wc -l`
    echo "Ilosc wierszy zaczynajacych sie na i w pliku $2 wynosi
$licz"
    ;;
  -e)
    licz=`grep ^e $2 | wc -l`
    echo "Ilosc wierszy zaczynajacych sie na e w pliku $2 wynosi
$licz"
    ;;
  *)
    echo "Nie rozpoznano opcji"
    ;;
esac

```

Ten sam przykład dla powłoki `tcsh` ma postać:

```

# Pierwszy wiersz musi zaczynać się od symbolu #
switch ( $1 )
  case -i | i:
    set licz = `grep ^i $2 | wc -l`
    echo "Ilosc wierszy zaczynajacych sie na i w pliku $2 wynosi
$licz"
    breaksw
  case -e | e:
    set licz = `grep ^e $2 | wc -l`
    echo "Ilosc wierszy zaczynajacych sie na e w pliku $2 wynosi
$licz"
    breaksw
  default:
    echo " Nie rozpoznano opcji"
    breaksw
endsw

```

Instrukcje iteracyjne

Język powłoki udostępnia również kilka metod zapętlenia programu. Najczęściej używa się do tego celu pętli `for`. Oprócz niej istnieją również inne pętle (jak `while` oraz `until`), ale nie różnią się one zasadniczo od pętli `for`.

Pętla for

Pętla `for` pozwala na kilkukrotne wykonanie grupy poleceń. `bash` i `pdksh` obsługują dwa warianty instrukcji `for`.

Pierwszy z nich ma następującą składnię:

```

for zm1 in lista
do
    polecienia
done

```

W tej formie polecenie `for` wykonywane jest dla każdej wartości z listy. Lista może być zmienną zawierającą słowa oddzielone spacjami albo też bezpośrednio podaną listą wartości. Podeczas każdego przebiegu pętli zmiennej `zm1` przypisywana jest kolejna wartość z listy, aż osiągnięty zostanie jej koniec.

Druga forma polecenia `for` ma składnię:

```
for zm1
do
    polecenia
done
```

W tym przypadku pętla wykonywana jest dla każdego wyrazu zawartego w zmiennej `zm1`. Gdy użyta jest ta forma, interpreter przyjmuje, że zmienna `zm1` zawiera wszystkie parametry pozycyjne, które zostały podane w wierszu poleceń. Zazwyczaj jest to równorównoznaczne z poniższym wyrażeniem:

```
for zm1 in "$@"
do
    polecenia
done
```

Odpowiednikiem takiej pętli w powłoce `tcsh` jest pętla `foreach`. Ma ona składnię:

```
foreach nazwa (lista)
    polecenia
end
```

Poniżej podano przykład zastosowania pętli `for` dla powłok `bash` i `pdksh`. Pobiera on jako parametry wywołania dowolną liczbę nazw plików tekstowych. Następnie wczytuje każdy z tych plików, zamienia w nim wszystkie małe litery na wielkie, a wynik zapisuje w pliku o nazwie takiej, jaką miał plik źródłowy, ale z rozszerzeniem `.caps`.

```
for plik
do
    tr a-z A-Z < $plik >$plik.caps
done
```

Ten sam przykład dla interpretera `tcsh` ma postać:

```
# foreach plik ($*)
    tr a-z A-Z < $plik > $plik.caps
end
```

Pętla while

Pętla `while` to kolejna instrukcja iteracyjna dostępna w językach powłoki. Powoduje ona wykonywanie bloku programu tak długo, dopóki określony warunek jest prawdziwy. Jej składnia w interpreterach `bash` i `pdksh` jest następująca:

```
while wyrazenie
do
    polecenia
done
```

W interpreterze `tcs` należy stosować składnię:

```
while (wyrazenie)
      polecenia
end
```

Przykładem zastosowania tej pętli może być program, który wypisuje na ekranie wszystkie argumenty wywołania, wraz z ich numerami. Oto wersja takiego programu dla `bash` i `pdksh`:

```
licz=1
while [ -n "$*" ]
do
    echo "Parametr numer $licz: $1"
    shift
    licz=`expr $licz + 1`
done
```

Jak przekonasz się czytając podrozdział „Polecenie `shift`”, polecenie `shift` służy do przesuwania parametrów przekazanych z wiersza poleceń o jeden w prawo.

Wersja tego programu dla powłoki `tcs` ma postać:

```
#
set licz = 1
while ( "$*" != "" )
    echo "Parametr numer $licz: $1"
    shift
    set licz=`expr $licz + 1`
end
```

Pętla `until`

Pętla `until` jest bardzo podobna do pętli `while`, z tym że zostaje przerwana w momencie, gdy warunek jest spełniony (czyli dokładnie na odwrót niż w przypadku pętli `while`). W powłokach `bash` i `pdksh` składnia instrukcji `until` ma postać:

```
until wyrazenie
do
    polecienia
done
```

Przykład podany dla pętli `while` można przepisać, używając pętli `until`, negując tylko warunek:

```
licz=1
until [ -z "$*" ]
do
    echo "Parametr numer $licz: $1"
    shift
    licz=`expr $licz + 1`
done
```

Jedyną różnicą jest fakt, że z polecenia `test` usunięta została opcja `-n` (która sprawdzała, czy tekst ma długość różną od zera) i zastąpiona opcją `-z`, która sprawdza, czy tekst ma zerową długość.

W praktyce pętla ta nie ma wielkiego znaczenia, ponieważ każda pętla `until` może zostać zapisana za pomocą instrukcji `while`. Z tego również powodu w powłoce `tcsch` nie ma jej odpowiednika.

Polecenie shift

Wszystkie trzy omawiane powłoki obsługują polecenie `shift`, które przesuwa aktualne wartości parametrów pozycyjnych o jeden w lewo. Przykładowo, jeśli aktualnymi wartościami parametrów pozycyjnych są:

```
$1=-r $2=plik1 $3=plik2
```

to po wykonaniu polecenia `shift` będą one miały wartości:

```
$1=plik1 $2=plik2
```

Podając po poleceniu `shift` liczbę, można również przesunąć parametry o więcej niż jedną pozycję; na przykład by przesunąć parametry pozycyjne o dwie pozycje, należy wydać polecenie:

```
shift 2
```

Polecenie `shift` jest bardzo użyteczne, gdy wykorzystujesz w swoim programie kilka argumentów wywołania, na przykład różne opcje. Ponieważ zwykle są one przetwarzane w jakiś rodzaju pętli, wygodnie jest po obsłudzeniu pierwszego z nich przesunąć wszystkie w lewo. Przykładowy skrypt spodziewa się dwóch opcji podanych w wierszu poleceń, po których następują odpowiednio nazwy pliku wejściowego i wyjściowego. Wczytuje następnie plik wejściowy, w którym zamienia wszystkie litery na wielkie, a wynik zapisuje w pliku wyjściowym. Skrypt działa w interpreterach `bash` i `pdksh`.

```
while [ "$1" ]
do
    if [ "$1" = "-i" ] then
        plikwe="$2"
        shift 2
    elif [ "$1"="-o" ]
    then
        plikwy="$2"
        shift 2
    else
        echo "Program $0 nie rozpoznaje opcji $1"
    fi
done
tr a-z A-Z $plikwe $plikwy
```

Polecenie select

Powłoka pdksh udostępnia bardzo użyteczne polecenie, którego nie obsługuje ani bash, ani tcsh. Jest to polecenie select. Od innych pętli różni się tym, że nie wykonuje bloku poleceń w zależności od tego, czy warunek jest spełniony, czy nie, ale służy głównie do generowania prostych menu tekstowych. Jego składnia jest następująca:

```
select pozycjamenu [in lista_pozycji]
do
    polecenia
done
```

Nawiązy kwadratowe oznaczają część opcjonalną.

Wykonując polecenie select, pdksh generuje numerowane menu dla wszystkich elementów w liście lista_pozycji. lista_pozycji może być zmienną zawierającą więcej niż jedną pozycję menu, np. poz1 poz2, ale może być również wpisana bezpośrednio, na przykład:

```
select pozycjamenu in poz1 poz2 poz3
```

Jeśli lista_pozycji nie jest podana, polecenie select używa parametrów pozycyjnych, podobnie jak polecenie for.

Po wybraniu jednej z pozycji menu (przez wpisanie odpowiedniej liczby), polecenie select zapisuje informację o wybranym elemencie w zmiennej pozycjamenu. Polecenia zawarte w bloku do – done mogą następnie wykonać odpowiednie operacje.

Poniższy przykład ilustruje wykorzystanie polecenia select. Wyświetla on trzy pozycje menu, a po tym, jak użytkownik wybierze jedną z nich, pyta o potwierdzenie. Jeśli użytkownik odpowie coś innego, niż t lub T, menu wyświetlane jest ponownie.

```
select pozmenu in poz1 poz2 poz3
do
    echo "Czy jesteś pewny, że chcesz wybrać $pozmenu?"
    read odp
    if [ $odp = "T" -o $odp =  "t" ]
    then
        break
    fi
done
```

W powyższym przykładzie wprowadzono kilka nowych poleceń. Polecenie read służy do pobierania informacji od użytkownika. Zapisuje to, co wpisze użytkownik, do zmiennej podanej jako jego parametr. Polecenie break służy do opuszczania pętli while, until, repeat, select lub for.

Instrukcja repeat

tcsh obsługuje instrukcję nie mającą odpowiednika ani w powłoce pdksh, ani bash. Jest to instrukcja repeat. Wykonuje ona pojedyncze polecenie zadaną ilość razy. Jej składnia jest następująca:

```
repeat ilość polecenie
```

Poniżej podano przykład wykorzystania instrukcji `repeat`. Pobiera on zestaw liczb z wiersza poleceń, po czym wyświetla odpowiadającą każdej liczbie ilość kropek.

```
#  
foreach num ($*)  
    repeat $num echo -n ".."  
    echo ""  
end
```



Każda pętla `repeat` może być zastąpiona pętlą `while` lub `for`. Czasem jednak składnia polecenia `repeat` jest wygodniejsza.

Funkcje

Języki powłoki pozwalają również definiować własne funkcje. Funkcje takie zachowują się podobnie jak definiowane w języku C czy innych językach programowania. Głównym powodem ich stosowania jest lepsza organizacja kodu programów. Kod pisany z wykorzystaniem funkcji jest o wiele bardziej czytelny, często również jest krótszy, ponieważ powtarzające się instrukcje można zebrać w jednej funkcji, wywoływanej w miejscu, w której miałyby się one pojawić. Powłoka `tcsh` nie obsługuje funkcji.

Składnia polecenia definiującego funkcję w interpreterach `bash` i `pdksh` jest następująca:

```
nazwafunkcji () {  
    polecenia  
}
```

`pdksh` obsługuje również składnię:

```
function nazwafunkcji {  
    polecenia  
}
```

Obie formy są identyczne pod względem działania.

Po zdefiniowaniu funkcji można ją wywołać poleceniem:

```
nazwafunkcji [param1 param2 param3 ...]
```

Zauważ, że do funkcji przekazać można dowolną ilość parametrów. Wewnątrz funkcji są one widziane jako parametry pozycyjne, tak, jakby zostały wprowadzone jako argumenty wywołania. Poniższy program powłoki zawiera kilka funkcji, z których każda obsługuje jedną z opcji podawanych w wierszu poleceń. Odczytuje on wszystkie pliki, których nazwy podano w wierszu polecień, a następnie, w zależności od podanej opcji, zamienia wszystkie litery w pliku na wielkie, małe lub też drukuje poszczególne pliki.

```

upper () {
    shift
    for i
    do
        tr a-z A-Z <$1 >$1.out
        rm $1
        mv $1.out $1
        shift
    done; }

lower () {
    shift
    for i
    do
        tr A-Z a-z <$1 >$1.out
        rm $1
        mv $1.out $1
        shift
    done; }
print () {
    shift
    for i
    do
        lpr $1
        shift
    done; }

blad () {
    echo "Skladnia polecenia $1: $1 <opcja> <pliki_wejsciowe>"
    echo ""
    echo "opcja moze przyjmowac nastepujace wartosci"
    echo "p      drukowanie plikow"
    echo "u      zamiana na wielkie litery"
    echo "l      zamiana na male litery"
case $1
in
    p | -p) print $@;;
    u | -u) upper $@;;
    l | -l) lower $@;;
    *)      blad $0;;
esac

```

Podsumowanie

W tym rozdziale przedstawiliśmy wiele aspektów programowania w języku powłoki. Zaawansowani użytkownicy Linuxa używają tego języka bardzo często.

Choć języki powłoki są dość potężne i łatwe do opanowania, zdarzają się problemy, do których rozwiązywania skrypty powłoki po prostu się nie nadają. Wtedy prawdopodobnie będziesz chciał użyć jednego z innych języków programowania dostępnych dla Linuxa.

Język `awk`, który jest bardzo przydatny przy wyszukiwaniu tekstów pasujących do wzorców oraz przetwarzaniu dużych zbiorów danych liczbowych, omówiony jest w rozdziale 25. „gawk”.

Perl, używany głównie do pisania szybkich skryptów oraz obsługi stron WWW, jest opisany w rozdziale 28. „Perl”.

Język Smalltalk/X, w którym tworzyć można aplikacje zorientowane obiektowo działające w środowisku X Window, przedstawiony jest w rozdziale 31. „Smalltalk/X”.

Rozdział 15.

FTP oraz Telnet

Tim Parker

W tym rozdziale:

- υ FTP
- υ Konfiguracja serwera FTP
- υ Obsługa FTP
- υ Protokół TFTP (Trivial file transfer protocol)
- υ Używanie programu Telnet

FTP oraz Telnet to dwa najbardziej użyteczne narzędzia służące do komunikacji pomiędzy komputerami UNIX-owymi lub linuxowymi (a także innymi systemami). FTP służy do przesyłania plików, natomiast Telnet do logowania się do innych systemów. W tym rozdziale omówimy metody używania obu tych narzędzi, wspomnimy również o protokole TFTP – uproszczonej wersji FTP.

FTP

Są dwa sposoby używania FTP oraz TFTP: z wiersza poleceń lub poprzez interfejs graficzny. Większość systemów linuxowych ograniczona jest do wiersza poleceń, chyba że zainstalowany jest program komercyjny czy shareware'owy udostępniający interfejs graficzny. Programy oparte na GUI są przeważnie spotykane w systemach Windows. Skoncentrujemy się więc na najczęściej używanej w systemach linuxowych wersji tekstuowej tych programów.

FTP (File Transfer Protocol, protokół transmisji plików) został zaprojektowany do przesyłania danych bez konieczności używania tak złożonych protokołów komunikacyjnych jak XMODEM, YMODEM, ZMODEM czy Kermit, jak również bez konieczności pełnego logowania się do systemu zdalnego. FTP zapewnia szybkie logowanie się, możliwość poruszania się po systemie plików (ograniczonego oczywiście prawami dostępu), oraz transmisji plików do i z systemu zdalnego.

Aby używać FTP, komputer, na którym pracujesz (klient), musi posiadać oprogramowanie klienta FTP. Jest ono jednym ze składników standardowego oprogramowania w systemach UNIX-owych, nie powinieneś więc mieć problemów ze zdobyciem go. Na drugim końcu połączenia – komputerze, z którego chcesz pobrać pliki (serwerze) – musi działać program, który obsługuje przychodzące polecenia FTP. Jest on nazywany serwerem FTP i musi działać bez przerwy, jeśli połączenia mają funkcjonować poprawnie.

Serwer FTP również jest zwykle rozprowadzany wraz z wielozadaniowymi systemami operacyjnymi, takimi jak Linux. Przeważnie nazywa się on `ftpd` (od ang. *ftp daemon*; nazwa „daemon” odnosi się do programu, który cały czas działa w tle). Jego uruchomienie następuje podczas ładowania systemu, chyba że administrator go wyłączy.

Jeśli próbujesz się połączyć z komputerem klasy PC (z systemem OS/2, Windows, NetWare, itp.), wówczas są małe szanse, by działał na nim serwer FTP. Spowodowane jest to faktem, że żaden z tych systemów nie obsługuje domyślnie protokołu TCP/IP. Administrator musi ręcznie go zainstalować. Jeśli na komputerze zdalnym nie działa program serwera FTP, nie można się z nim połączyć za pomocą programu klienta.

Konfiguracja serwera FTP

Zanim przejdziemy do łączenia się z innym systemem i przenoszenia plików, powinniśmy powiedzieć kilka słów o konfiguracji serwera FTP. Każdy komputer, który ma odpowiadać na polecenia FTP, musi być skonfigurowany jako serwer FTP. Nie jest to procedura skomplikowana.

Wszystkie programy, których będziesz potrzebował, wchodzą w skład dystrybucji Linuksa. Nie ma znaczenia to, czy chcesz udostępnić zasoby wszystkim logującym się, czy też ograniczać dostęp w zależności od hasła. Podstawowe czynności, które trzeba wykonać, są takie same.

Konfiguracja serwera FTP zaczyna się od wybrania nazwy węzła FTP. Nie jest ona niezbędna, ale ułatwi w przyszłości użytkownikom innych systemów odnalezienie Twojego systemu. Nazwy te mają zwykle format

`ftp.nazwa_domeny.typ_domeny`

`typ_domeny` to standardowe rozszerzenie DNS. Na przykład, jeśli węzeł FTP nazywa się
`ftp.tpci.com`

jest oczywiste, że jest to serwer FTP obsługujący domenę `tpci.com`.

Następnie należy uruchomić program rezydencyjny `ftpd`. Zwykle robi to automatycznie proces `inetd`, który obserwuje port poleceń TCP (kanał 21), rozpoznając nadchodzące żądania połączenia. Po ich wykryciu uruchamia program `ftpd`.



Upewnij się, że `ftpd` może zostać uruchomiony w razie potrzeby przez proces `inetd`, sprawdzając plik konfiguracyjny `/etc/inetd.config`. Powinien znajdować się w nim wiersz:

```
ftp stream tcp nowait rot /etc/ftpd  
ftpd
```

Jeśli go nie ma, należy go dopisać. W większości systemów linuxowych i UNIX-owych wiersz taki istnieje, ale może być zaznaczony jako komentarz - trzeba wtedy tylko usunąć odpowiedni symbol.

Wpis `ftp` w pliku konfiguracyjnym `inetd` powoduje, że za każdym razem, gdy nawiązywane jest nowe połączenie z portem FTP, uruchamiany jest program `ftpd`. Może on być również uruchamiany z opcją `-1`, która załącza rejestrację połączeń. Opcję tę należy jednak stosować ostrożnie, ponieważ pliki rejestracji rosną bardzo szybko.

Jeśli zamierzasz skonfigurować serwer FTP tak, by każdy użytkownik próbujący połączyć się z systemem posiadał własny identyfikator oraz hasło, musisz dla każdego z nich utworzyć nowe konto, tak jakby byli bezpośrednimi użytkownikami. Jeśli serwer ma być dostępny dla wszystkich, również należy utworzyć nowe konto. Nazywa się ono zwykle `anonymous`, `ftp` lub `guest`. Musisz również wybrać katalog domowy, który powinien być oddzielony od reszty systemu. Przykładowy wpis w pliku `/etc/passwd` może wyglądać tak:

```
ftp:*:400:51:Anonymous FTP access:/usr/ftp:/bin/false
```

Gwiazdka zamiast hasła powoduje, że nikt nie ma dostępu do tego konta. Identyfikator użytkownika (`400`) nie może pokrywać się z żadnym innym identyfikatorem w systemie. Identyfikator grupy (`51`) decyduje o tym, do której grupy użytkownik zostanie przypisany po zalogowaniu się.

Dla zapewnienia bezpieczeństwa dobrym pomysłem jest stworzenie osobnej grupy tylko dla użytkownika `ftp` (należy wprowadzić w tym celu odpowiednio zmiany do pliku `/etc/group`). Katalogiem domowym w powyższym przykładzie jest `/usr/ftp`, ale można wybrać dowolny inny katalog (powinien on, ze względów bezpieczeństwa, należeć do użytkownika `ftp`). Programem uruchamiającym się po zalogowaniu jest `/bin/false`, co również ma na celu zapewnienie większego bezpieczeństwa.

Obsługa FTP

Bez względu na to, czy pracujesz w systemie linuxowym, UNIX-owym, Windows czy też Macintosh, kiedy chcesz używać FTP, uruchamiasz program klienta FTP, podajesz nazwę węzła, z którym chcesz się połączyć, a następnie czekasz na nawiązanie połączenia. Po połączeniu możesz rozpocząć transfer plików.

Programy FTP oparte na interfejsie tekstowym zwykle uruchamiane są z nazwą lub numerem IP komputera, z którym chcesz się połączyć. Te oparte na GUI wyświetlają naj-

pierw okienko, z którego możesz wybrać polecenie Connect lub nazwę serwera FTP. Jeśli używasz nazwy komputera, system musi potrafić zamienić ją na numer IP.

Kiedy zostanie nawiązane połączenie FTP, trzeba się zalogować. Niektóre systemy umożliwiają logowanie anonimowe przy użyciu identyfikatorów `anonymous` lub `guest`. W większych sieciach, w których funkcjonują takie systemy jak Yellow Pages czy Network Information Services, identyfikator użytkownika jest ważny na każdym komputerze w sieci. Jeśli tak nie jest, Twój identyfikator musi być umieszczony w odpowiednim pliku w komputerze, z którym zamierzasz się połączyć (chyba że logujesz się anonimowo). Można użyć innego identyfikatora, niż tego, którego używasz w swoim systemie. By przenosić pliki pomiędzy systemami, musisz posiadać odpowiednie prawa dostępu w obu systemach.

Pamiętaj, że po zalogowaniu się do innego komputera przy użyciu FTP nie „przenosisz” się na ten komputer. Wszystkie polecenia odnoszą się do komputera, którego używasz. Odwrotną sytuację mamy w przypadku programu Telnet, co jest nieco mylące dla nie-doświadczonych użytkowników.



Pamiętaj, że wszystkie odniesienia do plików i katalogów dotyczą komputera, który zainicjował sesję FTP. Jeśli nie będziesz ostrożny, łatwo możesz pozbyć się zawartości swoich plików.

Połączenia FTP

Aby połączyć się z innym komputerem, musisz znać identyfikator użytkownika i hasło w tym komputerze, a w Twoim systemie musi działać program klienta FTP. To, co pojawią się będzie na ekranie po nawiązaniu połączenia, zależy od wersji oprogramowania i systemu operacyjnego serwera.

W systemach linuxowych, UNIX-owych i większości DOS-owych można uruchomić program FTP z nazwą komputera, z którym chcesz się połączyć, lub jego numerem IP. Po zalogowaniu się jesteś już połączony.

Polecenia FTP

Po połączeniu się z serwerem FTP, będziesz chciał poruszać się po jego systemie plików i przesyłać je. Dla użytkownika FTP dostępnych jest wiele poleceń, najczęściej używane zebrane są w tabeli 15.1. Programy z interfejsem graficznym posiadają odpowiednie elementy menu.

Podstawowymi poleceniami służącymi do przesyłania plików są polecenia `get` (pobierz) i `put` (wyślij). Pamiętaj o tym, że polecenia te są odniesione do komputera klienta, czyli polecenie `get` przenosi plik z serwera na Twój komputer, a `put` – z Twojego komputera na serwer.

Rozpatrzmy następujący przykład:

```
get autoexec.bat  
705 bytes received in 0.1 seconds (0.0 kbytes/s)
```

Tabela 15.1. Polecenia dostępne dla użytkownika FTP

Polecenie	Opis
ascii	Włączenie tekstowego trybu transmisji plików
binary	Włączenie binarnego trybu transmisji plików
cd	Zmiana katalogu na serwerze
close	Zakończenie połączenia
del	Usunięcie pliku na serwerze
dir	Wyświetlenie zawartości katalogu serwera
get	Pobranie pliku z serwera
hash	Wyświetlenie znaku # po przesłaniu każdego bloku
help	Wyświetlenie pomocy
lcd	Zmiana katalogu lokalnego
mget	Pobranie grupy plików z serwera
mput	Wysłanie grupy plików do serwera
open	Otwarcie połączenia
put	Wysłanie pliku do serwera
pwd	Wyświetlenie bieżącego katalogu na serwerze
quote	Bezpośrednie przesłanie polecenia wewnętrznego protokołu FTP
quit	Zakończenie sesji FTP

Użytkownik zalogowany do serwera FTP wydał polecenie `get`, by pobrać z serwera (pracującego w systemie Windows) plik `autoexec.bat`. Jak widać, klient FTP podaje informacje o przebiegu wykonania polecenia.

Polecenia `mget` i `mput` są podobne do `get` i `put`, ale dotyczą grup plików. Przykładowo polecenie

```
mget config.*
```

pobierze z serwera wszystkie pliki, których nazwa pasuje do wzorca `config.*`. Przed przesaniem każdego pliku zostaniesz poproszony o potwierdzenie.

Po systemie plików serwera poruszać się można używając poleceń `cd` i `pwd`. Pamiętaj, że w systemach UNIX-owych katalogi oznacza się symbolem `/`, a w systemach DOS-owych - `\`.

Jeśli nie posiadasz uprawnień do wykonania jakiegoś polecenia, na przykład prawa do czytania pliku przy próbie jego przesłania, zostanie wygenerowany komunikat o błędzie.

Tryby przesyłania plików

FTP powstał we wczesnym etapie rozwoju TCP/IP, gdy praktycznie wszystkie pliki zapisywane były w formacie ASCII. Kiedy trzeba było przesyłać plik binarny (dowolny plik nie spełniający wymogów standardu ASCII), tryb przesyłania danych musiał zostać zmieniony ręcznie. FTP pozwala na transmitowanie plików w kilku trybach, zależnie od systemu. Większość systemów obsługuje tylko dwa z nich: `binary` (binarny) oraz `ASCII` (pliki tekstowe w formacie ASCII). Niektóre stacje robocze obsługują jeszcze standard EBCDIC, a wiele sieci lokalnych posiada własne formaty, pozwalające na przyspieszenie transmisji (mogą one używać słów 32 – lub 64 – bitowych).

By przesyłać plik inny niż tekstowy, musisz zmienić tryb przesyłania na binarny poleceńiem `binary` (może ono zostać skrócone do formy `bin`). Do trybu tekstowego wrócić można, wydając polecenie `ascii`. Ważne jest, by pamiętać, jaki tryb jest aktualnie ustalony. Linuxowy FTP uruchamia się standardowo w trybie ASCII.

Transfer tekstowy przesyła pliki ASCII, w których wiersze są rozdzielone powrotem karetki i znakiem nowego wiersza, podczas gdy tryb binarny pozwala przesyłać dane bez żadnego formatowania. Tryb binarny jest szybszy od tekstu, a pozwala również na prawidłową transmisję plików tekstowych. W większości systemów domyślnie uruchamiany jest tryb tekstowy, ale wielu administratorów dla wygody użytkowników zmienia ten stan rzeczy. FTP nie umożliwia przesyłania atrybutów plików. Mimo tego niektóre programy FTP potrafią automatycznie rozpoznać typ pliku. Jeśli nie jesteś pewny, jakiego trybu przesyłania powinieneś użyć – użyj trybu binarnego.

Zwykle w sesji FTP nie są dostępne żadne skróty klawiaturowe (jak np. klawisz Tab do kończący polecenia). Oznacza to, że trzeba podawać nazwy plików w ich pełnym (i poprawnym) brzmieniu. Jeśli pomyliś się, otrzymasz komunikat o błędzie i musisz wprowadzić polecenie jeszcze raz. Jeśli uruchamiasz sesję FTP pod kontrolą systemu graficznego, zwykle masz możliwość kopiowania wcześniejszych polecień.

Anonimowy dostęp do FTP

FTP wymaga od użytkownika podania identyfikatora oraz hasła, zanim możliwe będzie korzystanie z zasobów serwera, ale istnieje też bardziej liberalna metoda udostępniania plików i katalogów – dostęp anonimowy. Eliminuje ona konieczność posiadania konta na serwerze, umożliwiając każdemu zalogowanie się podając identyfikator `anonymous` lub `guest`. Przykładowa sesja anonimowego połączenia FTP może wyglądać tak:

```
tpci> ftp uofo.edu
Connected to uofo.edu.
220 uofo.edu FTP server ready.
Name (uofo:username): anonymous
331 Guest login ok., send userID as password.
Password: tparker
220 Guest login ok, access restrictions apply.
ftp>
```

Jeśli serwer obsługuje anonimowe połączenia FTP, czasem zostaniesz poproszony o podanie hasła (którym może być Twój identyfikator lub adres e-mail) oraz wyświetcone zostanie ostrzeżenie o ograniczeniach dostępu. Jeśli chcesz pobrać jakiś plik z serwera, użyj polecenia `get`, tak jakbyś był zalogowany jako pełnoprawny użytkownik. Anonimowe połączenia FTP staje się coraz popularniejsze w Internecie.

Serwery obsługujące anonimowe połączenia FTP zwykle informują o tym zaraz po nawiązaniu połączenia, na przykład tak:

```
ftp sunsite.unc.edu
331 Guest login ok., send your complete e-mail address as password.
Enter username (default: anonymous): anonymous
Enter password : tparker@tpci.com
|FTP| Open
230-           WELCOME to UNC and SUN's anonymous ftp server
230-                           University of North Carolina
230-                           Office FOR Information Technology
230-                           sunsite.unc.edu
230 Guest login ok., access restrictions apply.
FTP>
```

Po zakończeniu logowania zobaczysz znak zachęty `FTP>`, który informuje, że serwer jest gotów do przyjmowania poleceń.

Po zalogowaniu się do niektórych systemów, wyświetlana jest krótka instrukcja dotycząca pobierania plików, nałożonych na anonimowego użytkownika ograniczeń oraz interesujących plików, które można znaleźć w systemie. Poniższy przykład pochodzi z węzła `sunsite.unc.edu`.

```
To get a binary file, type: BINARY and then: GET "File.Name" newfilename
To get a text file, type: ASCII and then: GET "File.Name" newfilename
Names MUST match upper, lower case exactly. Use the "quotes" as shown.
To get a directory, type: DIR. To change directory, type: CD "Dir.Name"
To read a short text file, type GET "File.Name" TT
For more, type HELP or see FAQ in gopher.
To quit, type EXIT or Control-Z.

230- If you e-mail to info@sunsite.unc.edu you will be sent help
Σ information
230- about how to use the different services sunsite provides.
230- We use the Wuarchive experimental ftpd. If you "get"
Σ <directory>.tar.Z
230- or <file>.Z it will compress and/or tar it on the fly. Using ".gz"
Σ instead
230- of ".Z" will use the GNU zip (/pub/gnu/gzip*) instead, a superior
230- compression method.
```

Większość anonimowych węzłów FTP pozwala tylko na pobieranie plików i nie można wysyłać do nich żadnych plików. Zwykle również bardzo ograniczone są możliwości poruszania się w systemie plików.

Protokół TFTP (Trivial file transfer protocol)

TFTP to uproszczona wersja protokołu FTP. Różni się od niego w dwóch zasadniczych aspektach: nie wymaga logowania się do serwera oraz wykorzystuje protokół UDP (User Datagram Protocol). Zwykle używa się go do pobierania plików z bardzo zajętych serwerów lub wtedy, gdy czas dostarczenia plików nie ma większego znaczenia. Bywa również przydatny, jeśli używasz terminalu nie posiadającego własnego dysku. Zasada jego działania jest nieco podobna do poczty elektronicznej: wysyłasz zlecenie, a po jakimś (trudnym do przewidzenia) czasie nadchodzi zamówiony plik.

W większości systemów plik może zostać wysłany poprzez TFTP tylko wtedy, gdy może być czytany przez wszystkich użytkowników. Ponieważ jednak zasady jego działania nie są jasno określone, większość administratorów nakłada na niego jeszcze większe ograniczenia bądź też całkowicie zabrania używania go.

Podstawowe instrukcje TFTP zebrane zostały w tabeli 15.2. Choć wyglądają podobnie jak polecenia FTP, różnią się zasadniczo pod tym względem, że działają bez istnienia bezpośredniego połączenia. Najbardziej rzuca się w oczy przypadek polecenia `connect`, które nie uruchamia połączenia, tylko pozwala określić serwer, z którego dane mają zostać pobrane.

Tabela 15.2. Polecenia dostępne dla użytkownika TFTP

Polecenie	Opis
<code>binary</code>	Włącza binarny tryb transmisji plików
<code>connect</code>	Ustawia adres serwera
<code>get</code>	Pobiera plik z serwera
<code>put</code>	Wysyła plik do serwera
<code>trace</code>	Wyświetla kody protokołu
<code>verbose</code>	Wyświetla więcej informacji podczas działania

TFTP pozwala na przesyłanie plików zarówno w trybie tekstowym, jak i binarnym. Podobnie jak Telnet i FTP, TFTP wymaga, by na serwerze działał odpowiedni proces (w przypadku maszyn UNIX-owych jest to proces `tftpd`) i program, zwykle zwany `tftp`. Ze względu na naturę tego protokołu, systemy takie jak Windows czy podobne systemy przeznaczone dla komputerów klasy PC nie obsługują go. W większości przypadków jest on używany do przesyłania danych pomiędzy systemami UNIX-owymi.

Prosta sesja TFTP z załączoną opcją `trace` przedstawiona jest w poniższym przykładzie.

```
>tftp
tftp> connect tpc1_hpws4
tftp> trace
Packet tracing on.
tftp> binary
```

```
Binary mode on.  
tftp> verbose  
Verbose mode on.  
tftp> status  
Connected to tpc1_hpws4.  
Mode: octet Verbose: on Tracing: on  
Rexmt-interval: 5 seconds, Max-timeout: 25 seconds  
tftp> get /usr/rmaclean/docs/draft1  
getting from tpc1_hpws4: /usr/rmaclean/docs/draft1 to /tmp/draft1 [octet]  
sent RRQ <file=/usr/rmaclean/docs/draft1, mode=octet>  
received DATA <block1, 512 bytes>  
send ACK <block=1>  
received DATA <block2, 512 bytes>  
send ACK <block=3>  
received DATA <block4, 128 bytes>  
send ACK <block=3>  
Received 1152 bytes in 0.2 second 46080 bits/s]  
tftp> quit
```

TFTP przydaje się, kiedy sieć jest bardzo zatłoczona lub serwer nie obsługuje połączeń FTP. Jednak w większości przypadków o wiele wygodniej jest używać FTP.

Używanie programu Telnet

Telnet pozwala zalogować się do innego serwera i pracować tak, jakbyś fizycznie siedział przed tym komputerem, umożliwiając dostęp do wszystkich jego zasobów. Jeśli na serwerze działa jednostka arytmetyczna o dużej wydajności, możesz używać jej zamiast swojego procesora. Jeśli do serwera podłączone są jakieś urządzenia, jak skaner, nagrywarka CD itp. możesz ich również używać. Oczywiście możesz też używać systemu plików.

Obsługa programu Telnet jest prosta, ponieważ problemami związanymi z nawiązaniem połączenia itp. zajmuje się sam protokół. Jako część procesu inicjalizacji, Telnet przesyła serię komunikatów pomiędzy klientem a serwerem dotyczących identyfikacji terminalu i specjalnych funkcji obsługiwanych przez terminal. Wszystko co musisz zrobić jako użytkownik, to podanie adresu komputera, z którym chcesz się połączyć, a następnie identyfikatora użytkownika i hasła.

Istnieje wiele implementacji programu Telnet dostępnych praktycznie dla każdego systemu operacyjnego. Dostępne są zarówno wersje komercyjne (zwykle rozprowadzane jako część pakietu TCP/IP), jak i darmowe albo shareware'owe. Wybór jednej z nich zależy od osobistych upodobań, ponieważ spełniają one te same zadania.

Uruchamianie programu Telnet jest bardzo proste. Wystarczy wydać w wierszu poleceń polecenie `telnet`, jako argument podając nazwę komputera, z którym chcesz się połączyć (system musi wówczas umieć przetłumaczyć ją na odpowiedni numer IP). Można również zamiast niej podać bezpośrednio numer IP. W niektórych systemach ze względów bezpieczeństwa na program Telnet nałożone są pewne ograniczenia – szczególnie powinieneś dowiedzieć się u administratora.

Polecenie `telnet` udostępnia wiele opcji, które pozwalają dostosować jego działanie do własnych potrzeb, ale są one dość rzadko używane w typowych sytuacjach. Poza tym obsługiwane opcje różnią się w zależności od wersji oprogramowania i rodzaju systemu operacyjnego. Jeśli chcesz zmodyfikować działanie tego programu, odsyłamy do dołączonej do niego dokumentacji. W większości przypadków wystarczą ustalenia domyślne.

Używanie programu Telnet w systemach graficznych

Jeśli chcesz połączyć się z systemem graficznym i wyświetlać grafikę na swoim komputerze, musisz „nauczyć” obie strony połączenia, jak to robić. Jeśli łączysz ze sobą komputery linuxowe, jest to bardzo proste, ponieważ system Linux umożliwia bezproblemowe przesyłanie danych, które mają zostać wyświetcone w okienkach. Jeśli na przykład chcesz połączyć się z systemem, w którym uruchomiony jest Motif, pierwszym krokiem będzie użycie polecenia `xhost`, pozwalającego zdalnemu komputerowi na otwieranie okien na Twoim terminalu:

```
xhost +
```

Polecenie to zezwala na otwieranie okien każdemu połączonemu z Twoim systemem komputerowym zdalnym. Jeśli nie takie są Twoje zamiary, po znaku + powinieneś podać nazwę komputera, którego zezwolenie dotyczy.

Po połączeniu się za pomocą programu Telnet z systemem opartym na GUI, musisz zlecić mu otwieranie wszystkich okienek na Twoim terminalu. W przypadku Linuxa robi się to przez ustawienie wartości zmiennej DISPLAY. Jeśli używasz powłoki C, wydaj polecenie:

```
setenv DISPLAY nazwa_twojego_komputera:0.0
```

W przypadku interpreterów `bash` lub `pdksh` należy odpowiednio dostosować składnię tego polecenia.

TN3270 i inne

Zależnie od systemu, do którego się logujesz, może okazać się, że Telnet nie potrafi prawidłowo obsługiwać terminalu. Zdarza się to najczęściej przy połączeniach ze stacjami roboczymi IBM i minikomputerami, które wymagają terminalu zgodnego z IBM 3270. Najprostsze wersje programu Telnet nie obsługują go, zwykle więc dostarczana jest wersja specjalna, o nazwie TN3270. Czasem również spotyka się program TN5250, który obsługuje terminal o bardziej zaawansowanych możliwościach niż TN3270. Oba te terminale obsługują pełną gamę kolorów.

TN3270 i TN5250 mogą być używane z każdym typem serwera, o ile potrafi on je emulować. Możesz na przykład używać TN3270, by otrzymać kolorowy terminal, podłączając się do serwera UNIX-owego, ponieważ standardowa wersja programu Telnet nie do-

starcza takiej możliwości. Większość wersji TN3270 i TN5250 pozwala zmieniać według życzenia kolory i czcionki wyświetlane na ekranie.

Podsumowanie

FTP i Telnet są w sieci lokalnej narzędziami, których wartości nie można przecenić, szczególnie jeśli różne aplikacje przechowywane są na różnych komputerach. Telnet jest bardzo łatwy do opanowania i używania. Wszystko, co musisz posiadać, to adres serwera, do którego zamierzasz się podłączyć, i połączenie z siecią.

Rozdział 22. „Instalacja i konfiguracja XFree86” dostarczy Ci informacji o możliwościach, jakie daje interfejs graficzny.

Zarządzanie systemem i procesami opisane jest w części czwartej, rozdział 32. „Podstawy administracji systemem”.

Wykaz ważniejszych węzłów FTP udostępniających oprogramowanie dla Linuxa znajdziesz w dodatku A pt. „Węzły FTP i grupy dyskusyjne dotyczące Linuxa”.

Część trzecia

Edycja i skład tekstu

W tej części:

- υ Edytory tekstu: vi i emacs
- υ groff
- υ geqn i gtbl
- υ TeX i LaTeX
- υ Drukowanie
- υ Linux i multimedia

Rozdział 16.

Edytory tekstu: vi i emacs

Peter MacKinnon

W tym rozdziale:

- υ Co to jest edytor tekstu?
- υ Funkcje edycyjne
- υ Edytor vi
- υ Edytor emacs

Co to jest edytor tekstu?

Edytor tekstu to chyba najważniejsze z narzędzi dostarczanych wraz z systemem Linux (i w zasadzie z każdym systemem operacyjnym). Używając go, można tworzyć i modyfikować pliki tekstowe, które w systemie obecne są między innymi jako:

- υ pliki użytkowników, takie jak na przykład `.login` czy `.cshrc`,
- υ pliki systemowe,
- υ programy powłoki,
- υ dokumenty,
- υ poczta elektroniczna.

To tylko niektóre z rozlicznych zastosowań plików tekstowych, z którymi zetkniesz się, pracując z Linuxem. Upraszczając nieco sprawę, edytor tekstu to program, który pozwala wstawiać, usuwać, przenosić tekst i przeszukiwać pliki tekstowe zawierające od kilku znaków do tysięcy wierszy.

Najpopularniejszymi edytorami tekstu używanymi w systemach linuxowych są programy `vi` oraz `emacs`. Są to edytory pełnoekranowe, tzn. wykorzystują one wszystkie wiersze i kolumny ekranu terminalu do wyświetlania zawartości pliku czy też informacji pomocniczych. Oba te edytory udostępniają ogromny zestaw poleceń edycyjnych. Podstawowy podzbiór tych poleceń można opanować w bardzo krótkim czasie, ale używanie poleceń umożliwiających wydobycie całej mocy drzewiącej w tych edytorach wymaga nieco praktyki. Mimo tego szybko stwierdzisz, że czas poświęcony na ich opanowanie nie był stracony.

Wybór edytora tekstu to w zasadzie kwestia gustu. Oba wspomniane programy są bardzo wydajne i potrafią obsłużyć plik o praktycznie dowolnych rozmiarach. Edytor `emacs` jest lepiej dostosowany do skomplikowanych operacji edytorskich i posiada pomoc dostępną w czasie pracy, ale w prostych przypadkach oba programy są równie dobre. Wybór jednego z nich należy do Ciebie.

Funkcje edycyjne

Choć w systemie Linux dostępnych jest kilka edytorów tekstu, a każdy z nich posiada inny interfejs, wszystkie potrafią wykonać w zasadzie te same zadania. Każdy edytor tekstu powinien udostępniać co najmniej funkcje omówione poniżej.

Wstawianie i usuwanie tekstu

Najbardziej podstawową funkcją edytora tekstu jest umożliwienie dopisywania i usuwania znaków wedle uznania użytkownika. Oznacza to również, że powinieneś mieć kontrolę nad ruchami kurSORA i jego położeniem w tekście.

Odczytywanie i zapisywanie plików

Ponieważ tekst, który edytujesz, zwykle przeznaczony jest do późniejszego wykorzystania, edytor tekstu potrafi zapisać go do pliku zewnętrznego. Kiedy zachodzi potrzeba wprowadzenia zmian w jakimś pliku, możesz wczytać jego zawartość do edytora. Wszystkie edytory potrafią obsługiwać standard ASCII, co oznacza, że plik utworzony za pomocą jednego edytora (na przykład `vi`) może być odczytany w innym (na przykład `emacs`).

Wyszukiwanie tekstu

Samodzielne przeglądanie tekstu wiersz po wierszu w poszukiwaniu konkretnego słowa jest co prawda świetnym ćwiczeniem koncentracji, ale nie każdy za tym przepada. Dlatego edytory tekstu posiadają zaawansowane możliwości wyszukiwania tekstu, zarówno

na podstawie ściśle określonego zestawu znaków, jak i bardziej ogólnych wzorców. Wzorce te mogą na przykład zawierać symbole wieloznaczne, takie jak * i ?, które następują dowolny tekst.

Edytory udostępniają również funkcję wyszukaj-i-zamień, która pozwala za pomocą pojedynczego polecenia podstawić dowolny tekst w miejsce każdego wystąpienia ciągu znaków pasującego do wzorca.

Kopiowanie i przenoszenie tekstu

Ponieważ nigdy nie ma pewności, czy tekst, który został zapisany w pliku, jest wersją ostateczną, edytory umożliwiają kopowanie, wycinanie i wstawianie bloków tekstu. Bloki te mogą mieć wielkość od kilku znaków do kilku stron. Różnica pomiędzy wycinaniem i kopowaniem polega na tym, że po wycięciu tekst przeniesiony do schowka jest usuwany ze swojej pierwotnej pozycji, natomiast po skopiowaniu pozostaje nienaruszony.

Bufory edycyjne

Bufory edycyjne to obszary pamięci, w których przechowywany jest edytowany tekst. Kiedy tworzysz nowy plik tekstowy, jego zawartość przechowywana jest w buforze edycyjnym przynajmniej do czasu zapisania go do pliku zewnętrznego. Bufory są również używane, gdy zachodzi potrzeba tymczasowego przeniesienia bloku tekstu do pamięci (na przykład przy kopiowaniu czy przenoszeniu). Wiele edytorów potrafi równocześnie obsługiwać kilka buforów edycyjnych.

Edytory opisane poniżej udostępniają wiele polecen, które nie zostały tu szczegółowo omówione. Z tego powodu gdy musisz przeprowadzić jakieś żmudne operacje edytorskie, powinieneś najpierw zajrzeć na strony `man` – może bowiem okazać się, że istnieją sposoby pozwalające łatwiej i szybciej wykonać zadanie. W miarę zdobywania doświadczenia poznasz wiele skrótów klawiaturowych i polecen, które uproszczą codzienną pracę.

Edytor vi

Edytor `vi` jest zainstalowany praktycznie w każdym istniejącym systemie UNIX-owym. Z tego powodu jest on w systemach UNIX-owych (a więc i Linuxie) uważany za edytor domyślny. Edytor `vi` jest nieco trudniejszy do opanowania niż `emacs`, ponieważ posiada dwa tryby pracy, a polecenia edycyjne mają bardzo zwięzłą formę. Warto go jednak poznać, szczególnie jeśli edytor `emacs` nie jest zainstalowany w Twoim systemie.

Uruchamianie edytora vi

Edytor vi można uruchomić, wpisując polecenie

```
vi
```

Ekran tekstowy zostanie wyczyszczony, a pierwsza kolumna wypełni się tyldami (~). Edytujesz teraz pusty plik, nie posiadający jeszcze nawet nazwy. Wpisywany tekst jest przechowywany w buforze edycyjnym, aż do czasu zapisania go na dysk. Za pomocą tyldy na początku wiersza edytor vi informuje użytkownika, że wiersz nie zawiera żadnego tekstu.

Nazwa pliku lub lista nazw plików, które chcesz edytować, może zostać przekazana do edytora vi jako argumenty wywołania:

```
vi plik1 plik2 plik3 ...
```

Zwykle jednorazowo edytuje się tylko jeden plik. Jeśli podasz całą listę plików, które zamierzasz edytować, vi umożliwia ich edycję w takiej kolejności, w jakiej zostały one podane.

Można również uruchomić edytor vi w następujący sposób:

```
vi +n nazwapliku
```

n oznacza tu numer wiersza, w którym (po otwarciu pliku) zostanie umieszczony kursor. Możliwość taka jest szczególnie przydatna dla programistów usuwających błędy w dużych plikach źródłowych i potrzebujących szybko przejść do wiersza zawierającego błąd.

Wpiszmy polecenie

```
vi piosenka  
i zobaczymy, co się stanie.
```

Tryby pracy edytora vi

W lewym dolnym rogu ekranu wyświetlony zostanie tekst:

```
"piosenka" 0 lines 0 characters
```

Informacje podawane w tym wierszu (jest to tzw. wiersz statusu) mówią, co właśnie zrobił lub robi edytor vi. W powyższym przykładzie otwarty został pusty bufor edycyjny, którego zawartość zostanie zapisana (o ile wydane zostanie odpowiednie polecenie) do pliku o nazwie piosenka.

W tej chwili aktywny jest tryb wydawania poleceń. Kwestią aktywnego trybu ma zasadnicze znaczenie podczas pracy z edytorem vi. W trybie wydawania poleceń wszystko, co wpisujesz, interpretowane jest jako polecenie. W trybie edycji wciśnięcie klawisza po-

woduje wstawienie do edytowanego tekstu odpowiedniego znaku (i oczywiście wyświetlenie go na ekranie).

W wierszu statusu wyświetlane są cztery spośród wielu dostępnych poleceń:

- / wyszukiwanie w przód,
- ? wyszukiwanie wstecz,
- : polecenie interpretowane przez ex (ex to osobny edytor oparty na wierszu poleceń, używany wewnętrz vi),
- ! wywołanie polecenia powłoki.

Polecenia te muszą zostać zatwierdzone klawiszem Enter. Pozostałe polecenia, na przykład wstawiające tekst, nie wymagają zatwierdzania.



Aby łatwiej było zorientować się, czy pracujesz w trybie wydawania poleceń, czy edycji, użyj polecenia set showmode, opisanego dokładniej w podrozdziale „Preferencje” w dalszej części tego rozdziału.

Wstawianie tekstu

Wstawmy teraz jakiś tekst, pamiętając o tym, że pracujemy w trybie wydawania poleceń. Dwa podstawowe polecenia edytora vi służące do tego celu to (ang. *insert*) – wstawiające tekst na lewo od kurSORA, oraz (ang. *append*) – wstawiające tekst na prawo od kurSORA. Wersje tych poleceń wywoływanie za pomocą wielkich liter działają podobnie, z tym że **I** wstawia tekst na początek, natomiast **A** na koniec wiersza, bez wzgledu na pozycję kurSORA.

Po wydaniu jednego z tych poleceń można rozpoczęć wprowadzanie tekstu. Zostanie on wyświetlony na ekranie.

Wciśnij klawisz a następnie wpisz:

```
Wlazl koteK<Enter>
na ploteK<Enter>
i mruga<Enter>
Ladna to<Enter>
piosenka<Enter>
niedluga.<Enter>
```

Tryb wstawiania tekstu opuścić można wciskając klawisz Escape. Zauważ, że polecenie nie jest wyświetlane na ekranie, nie trzeba również zatwierdzać go klawiszem Enter.

Zamykanie edytora vi

Po wprowadzeniu tekstu można zakończyć pracę edytora i obejrzeć utworzony plik. Polecenia zapisujące plik oraz kończące pracę edytora różnią się nieco od polecenia czy innych poleceń edycyjnych, ponieważ należy poprzedzić je dwukropkiem (:).

Zapisanie edytowanego pliku na dysk i zakończenie pracy edytora sprowadza się do wydania jednego polecenia. Wciśnij klawisz : – w lewym dolnym rogu ekranu pojawi się dwukropek. `vi` rozpoznał, że zamierzasz wydać polecenie interpretowane przez edytor `ex`, wyświetla więc wpisywane litery na ekranie. Wpisz `wq` i wciśnij Enter. `vi` poinformuje, że zapisał plik, poda liczbę zapisanych wierszy i zakończy działanie. Innym sposobem na zapisanie pliku i wyjście z programu jest użycie polecenia `zz`. Jedyna różnica polega na tym, że polecenie `zz` powoduje zapisanie pliku tylko wtedy, jeśli został on zmodyfikowany od czasu ostatniego zapisu.

Jeśli do otwartego pliku nie zostały wprowadzone żadne zmiany, można opuścić `vi`, wydając samo polecenie `:q`. Polecenie to nie zadziała, jeśli plik został zmodyfikowany. Jeśli jesteś pewny, że chcesz zakończyć pracę edytora bez zapisywania wprowadzonych zmian, użyj polecenia `:q!`, które wymusza zakończenie programu `vi`.

Upewnijmy się, czy edytowany plik został prawidłowo zapisany, oglądając jego zawartość za pomocą polecenia `cat`:

```
%cat piosenka
Wlazł kotek
na plotek
i mruga
Ladna to
piosenka
niedluga.
%
```

Plik zawiera tekst, który wpisałeś – żadnych niespodzianek.

Przesuwanie kurSORA

Do poruszania kursorem podczas pracy z edytorem `vi` służą cztery polecenia:

- h** przesuwa kurSOR jedną pozycję w lewo,
- j** przesuwa kurSOR jeden wiersz w dół,
- k** przesuwa kurSOR jeden wiersz do góry,
- l** przesuwa kurSOR jedną pozycję w prawo.

Polecenia te działają tylko w trybie wydawania poleceń. Większość implementacji `vi` obsługuje również klawisze kursora.

Możliwe jest też poruszanie się po edytowanym tekście większymi skokami. Poniżej podano kilka polecen pozwalających przesunąć kurSOR o więcej niż jeden wiersz:

Control+u przesuwa tekST o pół ekranu do góRy,

Control+d przesuwa tekST o pół ekranu w dól,

Control+f przesuwa tekST o cały ekran w dól,

Control+b przesuwa tekST o cały ekran do góRy.

Dokładna ilośĆ wierszy, o jaką przesuwany jest kurSOR, zależy od ustawień terminalu.

Innym sposobem poruszania się w tekście jest przejście do wiersza o znanyM numerze. Jeśli chcesz edytować piąty wiersz, wpisz w trybie wydawania polecen :5 albo `5G`. Samo polecenie `G` przesuwa kurSOR na koniec pliku. KurSOR nie poruszy się, jeśli zadany numer wiersza będzie poza dostępnym zakresem, na przykład po wydaniu polecenia :10 podczas edycji pliku, który składa się z ośmiu wierszy.

`vi` pozwala również przesuwać kurSOR słowo po słowie (słowo definiowane jest jako ciąg znaków otoczonych znakami białymi). Aby przenieść kurSOR na początek następującego słowa, wciśnij `w`, na początek poprzedniego – `b`.

Usuwanie tekstu

Edytor `vi` posiada polecenia służące do usuwania liter, wyrazów i całych wierszy. Usuwanie oznacza w tym przypadku, że znaki są „wycinane” z tekstu i przenoszone do bufora tekSTowego bez nazwy, z którego można je jeszcze odzyskać.

Aby usunąć słowo znajdujące się na prawo do kurSORa, użyj polecenia `dw`. Możesz również usunąć kilka słów za jednym zamachem, na przykład polecenie `4dw` usuwa cztery kolejne słowa w bieżącym wierszu.

Wiersze mogą być usuwane pojedynczo lub przez podanie zakresu. Aby usunąć bieżący wiersz, wydaj polecenie `dd`. Polecenie `4dd` usunie cztery kolejne wiersze. Polecenie `dG` usunie tekST do końca pliku.

Możesz też usuwać znaki w obu kierunkach w ramach bieżącego wiersza: polecenie `d^` usuwa wszystko do początku wiersza, `d$` (lub `D`) – do końca wiersza.

Do usuwania pojedynczych znaków służą polecenia `x` (usuwa znak na pozycji kurSORa) oraz `X` (usuwa znak na lewo od kurSORa). Oba te polecenia pozwalają również usuwać kilka znaków, np. polecenie `4x` usuwa znak na pozycji kurSORa i trzy następne.

Niezamierzone zmiany można odwołać poleceniem `u` (niestety, w większości wersji `vi` można odwołać tylko ostatnie polecenie).

Kopiowanie i przenoszenie tekstu

Skopiowanie fragmentu tekstu można rozbić na trzy etapy:

1. skopiowanie tekstu do bufora,
2. przeniesienie kurSORA w miejsce wstawiania tekstu,
3. wstawienie tekstu z bufora we wskazane miejsce.

Tekst można skopiować zarówno do bufora nazwanego, jak i nienazwanego. Bufor nie-nazwany to tymczasowe miejsce w pamięci, w którym przechowywany jest tekst. Każdy następny zapis niszczy jego poprzednią zawartość. `vi` posiada też 26 buforów nazwanych (ich nazwy są kolejnymi literami alfabetu).

Polecenia `yy` oraz `y` pozwalają skopiować wiersz tekstu do bufora. Mogą one być modyfikowane przez podanie liczby wierszy, które mają zostać skopiowane, np. polecenie `3yy` wydane podczas edycji pliku `piosenka` (gdy kurSOR jest w pierwszym wierszu) kopiuje do tymczasowego bufora następujący tekst:

```
Wlazł kotek  
na plotek  
i mruga
```

Tekst ten można również skopiować do bufora nazwanego; polecenie

```
"a3yy
```

skopiuje go do bufora o nazwie `a`. Cudzysłów na początku polecenia powoduje, że dotychczasowa zawartość bufora zostanie zamazana. Gdyby zamiast litery `a` wpisana została litera `A`, tekst zostałby dołączony na koniec bufora.

Jeśli przeniesiesz kurSOR na koniec pliku (np. używając polecenia `:$`), możesz wkleić zawartość bufora poleceniem `p` (na prawo od kurSORa) lub `P` (na lewo od kurSORa). Wersja tego polecenia dotycząca bufora o nazwie `a` ma postać "`ap`".

Do bufora można kopiować również pojedyncze słowa, służy do tego polecenie `yw` (które również współpracuje z buforami nazwanymi i pozwala na kopiowanie zadanej liczby słów).

Wyszukiwanie i zastępowanie tekstu

Wyszukiwanie tekstu może być w edytorze `vi` prowadzone zarówno w kierunku końca jak i początku pliku. Zawsze rozpoczyna się ono od bieżącej pozycji kurSORa, a po naciśnięciu końca (lub początku) pliku jest kontynuowane od jego początku (lub od końca, zależnie od kierunku przeszukiwania).

Dla zilustrowania tego zagadnienia posłużmy się stworzonym wcześniej plikiem. Aby poszukać w tekście słowa `mruga`, wpisz polecenie

```
/mruga
```

i wciśnij Enter. Zauważ, że treść polecenia jest wyświetlana w ostatnim wierszu ekranu. Po jej zatwierdzeniu kursor zostanie przeniesiony do miejsca, w którym po raz pierwszy występuje szukany tekst. Aby znaleźć miejsce kolejnego wystąpienia tego tekstu, wystarczy wydać polecenie `/`. Za każdym razem, gdy je wpiszesz, `vi` będzie próbował kontynuować ostatnio prowadzone szukanie w przód. Jeśli chcesz, aby wyszukiwanie było prowadzone w przeciwnym kierunku, powinieneś użyć polecenia `?`, działającego na tej samej zasadzie, co polecenie `/`. Spróbuj na przykład wydać polecenie

```
?o
```

Spowoduje ono wyszukanie pierwszego wystąpienia litery `o` poprzedzającego aktualną pozycję kurSORA. Jak można się spodziewać, wyszukiwanie może zostać powtórzone przez wydanie polecenia `?`. Polecenie `n` powoduje kontynuowanie procesu wyszukiwania w tym kierunku, w którym odbywało się ono ostatnio, natomiast `N` – w kierunku przeciwnym.

Jak wspomniano wcześniej, mechanizm wyszukiwania tekstów w edytorze `vi` jest dość potężny, głównie dzięki możliwości użycia symboli wieloznacznych. Symbole wieloznaczne w poleceniach wyszukiwania należy otaczać nawiasami kwadratowymi; aby na przykład znaleźć słowa zawierające ciąg `ot`, można wydać polecenie

```
/*]ot[*]
```

Pierwszym „trafionym” słowem w naszym tekście będzie `kotek`. Jeśli wciśniesz `n`, `vi` znajdzie następny pasujący do tego wzorca wyraz – czyli `plotek`. Można również podać zakres znaków, które mają być uwzględniane, np.

```
/[a-z]o
```

Znalezione zostaną słowa `kotek` oraz `to`. Ponieważ zakres znaków definiowany jest jako zakres odpowiadających im kodów ASCII, możliwe jest również podanie zakresu cyfr, np. `[0-9]`. Można również określić bezpośrednio zbiór znaków, które mają być uwzględniane, np.

```
/[Tt]o
```

W takim przypadku znalezione zostałyby wyrazy `to` oraz `To`.

Wyszukiwanie bez wykorzystania symboli wieloznacznych pozwala znaleźć tylko tekst identyczny z zadanym (dotyczy to również wielkości liter), więc nie ma chyba wątpliwości, że użycie symboli wieloznacznych jest czasem nieodzowne.

Jednym z częstszych zastosowań mechanizmu wyszukiwania jest zastępowanie znalezionej tekstu innym. Przykładowo, aby zamienić w tekście wszystkie wystąpienia słowa `plotek` słowem `krzeslo`, należy wydać polecenie

```
:%s/plotek/krzeslo/g
```

Litera `s` oznacza, że jest to polecenie wyszukiwania, symbol `%` – że przeszukiwany ma być cały plik, `płotek` – to tekst, który ma być wyszukany, natomiast `krzesło` to tekst, który ma go zastąpić. Litera `g` na końcu polecenia powoduje, że będzie ono wykonywane do chwili, gdy w tekście nie będzie już występowało słowo `płotek`. Bez niej `vi` zmieniłby tylko pierwsze odnalezione słowo. W poleceniu tym również można używać symboli wieloznacznych.

Preferencje

Zachowanie edytora `vi` może być dostosowane do wymagań użytkownika za pomocą opcji. Każda z nich jest inicjowana wartością domyślną, i można ją zmodyfikować używając wewnętrznego polecenia edytora `vi` o nazwie `set`. Musi być ono poprzedzone dwukropkiem i zatwierdzone klawiszem Enter, np. aby załączyć wyświetlanie numerów wierszy, należy wydać polecenie

```
:set number
```

Poniżej zebrano niektóre z dostępnych modyfikatorów polecenia `set`:

all	wyświetla listę wszystkich dostępnych opcji wraz z ich aktualnymi ustawieniami,
errorbells	załącza dźwiękowe informowanie o wystąpieniu błędu,
ignorecase	powoduje, że w poleceniach wyszukiwania nie będzie uwzględniana wielkość liter,
number	wyświetla z lewej strony numery wierszy (nie są one zapisywane w pliku),
showmode	powoduje, że w prawym dolnym rogu wyświetlana jest informacja o aktualnym trybie pracy.

Opcje można wyłączać, podając przed ich nazwą przedrostek `no`, np. aby wyłączyć numerowanie wierszy, należy wydać polecenie

```
:set nonumber
```

Polecenie `:set` bez żadnego modyfikatora powoduje wyświetlenie tych opcji, których wartości zostały zmienione przez użytkownika.

Wprowadzone modyfikacje są jednak tracone po opuszczeniu edytora `vi`. Aby tego uniknąć, można zapisać je w pliku inicjalizacyjnym edytora `vi` o nazwie `.exrc`, znajdującym się w katalogu domowym. Plik ten jest przetwarzany (o ile istnieje) przed każdym uruchomieniem edytora `vi`. Oto przykładowa zawartość takiego pliku:

```
set number
set errorbells
set showmode
```

Zauważ, że w pliku tym dwukropki poprzedzające polecenie `set` nie są wymagane.

Podsumowanie najważniejszych poleceń

W tabeli 16.1 zebrano najważniejsze polecenia edytora `vi` wraz z krótkimi objaśnieniami. Bardziej wyczerpujące informacje dotyczące tych i innych poleceń znajdziesz na stronach `man`.

Tabela 16.1. Podstawowe polecenia edytora vi

Polecenie	Znaczenie
i	Przejście do trybu wstawiania tekstu
h	Przesunięcie kurSORA o jedną pozycję w lewo
j	Przesunięcie kurSORA o jeden wiersz w dół
k	Przesunięcie kurSORA o jeden wiersz w górę

Tabela 16.1. cd. Podstawowe polecenia edytora vi

Polecenie	Znaczenie
l	Przesunięcie kurSORA o jedną pozycję w prawo
Control+f	Przesunięcie tekstu o jeden ekran w dół
Control+b	Przesunięcie tekstu o jeden ekran w górę
ndd	Usunięcie n wierszy
nyy	Skopiowanie n kolejnych wierszy do nienazwanego bufora
p	Wklejenie zawartości bufora na prawo od kurSORA
u	Odwołanie ostatnio wprowadzonej zmiany
:wq	Zapisanie pliku i zakończenie działania edytora vi
:q!	Zakończenie działania edytora vi bez zapisywania zmian
:set all	Wyświetlenie wartości wszystkich opcji
/tekst	Wyszukanie pierwszego wystąpienia słowa tekst

Edytor emacs

Wiele osób wybiera edytor `emacs` ze względu na dostępną w czasie pracy pomoc oraz możliwość tworzenia własnych poleceń edycyjnych. Dzięki temu, że można skonfigurować go tak, by automatycznie formatował kod źródłowy w językach takich jak C, C++,

Lisp i innych, edytor `emacs` jest szczególnie atrakcyjny dla programistów. Jest on również nieco łatwiejszy w obsłudze niż `vi` i udostępnia bogatszy zestaw poleceń.

Uruchamianie edytora `emacs`

Edytor `emacs` może zostać uruchomiony za pomocą polecenia

```
emacs
```

Nazwa pliku, który chcesz edytować, może zostać podana jako argument wywołania:

```
emacs <nazwa_pliku>
```

Jeśli jako argument programu `emacs` zostanie podana nazwa pliku, na ekranie wyświetlna zostanie jego zawartość. Dwa ostatnie wiersze ekranu terminalu zawierają informacje edytora: przedostatni zawiera nazwę edytowanego pliku i aktualne położenie kursora (`TOP, BOT, 20% itp.`), natomiast ostatni służy do wyświetlania komunikatów systemowych i pozwala na wprowadzanie niektórych poleceń.

Klawisze Control oraz Meta

Po uruchomieniu edytora `emacs` można zacząć wprowadzać tekst. Chciałbyś pewnie jednak umieć również poruszać kursorem. Zaraz omówimy ten drobiazg, ale najpierw wyjaśnijmy znaczenie dwóch klawisz: Control (z tym raczej nie ma problemów) oraz Meta. Większość poleceń edytora `emacs` dostępna jest za pomocą kombinacji klawisza Control i innego znaku, ale niektóre wymagają wcisnięcia klawisza Meta. Jeśli chcesz wydać polecenie wymagające użycia klawisza Control, musisz przytrzymać go podczas wpisywania następnego znaku, natomiast nie trzeba przytrzymywać klawisza Meta. W przypadku komputerów PC klawiszem Meta jest zwykle Alt.

Przesuwanie kurSORA

Poniżej przedstawiamy polecenia przesuwające kurSOR w edytorze `emacs`:

- Control+f** przesuwa kurSOR jedną pozycję do przodu,
- Control+b** przesuwa kurSOR jedną pozycję do tyłu,
- Control+p** przesuwa kurSOR do poprzedniego wiersza,
- Control+n** przesuwa kurSOR do następnego wiersza,
- Control+a** przesuwa kurSOR na początek wiersza,
- Control+e** przesuwa kurSOR na koniec wiersza.

Większość implementacji edytora `emacs` dla wygody zamiast pierwszych czterech poleceń pozwala używać klawiszy kurSORA. Stwórzmy teraz nowy plik o nazwie `plik2`:

```
emacs plik2
```

Po wpisaniu jakiegoś tekstu, możesz wrócić na jego początek, wciskając klawisze `Control+b`. Zwróć uwagę, jak kurSOR przeskakuje do poprzedniego wiersza po osiągnięciu początku wiersza bieżącego. W przeciwnym kierunku można poruszać się wciskając klawisze `Control+f`.

Innym sposobem poruszania się w tekście jest przesunięcie go o cały ekran do przodu za pomocą polecenia `Control+v`. Poleceniem `Meta+v` można przewinąć tekst o jeden ekran wstecz.

Podobnie jak `vi`, `emacs` traktuje ciąg znaków otoczony z obu stron znakami białymi jako słowo. Przesunąć kurSOR na początek następnego słowa można za pomocą polecenia `Meta+f`, na początek poprzedniego – `Meta+b`.

Zamykanie edytora `emacs`

Aby zapisać zawartość bufora edycyjnego do pliku, należy wydać polecenie `Control+x Control+s` (zauważ, jak jest ono wyświetlane w ostatnim wierszu ekranu). Opuścić edytor możesz wydając polecenie `Control+x Control+c`. Jeśli nie zapisałś wcześniej edytowanego pliku, `emacs` przed zakończeniem działania zapyta o potwierdzenie.

Usuwanie tekstu

Fragment tekstu można usunąć na kilka sposobów. Klawisz `Backspace` (lub `Delete`) służy do usuwania znaku na lewo od kurSORa. Polecenie `Control+d` usuwa znak znajdujący się na pozycji kurSORa, `Control+k` usuwa znaki od kurSORa do końca bieżącego wiersza. Można również usuwać pojedyncze słowa: polecenie `Meta+d` usuwa słowo, w którym aktualnie znajduje się kurSOR, `Meta+Delete` usuwa poprzednie słowo.

Jeśli zdarzy Ci się wydać jakieś polecenie edycyjne przez pomyłkę, możesz wrócić do poprzedniego stanu rzeczy kombinacją klawiszy `Control+x u`.



Jeżeli rozmyślisz się podczas wprowadzania polecenia, możesz się z niego wycofać kombinacją klawiszy `Control+g`.

Praca z wieloma plikami

Edytor `emacs` pozwala pracować z kilkoma plikami równocześnie, każdemu z nich przypisując osobny bufor edycyjny. Aby załadować plik do nowego bufora, wydaj polecenie `Control+x Control+f`. W ostatnim wierszu pojawi się tekst

Find file: ~/

Teraz należy podać nazwę pliku, który ma być edytowany. `emacs` potrafi po naciśnięciu klawisza Tab dokończyć za Ciebie rozpoczętą nazwę – na przykład jeśli wpiszesz `.log` i wciśniesz Tab, nazwa pliku zostanie uzupełniona do `.login` (o ile plik taki istnieje w katalogu domowym). Jeśli wynikiem dokańczania jest więcej niż jedna nazwa, możesz wybrać tę, o którą chodziło, naciskając ponownie klawisz Tab.

Po załadowaniu nowego pliku możesz przełączać się pomiędzy buforami edycyjnymi polecienniem Control+x b, po którym należy podać nazwę bufora (jest to nazwa pliku, który jest do niego załadowany; tu również obsługiwane jest dokończanie nazwy po naciśnięciu klawisza Tab).

Po dokonaniu zmian możesz zdecydować, że nie chcesz ich zachować. Zawartość bufora można usunąć polecienniem Control+x k. `emacs` zapyta o nazwę bufora, którego zawartość chcesz usunąć; wciskając Enter, wybierasz bufor bieżący. Następnie zostaniesz zapytany o potwierdzenie – powinieneś odpowiedzieć `yes` (jeśli jesteś pewny swojej decyzji) i wcisnąć Enter.



Jeśli pracujesz z dwoma buforami, wystarczy, że po wydaniu polecenia Control+x b wciśniesz po prostu klawisz Enter – `emacs` przełączy się na edycję drugiego bufora.

Kopiowanie i przenoszenie tekstu

Jeśli chcesz skopiować lub przenieść fragment tekstu, musisz najpierw zaznaczyć jego początek i koniec. Aby to zrobić, przejdź na początek wybranego fragmentu i wciśnij klawisze Control+Space. Końcem bloku jest punkt, w którym znajduje się kurSOR – przesuń go więc w żądane miejsce. Po wydaniu polecenia Meta+w tekst zostanie skopiowany do schowka, z którego można wkleić go w inne miejsce polecienniem Control+y. Można również przenieść tekst do schowka (Control+w), co spowoduje usunięcie go z pierwotnej lokalizacji.

Wypróbuj te polecenia na utworzonym wcześniej pliku. Użyj polecenia Meta+<, które przeniesie kurSOR na początek bufora. Po wciśnięciu klawiszy Control+Space zostanie zaznaczony początek bloku tekstu. Teraz wydaj polecenie Control+n, które spowoduje przejście do następnego wiersza. Przenieś zaznaczony tekst do schowka polecienniem Control+w, przesuń kurSOR na koniec bufora polecienniem Meta+>, a następnie wstaw tekst przeniesiony poprzednio do schowka wydając polecenie Control+y.

Wyszukiwanie i zastępowanie tekstu

Wyszukiwanie tekstu w przód i w tył możliwe jest dzięki poleceniom Control+s i Control+r. Polecenia te również w pewnym sensie korzystają z mechanizmu dokańczania: edytor

stara się odgadnąć, o co Ci chodzi już w trakcie ich wydawania. W tym przypadku, `emacs` po dopisaniu każdego znaku przenosi kurSOR do tekstu, który pasuje do podanego wzorca.

Zakres znajdowanych tekstów zawiera się w miarę wpisywania kolejnych liter. Po odnalezieniu fragmentu, o który chodziło, możesz zakończyć wyszukiwanie wciskając Enter lub którykolwiek z klawiszy sterujących kursorem.

Podobnie jak w edytorze `vi`, po napotkaniu końca pliku przeszukiwanie jest kontynuowane od jego początku (lub odwrotnie, zależnie od kierunku szukania). Jednak `emacs` informuje Cię o zaistniałej sytuacji, komunikując, że wyszukiwanie zakończyło się niepowodzeniem. Jeśli chcesz je kontynuować, powinieneś wcisnąć odpowiednio `Control+s` lub `Control+r`.

Dla zilustrowania zagadnienia wyszukiwania w edytorze `emacs` przejdźmy na koniec pliku `piosenka` i poszukajmy (wstecz) litery `o` – w tym celu należy wcisnąć `Control+r`, a następnie `o`. `emacs` przesunie kurSOR do litery `o` w wyrazie `piosenka`. Wpisz teraz `t`. Kursor przesunie się do litery `t` w wyrazie `plotek`. Możesz poprawiać wpisywany tekst klawiszami `Delete` lub `Backspace`. Zobacz, co się stanie, gdy usuniesz klawiszem `Delete` literę `t` i w jej miejsce wpiszesz `s`.

Polecenie `Meta+x` pozwala zastąpić tekst pasujący do wzorca dowolnym innym. Po jego wydaniu, `emacs` czeka na wprowadzenie tekstu, który należy odnaleźć. Następnie trzeba podać, na jaki tekst należy go zamienić. Jeśli `emacs` odnajdzie zadany tekst, zostaniesz zapytany o potwierdzenie zamiany.

Tryby edycji

Edytor `emacs` jest na tyle elastyczny, że potrafi dostosować się do różnych typów edytowanych dokumentów. Co więcej, każdy z buforów może być edytowany w innym trybie, dzięki czemu można łatwo dopasować sposób formatowania do typu dokumentu. Przykładowo, po wydaniu polecenia `Control+x m` uruchamiany jest tryb edycji poczty. W buforze pojawiają się pola `To:` oraz `Subject:`, jak również miejsce na samą wiadomość. `emacs` potrafi nawet wysłać ją za Ciebie (po wydaniu polecenia `Control+c Control+c`).

Edytor `emacs` posiada też tryby pracy przeznaczone do edycji plików zawierających kod źródłowy napisany w różnych językach programowania, np. w języku C. Jeśli zaledwie plik z rozszerzeniem `.c` (program w języku C) lub `.h` (plik nagłówkowy), bufor zostanie automatycznie ustawiony do pracy w trybie C. Ułatwia on w dużym stopniu przejrzyste formatowanie tekstu programu, obsługując poprawnie wcięcia w zależności od składni (np. zagnieżdżone pętle `for`).

Pomoc dostępna podczas pracy

Jedną z najprzyjemniejszych cech edytora `emacs` jest pomoc dostępna podczas pracy. Jeśli potrzebujesz skróconego przewodnika, wcisnij `Control+h t`. Jeżeli chcesz się do-

wiedzieć czegoś więcej na temat działania konkretnej kombinacji klawiszy, wciśnij Control+h, a następnie kombinację klawiszy, która Cię interesuje. Pomoc obejmuje bardzo wiele tematów; by zobaczyć je wszystkie, wywołaj tryb czytania dokumentacji (info) polecением Control+h i.

Podsumowanie najważniejszych poleceń

emacs, podobnie jak vi, posiada bardzo bogaty zestaw polecen edycyjnych. W tym rozdziale omówiliśmy tylko niewielką ich część. W tabeli 16.2 zebrane zostały te z nich, które są niezbędne do pracy w podstawowym zakresie. Pełniejsze informacje znaleźć można na stronach `man`.

Tabela 16.2. Podstawowe polecenia edytora emacs

Polecenie	Znaczenie
Control+b	Przesuwa kurSOR o jedną pozycję w lewo
Control+d	Usuwa znak na pozycji kurSora
Control+f	Przesuwa kurSOR o jedną pozycję w prawo
Control+g	Anuluje wydawane polecenie

Tabela 16.2. cd. Podstawowe polecenia edytora emacs

Polecenie	Znaczenie
Control+h	Uruchamia system pomocy
Control+n	Przesuwa kurSOR do następnego wiersza
Control+p	Przesuwa kurSOR do poprzedniego wiersza
Control+s	Wyszukuje tekst w przód
Control+v	Przewija tekst o jeden ekran do przodu
Meta+v	Przewija tekst o jeden ekran do tyłu
Control+x u	Cofa ostatnią zmianę
Control+x Control+c	Kończy pracę edytora emacs
Control+x Control+s	Zapisuje bufor do pliku

Podsumowanie

W systemie Linux dostępnych jest wiele edytorów tekstu. Najpopularniejszymi z nich są vi (jest to tak naprawdę inna nazwa edytora elvis) oraz emacs. Oba zapewniają podstawowe funkcje, takie jak wstawianie i usuwanie tekstu, odczytywanie i zapisywanie

plików, przeszukiwanie, kopianie i przenoszenie tekstu. `vi` posiada dwa tryby pracy: tryb wydawania poleceń oraz tryb edycji. `emacs` jest bardzo elastycznym i potężnym edytorem tekstu, nadającym się świetnie do wielu zadań (takich jak pisanie programów, dokumentów, poczty czy modyfikowanie plików systemowych).

Program `groff`, służący do formatowania tekstu, opisany jest w rozdziale 17. „groff”.

TeX i LaTeX, bardzo elastyczne systemy edycji i składu tekstu, opisane są w rozdziale 19. „TeX i LaTeX”.

O tym, jak skonfigurować system X, przeczytać możesz w rozdziale 22. „Instalacja i konfiguracja XFree86”.

Rozdział 17.

groff

Tim Parker

W tym rozdziale:

- υ Osadzanie poleceń
- υ Określanie wyglądu znaków
- υ Makropolecenia
- υ Zestaw makropoleceń `mm`

`groff` to wersja GNU programów `nroff` i `troff`, które są językami przeznaczonymi do formatowania tekstów, używanymi od lat w systemach UNIX-owych. System `groff` zawiera wersje programów `troff`, `nroff`, `eqn`, `tbl` i innych programów UNIX-owych służących do formatowania tekstu. Język `groff` używany jest głównie do komplikowania stron `man` zapisanych w formacie `groff/nroff` do postaci, w której mogą być one drukowane lub wyświetlane na ekranie terminala.

Język `nroff` był pierwotnie zaprojektowany do obsługi drukarek pracujących w trybie tekstowym, natomiast `troff` – do składu graficznego. Polecenia dostępne w obu tych językach są takie same, choć niektóre z nich (te, które nie mogą być obsłużone przez drukarkę tekstową) są ignorowane przez `nroff`. Zwykle nie używa się programów `nroff` i `troff` bezpośrednio, ale za pośrednictwem odpowiednio spreparowanych makropoleceń.

W większości zastosowań programy te zostały wyparte przez mające o wiele większe możliwości edytory i systemy składu tekstów. Ich głównym (o ile nie jedynym) zastosowaniem pozostała obsługa stron `man`.

Zarówno `nroff` jak i `troff` udostępniają wiele poleceń, których nigdy nie będziesz potrzebował. W tym rozdziale przyjrzymy się podstawowemu podzbiorowi polecen tych dwóch języków. Jeśli chcesz używać ich „na poważnie”, powinieneś sięgnąć do jednej z książek poświęconych temu tematowi.

Osadzanie poleceń

Tekst po przetworzeniu przez program `groff` niekoniecznie będzie miał taką formę, w jakiej został wpisany, na przykład połączone zostaną wszystkie wiersze. Plik zawierający tekst:

```
To bardzo ciekawe.  
Przyjrzenie się temu bliżej  
zajmie mi kilka godzin.
```

składa się z trzech wierszy, ale po sformatowaniu programem `groff` może wyglądać tak:

```
To bardzo ciekawe. Przyjrzenie się temu bliżej zajmie mi kilka godzin.
```

Podziały wierszy zależą tylko od układu strony. Dzięki takiemu rozwiązaniu nie trzeba przejmować się tym, czy w pliku źródłowym wszystko zostało prawidłowo sformatowane. Oczywiście wadą tego rozwiązania jest fakt, że tak naprawdę nigdy nie wiesz, co otrzymasz, dopóki tego nie zobaczysz.

Rzut oka na dowolny plik źródłowy programu `groff` pozwala stwierdzić, że jest to zwykły plik ASCII, zawierający tekst, który ma być wyświetlany, oraz polecenia zaczynające się od kropki, na przykład tak:

```
To tekst, który zostanie wyświetlony.  
To jeszcze więcej tekstu.  
.ps 14  
Powyższy wiersz jest poleceniem programu groff,  
ponieważ jej pierwszym znakiem jest kropka.
```

Większość poleceń języka `groff` zajmuje osobny wiersz, ale kilka z nich można osadzić w dowolnym miejscu, również wewnątrz wiersza tekstu. Takie polecenia zaczynają się od znaku \ (podobne rozwiązanie zastosowane jest w interpreterach poleceń powłoki). Oto przykład użycia poleceń osadzonych:

```
Ten \fBwiersz\fR zawiera dwa osadzone \fIpolecenia\fR języka groff.
```

Choć czasem użycie tego typu poleceń jest konieczne, większość poleceń zaczyna się od kropki w nowym wierszu.

Określanie wyglądu znaków

Język `groff` posiada kilka poleceń pozwalających kontrolować sposób, w jaki znaki zostaną wydrukowane czy wyświetcone. Są to polecenia dotyczące rozmiaru i typu czcionki oraz odstępów między wierszami.

Rozmiar czcionki i odstępy między wierszami

Rozmiar czcionki i odstępy między wierszami nie mają większego znaczenia, kiedy dokument jest wyświetlany na ekranie terminalu tekstowego. Ich użycie ma sens dopiero w przypadku dokumentacji drukowanej (albo oglądanej na ekranie graficznym). Rozmiar czcionki można ustalić za pomocą polecenia `.ps` (ang. *point size*):

```
To jest domyślna czcionka - 10 punktów.  
.ps 14  
To jest czcionka o rozmiarze czternaście punktów.  
.ps 20  
To rozmiar 20.  
.ps 6  
A to mała czcionka - rozmiar 6.
```



Punkt to 1/72 cala, więc czcionka o rozmiarze 32 jest wysoka na pół cala (ok. 1,27cm). Najczęściej używana czcionka o rozmiarze 12 punktów ma wysokość 1/6 cala (ok. 0,42cm). Różne wersje programu `groff` obsługują różne wielkości czcionek, ale większość z nich poprawnie drukuje i wyświetla czcionki o rozmiarach 6, 7, 8, 9, 10, 11, 12, 14, 16, 20, 24, 28 i 36 punktów. Jeśli wartość podana jako parametr polecenia `.ps` nie jest obsługiwana, jest ona zaokrąglana do najbliższej obsługiwanej. Domyślnie czcionka ma rozmiar 10 punktów. Polecenie `.ps` bez parametru powoduje przywrócenie poprzednio używanego rozmiaru czcionki.

W środku zdania rozmiar czcionki może zostać zmieniony poleceniem `\s`, po którym następuje liczba określająca nowy rozmiar:

```
To jest 10 punktów, a \s20to jest dwadzieścia, \s10to jest z powrotem 10.
```

Polecenie `\s0` powoduje powrót do poprzednio używanego rozmiaru. Obsługiwane są również zmiany względne, można więc osadzać polecenia typu `\s+2` czy `\s-2`, choć możliwe jest podanie tylko jednej cyfry, z czego wynika że w ten sposób nie da się zmienić rozmiaru czcionki o więcej niż 9 punktów.

Odstępy pomiędzy wierszami nie są wiązane automatycznie z rozmiarem czcionki, dla tego jeśli rozmiar czcionki został zmieniony, należy również odpowiednio skorygować odstępy. Odstępy pomiędzy wierszami powinny być o ok. 20% większe od wysokości czcionki. Domyślnie jest to 11 punktów.

Odstępy między wierszami kontrolowane są za pomocą polecenia `.vs` (ang. *vertical spacing*). W poniższym przykładzie zmieniony został rozmiar czcionki oraz wielkość odstępów między wierszami, tak by poszczególne wiersze nie nakładały się na siebie:

```
To jest czcionka 10 punktów, odstęp między wierszami - 11.  
.ps 12  
.vs 14  
To jest czcionka 12 punktów, odstęp między wierszami - 14..
```

Polecenie `.vs` bez parametrów powoduje przywrócenie poprzednio używanej wielkości odstępu.

Jeśli chcesz z jakiegoś powodu wymusić odstęp pomiędzy blokami tekstu, użyj polecenia `.sp` (ang. *space*). Użyte bez argumentów, polecenie to powoduje wstawienie pustego wiersza. Może również przyjmować jeden argument – wielkość odstępu, podaną w calach (przyrostek `i`) lub w punktach (przyrostek `p`):

```
To standardowa czcionka 10p z odstępami między wierszami 11p  
.sp  
Powyżej jest pusty wiersz.  
.sp 3.5i  
Nad tym wierszem jest 3.5 cala odstępu.
```

Jak widać w powyższym przykładzie, w większości poleceń języka `groff` można używać ułamków dziesiętnych.

Czcionki

Zmiana rodzaju czcionki wymaga użycia polecenia `.ft` (ang. *font type*). W pierwszych latach istnienia systemu `groff` dostępne były tylko cztery rodzaje czcionek: Roman, Roman wytłuszczone (ang. *bold*), Roman kursywa (ang. *italic*) i zestaw znaków specjalnych. Inne czcionki musiały być specjalnie ładowane do maszyny do składu. Z tego powodu domyślną czcionką jest czcionka Roman.

Aby zastosować czcionkę wytłuszczoną (ang. *bold*), użyj polecenia `.ft B`. Polecenie `.ft I` powoduje, że tekst będzie pisany kursywą (ang. *italic*). Do zwykłej czcionki wrócić można wydając polecenie `.ft R`, choć w większości systemów wystarczy samo polecenie `.ft`:

```
To zwykła czcionka Roman.  
.ft B  
To czcionka wytłuszccona.  
.ft I  
To kursywa.  
.ft  
A to znów zwykła czcionka.
```

Można przełączać się pomiędzy różnymi rodzajami czcionki również w obrębie jednego wiersza, za pomocą polecenia `\f`, po którym następuje litera `I`, `B` lub `R`:

```
To zwykła czcionka, \fBto wytłuszczena, \fIkursywa \fTRa to znów zwykła Σ  
czcionka.
```

Ponieważ dawniej większość drukarek nie obsługiwała podkreśleń, tekst podkreślony był zamieniany na kursywę. Dlatego polecenie podkreślenia `.ux` (ang. *underline*) powoduje, że w następnych wierszy pisanych będzie kursywa.

Ponieważ dziś mamy do dyspozycji również czcionki inne niż Roman, musi być jakaś możliwość zmiany kroju pisma. Służy do tego polecenie `.fp`, które wymaga podania jednego parametru – liczby oznaczającej pozycję, na której czcionka została zainstalowana w maszynie do składu (to rozwiązanie pochodzi z bardzo zamierzchłych czasów).

Przykładowo, jeśli czcionka Helvetica była zamontowana na trzeciej pozycji i odnosiliśmy się do niej przez literę H, polecenie

```
.fp 3 H
```

instruowało maszynę, by przełączyła się na font Helvetica w pozycji trzeciej. Język groff nadal trzyma się tej archaicznej składni.

Wcięcia i długość wiersza

Domyślnie wiersze tekstu mają długość 6,5 cala. Aby zmienić tę wartość, należy użyć polecenia .ll (ang. *line length*) pobierającego jeden argument. Przykładowo, polecenie

```
.ll 7i
```

powoduje, że wiersze będą miały długość 7 cali. Maksymalna akceptowana długość to zwykle ok. 7,5 cala, więc jeśli chcesz użyć szerszego papieru, musisz odpowiednio przesunąć lewy margines za pomocą polecenia .po (ang. *page offset*). Polecenie .po 0 ustawia lewy margines najdalej jak się da.

Do ustawiania wcięć służy polecenie .in (ang. *indent*), również pobierające jeden argument, który określa wielkość wcięcia, wyrażoną w calach lub punktach:

```
Tekst bez wcięcia.  
.in 0.75i  
Tekst wcięty o trzy czwarte cala.
```

Aby przesunąć prawy margines, dzięki czemu można uzyskać efekt wyróżnienia bloku tekstu, powinieneś zastosować omówione wcześniej polecenie .ll.

```
To zwykły tekst, który jest wystarczająco długi  
żeby osiągnąć koniec wiersza.  
.in 1i  
.ll -1i  
Ten tekst jest wcięty jeden cal od lewej,  
natomiast prawy margines jest przesunięty  
o cal w prawo w porównaniu do standardowych ustawień.  
.in -1i  
.ll +1i  
A to znowu zwyczajne ustawienia. Poprzedni blok został  
wyróżniony z całego tekstu.
```

W tym przykładzie użyte zostały względne wartości przesunięć. Dzięki temu można osiągnąć zamierzony efekt bez znajomości rozmiaru strony. Aby powrócić do poprzednich ustawień, można również użyć polecień .ll i .in bez argumentów.

Raz ustawione rozmiary wcięcia i długość wiersza pozostają w mocy aż do następnego polecenia zmieniającego je. Często jednak zdarza się, że trzeba wciąć tylko jeden wiersz tekstu. Do tego celu służy polecenie .ti (ang. *temporary indent*):

```
To koniec akapitu, po którym nastąpi nowy akapit. Można go wyróżnić wcinając  
pierwszy wiersz.  
.ti 2i  
To jest nowy akapit, którego pierwszy wiersz jest wcięty o cal w lewo, ale  
następne już nie.
```

Tabulatory używane są wtedy, gdy tekst ma być wyświetlany w kolumnach. Domyślnie są one ustawione co pół cala. Rozmieszczenie tabulatorów można zmienić wydając polecenie `.ta` (ang. *tab*)

```
.ta 1i 2i 3i 4i 5i 6i
```

Powyzsze polecenie ustawia tabulatory co jeden cal, od lewej do prawej – podobnie jak w maszynie do pisania. Jeśli jednak chcesz sformatować tabelę, o wiele lepiej do tego celu nadają się makropolecenia `gtbl`, które zostaną omówione w następnym rozdziale.

Inne polecenia

System `groff` udostępnia jeszcze wiele poleceń pozwalających na ustawianie wielkości pojedynczych znaków, wstawianie równań i liter greckich itp. Ponieważ jednak mało prawdopodobne jest, by ktoś chciał używać go do tego typu zadań, pominiemy te zagadnienia. Jeśli jesteś zainteresowany tymi możliwościami, przejrzyj strony `man` dotyczące programu `groff` lub poszukaj jakieś książki poświęconej w całości temu tematowi.

Makropolecenia

Makropolecenie to skrócony zapis zestawu poleceń (lub znaków). Wiele poleceń wykorzystywanych do tworzenia stron `man` to makropolecenia. Oto praktyczny przykład ich zastosowania: założmy, że każdy akapit ma zaczynać się od pustego wiersza, a pierwszy wiersz tekstu ma być wcięty o pół cala. Polecenia pozwalające na uzyskanie takiego efektu mają postać:

```
.sp
.ti +0.5i
```

Zamiast wpisywać je na początku każdego akapitu, można zdefiniować makropolecenie (którego nazwa może być nawet jednoliterowa) i używać go.

Makropolecenie definiuje się poleceniem `.de`, po którym następuje jego nazwa. Definicja makropolecenia z powyższego przykładu może mieć więc następującą postać (powinna znaleźć się gdzieś na początku pliku źródłowego):

```
.de PP
.sp
.ti +0.5i
..
```

Ostatni wiersz, składający się z dwóch kropek, oznacza koniec definicji makropolecenia. Od tej chwili użycie polecenia `.PP` będzie powodowało wykonanie skojarzonego z nim makropolecenia.



Upewnij się, że nazwa makropolecenia nie jest jedną z nazw zarezerwowanych dla poleceń `groff`; w przeciwnym przypadku makropolecenie nie zostanie wykonane.

Makropolecenia mm

Pakiet `mm` (ang. *memorandum macros*) nie jest częścią ani programu `troff`, ani `nroff`, choć oba te programy potrafią z nim współpracować. Program `mm` wczytuje plik źródłowy i, podobnie jak `groff`, w oparciu o niego generuje sformatowany tekst. Większość z dostępnych w pakiecie `mm` makropoleceń służy do formatowania stron `man`. Wielu użytkowników uważa, że polecenia programów `nroff` i `troff` są zbyt dziwaczne i skomplikowane, więc używają `mm`.

Aby wstawić polecenie do pliku źródłowego, powinieneś – podobnie jak w `troff` – rozpocząć je od kropki w nowym wierszu. Makropolecenia `mm` są dość proste i łatwo nauczyć się korzystania z nich. Przyjrzyjmy się najważniejszym z nich.

Akapity i nagłówki

Podobnie jak `groff`, `mm` ignoruje „formatowanie” wprowadzone ręcznie w pliku źródłowym (takie jak znaki nowego wiersza czy tabulacji). Aby wymusić przejście do nowego akapitu, należy wydać polecenie `.P.` Powoduje ono przejście do nowego wiersza i wstawienie jednego pustego wiersza. Akapity są zwykle wyrównane do lewej.

Nagłówki formatuje się polecienniem `.H`. Przykładowo, polecenie

```
.H To jest nagłówek
```

powoduje przejście do nowego wiersza, zastosowanie wytłuszczenia do tekstu `To jest nagłówek` oraz pozostawienie niewielkiego odstępu pomiędzy nagłówkiem i następującym po nim tekstem.

Możliwe jest używanie siedmiu poziomów nagłówków: 1 to poziom najwyższy, 7 – najniższy. Numer poziomu nagłówka podaje się jako argument polecenia `.H`:

```
.H 2 To jest nagłówek drugiego poziomu
```

Makropolecenie generujące nagłówki numeruje je automatycznie; można temu zapobiec wydając polecenie `.HU` (ang. *heading unnumbered*). Aby rozpocząć numerowanie od zadanej liczby (np. po przejściu do następnej części dokumentu), powinieneś wydać polecenie `.nr` (ang. *number register*), po którym następuje poziom nagłówka i numer, od którego należy zacząć liczyć. Polecenie

```
.nr H2 1
```

powoduje numerowanie nagłówków drugiego poziomu od jedynki.

Wyliczenia

Za pomocą makropoleceń `mm` można wygodnie formatować listy i wyliczenia. Do tego celu przeznaczone są polecenia `.LI` (ang. *list*) i `.LE` (ang. *list end*). Następujący fragment pliku źródłowego

```
.LI
pozycja 1
.LI
pozycja 2
.LI
pozycja 3
.LE
```

zawiera definicję listy, przed której elementami pojawią się niewielkie kółka. Jeśli wolałbyś, aby zamiast kólek pojawiły się myślniki – użij polecenia `.DL` (ang. *dash list*). Polecenie `.ML` pozwala wyróżniać pozycje listy dowolnie wybranym znakiem.

Jeśli chcesz utworzyć listę numerowaną, użyj polecenia `.AL`. (ang. *automatic list*). Jeśli polecenie to zostanie wywołane bez argumentów, pozycje listy będą numerowane kolejnymi cyframi arabskimi. Jeśli jego argumentem jest `A` – pozycje listy będą oznaczane kolejnymi literami alfabetu, jeśli `I` – cyframi rzymskimi.

Można również w razie potrzeby zagnieźdzać listy. Następujący plik źródłowy:

```
.AL I
.LI
groff
.AL
.LI
Makropolecenia
.LI
mm
.LE
.LI
gtbl
.LI
geqn
.LE
```

da po sformatowaniu listę o następującej postaci:

```
I. groff
    1. Makropolecenia
    2. mm
II. gtbl
III. geqn
```

Należy zwrócić uwagę na to, czy każda rozpoczęta lista jest zakończona za pomocą polecenia `.LE`. Odrobina praktyki pozwoli Ci łatwo tworzyć tego typu konstrukcje. Jednak nawet niezbyt skomplikowana lista wymaga całkiem sporej liczby poleceń.

Zmiana rodzaju czcionki

Zmiana rodzaju czcionki za pomocą makropoleceń `mm` jest bardzo łatwa. Polecenie `.B` (ang. *bold*) pogrubia czcionkę aż do wystąpienia polecenia `.R` (ang. *restore*), podobnie działa polecenie `.I` (ang. *italic*), załączające kursywę. Jeśli chcesz wyróżnić tylko jedno słowo, możesz je wpisać bezpośrednio po odpowiednim poleceniu, jak w poniższym przykładzie:

```
To jest zwykły tekst.  
.B  
To tekst wytłuszczony.  
To również.  
.R  
To znów zwykły tekst.  
A to pojedyncze  
.B pogrubione  
słowo.
```

Jeśli wyróżnione zostało tylko jedno słowo, nie jest konieczne użycie polecenia .R.

Zmiana czcionki w obrębie jednego wiersza może być przeprowadzona dokładnie w taki sam sposób, jak w groff:

```
To jest tekst \fIpisany kursywa aż do \fRtego miejsca.
```

Przypisy

Do formatowania przypisów służą polecenia .FS (ang. *footnote start*) i .FE (ang. *footnote end*). Wszystkie przypisy wchodzące w skład strony będą zebrane razem i wydrukowane w jej dolnej części. Przypisy są automatycznie numerowane, chyba że do ich oznaczenia wybrany zostanie inny symbol.

```
To jest zwykły tekst  
.FS  
To jest przypis wraz z właściwym numerem.  
.FE  
To znów zwykły tekst.  
.FS *  
To przypis oznaczony gwiazdką.  
.FE  
I jeszcze raz zwykły tekst. W dolnej części strony wydrukowane zostaną dwa przypisy:  
jeden z numerem i jeden z gwiazdką.
```

Do wyróżnienia przypisu możesz użyć każdego dostępnego w groff symbolu, nie wyłączając liter greckich itp.

Podsumowanie

Jak się prawdopodobnie domyślasz, o wielu aspektach pracy z systemami groff i mm nawet nie wspomnieliśmy. Omówiliśmy tylko najbardziej podstawowe zagadnienia, ponieważ groff jest w dzisiejszych czasach używany dość rzadko. Jak wspomnieliśmy wcześniej, więcej na temat tego systemu formatowania tekstów możesz dowiedzieć się ze stron man i poświęconych mu książek.

Inne narzędzia do formatowania tekstu – geqn i gtbl – omówione są w rozdziale 18. „geqn i gtbl”.

TeX i LaTeX, dwa potężne systemy składu, przedstawia rozdział 19. „TeX i LaTeX” oraz rozdział 20. „Drukowanie”.

Rozdział 21. „Linux i multimedia” omawia zagadnienia związane z możliwościami obsługi urządzeń multimedialnych.

Rozdział 18.

geqn oraz gtbl

Tim Parker

W tym rozdziale:

- υ geqn
- υ gtbl

Teraz, kiedy znasz już podstawy systemu `groff`, przyjrzymy się dwóm jego rozszerzeniom: `geqn` oraz `gtbl`. W tym rozdziale dowiesz się:

- υ co to jest `geqn` oraz `gtbl`?
- υ jak w łatwy sposób formatować złożone równania?
- υ jak formatować tabele?

W poprzednim rozdziale pokazaliśmy, w jaki sposób używa się systemu `groff` do tworzenia sformatowanych dokumentów tekstowych nadających się zarówno do wydrukowania, jak i do oglądania na ekranie graficznym. Niestety, tworzenie bardziej skomplikowanych struktur w systemie `groff`, takich jak tabele czy równania, nie jest łatwe. Opracowano więc zestaw makropoleceń ułatwiających te zadania.

Programy `gtbl` i `geqn` są preprocesorami, co oznacza, że pliki źródłowe powinieneś tworzyć tak, jak robiłeś to nie używając tych programów. Można jednak wprowadzać do nich również polecenia programów `gtbl` i `geqn`, które zostaną rozwinięte w polecenia rozpoznawane przez `groff`. Pozostała część pliku źródłowego, nie zawierająca poleceń `gtbl` i `geqn`, jest bez zmian przesyłana do programu `groff`.

geqn

Preprocesor `geqn` jest zaprojektowany do formatowania złożonych równań i drukowania symboli specjalnych. Jego użycie jest potrzebne tylko w przypadku, gdy formatujesz tekst zawierający takie elementy.

Choć `groff` umożliwia formatowanie prostych równań, na pewno nie nadaje się do tworzenia równań zajmujących wiecej niż jeden wiersz. `geqn` jest łatwy do opanowania, a w wielu sytuacjach wręcz niezastąpiony. Polecenia programu `geqn` są zwykle skrótami słów angielskich, dzięki czemu łatwiej je zapamiętać.

Uruchamianie `geqn`

Program `geqn` musi być wywołany przed programem `groff`. Zwykle dokonuje się tego poprzez wydanie polecenia:

```
geqn nazwa_pliku | groff
```

Powoduje ono przetworzenie pliku `nazwa_pliku` przez program `geqn` oraz przesłanie wyników jego działania na wejście programu `groff`. Polecenie

```
geqn plik1 plik2 plik3 | groff
```

powoduje przetworzenie kolejno wszystkich trzech plików.

Pamiętaj, że wiele terminali nie potrafi prawidłowo wyświetlać równań (muszą to być konsole graficzne lub umożliwiające użycie odpowiedniej czcionki). Możliwe, że aby obejrzeć efekty swojej pracy, będziesz musiał je wydrukować.

Równania

Równanie można wstawić informując program `geqn` w którym miejscu zaczyna się ono i kończy za pomocą poleceń `.EQ` (ang. *equation*) oraz `.EN` (ang. *equation end*). Tekst pomiędzy tymi poleceniami traktowany jest jako równanie. Polecenie:

```
.EQ  
b=c*(d+x)  
.EN
```

powoduje sformatowanie równania `b=c*(d+x)`.

Jeśli wpiszesz wiersz `b=c*(d+x)` nie zaznaczając go jako równania, czyli w efekcie przekazując go bezpośrednio do programu `groff`, efekt wyjściowy będzie inny, ponieważ `groff` nie potrafi prawidłowo zinterpretować znaków wchodzących w skład równania.

Równania można numerować – często robi się to w tekstach technicznych – przez podanie odpowiedniego numeru po poleceniu `.EQ`, na przykład:

```
.EQ 15  
b=c*(d+x)  
.EN
```

powoduje umieszczenie przed równaniem numeru 15.

Indeks górnny i dolny

Aby w równaniu zamieścić indeks górnny lub dolny, należy użyć polecień `sup` i `sub`. Słówka te muszą być z obu stron otoczone spacjami. Przykładowo, polecenie:

```
E=mc sup 2
```

daje na wyjściu słynne równanie Einsteina.

Aby zaznaczyć koniec tekstu, który ma być zawarty w indeksie, należy wstawić znak spacji lub tylde (~). Na przykład polecenie:

```
x=(z sup 2)+1
```

daje na wyjściu:

```
x=(z^2)+1
```

a prawdopodobnie nie o to chodziło. Zamiast tego, powinieneś wprowadzić któreś z poleceń:

```
x=(z sup 2 )+1  
x=(z sup 2~)+1
```

Spacja lub tylde oznacza koniec indeksu górnego. Oba powyższe polecenia dają na wyjściu równanie:

```
x=(z^2)+1
```

Możesz również uzyskać indeks górnny w indeksie górnym oraz dolny w dolnym:

```
y sub x sub 3
```

lub zastosować oba indeksy równocześnie:

```
x sub 3 sup 18
```

Ponieważ spacja jest używana do oznaczania końca tekstu wchodzącego w skład indeksu górnego lub dolnego, może to spowodować drobne problemy w przypadku, gdy trzeba jej użyć do rozdzielenia symboli. Sposobem na obejście tych problemów jest zastosowanie nawiasów klamrowych, w których zawrzeć należy cały tekst należący do danego indeksu:

```
w sup {x alpha y}
```

Powyższy przykład ilustruje równocześnie możliwość wykorzystania liter greckich. Nawiązy klamrowe można zagnieździć:

```
omega sub { 2 pi r sup { 2 + rho }}
```

Wypróbuj sam te polecenia i obejrzyj efekty ich działania.



Choć może się wydawać, że litery greckie są przydatne tylko w skomplikowanych równaniach, często wygodnie jest ich używać również w prostych formułach. Warto wiedzieć, jak je uzyskać i jak wyglądają na papierze.

Ułamki

Do tworzenia prawidłowo sformatowanych ułamków służy polecenie `over. geqn` automatycznie dostosowuje długość linii oddzielającej licznik i mianownik. Na przykład polecenie

```
a=2b over {3c alpha}
```

da w wyniku równanie, po którego prawej stronie jest ułamek – tak, jakby zostało napisane na papierze.

Można oczywiście używać równocześnie indeksów i ułamków, co pozwala na uzyskanie bardziej złożonych równań:

```
{alpha + beta +gamma sup 3} over {3 sub {4 + alpha}}
```

W takim przypadku najpierw przetwarzane są polecenia `sub` i `sup`, a potem `over`, co daje zamierzony efekt.

Pierwiastek kwadratowy

Aby uzyskać symbol pierwiastka kwadratowego, należy użyć polecenia `sqrt - geqn` zadba o to, by symbol pierwiastka prawidłowo otaczał cały tekst, który ma znaleźć się „pod pierwiastkiem”. Bardzo duże pierwiastki (szczególnie o długości przekraczającej jeden wiersz tekstu) nie wyglądają jednak najlepiej po wydrukowaniu – w takim przypadku warto rozważyć zapisanie pierwiastka jako potęgi o wykładniku 0,5.

Polecenie `sqrt` jest łatwe w użyciu, poniższy przykład:

```
sqrt a+c -1 over sqrt {alpha + beta}
```

powoduje „nałożenie” pierwiastków na wyrażenia `a+c` oraz `alpha + beta`.

Sumy, teoria zbiorów i całki

Aby uzyskać symbol sumy, użyj polecień `sum, from` i `to` (dwa ostatnie definiują granice sumowania). Użycie polecenia:

```
sum from x=1 to x=100 x sup 2
```

powoduje wstawienie symbolu sumy kwadratów liczby x dla x zmieniającego się od 1 do 100. Można również wydać polecenie:

```
sum from x=1 to {x= inf} x sup 2
```

dające podobne efekty, z tym, że górną granicą sumowania jest nieskończoność. Klamry zapewniają prawidłową interpretację polecenia. Jeśli któryś z elementów `from` lub `to` nie jest podany, nie jest on drukowany.

Używanie symbolu całki możliwe jest dzięki poleceniu `int`. Jego składnia jest taka sama jak składnia polecenia `sum`, np.:

```
int from x=1 to {x= inf} x sup 2
```

Drukowanie symbolu granicy możliwe jest za pomocą polecenia `lim`:

```
lim from n=1 xy sup 3 =9
```

Inne zarezerwowane słowa to `union` i `inter`, używane w teorii zbiorów (odpowiednio unia i przecięcie zbiorów).

Nawiasy i kolumny

Kiedy równania stają się bardziej skomplikowane, konieczne staje się użycie kilku poziomów nawiasów. Aby uzyskać taki efekt, należy zastosować polecenia `left` i `right`:

```
left { b over d+1 right } = left ( alpha over {beta+ gamma} right )
```

W powyższym przykładzie po lewej stronie równania znajdzie się ułamek w nawiasie klamrowym, a po prawej wyrażenie w nawiasach okrągłych. Można oczywiście zagnieździć nawiasy – `geqn` sam odpowiednio dostosuje ich wysokość. Nawiasy klamrowe i kwadratowe są zwykle większe niż okrągłe.

Oznaczenia sufitu i podlogi uzyskać można za pomocą poleceń `left floor`, `right floor`, `left ceiling` i `right ceiling`, na przykład:

```
left ceiling x over alpha right ceiling > left floor beta over 2 right floor
```

Aby uzyskać kolumnę elementów, należy użyć polecenia `pile`. Poniższy przykład daje kolumnę trzech elementów otoczoną dużymi nawiasami kwadratowymi:

```
X = left [ pile {a above b above c } right ]
```

Macierze

Tworzenie macierzy wymaga nieco więcej wysiłku. Mógłbyś prawdopodobnie zrobić to za pomocą polecenia `pile`, ale jeśli wysokości elementów są różne, nie znajdą się one w jednej linii. Z tego powodu należy używać polecenia `matrix`. Ogólnie jego składnia jest następująca:

```
matrix {
  ccol { elementy }
  ccol { elementy }
}
```

Polecenie to powoduje utworzenie macierzy, której elementy są wyśrodkowane. Jeśli chcesz dosunąć je do lewej lub prawej strony, użyj zamiast `ccol` słowa `lcol` lub `rcol`. Przykładowe polecenie:

```
matrix {
  ccol { x sub 11 above x sub 12 }
  ccol { x sub 21 above x sub 22 }
}
```

generuje macierz o wymiarach 2x2 o następującej postaci:

$$\begin{matrix} x_{11} & x_{21} \\ x_{12} & x_{22} \end{matrix}$$

Wszystkie kolumny macierzy muszą składać się z takiej samej liczby elementów.



Niektóre pakiety matematyczne dopuszczają tworzenie macierzy z różną ilością elementów w kolumnach. Nie jest to podejście właściwe, ponieważ taki twór przestaje być macierzą. Wszystkie kolumny macierzy powinny składać się z takiej samej liczby elementów, podobnie jak wiersze

Cudzysłowy

Ciąg znaków umieszczony w cudzysłowie nie jest interpretowany przez `geqn`. Takie rozwiązanie

przydaje się, gdy zachodzi potrzeba wypisania tekstu będącego nazwą zastrzeżoną, np.:

```
italics "beta" = beta + gamma
```

W powyższym przykładzie zamiast greckiej litery beta przed znakiem równości pojawi się tekst `beta`.

Zmiana czcionki

Rodzaj i rozmiar czcionki można zmieniać bardzo podobnie jak w programie `groff`. Czcionką domyślną jest czcionka Roman o wysokości 10 punktów. Jeśli chcesz, aby litery były wytłuszczone, powinieneś wydać polecenie `bold`. Polecenie `italic` powoduje, że tekst będzie pisany kursywą.

```
x = y bold alpha
```

Polecenie `fat` powoduje drukowanie liter poszerzonych. Polecenia te dotyczą tylko tekstu następującego bezpośrednio po nich, więc jeśli chcesz zastosować je do większej partii tekstu, musisz umieścić ją w nawiasach klamrowych:

```
x = y*2 bold { alpha + beta }
```

Zmiana rozmiaru czcionki możliwa jest dzięki poleceniu `size`:

```
size 16 { alpha + beta }
```

Dopuszczalne są również zmiany względne, np. `size +2`.

Jeśli chcesz, aby zmiany dotyczyły całego równania, możesz na początku bloku równania użyć poleceń `gsize` (ang. *global size*) i `gfont` (ang. *global font*):

```
.EQ  
gsize 14  
gfont H  
.....
```

Dzięki temu łatwo sformatować równanie w wymagany sposób.

Używanie programu `geqn`

Jak widzisz, stosowanie programu `geqn` do formatowania równań nie jest szczególnie trudne, a daje bardzo dobre efekty. Powinieneś pobawić się chwilę tym systemem, aby nabrać wprawy. Oprócz przedstawionych tu najbardziej podstawowych poleceń, dostępnych jest jeszcze wiele innych. Ich opis znaleźć możesz na stronach `man` i w dobrych książkach omawiających system `groff`.

gtbl

Program `gtbl` został opracowany po to, by ułatwić formatowanie tabel i wyliczeń składających się z kilku kolumn. Jego polecenia nie są trudne w użyciu, ale wyglądają dość dziwacznie – najlepszą metodą zapoznania się z nimi jest przestudiowanie kilku przykładów.

Aby użyć `gtbl`, w pliku źródłowym powinieneś zatrzymać dwa specjalne polecenia: `.TS` (ang. *table start*) i `.TE` (ang. *table end*), a pomiędzy nimi dane dotyczące tabeli. Polecenia te zostaną następnie przetłumaczone przez program `gtbl` na postać rozpoznawaną przez `groff`.

Każda z tabel jest niezależna od innych, co oznacza, że dla każdej z osobna trzeba podać informacje dotyczące formatowania; nie można założyć, że zostaną użyte poprzednie wartości. Tabela musi zawierać trzy typy informacji: dane, które mają znaleźć się w tabeli, opcje kontrolujące zachowanie `gtbl` oraz polecenia formatujące tabelę. Ogólnie, składnia polecenia powodującego wstawienie do dokumentu tabeli jest następująca:

```
.TS
opcje;
format.
dane
.TE
```

Przyjrzyjmy się każdemu z elementów tego polecenia z osobna, by później połączyć je i utworzyć prawidłowo sformatowaną tabelę.

Uruchamianie programu gtbl

Ponieważ `gtbl` jest preprocesorem, należy go wywołać z argumentem określającym nazwę pliku źródłowego, zaś efekty jego działania przesyłać na wejście właściwego programu formatującego:

```
gtbl plikzrodlowy | groff
```

Można również użyć obu preprocesorów `geqn` i `gtbl` równocześnie, na przykład tak:

```
geqn plikzrodlowy | gtbl | groff
```

Opcje

Po poleceniu `.ts` może wystąpić jeden wiersz zawierający opcje, które dotyczą całej tabeli. Jeśli podajesz więcej niż jedną opcję, musisz rozdzielić je spacjami, przecinkami lub tabulatorami, natomiast po ostatniej musi wystąpić średnik. `gtbl` obsługuje następujące opcje:

center	powoduje wyśrodkowanie zawartości tabeli (domyślnie – wyrównywanie do lewej),
expand	poszerza tabelę do aktualnej długości wiersza,
box	otacza tabelę ramką,
allbox	otacza każdy element tabeli ramką,
doublebox	otacza tabelę podwójną ramką,
tab (n)	używa znaku <code>n</code> zamiast tabulatora jako separatora danych,
linesize (n)	używa rozmiaru <code>n</code> punktów dla poszczególnych wierszy,
delim (mn)	używa znaków <code>m</code> i <code>n</code> jako ograniczników równań.

Kiedy `gtbl` próbuje sformatować tabelę, stara się zmieścić ją na jednej stronie, nawet jeśli musi pozostawić poprzednią stronę częściowo pustą. Może to czasem być przyczyną problemów, ponieważ zdarza się, że `gtbl` źle oblicza rozmiar tabeli przed jej wygenerowaniem, szczególnie jeśli są w niej osadzone polecenia zmieniające rozmiar czcionki lub odstępy między wierszami. Aby uniknąć tych problemów, niektórzy użytkownicy

otaczają całą tabelę makropoleceniem `.DS` (ang. *display start*) i `.DE` (ang. *display end*). W przypadku prostych tabel nie pojawiają się żadne problemy.

Format

Sekcja opisująca format tabeli determinuje sposób wyświetlania poszczególnych kolumn. Każdy wiersz sekcji formatu odpowiada wierszowi wygenerowanej tabeli. Jeśli podanych jest mniej wierszy formatu, niż wierszy danych, pozostała część danych zachowuje ostatnio zdefiniowany format. Pozwala to na łatwe tworzenie nagłówków w innym formacie i użycie jednolitego formatowania reszty tabeli. Sekcja formatu musi kończyć się kropką.

Każdy wiersz tej sekcji zawiera litery kluczowe odnoszące się do kolumn tabeli. Powinny być one rozdzielone spacjami albo znakami tabulacji, ponieważ poprawia to czytelność.

Wielkość tych liter nie ma znaczenia (można używać zarówno wielkich, jak i małych liter). Oto ich funkcje:

- l** wyrównanie pola do lewej,
- r** wyrównanie pola do prawej,
- c** wyśrodkowanie pola,
- n** wyrównanie danych numerycznych do kropki,
- a** wyrównanie do lewej tak, by najdłuższy wpis był wyśrodkowany,
- s** zastosowanie do kolumny bieżącej formatu poprzedniej kolumny.

Podana poniżej przykładowa sekcja formatu zawiera jedną literę dla każdej kolumny; ostatni wiersz definiuje format obowiązujący do końca tabeli:

c s s
l n n.

W tym przykładzie, w pierwszym wierszu tabeli pierwsza, druga i trzecia kolumna jest wyśrodkowana (litera `s` powoduje, że dana kolumna ma taki sam format, jak kolumna poprzednia). W drugim wierszu i wszystkich następnych pierwsza kolumna jest wyrównana do lewej, a druga i trzecia zawiera dane numeryczne wyrównane do kropki. Koniec sekcji oznaczony jest kropką. Jeśli chcesz, możesz wszystkie te dane umieścić w jednym wierszu, na przykład tak:

c s s, l n n.

Tabela sformatowana w ten sposób wyglądać będzie mniej więcej tak:

Wyśrodkowany nagłówek

Dana1	12.23	231.23
Dana2	3.23	45.2
Dana3	45	124.344
Dana4	3.2	2.3

Dane liczbowe wyrównane są tak, by kropki dziesiętne znajdowały się w jednej linii. Czasem jednak będziesz chciał wymusić zmianę formatu dla konkretnej danej. Aby przesunąć kropkę dziesiętną, użyj symbolu specjalnego \&. Nie jest on drukowany w tabeli. Spójrzmy na przykładowe dane (pierwsza kolumna to dane źródłowe, druga przedstawia, jak zostaną one sformatowane):

14.5	14.5
13	13
1.253	1.253
3\&1.21	31.21
53.2	53.2
6\&2.23	62.23

Można zauważyć, że liczby ułożone są tak, by kropki dziesiętne były jedna pod drugą, za wyjątkiem tych, w których użyto symbolu \&. Jeśli dana nie zawiera kropki dziesiętnej, program gtbl przyjmuje, że występuje ona po ostatniej cyfrze.

Poniżej zebrane zostały inne symbole specjalne, które mogą być używane (w sekcji formatu) do nietypowego formatowania tabeli, czyniąc ją nieco bardziej atrakcyjną:

- linia pozioma zamiast danych;
- = podwójna linia pozioma zamiast danych;
- | pomiędzy danymi, wstawia między kolumnami pionową linię; przed pierwszą daną w wierszu wstawia linię po lewej stronie tabeli; po ostatniej danej wstawia linię po prawej stronie tabeli;
- || pomiędzy danymi, wstawia podwójną pionową linię między kolumnami;
- e/E ustawia równe szerokości kolumn; wszystkie kolumny, dla których ustawiona jest opcja e będą miały równe szerokości;
- f/F x powoduje użycie zadanego rodzaju lub rozmiaru czcionki (x to nazwa czcionki lub jej rozmiar);
- n x określa wielkość odstępów pomiędzy kolumnami;

- p/P x** zmienia rozmiar wpisu zgodnie z następującą po nim liczbą; obsługiwane są również zmiany względne;
- t/T** powoduje, że szerokie obiekty będą zaczynać się od samej góry wiersza (zwykle są one wyśrodkowane w pionie);
- v/V** ustawia odstęp między wierszami;
- w/W** ustawia minimalną szerokość kolumny.

Porządek tych znaków w wierszu formatu nie jest istotny. Przykładowy wpis

```
np14w(2.5i) fi
```

formatuje pole liczbowe (`n`), pisane kursywą (`fi`) o wysokości 14 punktów (`p14`) i minimalnej szerokości kolumny 2.5 cala (`w(2.5i)`).

Może również zajść potrzeba zmiany formatu tabeli gdzieś w jej środku – na przykład by wstawić podsumowanie. Jeśli musisz to zrobić, zastosuj polecenie `.T&` (ang. *table continue*).

Dane

Dane, które mają znaleźć się w tabeli, wpisywane są po określeniu formatu. Poszczególne kolumny oddzielać należy tabulatorami albo innymi znakami określonymi za pomocą opcji `tab`. Każdy wiersz danych odpowiada jednemu wierszowi tabeli. Długie wiersze danych można kontynuować dla wygody w nowym wierszu pliku źródłowego wstawiając na końcu poprzedniego wiersza symbol `\`.

Każdy wiersz rozpoczynający się od kropki jest traktowany jak polecenie `groff` i nie jest przetwarzany. Jeśli wiersz danych składa się tylko ze znaków podkreślenia bądź równości, jest traktowany tak, jakby miał szerokość całej tabeli.

W tabeli można osadzać bloki tekstu używając poleceń `T{` (początek tekstu) oraz `}T` (koniec tekstu). Dzięki temu możliwe jest umieszczanie w tabelach tekstów, które trudno byłoby wprowadzić jako tekst rozdzielany znakami tabulacji.

Przykłady

Najlepszym sposobem na zrozumienie zasady działania `gtbl` jest przestudiowanie kilku przykładów. Oto polecenie formatujące prostą tabelkę:

```
.TS
doublebox;
c c c, l l n.
Imie      Wydział     Telefon
Michał    8A          2245
Piotr     6D          1246
Pawel     2C          6436
```

```
Anna      1A      1363  
.TE
```

Wszystkie dane rozdzielone są znakami tabulacji. Wygenerowana tabela składa się z trzech kolumn; pierwszy wiersz zawiera wyśrodkowany tekst. W następnych wierszach pierwsza i druga kolumna wyrównana jest do lewej, natomiast trzecia ma format numeryczny. Cała tabela otoczona jest podwójną ramką.

Odrobinę bardziej złożony przykład przedstawia tabelę zawierającą tytuł, po którym następują dane. Każdy element tabeli jest otoczony ramką.

```
.TS  
allbox;  
c s s  
c c c  
n n n .  
Wyniki  
Wschod    Zachod    Polnoc  
15        12        14  
12        12        18  
36        15        24  
.TE
```

Spróbuj sam wpisać te przykłady, aby przekonać się, jakie są wyniki działania poszczególnych poleceń. Kiedy zrobisz pierwszy krok, używanie `gtbl` przestaje być takie straszne.

Podsumowanie

Choć dostępne dziś graficzne edytory tekstu spowodowały znaczne zmniejszenie się popularności programów `geqn` i `gtbl`, niektórzy zatwardzali użytkownicy UNIX-a wciąż ich używają. Zdarzyć się również może, że nie będziesz mógł w swoim edytorze tekstu sformatować równania dokładnie tak, jakbyś sobie tego życzył. Wtedy będziesz zmuszony do powrotu do korzeni. Innym powodem używania tych programów jest fakt, że graficzne edytory tekstu są dość drogie, więc `geqn` i `gtbl` są świetną alternatywą dla tych użytkowników, którzy z ich możliwości korzystać muszą rzadko, więc nie mają potrzeby kupowania drogiego edytora.

`gawk`, poręczny i szybki język programowania dla zaawansowanych użytkowników i administratorów, omówiony jest w rozdziale 25. „`gawk`”.

Perl, inny język programowania, który jest bardzo popularny w skryptach dla stron WWW, przedstawiony jest w rozdziale 28. „`Perl`”.

Smalltalk/X, oparta na systemie X implementacja zorientowanego obiektowo języka programowania, omówiony jest w rozdziale 31. „`Smalltalk/X`”.

Rozdział 19.

TeX i LaTeX

Peter MacKinnon

W tym rozdziale:

- υ Edycja a skład tekstu
- υ TeX
- υ LaTeX – rozszerzenie systemu TeX
- υ VirTeX i IniTeX

TeX (nazwę tę należy wymawiać *tech*) jest systemem formatowania tekstu opracowanym przez Donalda Knutha. Pozwala on na tworzenie profesjonalnie wyglądających dokumentów przez osadzanie poleceń formatujących systemu TeX w pliku ASCII. Plik tekstowy może następnie zostać skompilowany do formatu `.dvi` (ang. *device independent file*, plik niezależny od urządzenia), który może być oglądzany na ekranie graficznym za pomocą programu `xdv`i lub zamieniony na format postscriptowy i wydrukowany.

TeX to program o bardzo dużych możliwościach. Pozwala łatwo ustawać parametry składu, takie jak rozmiar czcionki, rozmiar strony czy odstępy między wierszami. Działa również jako język programowania, pozwalając tworzyć makropolecenia definiujące bardziej abstrakcyjne jednostki tekstu, jak dokumenty, nagłówki czy akapity. Dzięki temu zamiast spędzać godziny na formatowaniu tekstu możesz skupić się na samej jego treści. Kluczową sprawą dla inżynierów i naukowców jest fakt, że TeX doskonale radzi sobie z formatowaniem nawet bardzo złożonych równań i formuł matematycznych.

Edycja a skład tekstu

Wartość użytkowa dokumentu może być ograniczona przez jego wygląd. Rozważmy dwa dokumenty: pierwszy z nich jest dobrze zorganizowany, z jasno zdefiniowanymi jednostkami tekstu, takimi jak rozdziały, nagłówki i akapity; drugi nie jest podzielony na akapity. Pierwszy jest o wiele łatwiejszy do przeczytania. Dzięki temu, że nie trzeba

koncentrować się na tym, by nie „zgubić się” w tekście, można skoncentrować się na jego treści. Mimo wysiłków autora w kierunku stworzenia jak najbardziej wartościowego dzieła, czytelnik może zginąć w bałaganie typograficznym.

Przy publikowaniu książek autorzy zwykle są odpowiedzialni tylko za ich treść. Szatę graficzną opracowuje ktoś zupełnie inny, następnie do akcji wkraczają specjaliści od składu. TeX bierze na siebie projektowanie i skład tekstu, pozwalając Ci być swoim własnym wydawcą. Daje pełną kontrolę nad wyglądem publikowanego materiału, pozwalając w dalszym ciągu skupić się na jego treści.

TeX

Plik wejściowy dla programu TeX może być przygotowany za pomocą dowolnego edytora tekstu, jak `vi` czy `emacs`. Wpiszmy na przykład do pliku `arkana.tex` następujący tekst:

```
Czy uważacie, ze Alfred Hitchcock odniósł by tak wielki sukces jako reżyser  
gdyby w jego filmach nie wystąpiły takie gwiazdy jak Cary Grant  
czy James Stewart? Trudno odpowiedzieć na to pytanie...  
\bye
```

Po zapisaniu tego pliku można użyć programu TeX, aby przekonwertować go do formatu `.dvi`:

```
$ tex arcana
```

Plik wyjściowy, `arkana.dvi`, zawiera wpisany wcześniej tekst. Plik ten można oglądać lub drukować za pomocą urządzeń wyjściowych różnego typu. Przykładowo, jeśli chcesz go wydrukować na drukarce postscriptowej, powinieneś wydać polecenie (zakładamy, że domyślna drukarka obsługuje język PostScript):

```
$ dvi2ps arkana.dvi | lpr
```

Jeśli chcesz najpierw sprawdzić, jak dokument będzie wyglądał, użyj aplikacji `xdvi` pracującej w systemie X:

```
$ xdvi arkana.dvi &
```

Polecenie `tex` tworzy również plik `arkana.log`, w którym zawarte są wszystkie komunikaty o błędach i ostrzeżenia, oraz inne informacje, takie jak ilość stron dokumentu wyjściowego. Główną zaletą formatu `.dvi` jest jego całkowita przenośność, w szczególności do innych systemów UNIX-owych.

Proste formatowanie tekstu

Większość pracy przy tworzeniu dokumentu systemu TeX to wpisywanie jego treści. Jak pokazano wcześniej, łatwo jest stworzyć dokument nie posiadający żadnego spe-

cjalnego formatowania. Jednym polecienniem, które trzeba w tym celu dodać do pliku tekstopożego, jest polecenie `\by`, oznaczające koniec dokumentu. Zaczyna się ono od jednego ze znaków specjalnych rozpoznawanych przez system TeX; są to: `\, {, }, -, #, $, %, ^, &` oraz spacja. Ich znaczenie wyjaśnimy w dalszej części tego rozdziału.

Główną zaletą systemu TeX jest jego „inteligentne” rozpoznawanie bloków tekstu. Słowem jest dowolna sekwencja znaków otoczona z obu stron znakami białymi. Ilość znaków białych pomiędzy wyrazami nie ma znaczenia, TeX traktuje je tak samo jak pojedynczą spację. Zdania są rozpoznawane po tym, że po ostatnim wyrazie następuje kropka, znak zapytania, wykrzyknik lub dwukropki. Akapit rozpoznawany jest dzięki pustemu wierszowi po zakończonym zdaniu. Dodatkowe puste wiersze są ignorowane. Przykładowy tekst:

```
Rozgrywki piłkarskiej ekstraklasy
będą zawieszone do 19 sierpnia - postanowili wczoraj prezesi klubów I i
II ligi.
Zagraja dopiero, gdy PZPN podpisze umowę
w sprawie powstania Autonomicznej Ligi Polskiej.
```

Szefowie klubów zrzeszeni w Lidze, która wczoraj powołano, mają sami decydować o terminarzu i regulaminie rozgrywek.

Zostanie sformatowany w identyczny sposób jak tekst:

```
Rozgrywki piłkarskiej ekstraklasy będą zawieszone do 19 sierpnia -
postanowili wczoraj prezesi klubów I i II ligi. Zagrają dopiero, gdy PZPN
podpisze umowę w sprawie powstania Autonomicznej Ligi Polskiej.
```

Szefowie klubów zrzeszeni w Lidze, która wczoraj powołano, mają sami decydować o terminarzu i regulaminie rozgrywek.

W pliku źródłowym można również używać komentarzy. Komentarz zaczyna się od znaku `%` i nie jest umieszczany w pliku wyjściowym. Tekst:

```
Początek tekstu % kilka słów komentarza
% może niepotrzebnego...
i koniec tekstu.
```

Zostanie potraktowany tak samo, jak tekst:

Początek tekstu i koniec tekstu.

Polecenie `\par` powoduje przejście do następnego akapitu, daje więc taki sam efekt jak wstawienie pustego wiersza. Tekst:

Pierwszy akapit. \par Drugi akapit.

zostanie sformatowany następująco:

```
Pierwszy akapit.
Drugi akapit.
```

Polecenie `\noindent` zapobiega wcięciu pierwszego wiersza nowego akapitu:

```
Pierwszy akapit.
```

```
\noindent Drugi akapit.
```

Powyższy tekst zostanie sformatowany jako:

```
Pierwszy akapit.  
Drugi akapit.
```

Aby wymusić wstawienie dodatkowej spacji, należy poprzedzić ją znakiem `\`, który zapobiegnie interpretowaniu jej przez TeX; tekst źródłowy:

```
Myślę ze potrzebuje jednej albo dwu \ \ dodatkowych spacji.  
Jestem tego pewny.
```

zostanie sformatowany tak:

```
Myślę ze potrzebuje jednej albo dwu     dodatkowych spacji.  
Jestem tego pewny.
```

Czcionki

Czcionka to zestaw znaków charakteryzujących się podobnym stylem i wielkością. Domyslną czcionką w systemie TeX jest czcionka Roman. Ustawienie to można zmienić, używając wewnętrznych nazw przypisanych do czcionek ładowanych z osobnych plików. Można również dodawać nowe definicje czcionek. Domyślnie w systemie TeX można używać definicji `\rm` (czcionka Roman), `\tt` (czcionka maszyny do pisania), `\bf` (czcionka wytłuszczonej), `\sl` (czcionka pochylona) i `\it` (kursywa). TeX używa wybranej czcionki aż do następnego zmieniającego ją polecenia. Tekst:

```
To jest czcionka Roman, \tt to czcionka maszyny do pisania, \it to  
kursywa, \rm a to znow Roman.
```

zostanie sformatowany następująco:

```
To jest czcionka Roman, to czcionka maszyny do pisania, to kursywa,  
a to znow Roman.
```

Aby zdefiniować nową czcionkę (ściślej: alias – przyp. *thum.*) o zadanej wielkości, należy użyć polecenia:

```
\font \nazwa_czcionki=czcionka_pomocnicza
```

Aby użyć 12-punktowej czcionki Roman, można zmienić definicję `\rm` tak, aby używała czcionki pomocniczej `cmr12`:

```
\font\rm=cmr12
```

```
Zmieniamy czcionkę z dwunastki \rm na 10-punktową.
```

Powyższy tekst źródłowy zostanie sformatowany tak:

```
Zmieniamy czcionke z dwunastki na 10-punktowa.
```

W każdym zestawie znaków dostępnych jest do 256 różnych symboli, między innymi cyfry, małe i wielkie litery, znaki przestankowe i operatory arytmetyczne, których używa się najczęściej. Symbole nie mające reprezentacji na klawiaturze mogą być wstawiane dzięki poleceniu `\char`. Parametrem tego polecenia jest liczba całkowita, oznaczająca położenie żądanego symbolu w tablicy znaków. Przykładowy tekst:

```
TeX zinterpretuje znak \char 37 jako początek komentarza,  
ale zignoruje znak \char 43.
```

sformatowany zostanie jako:

```
TeX zinterpretuje znak % jako początek komentarza, ale zignoruje znak +.
```

Kontrolowanie odstępów

Pokazaliśmy już, jak można wstawić do formatowanego dokumentu kilka dodatkowych spacji. Zobaczmy teraz, w jaki sposób dokładniej kontrolować odstępy międzywyrazowe w większych fragmentach tekstu. TeX posiada kilka poleceń służących do tego celu i rozpoznających następujące jednostki:

Jednostka	Znaczenie
em	Szerokość litery M (zależnie od wybranej czcionki)
in	Cale
pt	Punkty (1 cal to 72 punkty)
mm	Milimetry (1 cal to 25.4 milimetra)

Właściwy argument polecenia jest liczbą określającą wielkość odstępu wyrażoną w jednej z powyżej podanych jednostek.

Polecenie `\hskip` pozwala wstawić odstęp w poziomie pomiędzy fragmentami wiersza:

```
\tt Odstęp \hskip 0.5in aż do tego miejsca.
```

Powyższy tekst zostanie sformatowany mniej więcej tak:

```
Odstęp aż do tego miejsca.
```

Możesz również podać liczbę ujemną, co spowoduje przesunięcie tekstu w lewo.

Polecenie `\hfil` wstawia poziomy odstęp w akapicie wtedy, jeśli jest na niego miejsce. Interesujący jest fakt, że TeX wstawia to polecenie niejawnie do każdego akapitu. Pamiętając o tym, możesz używać go do przesuwania tekstu w lewo, w prawo lub środkowania go pomiędzy marginesami, na przykład tak:

```
\noindent \hfil Tekst wyśrodkowany. \par
```

Daje to w wyniku akapit sformatowany następująco:

```
Tekst wyśrodkowany.
```

Polecenie `\vskip` służy do wstawiania odstępu w pionie pomiędzy akapitami. Aby wstawić odstęp o wysokości 4 cm przed i po bieżącym akapicie, należy użyć polecenia:

```
\vskip 40mm
```

TeX umożliwia również stosowanie małych, średnich i dużych odstępów za pomocą polecień

```
\smallskip, \medskip i \bigskip.
```

Pionowym odpowiednikiem polecenia `\hfil` jest `\vfil`, które wstawia odstępy w pionie pomiędzy akapitami pod warunkiem, że jest na to dość miejsca. TeX wstawia to polecenie domyślnie na koniec dokumentu.

Można również wymusić przejście do nowego wiersza lub strony za pomocą polecenia `\break`. Jeśli występuje ono wewnątrz akapitu, powoduje przejście do nowego wiersza, zaś pomiędzy akapitami – przejście do nowej strony. Z kolei za pomocą polecenia `\nobreak` można zaznaczyć obszary dokumentu, które nie mogą zostać rozdzielone.

Układ strony

Strona składa się z nagłówka, stopki i treści. Nagłówek i stopka zwykle zawierają takie informacje, jak numer strony, tytuł rozdziału czy nagłówek podrozdziału. Zmieniając sposób wyświetlania tych informacji, projektujesz końcowy wygląd dokumentu.

Polecenia `\headline` oraz `\footline` wymagają jednego argumentu, który określa zarówno stopkę i nagłówka. Mogą się w nim zawierać na przykład takie elementy, jak numer strony i polecenie `\hfil`:

```
\headline={\hfil \the\pageno}
\footline={\hfil}
```

Powyzsze polecenia tworzą dla każdej strony wyrównany do prawego marginesu nagłówek zawierający numer strony oraz pustą stopkę.

Szerokość akapitu (czyli odstęp między marginesami) można zmieniać za pomocą polecenia `\hsize`. Przykładowy tekst:

```
\hsize=2in
To jest tekst szeroki na dwa cale, ale równie dobrze mógłby być szerszy
lub węższy.
```

zostanie sformatowany mniej więcej tak:

```
To jest tekst szeroki na dwa
cale, ale równie dobrze
mogliby byc szerszy lub
węższy.
```

Marginesy mogą być przesuwane do wewnątrz lub na zewnątrz za pomocą polecień `\leftskip` i `\rightskip`. Użycie ich z parametrami o wartości dodatniej powoduje przesunięcie marginesów do wewnątrz, wartość ujemna przesuwa je na zewnątrz. W podobny sposób, poleceniem `\parindent`, kontrolowane są wcięcia.

Polecenia `\baselineskip` i `\parskip` pozwalają ustawać odstępy pomiędzy wierszami (interlinię) i akapitami, na przykład:

```
\baselineskip=0.15in  
\parskip=0.3in
```

Interlinia określa odstęp pomiędzy liniami bazowymi (ang. *baseline*; jest to linia będąca dolną granicą rysunku liter bez dolnych wydłużen – przyp. tłum.) w kolejnych wierszach.

Grupowanie

TeX używa wybranej przez użytkownika czcionki czy stylu tekstu aż do wprowadzenia kolejnej zmiany, co nie zawsze jest wygodne. Grupowanie pozwala zastosować zmiany lokalnie w obrębie bloku tekstu. Po zakończeniu przetwarzania bloku (grupy) przywrocane jest pierwotne formatowanie.

Istnieją dwa sposoby grupowania tekstu. Jednym z nich jest użycie polecień `\begingroup` i `\endgroup`, drugim – użycie nawiasów klamrowych. Choć oba spełniają dobrze swoją rolę, nawiasy klamrowe używane są również do przekazywania parametrów polecen, dlatego należy ich używać z dużą ostrożnością.

Ilustracją grupowania niech będzie tekst

```
Zobaczmy jak działa \begingroup \it grupowanie {\bf i do czego } może się  
\endgroup przydać?
```

który zostanie sformatowany następująco:

```
Zobaczmy jak działa grupowanie i do czego może się przydać?
```

Jak widać w powyższym przykładzie, grupy można zagnieżdżać.

Symbole matematyczne

Jedną z najważniejszych zalet TeX-u jest możliwość wygodnego formatowania złożonych formuł matematycznych. Głównie z tego powodu stał się on tak popularny w środowiskach naukowych.

TeX odróżnia równania, które mają pojawić się w tekście od tych, które muszą znaleźć się w osobnym wierszu. Aby zaznaczyć pierwszy rodzaj równania, należy otoczyć je znakami \$, na przykład tak:

```
Rozwiązaniem równania $2+3=x$ jest liczba $x=5$.
```

Po sformatowaniu tekst powyższy przyjmie postać:

```
Rozwiązaniem równania 2+3=x jest liczba x=5.
```

Aby równanie zostało umieszczone w osobnym wierszu, należy otoczyć je podwójnymi znakami \$:

Rozwiązaniem równania $2+3=x$ jest liczba $x=5$.

Powyższy tekst po sformatowaniu będzie wyglądał następująco:

```
Rozwiązańiem równania  
2+3=x  
jest liczba  
x=5.
```

Tabela 19.1 przedstawia niektóre z symboli matematycznych generowanych przez system TeX, wraz z poleceniami je wywołującymi i opisem.

Tabela 19.1. Niektóre symbole matematyczne generowane przez system TeX

Symbol	Polecenie	Znaczenie
π	\pi	Liczba pi
\sum	\sum	Symbol sumy
{	\{	Lewa klamra
}	\}	Prawa klamra
\int	\int	Symbol całki
\leq	\leq	Mniejsze lub równe
\geq	\geq	Większe lub równe
\neq	\neq	Różne od
•	\bullet	Kółeczko
...	\ldots	Wielokropek
◆	\diamond	Romb
Δ	\Delta	Delta

W systemie TeX równania tworzone są na podstawie bieżących czcionek. Zmiana rodzaju czcionki ma wpływ tylko na wygląd liter i cyfr.

Wstawianie wykresów

Do dokumentu można również wstawić wykresy narysowane poza programem TeX. Będą one „pływły” z tekstem, co oznacza, że TeX zadba, by pojawiły się przy akapicie, który się do nich odnosi. Rozwiązanie takie uwalnia autora od martwienia się o to, gdzie dokładnie zostanie wstawiony rysunek.

Aby wstawić wykres, który ma zostać umieszczony na początku strony, należy użyć polecenia:

```
\topinsert wykres \endinsert
```

`wykres` to odniesienie do pliku zewnętrznego lub wewnętrznej definicji. TeX umieści go na początku następnej strony, zawierającej odpowiednio dużo miejsca.

Można również wstawić wykres na osobnej stronie, używając polecenia:

```
\pageinsert wykres \endinsert
```

Makropolecenia

Dzięki makropoleceniom TeX ma ogromne możliwości dostosowywania się do potrzeb użytkownika. Pozwalają one na tworzenie nowych poleceń przez grupowanie istniejących poleceń oraz tekstu. Po zdefiniowaniu, makropolecenie może zostać użyte w innym fragmencie dokumentu, bądź to dla zastąpienia powtarzających się bloków tekstu, bądź wykonania serii poleceń.

Makropolecenie definiuje się za pomocą polecenia:

```
\def\nazwa_makropolecenia { treść makropolecenia }
```

Tak zdefiniowane makropolecenie może zostać wywołane przez wpisanie polecenia `\nazwa_makropolecenia`. Oto przykład:

```
\def\brg{burger}
Ham\brg, cheese\brg, fish\brg.
```

Powyższy tekst po sformatowaniu przyjmie postać:

Hamburger, cheeseburger, fishburger.

Wewnątrz definicji makropolecenia można również wywołać inne makropolecenie, na przykład tak:

```
\def\hbg{Ham\brg}
```



Trzeba uważać na rekurencyjne definicje makropoleceń - czyli makropolecenia, których definicje odnoszą się do nich samych. Mogą one spowodować, że TeX wpadnie w nieskończoną pętlę. Przykładem takiego makropolecenia (nie powinieneś oczywiście go używać) może być:

```
\def\rekur{\rekur}
\rekur
```

Makropolecenia mogą również pobierać argumenty, jeśli w ich definicji podano listę parametrów formalnych. Definicja makropolecenia ma wówczas składnię:

```
\def \nazwa_makropolecenia lista_parametrów_formalnych {treść
makropolecenia}
```

lista_parametrów_formalnych ma postać #1, #1#2, #1#2#3 itd., w zależności od liczby parametrów. Oto przykład:

```
\def\param#1{To jest #1 wywołanie makropolecenia.}
\param{pierwsze}\break
\param{drugie}\break
\param{trzecie}
```

Na jego podstawie TeX wygeneruje tekst:

```
To jest pierwsze wywołanie makropolecenia.
To jest drugie wywołanie makropolecenia.
To jest trzecie wywołanie makropolecenia.
```

Każdy z przekazywanych parametrów musi zostać osobno otoczony klamrami, np. powyższy tekst:

```
\def\siostra#1#2{Moja #1 siostra ma na imię #2 \break}
\siostra{najstarsza}{Anna}
\siostra{średnia}{Mariola}
\siostra{najmłodsza}{Monika}
```

po sformatowaniu przyjmie postać:

```
Moja najstarsza siostra ma na imię Anna
Moja średnia siostra ma na imię Mariola
Moja najmłodsza siostra ma na imię Monika
```

Należy również uważać, by w definicji makropolecenia podać odpowiednią liczbę parametrów. Definicja:

```
\def\blad#1{Blad ze względu na #2}
```

jest niepoprawna, ponieważ występuje w niej odwołanie do drugiego parametru formalnego, który nie jest przekazywany.

Makropolecenie może zostać przeddefiniowane w obrębie dokumentu – wywoływana będzie zawsze wersja „najświeższa”. Makropolecenia zdefiniowane w obrębie grupy są widoczne tylko w grupie, w której zostały zdefiniowane.

Oto przykład użycia bardziej wyrafinowanego makropolecenia, wykorzystującego możliwość zagnieżdżania makropoleceń i ich redefiniowania:

```
\def\hej{Hej\def\hej{\hej}}
\hej, \hej, \hej.
```

Efektem przetworzenia takiego fragmentu pliku źródłowego jest tekst:

```
Hej, hej, hej.
```

Podobnie jak w przypadku większości tematów poruszonych w tej książce, pokazaliśmy tylko wierzchołek góry lodowej. Jednak gdy znasz już podstawy pracy z makropoleciami, możemy przyjrzeć się najpopularniejszemu chyba ich zestawowi, ułatwiającym w znacznym stopniu tworzenie dokumentów. Zestaw ten nazywa się LaTeX.

LaTeX - rozszerzenie systemu TeX

LaTeX to zestaw makropoleceń opartych na poleceniach systemu TeX, wzbogacających go o obsługę bardziej abstrakcyjnych struktur tekstu. Jest to w zasadzie biblioteka stylów, która umożliwia zachowanie jednolitości formatowania i składu dokumentów. Makro-polecenia systemu LaTeX pozwalają autorowi myśleć o tekście w sposób bardziej ogólny, na przykład zamiast wydawać polecenie zmieniające rodzaj czcionki na pochyloną o wysokości 8 punktów można po prostu zażądać wyróżnienia danego fragmentu, nie martwiąc się o szczegóły techniczne. Nazwy makropoleceń systemu LaTeX są bardziej adekwatne do tego, jak autor myśli o tworzonym tekście.

Ponieważ LaTeX jest rozszerzeniem systemu TeX, nie będziesz miał żadnych kłopotów z obsługiwaniem go, o ile masz nieco wprawy w posługiwaniu się TeX-em. Polecenia dotyczące ustawiania odstępów działają identycznie jak w systemie TeX. Tak samo obsługiwane są również znaki specjalne, nie wyłączając symbolu % oznaczającego początek komentarza.

Główne różnice pomiędzy systemem TeX i LaTeX ujawniają się w miarę poznawania możliwości nowych makropoleceń pozwalających w bardzo wygodny sposób sformatować dokument.

Definicja dokumentu systemu LaTeX

Każdy dokument tworzony za pomocą systemu LaTeX rozpoczyna się od polecenia \documentclass. Jego parametrem jest klasa tworzonego dokumentu. Podstawowe klasy dokumentów zebrane w tabeli 19.2.

Tabela 19.2. Podstawowe klasy dokumentów LaTeX-u

Klasa dokumentu	Opis
article	Używana do tworzenia krótkich raportów, prezentacji, artykułów naukowych itp.
book	Używana do formatowania całych książek
report	Używana do tworzenia raportów, zawierających kilka rozdziałów, tez itp.

Aby utworzyć najprostszy dokument systemu LaTeX, po prostu wstaw kilka słów pomiędzy polecenia \begin{document} oraz \end{document}, np.:

```
\documentclass{article}  
\begin{document}  
To jest krociutki dokumencik.  
\end{document}
```

Aby sformatować ten dokument, należy wydać polecenie (zakładając, że jest on zapisany w pliku o nazwie doc1):

```
% latex doc1
```

Spowoduje ono wygenerowanie pliku `.dvi` i pliku zawierającego informacje o przebiegu komplikacji, dokładnie tak samo jak podczas używania systemu TeX. Plik `.dvi` może zostać przekonwertowany do formatu postscriptowego, można go również obejrzeć za pomocą programu `xdv`.

Przy podawaniu klasy dokumentu można również określić wartości pewnych opcji; składnia polecenia `\documentclass` jest w takim przypadku następująca:

```
\documentclass[opcje]{klaś_dokumentu}
```

Najczęściej używane opcje zebrane w tabeli 19.3.

Tabela 19.3. Najczęściej używane opcje polecenia `\documentclass`

Opcja	Znaczenie
<code>10pt, 11pt, 12pt itd.</code>	Określa rozmiar czcionki dokumentu (domyślnie 10pt)
<code>fleqn</code>	Wyświetla równania dosunięte do lewego marginesu, a nie wyśrodkowane
<code>leqno</code>	Numeruje równania z lewej strony
<code>letterpaper, a4 paper</code>	Określa rozmiar papieru (domyślnie letterpaper)
<code>openright, openany</code>	Rozpoczyna nowy rozdział na prawej lub na dowolnej stronie
<code>titlepage, notitlepage</code>	Powoduje utworzenie strony tytułowej lub zabrania jej tworzenia
<code>twocolumn</code>	Dzieli strony na dwie kolumny
<code>twoside, oneside</code>	Generuje dwu lub jednostronnie drukowany dokument

Poszczególne klasy dokumentów różnią się co do wartości domyślnych tych opcji. Przykładowo, artykuły i raporty są domyślnie drukowane jednostronnie, natomiast książki – dwustronnie. Artykuły nie uwzględniają opcji dotyczących stron tytułowych i rozpoczynania rozdziałów, ponieważ rozdziały nie są w nich definiowane. Jak widać, dokumenty poszczególnych klas formatowane są zgodnie z oczekiwaniemi.

Pakiety

LaTeX pozwala również na dołączanie zewnętrznych pakietów makropoleceń, co umożliwia dalsze rozszerzanie jego funkcjonalności. Służy do tego polecenie `\usepackage{nazwa_pakietu}`. Przykładami takich pakietów mogą być pakiety `doc` (który wspiera pisanie dokumentacji) czy `makeidx` (ułatwiający generowanie indeksów).

Można również – za pomocą polecenia `\pagestyle` – samodzielnie zdefiniować styl strony, który zostanie zastosowany do formatowania dokumentu. Tabela 19.4 przedstawia najczęściej używane style.

Tabela 19.4. Najczęściej używane style strony (polecenie `\pagestyle`)

Styl	Opis
<code>empty</code>	Ustawia pusty nagłówek oraz stopkę
<code>headings</code>	Drukuje aktualny tytuł rozdziału oraz numer strony w nagłówku,

<p>stopkę pozostawiając pustą</p> <p><code>plain</code></p>	<p>Drukuje wyśrodkowany numer strony jako stopkę (styl domyślny)</p>
---	--

Możesz również różnicować strony pod względem stylu w obrębie dokumentu za pomocą polecenia `\thispagestyle{...}`. Zmiany dotyczą wówczas tylko bieżącej strony.

Znaki narodowe

LaTeX obsługuje znaki narodowe, na przykład różnego typu akcenty i umlauty. Po szczególnie litery są generowane przy użyciu różnych poleceń. Przykładowo, by uzyskać literę ö należy wpisać polecenie `\\"o`. Za pomocą polecenia `\frenchspacing` można polecić, by po kropce nie była wstawiana dodatkowa spacja. Niestety, jedynym sposobem na uzyskanie polskich „ogonków” jest załadowanie międzynarodowego zestawu czcionek, dostępnego zwykle w katalogu o nazwie `dd (przyp. tłum.)`.

Struktura dokumentu systemu LaTeX

LaTeX posiada polecenia, dzięki którym łatwo jest zachować jasną i przejrzystą strukturę dokumentu, co ułatwia jego odbiór czytelnikowi. Dla dokumentów klasy `article` dostępne są polecenia:

```
\section
\subsection
\subsubsection
\paragraph
\subparagraph
\appendix.
```

Polecenia te muszą być deklarowane przed tekstem, którego dotyczą, a ich argumentami (za wyjątkiem polecenia `\appendix`) są nagłówki poszczególnych bloków tekstu. LaTeX zajmuje się szczegółami: ustawia odpowiednie odstępy, numerowanie rozdziałów oraz czcionkę. Polecenie `\appendix` oznacza rozdziały kolejnymi literami alfabetu.

Dla dokumentów klas `report` oraz `book` dostępne są dwa dodatkowe polecenia: `\part` oraz `\chapter`. `\part` pozwala na wstawienie kolejnej części dokumentu bez ingerowania w numerację rozdziałów. Możesz zapobiec wstawianiu nagłówka do spisu treści przez wpisanie po poleceniu

```
\section znaku *:
\section*{Ten rozdział nie pojawi się w spisie treści}
```

Aby dodać do dokumentu tytuł, należy go zdefiniować, przekazując odpowiedni argument do polecenia `\title`, a potem wywołać polecenie `\maketitle`:

```
...
\title {Wyznania użytkownika systemu LaTeX}
\author {Ja}
\date
```

```
\begin{document}
\maketitle
...
```

Aby wstawić do dokumentu spis treści, wydaj polecenie `\tableofcontents` w miejscu, w którym ma się on pojawić. Przetworzenie dokumentu wymagać będzie wtedy dwóch przebiegów: jednego dla zanotowania wszystkich nagłówków, drugiego dla utworzenia na ich podstawie spisu treści.

Dodawanie innych elementów strukturalnych

Do dokumentu można również wstawać odsyłacze, które wiążą dany element (jak tekst, wykres czy tabela) z tekstem w innym miejscu dokumentu. Punkt, do którego chcesz się odnosić, określany jest poleceniem `\label`, a odnośnik – poleceniem `\ref` lub `\pageref`.

Wykorzystując polecenie `\footnote`, można łatwo dodawać przypisy: po prostu jako jego argument należy przekazać tekst, który ma być treścią przypisu.

Struktura dokumentu może również zostać wzbogacona za pomocą środowisk. Są to zdefiniowane wcześniej formaty tekstu. Można ich użyć za pomocą polecień `\begin{nazwa_srodowiska}` i `\end{nazwa_srodowiska}`. Niektóre z predefiniowanych w systemie LaTeX środowisk zebrane w tabeli 19.5.

Tabela 19.5. Predefiniowane środowiska

Nazwa	Opis
<code>center</code>	Wyśrodkowanie tekstu
<code>description</code>	Akapit opisowy
<code>enumerate</code>	Wyliczenie
<code>flushleft</code>	Wyrównanie do lewego marginesu
<code>flushright</code>	Wyrównanie do prawego marginesu
<code>itemize</code>	Proste wyliczenie
<code>quote</code>	Cytat nie dłuższy niż jeden akapit
<code>quotation</code>	Cytat dłuższy niż jeden akapit
<code>tabular</code>	Tabela z opcjonalnymi separatorami kolumn i wierszy
<code>verbatim</code>	Tekst pisany czcionką maszyny do pisania (używane np. do prezentacji kodu programu)
<code>verse</code>	Kontrola podziału na wiersze (np. w poezji)

Praca z tabelami i wykresami

Wstawianie pływających wykresów i tabel możliwe jest dzięki środowiskom `table` oraz `figure`. Przykładowy wykres można zdefiniować następująco:

```
\begin{figure} [!hbp]
\makebox[\textwidth]{\framebox[2in]{\rule{Opt}{2in}}}
\end{figure}
```

Oczekiwane położenie wykresu można podać przekazując odpowiednie argumenty do polecenia `\begin{figure}`. LaTeX stara się rozmieścić obiekty pływające zgodnie z tymi preferencjami, uwzględniając jednak wewnętrzne wytyczne, takie jak na przykład maksymalna ilość obiektów pływających na stronie. W tym przypadku, LaTeX będzie próbował umieścić wykres przy odpowiadającym mu tekście (`h`), na dole następnej strony (`b`) lub, jeśli i to będzie niemożliwe, na oddzielnej stronie z obiektami pływającymi (`p`). Symbol `!` powoduje, że LaTeX nie będzie próbował umieszczać rysunku w sposób domylny (co może powodować konflikt z innymi specyfikatorami).

Tabele i wykresy mogą być opisywane za pomocą polecenia `\caption`, które należy wydać wewnątrz odpowiedniego środowiska.

Przedstawiliśmy jedynie najbardziej podstawowe informacje dotyczące systemu LaTeX, ale mamy nadzieję, że dadzą Ci one podstawy, dzięki którym będziesz mógł nadawać dokumentom bardziej atrakcyjny wygląd. Zauważysz na pewno, że LaTeX jest nieco wygodniejszy w obsłudze niż TeX, ponieważ ukrywa przed autorem rozpraszające szczegóły dotyczące formatowania dokumentu.

VirTeX i IniTeX

Są to dwa związane z systemem TeX programy, pełniące nieco odmienne funkcje. IniTeX służy do tworzenia plików formatu o rozszerzeniu `.fmt`, zawierających definicje czcionek i makropoleceń. Program VirTeX służy do szybkiego ładowania tych prekompliowanych plików (potrafi robić to o wiele szybciej niż TeX). Aby użyć pliku formatu, powinieneś wydać polecenie:

```
$ virtex \&mojformat.fmt plik.tex
```

Znak `\&` jest niezbędny, by VirTeX rozpoznał, że powinien najpierw załadować plik formatu. Ponieważ symbol ten jest znakiem specjalnym powłoki, musi być poprzedzony symbolem `\`, co pozwala uniknąć jego interpretacji. VirTeX nie może być używany do tworzenia plików `.fmt`, ale potrafi je bardzo szybko wczytywać.

Podsumowanie

TeX to system formatowania dokumentów dla Linuxa, który pozwala autorom własnoręcznie tworzyć dokumenty wysokiej jakości. Pliki utworzone przez system TeX są w dużej mierze przenośne. System ten obsługuje wiele zaawansowanych technik składu i jest szczególnie wygodny do formatowania złożonych formuł matematycznych. Posiada makropolecenia, które mogą zostać użyte do dalszego rozszerzania jego możliwości. LaTeX, jedno z najpopularniejszych rozszerzeń systemu TeX, pozwala w łatwy i przestrzesty sposób nadać dokumentowi spójną strukturę.

Jeśli chcesz nauczyć się programować w języku `gawk`, poręcznym i łatwym języku używanym przez zaawansowanych użytkowników i administratorów, przejdź do rozdziału 25. „`gawk`”.

Programowanie w języku Perl, bardzo rozpowszechnionym wśród autorów stron WWW, omówione jest w rozdziale 28. „Perl”.

Zorientowany obiektowo i obsługujący interfejs graficzny język Smalltalk/X jest opisany w rozdziale 31. „Smalltalk/X”.

Rozdział 20.

Drukowanie

Tim Parker

W tym rozdziale:

- υ Konfiguracja drukarki
- υ Polecenia służące do drukowania

W poprzednich rozdziałach wspomnieliśmy kilkakrotnie o drukowaniu; temat ten będzie również w jednym z następnych rozdziałów omówiony z punktu widzenia administratora systemu. W tym rozdziale dowiesz się, jak można wydrukować dokument w systemie Linux, jakie urządzenia służą do drukowania w systemie linuxowym oraz jak skonfigurować drukarkę.

Trzeba pamiętać o tym, że wiele procedur drukowania zostało „w spadku” po trzydziestu latach istnienia UNIX-a. W zamierzchłych czasach drukarki były dostępne tylko w większych systemach i właśnie w tym duchu pisano programy je obsługujące. Choć zostały one przystosowane do dzisiejszych realiów, nadal zawierają wiele anachronizmów.

Konfiguracja drukarki

W większości wersji Linuxa użytkownik ma możliwość skonfigurowania drukarki już przy pierwszej instalacji systemu z poziomu programu instalacyjnego. W takim przypadku można zwykle wybrać jeden spośród modeli drukarek obsługiwanych przez system Linux. Należy podać również nazwę portu, do którego jest ona podłączona. Jeśli skonfigurowałś drukarkę w ten sposób, powinna ona działać prawidłowo. Jeśli pominąłeś ten fragment instalacji albo Twоя wersja Linuxa nie daje takiej możliwości, musisz zainstalować drukarkę ręcznie.

Zanim rozpocznesz instalowanie drukarki, powinieneś zdobyć dwie informacje. Pierwszą z nich jest typ drukarki (oraz dane o jej kompatybilności z innymi modelami). Informacja o kompatybilności z innymi modelami może okazać się przydatna, ponieważ nie wszystkie typy drukarek są obsługiwane bezpośrednio. Nawet jeśli kupiłeś właśnie najnowszy mo-

del drukarki ze sterownikami dla systemu Windows, prawdopodobnie producent i tak nie dostarczył sterowników dla Linuxa. Na szczęście większość dostępnych dziś na rynku drukarek jest kompatybilna ze starszymi modelami, dla których sterowniki linuxowe zostały już opracowane. Co prawda nie będzie wówczas można wykorzystywać wszystkich możliwości urządzenia, ale przynajmniej będzie można z niego w ogóle korzystać.

Drugą informacją niezbędną do skonfigurowania drukarki jest nazwa portu, do którego drukarka jest podłączona. Drukarki są zwykle podłączane do jednego z trzech typów portów: równoległego, szeregowego lub sieciowego (tylko w większych systemach). W przypadku portu równoległego trzeba tylko wiedzieć, czy używasz portu pierwszego, czy drugiego (LPT1 czy LPT2; w niektórych systemach dostępny jest tylko jeden port równoległy, wówczas oznaczany jest on jako LPT1), ponieważ wszystkie pozostałe parametry transmisji pozostają stałe.

Drukarki dołączane do portu szeregowego są także dość popularne – do połączenia niezbędnego jest wówczas kabel typu RS-232. W takim przypadku trzeba znać jeszcze dwie informacje: prędkość przesyłania danych (zwykle 9600 bodów) i ilość transmitowanych bitów (przeważnie 8). Drukarka i port szeregowy muszą być skonfigurowane w ten sam sposób, w przeciwnym przypadku urządzenie nie będzie działało prawidłowo.

Porty sieciowe są łatwiejsze w konfiguracji niż porty szeregowe. Każda drukarka w sieci posiada własny adres IP, który będzie potrzebny podczas instalacji; oprócz niego musisz również podać, jakiego sterownika należy użyć.

Nazwy portów

W systemie Linux obowiązuje inne nazewnictwo portów niż w systemach DOS i Windows. Dokładniej nazwom urządzeń przyjrzymy się w rozdziale 33. „Urządzenia”. DOS-owy port LPT1 to w Linuxie `/dev/lp0`; port LPT2 – `/dev/lp1`. Początkowa część nazwy urządzenia – `/dev` – informuje system, w którym katalogu ma szukać informacji dotyczących danego urządzenia, natomiast właściwa nazwa urządzenia to odpowiednio `lp0` i `lp1`.

Podobnie rzecz ma się w przypadku portów szeregowych: DOS-owy port COM1 to w systemie Linux `/dev/ttys0`, COM2 – `/dev/ttys1` itd. Linux umożliwia stosowanie wielu portów szeregowych, a konwencje dotyczące ich nazewnictwa mogą być nieco inne w przypadku zastosowania karty multiport, ale bezpiecznie jest używać nazw urządzeń `/dev/ttys0` i `/dev/ttys1`, oznaczających porty szeregowe wbudowane w płytę główną.



Każdy port szeregowy ma w Linuxie dwie nazwy. Przykładowo, port o DOS-owej nazwie COM1 to w systemie Linux zarówno `/dev/ttys0`, jak i `/dev/ttysua0`. Druga postać nazwy używana jest do komunikacji dwukierunkowej, np. z modemem, i nie powinna być używana do współpracy z drukarką.

Sterowniki drukarek

Każda drukarka w systemie linuxowym musi być skojarzona z odpowiednim sterownikiem. Sterownik umożliwia systemowi porozumiewanie się z urządzeniem, tłumacząc polecenia typu „wysuń stronę” na ciągi znaków charakterystyczne dla danego modelu drukarki.

Do dystrybucji Linuxa dołączany jest zestaw sterowników drukarek; dodatkowe sterowniki znaleźć można w węzłach FTP i na stronach WWW. Użycie sterownika niedostosowanego do danego modelu drukarki może spowodować, że drukarka będzie się bardzo dziwnie zachowywać.

Wszystkie informacje dotyczące drukarki i komunikacji z nią przechowywane są w pliku `/etc/printcap`. Zwykle plik ten jest dość duży i tylko użytkownik `root` może go modyfikować. Niektóre systemy linuxowe nie pozwalają modyfikować tego pliku ręcznie (uparty administrator może oczywiście obejść to zabezpieczenie), w zamian oferując specjalne programy narzędziowe. Rysunek 20.1 przedstawia reakcję systemu RedHat 5.0 na próbę wyświetlenia zawartości tego pliku poleceniem `more`. W tym przypadku system doradza użycie do modyfikacji pliku `/etc/printcap` programu opartego na interfejsie X. Nie wszystkie wersje Linuxa są tak ostrożne, pozwalając administratorowi modyfikować ten plik bez ograniczeń.

Rysunek 20.1.

Niektóre systemy nie pozwalają na bezpośrednią edycję pliku `/etc/printcap`

```
[root@localhost /etc]# more printcap
#
# Please don't edit this file directly unless you know what you are doing!
# Be warned that the control-panel printtool requires a very strict format!
# Look at the printcap(5) man page for more info.
#
# This file can be edited with the printtool in the control-panel.

[root@localhost /etc]#
```

Można oczywiście wpisać samodzielnie do pliku `/etc/printcap` wszystkie kody sterujące drukarki, jak robiono to dawniej, ale jest to proces długotrwały i bardzo podatny na błędy. Zamiast tego o wiele wygodniej jest poszukać sterownika do modelu kompatybilnego z Twoim typem drukarki.

Oprócz sterownika w procesie drukowania biorą udział jeszcze inne programy, szczególnie jeśli drukarka obsługuje jeden z języków opisu strony, takich jak PCL czy PostScript. W takim przypadku informacje najpierw przesyłane są do odpowiedniego programu (filtru) tłumaczącego dane, które mają zostać wydrukowane, na format zrozumiały dla drukarki. Następnie za pośrednictwem sterownika są one przesyłane do drukarki.

Jeśli posiadasz drukarkę obsługującą więcej niż jeden język opisu strony (np. Hewlett-Packard LaserJet) możesz zdefiniować więcej niż jedno urządzenie drukarki (np. jedno używające filtra PCL i sterownika HP LaserJet i drugie, używające filtra PostScript i tego samego sterownika), dzięki czemu możesz w każdej chwili wybrać, którego języka chcesz używać. Linux pozwala definiować dowolną ilość urządzeń drukarek i przypisywać im dowolne kombinacje portów i filtrów. Przy drukowaniu trzeba podać, której definicji drukarki chcesz użyć – zostanie wtedy użyty określony w niej filtr, sterownik i port. Niektóre drukarki (w szczególności starsze modele drukarek igłowych) nie potrzebują żadnych filtrów, więc ich sterowniki mogą przyjmować dane bezpośrednio z aplikacji.

Polecenia służące do drukowania

Ponieważ Linux oparty jest na systemie UNIX, w posiadanej przez Ciebie wersji systemu może działać kilka poleceń i programów użytkowych służących do obsługi procesu drukowania. Najpopularniejszymi z tych programów są `lpr`, `lpd` i `lpq`. Skonfigurowanie ich tak, by działały w pełni poprawnie, nie jest ani łatwe, ani przyjemne i często zdarza się, że zajmuje więcej czasu niż konfiguracja całej reszty systemu.

Kiedy drukujesz plik (na przykład pod kontrolą edytora `emacs`), w tle uruchamiane są programy obsługi drukowania. Kopiuję one dane, które mają zostać wydrukowane, do specjalnego katalogu pełniącego funkcję kolejki (spool), zachowując je w pliku. Dzięki temu aplikacja, która wydała zlecenie drukowania, nie musi czekać na zakończenie tego procesu, a system może zarządzać zadaniami drukowania według ich priorytetu, typu drukarki, wymaganego papieru itp. Katalog, w którym przechowywane są dane czekające na wydrukowanie, to zwykle `/usr/spool/lp` lub `/var/spool/lp`. W katalogu tym znajdują się osobne podkatalogi dla każdej definicji drukarki, na przykład

```
/usr/spool/lp/laserjet czy /usr/spool/lp/bubblejet.
```

Kiedy wydajesz zlecenie wydrukowania pliku, Linux przydziela mu numer identyfikacyjny. Zwykle proces ten nie jest widoczny, szczególnie jeśli drukujesz pod kontrolą jakiejś aplikacji, ale przy drukowaniu z wiersza poleceń można zauważyc następujące komunikaty:

```
$ lp plik1.txt
request id is hplj_307 (1 files)
```

W tym przypadku numer identyfikacyjny zlecenia to `hplj_307`, co oznacza, że jest to 307 dokument od momentu uruchomienia licznika (zainstalowania drukarki). Po numerze identyfikacyjnym następuje liczba plików, które oczekują na wydrukowanie. W poniższym przykładzie drukować będziemy grupę plików:

```
lp tekst*.doc
request id is hplh_308 (12 files)
```

Nawet, jeśli zlecono drukowanie dwunastu plików, przydzielany jest im tylko jeden numer identyfikacyjny. Wszystkie pliki, które chcemy wydrukować, potraktowane zostały jako pojedyncze zlecenie.

Po wysłaniu przez aplikację zlecenia drukowania odpowiednie pliki są kopiowane do katalogu kolejki. Działający w tle program obsługi drukarki realizuje kolejno zlecenia. Program obsługi drukarki działa przez cały czas, sprawdzając czy w katalogu kolejki nie pojawiły się nowe pliki. Jeśli odnajdzie zlecenie, które nie zostało jeszcze obsłużone, sprawdza, czy określona drukarka jest dostępna i wysyła do niej dane. Jeśli drukarka nie jest dostępna, drukowanie jest wstrzymywane do czasu, aż będzie mogło zostać zrealizowane. Program obsługi drukarki może również na żądanie użytkownika przydzielać zleceniom priorytety.

W systemach linuxowych dostępnych jest wiele poleceń, które pozwalają sprawdzić, jakie pliki czekają na wydrukowanie, oraz ustawać je w żądanej kolejności. Nazwy tych poleceń są zależne od wersji oprogramowania, więc musisz sam sprawdzić, które z nich działają w Twoim systemie.

Aby obejrzeć listę zleceń oczekujących na wydrukowanie (zapisanych w katalogu kolejki), wydaj polecenie `lpq` lub `lpstat`. `lpq` wyświetla informacje o wszystkich plikach oczekujących na wydrukowanie, wraz z ich pozycją w kolejce, dzięki czemu można oszacować, kiedy dany plik zostanie wydrukowany. W niektórych systemach do tego samego celu służy polecenie `lpstat`. Program `lpstat` wywołany z opcją `-t` pokazuje również informacje o statusie wszystkich dostępnych drukarek. W innych systemach do tego celu służy polecenie `lpc`.

Aby usunąć zlecenie z kolejki, w większości systemów linuxowych należy użyć polecenia `lprm`, po którym należy podać identyfikator zlecenia, które ma zostać usunięte, np.:

```
lprm hplj_308
```

W innych systemach do tego celu służy polecenie `cancel`, np.:

```
cancel hplj_308
```

Po wydaniu jednego z tych poleceń przeważnie wyświetlany jest komunikat potwierdzający usunięcie zadania z kolejki:

```
request hplj_308 canceled
```

Podsumowanie

W tym rozdziale omówiliśmy, w jaki sposób Linux obsługuje drukarki i ich sterowniki oraz jak działają filtry. W wielu wersjach Linuksa dostępne są skrypty instalujące drukarki, więc uciążliwy i rzadko zakończony sukcesem proces ręcznej instalacji drukarki, konfiguracji katalogu kolejki i ustalania praw dostępu może zostać pominięty.

Aby dowiedzieć się więcej o programach GhostScript i Ghostview, które umożliwiają obsługę języka PostScript w systemach linuxowych, przejdź do rozdziału 24. „GhostScript”.

Urządzenia i sterowniki opisane są dokładniej w rozdziale 33. „Urządzenia”.

Używanie drukarek w sieci omówione jest w rozdziale 37. „Praca w sieci”.

Rozdział 21.

Linux i multimedia

Tim Parker

W tym rozdziale:

- υ Karty dźwiękowe
- υ Dżojstiki

Jednym z ograniczeń pierwszych wersji Linuxa był brak możliwości obsługi kart dźwiękowych, dżojstików i innych urządzeń multimedialnych. Kilka ostatnich wersji systemu zmieniło tę sytuację zasadniczo, teraz Linux jest równie multimedialny jak Windows 98. Ten rozdział pomoże Ci skonfigurować system do zastosowań multimedialnych.

Karty dźwiękowe

Karta dźwiękowa potrafi o wiele więcej niż tylko wydawać dźwięki przy starcie Windows czy wtedy, gdy zrobisz coś nie tak. Jest ona niezastąpiona przy grach komputerowych, ponieważ potrafi generować zarówno muzykę, jak i efekty dźwiękowe. Może być używana do nagrywania i odtwarzania plików dźwiękowych. Pozwala odsłuchać płytę CD Audio w napędzie CD-ROM, dzięki czemu możesz słuchać muzyki podczas pracy. Może również spełniać rolę interfejsu pomiędzy systemem a niektórymi urządzeniami, na przykład łańcuchem SCSI czy napędem CD-ROM.

Wszystkie karty dźwiękowe pracują cyfrowo: przesyłane do nich pakiety danych powodują, że ich układy generują ton o żądanej częstotliwości, głośności i innych parametrach. Mogą one generować dźwięk o różnych częstotliwościami próbkowania: im większa częstotliwość próbkowania, tym lepsza jest jakość dźwięku. Większość odtwarzaczy CD próbuje dźwięk z częstotliwością 44,1 kHz przy 16 bitach danych przypadających na jedną próbke. Po prostych obliczeniach daje to około 600 megabajtów danych na godzinę wysokiej jakości dźwięku. Starsze karty dźwiękowe nie miały takich możliwości, oferując tylko 8 bitów na próbke przy częstotliwości próbkowania 8 kHz. Różnica jest bardzo wyraźna, jeśli do odsłuchu takiego dźwięku użyjesz dobryj jakości sprzęt, mniej zauważalna – gdy używasz do tego celu głośniczka wbudowanego w Twój PC.

Plik dźwiękowe dostępne są dziś w wielu różnych formatach. Różnią się one częstotliwością próbkowania, liczbą bitów przypadających na jedną próbkę, zastosowaniem algorytmu kompresji, liczbą zapisanych kanałów (dźwięk monofoniczny wymaga tylko jednego kanału, stereofoniczny – dwóch) itd. Najczęściej spotykane formaty plików dźwiękowych zebrane są w tabeli 21.1.

Tabela 21.1. Popularne formaty plików dźwiękowych

Rozszerzenie nazwy pliku	Liczba kanałów	Długość jednej próbki	Częstotliwość próbkowania	Komentarz
.au	2	8 bitów	8 kHz	Format UNIX SUN
.mod	2	8 bitów	zależna od użytego algorytmu	
.raw	2	8 bitów	22 kHz	
.voc	1	8 bitów	22 kHz	
.wav	1	8 bitów	22 kHz	
.wav	2	16 bitów	44 kHz	

Z nowszymi wersjami Linuxa dostarczanych jest wiele sterowników do kart dźwiękowych. Wszystkie one są napisane przez użytkowników, którzy następnie zechcieli udostępnić je innym (sterowniki rozprowadzane z kartami muzycznymi przeważnie przeznaczone są dla systemów DOS i Windows i nie będą pracować poprawnie w systemie Linux). Nawet, jeśli nie posiadasz sterownika dla swojej karty, najprawdopodobniej będzie ona kompatybilna z którymś z modeli kart, dla którego sterowniki są dostępne. Przykładowo, wiele kart jest kompatybilnych z kartami Sound Blaster lub Sound Blaster Pro, więc ich sterowniki powinny (ale nie muszą) działać.

Używanie wbudowanego głośniczka

Jeśli nie posiadasz karty dźwiękowej, nie oznacza to, że Twój komputer nie potrafi wydawać dźwięków. Dostępny jest również sterownik obsługujący głośniczek wbudowany w prawie każdy komputer PC. Mimo że jakość dźwięku nie jest najlepsza, pozwala jednak uzyskać dźwięk wystarczający by np. zagrać w Dooma. Jeśli nie boisz się kabellków, możesz również użyć zewnętrznego głośnika, co pozwoli uzyskać nieco lepsze brzmienie. Jeśli chcesz używać tego sterownika, musisz zdobyć go gdzieś w Internecie, ponieważ nie jest on dołączany do standardowej dystrybucji Linuxa.



Jeśli chcesz załadować najnowszą wersję sterownika dla wbudowanego w PC głośniczka, poszukaj go pod adresem
<ftp.informatik.hu-berlin.de/pub/linux/hu-sound>
lub w jednym z węzłów wymienionych w dodatku A „Węzły FTP i grupy dyskusyjne poświęcone Linuxowi”.

Wraz ze wspomnianym sterownikiem dostarczany jest zestaw instrukcji, które pomogą Ci w jego instalacji. Dołączony jest również plik pozwalający na przetestowanie konfiguracji.

Konfiguracja karty dźwiękowej

Większość wersji Linuxa zawiera skrypt, który znacznie ułatwia konfigurację karty dźwiękowej. Zwykle możesz wybrać jej typ z menu wyświetlanego podczas instalacji, a program instalacyjny zajmie się całą resztą. Jeśli tak jest w Twoim przypadku, karta powinna być zainstalowana poprawnie i możesz ominąć ten podrozdział. Jeśli Linux nie zapytał o kartę dźwiękową lub też po zainstalowaniu nie działa ona prawidłowo, musisz przeprowadzić proces jej instalacji tak, jak opisano poniżej.



Niektóre dystrybucje Linuxa zawierają programy instalujące kartę dźwiękową, do których nie ma jednak dostępu aż do momentu pełnego zainstalowania systemu. RedHat 5.0 zawiera program `sndconfig`, który potrafi skonfigurować większość kart kompatybilnych z kartami Sound Blaster. Sprawdź w dokumentacji dołączonej do używanej przez Ciebie dystrybucji, czy przypadkiem nie oferuje ona tego typu ułatwień.

Pierwszym krokiem konfiguracji karty dźwiękowej w systemie linuxowym jest jej zainstalowanie (zgodnie z informacjami zawartymi w jej dokumentacji). Aby zainstalować kartę dla systemu Linux, nie są wymagane żadne szczególne zabiegi – należy tylko upewnić się, że ma ona przypisany odpowiedni numer przerwania, kanału DMA i adres wejścia/wyjścia. Jeśli karta jest już zainstalowana w systemie DOS lub Windows, możesz uzyskać informacje konfiguracyjne, używając Panelu Sterowania czy jakiegoś programu użytkowego. Aby ustawić te parametry w niektórych starszych modelach kart dźwiękowych, niezbędne jest odpowiednie przełączenie zworek czy mikroprzełączników. Zanotuj wymienione wyżej dane (IRQ, DMA i adres I/O) – mogą być one potrzebne podczas konfiguracji karty w systemie Linux.



Jeśli używasz karty kompatybilnej ze standardem Plug and Play, powinieneś najpierw zainstalować ją w systemie DOS lub Windows, ponieważ Linux w większości przypadków nie potrafi ustawać parametrów takich urządzeń. Następnie zanotuj ustawienia konfiguracyjne karty, po czym najlepiej wyłącz używanie PNP w BIOS-ie (jeśli tego nie zrobisz, może się zdarzyć, że karta uruchamiać się będzie z innymi ustawieniami). Zwykle w programie konfiguracyjnym BIOS-u służy do tego celu opcja taka jak `PNP OS?`, której domyślną wartością jest `Yes`. Powinieneś ją wyłączyć.

Po zainstalowaniu karty dźwiękowej należy dołączyć do jądra systemu odpowiedni sterownik. Nie przejmuj się, jeśli nigdy tego nie robiliś – nie jest to procedura szczególnie trudna. Prawdopodobnie nie miałeś potrzeby kompilować jądra systemu do tej pory, po-

nieważ większość dystrybucji zawiera skompilowane wersje jądra, ale nie umożliwiają one obsługi kart dźwiękowych. Należy ponownie skompilować jądro i wszystkie potrzebne sterowniki (włączając w to sterownik karty dźwiękowej), a następnie dołączyć sterowniki do jądra systemu.

Jeśli pobrałeś sterownik do karty dźwiękowej z sieci, prawdopodobnie dołączony jest do niego plik README lub coś w tym rodzaju. Zapoznaj się najpierw z informacjami w nim zawartymi, ponieważ niektóre sterowniki wymagają przeprowadzenia jakiś dodatkowych operacji przed komplikacją jądra albo wymagają ustawienia specyficznych numerów przerwań czy kanałów DMA.

Każda z wersji Linuxa zawiera nieco inny skrypt pozwalający na przekompilowanie jądra. Zawsze jednak można uruchomić go w następujący sposób:

1. zaloguj się jako `root`,
2. zmień katalog bieżący na `/usr/src/linux`,
3. wydaj polecenie `make config`.

Niektóre wersje Linuxa posiadają również przeznaczone do tego celu programy oparte na systemie menu lub interfejsie X. Należy wówczas wydać polecenie `make menuconfig` lub `make xconfig`.

Podczas rekompilacji jądra systemu musisz być zalogowany jako `root` – w przeciwnym przypadku może się okazać, że nie posiadasz praw dostępu do odpowiednich plików i programów. W większości wersji Linuxa plik `makefile` służący do komplikowania jądra znajduje się w katalogu `/usr/src/linux`, jeśli jednak tak nie jest – sprawdź, gdzie powinieneś go szukać, w dołączonej do dystrybucji dokumentacji.

Po uruchomieniu skryptu czy programu konfigurującego jądro systemu zostaniesz zapytany o to, jakie komponenty mają zostać dołączone do kernela (rys. 21.1). Odpowiedzi domyślne są prawie zawsze dobrym rozwiązaniem, chyba że masz pewność, że chcesz skonfigurować jądro w jakiś specyficzny sposób. Dobrym przykładem jest konfiguracja karty dźwiękowej. Program konfiguracyjny pyta, czy powinieneś dołączyć do jądra systemu obsługę tego urządzenia. Odpowiedzią domyślną jest `no`, ale jeśli ją posiadasz, powinieneś odpowiedzieć `yes`. Zostanie Ci następnie zadana cała seria pytań dotyczących konfiguracji tego urządzenia (rys. 21.2).

Rysunek 21.1.

Skrypt konfiguracyjny jądra systemu pyta, jakie komponenty jądra należy zainstalować

```
nxtterm
[root@localhost linux]# cd /usr/src/linux
[root@localhost linux]# make config
rm -f include/asm
( cd include ; ln -sf asm-i386 asm)
/bin/sh scripts/Configure arch/i386/config.in
#
# Using defaults found in arch/i386/defconfig
#
* Code maturity level options
*
Prompt for development and/or incomplete code/drivers (CONFIG_EXPERIMENTAL) [Y/n]
?/
```

Rysunek 21.2.

Skrypt konfiguracyjny zadaje serię pytań dotyczących zainstalowanej karty dźwiękowej

```
nxtterm:
Configuring Sound Support
ProAudioSpectrum 16 support [N/y/?]
Sound Blaster (SB, SBPro, SB16, clones) support [N/y/?] y
Gravis Ultrasound support [N/y/?]
MPU-401 support (NOT for SB16) [N/y/?]
6890 UART Midi support [N/y/?]
PS3 ECHO-AD12111 support [N/y/?]
Microsoft Sound System support [N/y/?]
Ensoniq SoundScape support [N/y/?]
MediaTrix Audiatrix Pro support [N/y/?]
Support for MM16 and/or Mozart based cards [N/y/?]
Support for Crystal CS4232 based (FnP) cards [N/y/?]
Support for Turtle Beach Wave Front (Maui, Tropez) synthesizers [N/y/?]
/dev/dsp and /dev/audio support [Y/n/?]
MIDI interface support [Y/n/?]
FM synthesizer (YN3812/DPL-3) support [Y/n/?]
Support for the SG NX Pro mixer [N/y/?]
Support for the MV Jazz16 (ProSonic etc.) [N/y/?]
Do you have a Logitech SoundMan Games [N/y/?]
```

Powinieneś odpowiedzieć **n** na pytania, czy zainstalować sterowniki dla kart dźwiękowych, których nie posiadasz, natomiast **y** w odpowiedzi na pytanie, czy chcesz zainstalować sterownik do zainstalowanego modelu karty. Jeśli Linux daje Ci możliwość automatycznego wykrycia karty dźwiękowej, możesz z niej skorzystać, ale procedura ta dość często zawodzi.



Jeśli konfigurowałeś już wcześniej kartę dźwiękową, program konfiguracyjny zapyta, czy użyć poprzednich ustawień, na przykład tak:

Old configuration exists in /etc/soundconf. Use it [Y/n] ?

Jeśli ustawienia te były prawidłowe, możesz odpowiedzieć twierdząco, jeśli karta nie działała poprawnie, powinieneś odpowiedzieć **n i skonfigurować ją jeszcze raz.**

Po przebrnięciu przez dość długą listę typów kart dźwiękowych będziesz musiał odpowiedzieć na kilka bardziej ogólnych pytań, np.

/dev/dsp and /dev/audio support [N/y] ?

Na to pytanie dobrze jest odpowiedzieć twierdząco, ponieważ wiele programów wykorzystujących kartę dźwiękową używa urządzeń `/dev/dsp` oraz `/dev/audio`. Musisz również zdecydować, czy chcesz mieć dostęp do interfejsu MIDI oraz syntezy FM:

```
MIDI interface support [N/y]
FM synthesis (YM2813/OPL-3) support [N/y]
```

Jeśli Twoja karta obsługuje interfejs MIDI, na pierwsze pytanie powinieneś odpowiedzieć twierdząco. Większość kart posiada syntezator FM, więc na drugie pytanie możesz odpowiedzieć twierdząco prawie zawsze (niektóre karty posiadają zarówno interfejs MIDI jak i syntezator FM).

Ostatni zestaw pytań dotyczących karty dźwiękowej związany jest z ustawieniami sprzętowymi: numerem używanego przerwania, kanału DMA i adresem portu I/O. Powinieneś w odpowiedzi na nie podać zanotowane wcześniej wartości:

```
I/O base for SB
The default value is 220
Enter the value: 200
```

```
SoundBlaster IRQ
The default value is 7
Enter the value: 5
```

```
SoundBlaster DMA
The default value is 1
Enter the value: 1
```

Możliwe, że pytań będzie jeszcze więcej – zależy to od rodzaju karty dźwiękowej i sterownika. Dane wprowadzaj uważnie, bo pomyłka oznacza konieczność wprowadzenia ich od początku.

Na zakończenie procesu konfigurowania karty dźwiękowej będziesz miał możliwość zapisania informacji konfiguracyjnych:

```
The sound card has been configured.
Save copy of this configuration in /etc/soundconf [Y/n]
```

Dobrze jest skorzystać z tej możliwości, dzięki temu bowiem przy ponownym konfigurowaniu kernela nie będziesz musiał wprowadzać wszystkich informacji od nowa.

Po zebraniu informacji dotyczących pozostałych komponentów jądra system rozpoczęna proces komplikacji. Może on zająć od kilku do kilkudziesięciu minut, zależnie od liczby sterowników i szybkości komputera.

Niestety, to jeszcze nie wszystko. Po przekompilowaniu jądra trzeba skonfigurować pliki urządzeń. Proces ten może być opisany w pliku `README` dostarczonym wraz ze sterownikiem do karty. Jeśli tak nie jest, zamiast ręcznie tworzyć plik urządzenia, zajrzyj do pliku `/usr/src/linux/drivers/sound/Readme.linux`. Na samym końcu zawiera on skrypt, który zrobi to za Ciebie. Wytnij odpowiedni fragment, zapisz go w innym pliku, nadaj mu atrybut wykonywalności i uruchom. Po rekompilacji jądra i zainstalowaniu sterowników należy ponownie uruchomić system.

Skrypt, który należy uruchomić (wyjęty z pliku `Readme.linux`), wygląda mniej więcej tak (część komentarzy została usunięta):

```

#!/bin/sh
AUDIOPERMS=622
# Create the devices
#      Mixer devices
if [ -e /dev/mixer ]; then
    rm -f /dev/mixer
fi

if [ -e /dev/mixer0 ]; then
    rm -f /dev/mixer0
fi

mknod -m 666 /dev/mixer0 c 14 0
ln -sf /dev/mixer0 /dev/mixer

if [ -e /dev/mixer1 ]; then
    rm -f /dev/mixer1
fi
mknod -m 666 /dev/mixer1 c 14 16
#      Sequencer      (14, 1)
#
if [ -e /dev/sequencer ]; then
    rm -f /dev/sequencer
fi
mknod -m 666 /dev/sequencer c 14 1

if [ -e /dev/patmgr0 ]; then
    rm -f /dev/patmgr0
fi
mknod -m 666 /dev/patmgr0 c 14 17
if [ -e /dev/patmgr1 ]; then
    rm -f /dev/patmgr1
fi
mknod -m 666 /dev/patmgr1 c 14 33

#      music     (14, 8)
#
if [ -e /dev/music ]; then
    rm -f /dev/music
fi

mknod -m 666 /dev/music c 14 8
if [ -e /dev/sequencer2 ]; then
    rm -f /dev/sequencer2
fi
ln -s /dev/music /dev/sequencer2

#      Midi devices
#
if [ -e /dev/midi ]; then
    rm -f /dev/midi           # Old name. Don't use it
fi
if [ -e /dev/midi00 ]; then
    rm -f /dev/midi00
fi
mknod -m 666 /dev/midi00 c 14 2
ln -sf /dev/midi00 /dev/midi

if [ -e /dev/midi01 ]; then
    rm -f /dev/midi01
fi
mknod -m 666 /dev/midi01 c 14 18

```

```
if [ -e /dev/midi02 ]; then
    rm -f /dev/midi02
fi
mknod -m 666 /dev/midi02 c 14 34

if [ -e /dev/midi03 ]; then
    rm -f /dev/midi03
fi
mknod -m 666 /dev/midi03 c 14 50
#
#      DSP          (14, 3)
#
if [ -e /dev/dsp ]; then
    rm -f /dev/dsp
fi
if [ -e /dev/dsp0 ]; then
    rm -f /dev/dsp0
fi
mknod -m $AUDIOPERMS /dev/dsp0 c 14 3
ln -s /dev/dsp0 /dev/dsp

#
#      DSPW          (14, 5)
#
if [ -e /dev/dspW ]; then
    rm -f /dev/dspW
fi
if [ -e /dev/dspW0 ]; then
    rm -f /dev/dspW0
fi
mknod -m $AUDIOPERMS /dev/dspW0 c 14 5
ln -s /dev/dspW0 /dev/dspW

if [ -e /dev/dspW1 ]; then
    rm -f /dev/dspW1
fi
mknod -m $AUDIOPERMS /dev/dspW1 c 14 37

#
#      SPARC compatible /dev/audio  (14, 4)
#
if [ -e /dev/audio ]; then
    rm -f /dev/audio
fi
if [ -e /dev/audio0 ]; then
    rm -f /dev/audio0
fi
mknod -m $AUDIOPERMS /dev/audio0 c 14 4
ln -s /dev/audio0 /dev/audio

#
#      DSP1          (14, 19) /dev/dsp for the second soundcard.
#                                         Also the SB emulation part of the
#                                         PAS16 card.
#
if [ -e /dev/dsp1 ]; then
    rm -f /dev/dsp1
fi
mknod -m $AUDIOPERMS /dev/dsp1 c 14 19
#
#      SPARC audio1  (14, 20)
```

```

#
#                                /dev/audio for the second soundcard.
#
#                                Also the SB emulation part of the
#                                PAS16 card.
#
#                                if [ -e /dev/audiol ]; then
#                                    rm -f /dev/audiol
#                                fi
#                                mknod -m $AUDIOPERMS /dev/audiol c 14 20
#                                #
#                                /dev/sndstat    (14,6)  For debugging purposes
#                                #
#                                if [ -e /dev/sndstat ]; then
#                                    rm -f /dev/sndstat
#                                fi
#                                mknod -m 666 /dev/sndstat c 14 6
#                                exit 0

```

Problemy z instalacją i konfiguracją karty dźwiękowej

Podobnie jak w przypadku każdego urządzenia, zdarza się, że podczas instalacji karty dźwiękowej występują problemy. Jeśli Twoja karta działa bez zarzutu, możesz oczywiście pominąć ten podrozdział.

Jeśli po skompilowaniu nowego kernela nadal nie masz możliwości korzystania z karty dźwiękowej, a podczas uruchamiania systemu nie pojawiają się żadne informacje jej dotyczące, oznacza to najprawdopodobniej, że nadal uruchamiane jest stare jądro systemu. Szczegóły dotyczące aktualnie używanego jądra uzyskać możesz wydając polecenie

```
uname -a
```

Sprawdź, czy data i czas komplikacji są prawidłowe. Jeśli nie, będziesz prawdopodobnie musiał użyć do zainstalowania nowego jądra programu LILO, a w ostateczności powtórzyć komplikację.



Plik /proc/version w większości wersji Linuxa zawiera te same informacje, które dostępne są dzięki poleceniu uname - a, więc by sprawdzić wersję jądra, możesz również przejrzeć jego zawartość.

Jeżeli używasz nowo skompilowanej wersji jądra, a pomimo tego karta dźwiękowa nie działa, powinieneś sprawdzić, czy odpowiednie sterowniki zostały w ogóle dołączone do kernela. Plik /proc/devices zawiera nazwy urządzeń znakowych i blokowych, które są obsługiwane. Poszukaj sekcji sound (zwykle ma ona numer 14). Jeśli jej nie ma, oznacza to, że sterowniki karty dźwiękowej nie zostały dołączone do kernela i musisz ponowić ich instalację.

Jeśli wpis w pliku /proc/devices jest prawidłowy i podczas uruchamiania systemu wyświetlane są informacje dotyczące karty dźwiękowej (informacje wypisywane podczas uruchamiania systemu można obejrzeć w każdej chwili używając polecenia dmesg lub przeglądając plik /var/adm/messages), możliwe, że Twoja karta nie jest kompaty-

bilna z zainstalowanym sterownikiem. Zdarza się to czasem w przypadku klonów kart Sound Blaster. Sprawdź, czy pliki README nie zawierają informacji o problemach z posiadanym przez Ciebie typem karty. Możesz również szukać pomocy na stronach internetowych producenta karty.

Plik `/dev/sndstat` również może zawierać informacje o przebiegu procesu instalacji karty dźwiękowej. Jeśli plik ten nie istnieje, oznacza to, że zapomniałeś uruchomić skrypt instalujący sterowniki (wyjęty z pliku `/usr/src/linux/drivers/sound/Readme.linux`). Wytnij odpowiedni fragment z pliku `Readme.linux`, zapisz go w osobnym pliku, przypisz mu prawo do wykonywania i uruchom – plik `/dev/sndstat` powinien zostać wygenerowany.

W pliku `/dev/sndstat` mogą znajdować się informacje o napotkanych problemach, które powinny pomóc Ci je rozwiązać, albo przynajmniej zidentyfikować. Najczęściej pojawiające się komunikaty wraz z sugerowanymi rozwiązaniami problemów przedstawia tabela 21.2.

Tabela 21.2. Komunikaty o błędach podawane w pliku `/dev/sndstat`

Komunikat	Prawdopodobna przyczyna i sugerowane rozwiązanie problemu
No such device	Kernel nie zawiera sterowników dla karty dźwiękowej; skompiluj go ponownie
No such device or address	Sterownik nie potrafił odnaleźć karty dźwiękowej w podanej przez Ciebie konfiguracji; sprawdź dane konfiguracyjne karty

Programy obsługujące kartę dźwiękową

Istnieje sporo aplikacji, które potrafią zrobić użytku z karty dźwiękowej, przeznaczonych do dość różnorodnych zastosowań. Większość z nich dostępna jest w węzłach FTP; niektóre wchodzą również w skład dystrybucji rozprowadzanych na płytach CD-ROM. Jeśli chcesz sprawdzić, czy dany program jest zainstalowany w Twoim systemie bądź też znajduje się na dysku CD-ROM, możesz użyć polecenia `find`.

Nie zamierzamy przyglądać się każdej z tych aplikacji, ponieważ jest ich kilkadziesiąt, a co miesiąc powstają nowe. Omówimy tylko kilka reprezentatywnych przykładów. Po pozostałe programy obsługujące kartę dźwiękową znajdziesz w węzłach FTP.

vplay, vrec, splay oraz srec

Program `vplay` i trzy pozostałe wchodzą w skład pakietu `snd-util`. Ich autorami są Hannu Savolainen i Michael Beck. Są to proste programy służące do nagrywania i odsłuchiwanego plików muzycznych, napisane we wczesnej fazie rozwoju Linuxa, ale dobrze spełniające swoją rolę do dzisiaj, o ile chcesz korzystać z wiersza poleceń zamiast interfejsu graficznego. Program `srec` pozwala na zapisywanie plików dźwiękowych, `splay` – na ich odtwarzanie (do nagrywania niezbędne jest urządzenie wejściowe, takie jak mikrofon albo odtwarzacz CD).



Najświeższa wersja pakietu `snd-util` powinna być dostępna w węźle FTP:

```
ftp://sunsite.unc.edu/pub/Linux/apps/sound/  
snd-util-x.x.tar.gz,  
gdzie x.x to numer wersji.
```

Czas trwania nagrania oraz częstotliwość próbkowania mogą zostać przekazane do programu `srec` jako argumenty wywołania. Dźwięk jest zapisywany w formacie `.raw`. Aby nagrać dwudziestosekundowy plik, używając częstotliwości próbkowania 21 kHz i zapisać dane do pliku `przyklad.raw`, powinieneś wydać polecenie:

```
srec -t 20 -s 21000 przyklad.raw
```

Jeśli chcesz później odtworzyć ten plik za pomocą programu `splay`, wydaj polecenie:

```
splay -s 21000 przyklad.raw
```

Jeżeli wybierzesz złą częstotliwość próbkowania, zorientujesz się od razu, ponieważ dźwięk nie będzie przypominał nagrywanego.

Polecenia `vplay` i `vrec` oparte są na poleceniach `splay` oraz `srec` i wzbogacają je o możliwość nagrywania i odtwarzania dźwięku w formacie `.wav` i `.voc` (używanym przez karty SoundBlaster).

WAVplay

WAVplay to oparty na interfejsie graficznym program do zapisywania i odtwarzania dźwięku, którego autorem jest Andre Fuechsel. Program ten pozwala również oglądać nagrany dźwięk w postaci wykresu. Można zmieniać częstotliwość próbkowania i liczbę bitów przypadających na jedną próbkę, wybierając odpowiednią opcję oferowaną przez interfejs graficzny. Możliwe jest nagrywanie zarówno dźwięku mono, jak i stereo. Program WAVplay obsługuje tylko pliki w formacie `.wav`.



Najświeższa wersja programu `WAVplay` powinna być dostępna w węźle FTP:

```
ftp://sunsite.unc.edu/pub/Linux/apps/sound/players/  
wavplayxxxx.tar.z,  
gdzie xxxx to numer wersji.
```

Sound Studio

Program Sound Studio został napisany przez Paula Sharpe. Jest to aplikacja systemu X, zawierająca praktycznie wszystkie narzędzia potrzebne przy nagrywaniu i odtwarzaniu plików dźwiękowych. Pozwala nagrywać, edytować i odtwarzać pliki w kilku różnych formatach. Napisane w języku Tcl/Tk (z niewielkimi wstawkami w języku C), Sound Studio jest naprawdę profesjonalnym programem.



Najświeższa wersja programu Sound Studio ma numer 0.21 i powinna być dostępna w węźle FTP:

<ftp://sunsite.unc.edu/pub/Linux/apps/sound/players/studio.0.2.3.tar.gz>

MixViews

MixViews to program podobny do Sound Studio, pierwotnie jednak zaprojektowany dla systemu UNIX, a następnie przeniesiony na platformę linuxową. Jego autorem jest Douglas Scott. Program ten obsługuje pliki dźwiękowe w większości znanych formataw. Dodatkowo pozwala na filtrowanie, skalowanie i mieszanie plików dźwiękowych.



Najświeższa wersja programu MixViews powinna być dostępna w węźle FTP:

<ftp://ftp.ccmrc.ucsbs.edu/pub/MixViews/source>

Dżojstiki

Niektóre gry o wiele wygodniej niż klawiaturą obsługują się za pomocą dżojstikiem. Ponieważ port dżojstika dostępny jest w większości komputerów PC (bądź to na płycie głównej, bądź na karcie dźwiękowej czy karcie I/O), warto byłoby z niego skorzystać. Niestety, odpowiedni sterownik nie jest wbudowany w Linuxa, co nie oznacza, że nie jest w ogóle dostępny – kilka sterowników zostało opracowanych. Dżojstik może być oczywiście wykorzystany tylko w tych grach, które potrafią go obsługiwać.



Najświeższa wersja sterowników dla dżojstika ma numer 0.8.0 i powinna być dostępna w zasadzie w każdym z węzłów FTP wymienionych w dodatku A, np.:

<ftp://sunsite.unc.edu/pub/Linux/kernel/patches/console/joystick-1.2.14.tar.gz>

Sterownik ten jest rozprowadzany jako ładowalny moduł jądra (ang. *loadable kernel module*), więc do jego zainstalowania niezbędny będzie odpowiedni program. Jest on dostarczany praktycznie z każdą dystrybucją Linuxa. Sterowniki ładowalne mogą być ładowane do pamięci i usuwane z niej przez poszczególne aplikacje, dzięki czemu zajmują pamięć tylko wtedy, gdy są potrzebne, i nie wymagają rekompilacji jądra.

Aby zainstalować sterownik dżojstika, rozpakuj plik zawierający go za pomocą programów gunzip i tar. Sprawdź, czy plik joystick.h zawiera prawidłowe informacje (większość dżojstików używa adresu I/O 201). Po skompilowaniu sterownika i uruchomieniu skryptu z nim dostarczonego utworzone zostaną odpowiednie urządzenia. Dżojstik stanie się dostępny dla Ciebie (i dla Dooma!). Większość wersji sterownika zawiera również prosty program umożliwiający sprawdzenie, czy działa on poprawnie.

Podsumowanie

W tym rozdziale przyjrzaliśmy się problemom związanym z konfiguracją i instalacją karty dźwiękowej i dżojstika. Ponieważ w Twoim systemie jest już zainstalowany napęd CD-ROM, możesz teraz używać wszystkich programów multimedialnych dostępnych dla systemu Linux.

Aby dowiedzieć się, jak zainstalować system X, który umożliwia używanie programów do obróbki dźwięku posiadających interfejs graficzny, zajrzyj do rozdziału 22. „Instalacja i konfiguracja XFree86”.

O programie Wabi, który pozwala uruchamiać programy napisane dla Microsoft Windows pod kontrolą systemu X, możesz dowiedzieć się więcej z rozdziału 23. „Wabi”.

Proces komplikowania jądra systemu omówiony jest w rozdziale 57. „Praca z jądem systemu”.

Część czwarta

Graficzne interfejsy

użytkownika

W tej części:

- υ Instalacja i konfiguracja XFree86
- υ Wabi
- υ Ghostscript i Ghostview

Rozdział 22.

Instalacja i konfiguracja

XFree86

Tim Parker

W tym rozdziale:

- υ Co to jest XFree86?
- υ Dystrybucja oprogramowania XFree86
- υ Konfiguracja XFree86
- υ Pliki Xconfig i XF86Config
- υ Plik .xinitrc

Większość użytkowników Linuxa chce używać wchodzącego w skład wielu dystrybucji systemu X, który udostępnia graficzny interfejs użytkownika. Wersja rozpowszechniana z Linuxem została zaprojektowana w MIT i nazywa się **XFree86**. Jest ona dostępna również dla niektórych innych platform UNIX-owych. Jest przystosowana do współpracy z wieloma urządzeniami instalowanymi w komputerach PC.

Na rynku istnieją przynajmniej dwie główne wersje systemu **XFree86** dla Linuxa: w skład niektórych dystrybucji wchodzi wersja 2.X, która jest nieco starsza od wersji 3.X oferowanej w większości nowszych dystrybucji. W tym rozdziale przyjrzymy się instalowaniu i konfigurowaniu obu tych wersji, choć większość przykładów dotyczyć będzie wersji 3.X.



Ważne jest, byś zrozumiał cały proces instalacji **XFree86** przed jego rozpoczęciem. W niektórych przypadkach nieprawidłowa konfiguracja może skończyć się nawet uszkodzeniem sprzętu (konkretnie karty graficznej).

nie monitora lub karty graficznej).

Co to jest XFree86?

XFree86 to dostępna na licencji public domain wersja systemu X11, zaprojektowana przez MIT. Prawa autorskie do niej posiada MIT Consortium. Ze względu na to, że twórcy systemu Linux starają się, aby w skład systemu nie wchodziły elementy wymagające posiadania odrębnej licencji, nie jest ona częścią systemu operacyjnego. Działa na komputerach klasy PC nie tylko pod kontrolą Linuxa, ale i z innymi systemami UNIX-owymi.

Obecnie spotyka się kilka wersji XFree86. Najpopularniejsze są wersje 2.X, oparte na standardzie X11 Release 5 (w skrócie X11R5). Najnowsze wersje mają numery 3.X i oparte są na X11 Release 6 (X11R6). Wersje o innych numerach pobocznych to przeważnie wersje poprawione i nieznacznie zmienione; mogą być one instalowane tylko na istniejącą już wersję o takim samym numerze głównym. Przykładowo, jeśli posiadasz wersję XFree86 v2.1 i załadowałeś z Internetu wersję 2.1.1, musisz ją instalować na istniejącej wersji 2.1, a nie jako samodzielny system, ponieważ wersje te nie zawierają wszystkich plików potrzebnych do działania systemu.



Nie używaj wersji XFree86 o numerze 2.0. Zawiera ona kilka poważnych błędów. Zamiast niej użyj wersji 2.1 lub 2.1.1. Wersje 1.0 nie są nawet w połowie tak przyjemne w obsłudze i stabilne jak 2.X i 3.X, więc ich również nie powinieneś używać.

Teraz trochę o terminologii. Oficjalną nazwą omawianego interfejsu graficznego jest X. Często i niepoprawnie nazywa się go również X Window lub X Windows (co zdecydowanie zbyt mocno pachnie produktami firmy Microsoft). Zwykle terminy X, X11, XFree86 i X Window mogą być używane zamiennie, ale raczej wystrzegaj się nazwy X Windows. Jej użycie jest jednym z prostszych sposobów rozzłoszczenia weterana systemów UNIX-owych i pokazania się z nie najlepszej strony.

Na początku powstawania systemu XFree86 narosło wokół niego kilka problemów, głównie z powodu braku dostatecznie szczegółowych informacji z X Consortium (instytucji kontrolującej standard X). By je rozwiązać, zawiązano instytucję o nazwie XFree86 Project Inc., która weszła w skład X Consortium, co pozwoliło na dostęp do informacji o nowych wersjach standardu X jeszcze przed ich opublikowaniem. XFree86 jest teraz znakiem handlowym XFree86 Project Inc.

Wiele linuxowych wersji XFree86 zawiera katalogi i odniesienia do produktu o nazwie X386. Była to wcześniejsza wersja X11R5 dla komputerów PC. W spadku po niej

XFree86 przejął nazewnictwo plików i katalogów. Są to dziś jednak dwa zupełnie oddzielne systemy.

XFree86 wymaga co najmniej 8 MB pamięci RAM i 16 MB całkowitej pamięci wirtualnej. Innymi słowy, jeśli posiadasz 8 MB pamięci RAM, potrzebujesz jeszcze co najmniej 8 MB partycji wymiany. Jeśli posiadasz 16 MB RAM-u, użycie partycji wymiany nie jest konieczne, ale jest zalecane ze względów bezpieczeństwa, szczególnie jeśli zamierzasz używać aplikacji wymagających dużych ilości pamięci. Jeśli planujesz wykorzystywać system X naprawdę intensywnie, powinieneś mieć w swoim systemie co najmniej 32 MB pamięci wirtualnej (najlepiej co najmniej 16 MB fizycznej pamięci RAM i co najmniej 16 MB partycji wymiany).

Wersje 2.X systemu XFree86 mogą być skonfigurowane tak, by działały już przy 4 MB pamięci RAM, lecz zarówno konfiguracja, jak i późniejsze działanie są wolne i raczej nie warto zwracać sobie nimi głowy. Wersje 3.X nie działają poprawnie przy 4 MB RAM-u (tzn. da się je uruchomić, ale działają tak wolno, że trudno to nazwać działaniem poprawnym). Działają dobrze już przy 8 MB pamięci RAM, ale zalecane jest 16 lub 32 MB.

Dystrybucja oprogramowania XFree86

Większość wersji XFree86 dostarczanych jest jako część dystrybucji Linuxa na płytach CD-ROM. W tym rozdziale przyjmiemy, że posiadasz właśnie taki CD-ROM (choć podane dalej informacje stosują się również prawie w całości do wersji rozprowadzanych przez sieć czy na dyskietkach).

Oprogramowanie XFree86 zwykle umieszczane jest w katalogach o nazwach `x1`, `x2`, `x3` itd. lub w archiwach o podobnych nazwach. Aplikacje dla tego systemu umieszczane są w katalogach `xap1`, `xap2` itd. lub też w archiwach, których nazwy zaczynają się od litery X. Zwykle w każdym katalogu znajduje się plik tekstowy opisujący zastosowanie zawartych w nim plików.

Zanim zainstalujesz system X, powinieneś sprawdzić, czy będzie on działał poprawnie z już zainstalowanym oprogramowaniem. Zwykle wymagana jest odpowiednia wersja jądra, biblioteki `libc` oraz `ld.so`. Plik tekstowy dołączony do dystrybucji powinien rozwiać wszystkie wątpliwości. Jeśli instalujesz system X dostarczony na tym samym CD-ROM-ie, co reszta systemu, możesz spokojnie darować sobie sprawdzanie wersji.

Oprogramowanie może zostać zainstalowane ręcznie przez rozpakowanie każdego z archiwów. Rozpakowane pliki muszą znaleźć się w określonych katalogach. Jest to jednak długotrwały i nudzący proces, o wiele wygodniej jest więc użyć programu instalacyjnego, takiego jak `setup`.

Katalogi używane przez wersje 2.X systemu XFree86 pokrywają się w zasadzie z tymi używanymi w X386. W większości systemów katalogiem głównym jest `/usr/X386`.

Aby zachować spójność, wersje X11R5 i X11R6 zwykle tworzą odpowiednie dowiązania, dzięki czemu pliki mogą zawsze zostać łatwo zlokalizowane.



Wersje 3.X nie znajdują się już w katalogu /usr/X386. Zostały przeniesione do katalogu o nazwie /usr/X11R6. Jeśli aktualniasz starszą wersję X, pamiętaj o tym i po instalacji albo utwórz odpowiednie dowiązania, albo całkowicie usuń starą wersję. Pamiętaj też o zmianie ścieżek przeszukiwania.



Aby uprościć i ujednolicić strukturę katalogów, zwykle tworzone jest dowiązanie o nazwie usr/X11, wskazujące na katalog zawierający system X. Sprawdź, jakie dowiązania są stworzone w Twoim systemie, sprawdź również czy zmienna środowiskowa PATH zawiera odniesienie do tego katalogu.

Wybór serwera X

Przed zainstalowaniem systemu XFree86 musisz zdecydować, jakiego typu serwera będziesz używał. Serwery XFree86 to rodzaj sterowników dla kart graficznych. Wybór złego typu sterownika może spowodować nieprawidłowe zachowanie się systemu (a w niektórych przypadkach nawet uszkodzenie karty graficznej lub monitora). Upewnij się więc, że wybrany sterownik odpowiada zainstalowanemu w Twoim komputerze sprzętowi.

Istnieje kilka typów serwerów XFree86, a ich nazwy mówią, dla jakich typów kart graficznych są one przeznaczone. Wraz z większością wersji dostępne są serwery obsługujące następujący sprzęt:

- XF86_Mono** monochromatyczne karty graficzne (serwer ogólny);
- XF86_VGA16** 16 kolorowe karty VGA (serwer ogólny);
- XF86_SVGA** kolorowe karty SVGA (serwer ogólny);
- XF86_S3** akcelerowany serwer dla kart opartych na S3;
- XF86_Mach8** akcelerowany serwer dla kart opartych na Mach8;
- XF86_Mach32** akcelerowany serwer dla kart opartych na Mach32;
- XF86_8514** akcelerowany serwer dla kart opartych na 8514/A.

Serwery ogólne nie zawierają instrukcji specyficznych dla jakiegoś rodzaju kart, powinny więc działać ze wszystkimi kartami spełniającymi dane założenia. Na przykład, serwer XF86_S3 zawiera kod, który działa poprawnie tylko w przypadku użycia karty

opartej na układach S3. Zajrzyj do dokumentacji swojej karty graficznej (lub użyj jakiegoś programu diagnostycznego), aby dowiedzieć się, jakie układy zainstalowane są w Twojej karcie. Zdarza się, że do wersji XFree86 dołączone są również inne typy serwerów, więc powinieneś poszukać w ich dokumentacji informacji o kompatybilności z używanym przez Ciebie sprzętem.

Serwery ogólne działają z większością kart obsługujących standard VGA i SVGA. Ponieważ zawierają one tylko najbardziej podstawowe instrukcje pozwalające na komunikowanie się z kartą, nie będą dostępne żadne rozszerzenia oferowane przez dany model karty. Z tego właśnie powodu opracowane zostały serwery akcelerowane, specyficzne dla określonego typu urządzeń, ale za to pozwalające wykorzystać w pełni ich możliwości.



Zainstalowanie nieprawidłowego sterownika karty graficznej może spowodować uszkodzenie jej i monitora! Jeśli nie masz pewności, którego serwera powinieneś użyć, użyj jednego z typów ogólnych. Większość modeli kart graficznych potrafi z nimi bez problemów współpracować.

Większość dystrybucji `XFree86` zawiera predefiniowane w plikach konfiguracyjnych ustawienia dla standardowych kart VGA. Są to ustawienia bezpieczne, ale mimo wszystko powinieneś przejrzeć pliki konfiguracyjne przed pierwszym uruchomieniem systemu X.

Nazwa używanego serwera jest zmieniana przez modyfikację dowiązania symbolicznego o nazwie `/usr/X386/bin/X`. Możesz w każdej chwili zmienić typ serwera przez określenie nowej nazwy pliku, na który dowiązanie to będzie wskazywać. Przykładowo: jeśli używasz serwera VGA, a w systemie zainstalowana jest karta SVGA, możesz zmienić serwer za pomocą polecień:

```
rm /usr/X386/bin/X  
ln -s /usr/X386/bin/XF86_SVGA /usr/X386/bin/X
```

Pierwsze polecenie usuwa istniejące dowiązanie, drugie tworzy je na nowo, z tym że teraz wskazuje ono na serwer `/usr/X386/bin/XF86_SVGA`. Nazwy katalogów mogą być nieco inne, w zależności od używanej wersji (jeśli utworzone zostały odpowiednie dowiązania, nie ma to żadnego znaczenia).

Ręczna instalacja XFree86

Jak wspomniano wcześniej, system XFree86 może zostać zainstalowany ręcznie. Dzięki temu możliwe jest umieszczenie plików w innych niż domyślne lokalizacjach. Niektórzy użytkownicy przeprowadzają instalację ręczną, ponieważ pozwala im ona zorientować się dokładnie, co dzieje się w każdym jej etapie. Jest to doskonała metoda na poznanie systemu XFree86 w całej jego złożoności, ale może zabrać naprawdę dużo czasu.

Aby ręcznie zainstalować system X, musisz rozpakować odpowiednie archiwa do właściwych katalogów poleceniem `gzip`. Proces ten nie jest skomplikowany.

- ⦿ Zaloguj się jako `root`.
- ⦿ Utwórz katalog `/usr/X386`. Może on już istnieć – niektóre programy instalacyjne tworzą go automatycznie.
- ⦿ Przejdź do katalogu `/usr/X386`.
- ⦿ Za pomocą polecień `gzip` i `tar` rozpakuj i zainstaluj wszystkie pliki wchodzące w skład pakietu X. Składnia polecenia, które należy wydać w tym celu, jest następująca:

```
gzip -dc rozpakowywany_plik | tar xvof -
```
- ⦿ Powtórz ten proces dla każdego ze składników pakietu X, który chcesz zainstalować. Będziesz tym celu musiał przejść do każdego z katalogów wchodzących w skład dystrybucji i rozpakować wszystkie zawarte w tych katalogach archiw.

Opcje programu `tar` użyte w podanym wyżej poleceniu powodują, że plikom przypisywany jest odpowiedni właściciel oraz że na ekranie wyświetlane są dodatkowe informacje. Po rozpakowaniu wszystkich plików do odpowiednich katalogów możesz kontynuować proces konfiguracji.

Instalowanie XFree86 za pomocą skryptu

Większość użytkowników preferuje jednak instalację automatyczną. Jest to metoda szybsza i mniej podatna na błędy. Z tego powodu większość dystrybucji oprogramowania X zawiera odpowiednie programy instalujące.

Program instalacyjny systemu Linux zwykle daje możliwość automatycznego zainstalowania XFree86. Jeśli zdecydowałeś się na takie rozwiązanie, wszystkie pliki są już na właściwych miejscach. Jeśli podczas instalacji nie miałeś możliwości zdecydowania, czy chcesz zainstalować system X, odpowiednie pliki mogły zostać zainstalowane automatycznie. Sprawdź, czy w katalogach `/usr/X386/bin` lub `/usr/X11R5/bin` znajdują się pliki. Jeśli jest ich tam dość sporo, oznacza to, że system X został zainstalowany.

Nawet jeśli system XFree86 został zainstalowany automatycznie, nie oznacza to, że można go już używać. Należy go jeszcze odpowiednio skonfigurować za pomocą takich, programów jak `ConfigFX86` czy `fx86config`, lub też edytując ręcznie plik `Xconfig` albo `XF86Config` (w zależności od wersji XFree86). W przypadku instalacji automatycznej pliki te są zwykle skonfigurowane tak, by można było uruchomić system X z kartami VGA lub SVGA, nie zaszkodzi jednak sprawdzić, jakie dane znajdują się w tych plikach przed uruchomieniem XFree86.

Zmienna środowiskowa PATH

Ważne jest, by pliki wykonywalne systemu X były dostępne w ścieżce przeszukiwania, określonej wartością zmiennej `path` lub `PATH` (w zależności od interpretera poleceń). Miejsce, w którym znaleźć można definicję tej zmiennej, również zależy do tego, jakiego interpretera poleceń używasz. Do definicji tej zmiennej należy dodać katalogi `/usr/X386/bin` (dla wersji 2.X) albo `/usr/X11R6/bin` (dla wersji 3.X).

Przykładowo, jeśli używasz powłoki `bash`, polecenia inicjalizujące zmienne środowiskowe zapisane są w pliku `.profile` znajdującym się w katalogu domowym każdego użytkownika. Jeśli jesteś zalogowany jako `root`, możesz również posiadać własną kopię tego pliku lub używać systemowego pliku domyślnego `/etc/profile` (nazwa tego pliku nie zaczyna się od kropki – konwencja ta oznacza, że ten plik konfiguracyjny jest dostępny dla wszystkich użytkowników).

Jeśli katalog zawierający pliki wykonywalne systemu X nie wchodzi w skład definicji ścieżki przeszukiwania, należy ją zmodyfikować. Przykładowy wpis w pliku `.profile` może wyglądać tak:

```
PATH="/sbin:/usr/bin:/bin:/usr/X11/bin:/usr/openwin/bin"
```

Składnia polecenia definiującego ścieżki przeszukiwania jest nieco inna w przypadku powłoki `tcsh`. Plikiem konfiguracyjnym tego interpretera jest plik o nazwie `.login` lub `.csh.login`, a odpowiednie polecenie ma postać:

```
set path = ( /sbin /usr/bin /bin /usr/X11/bin /usr/openwin/bin . )
```

Oczywiście nazwy katalogów w Twoim systemie mogą być nieco inne od przykładowych, ale po tym, jak dodasz do nich ścieżkę dostępu do katalogu XFree86 i zalogujesz się ponownie, pliki wykonywalne systemu X powinny być dostępne dla interpretera poleceń.

Konfigurowanie XFree86

Zanim będzie można uruchomić `XFree86`, należy wprowadzić jeszcze kilka informacji konfiguracyjnych. Jest to część procesu instalacji sprawiająca użytkownikom najwięcej kłopotów, a poprawne wprowadzenie wszystkich danych do plików konfiguracyjnych tak, by system X działał bez zarzutu, może być dość złożone. Poniżej zamieścimy więc instrukcje pozwalające krok po kroku przejść przez ten proces.

Wraz z wieloma dystrybucjami Linuxa dostarczany jest program `ConfigX86` lub `fx86config`¹. Upraszczca on znacząco proces konfiguracji, o ile w systemie zainstalowa-

¹ Niektóre z dystrybucji RedHat zawierają również skrypt `xconfigurator`, spełniający podobną rolę jak wymienione programy (*przyp. tłum.*).

ny jest jeden z obsługiwanych modeli kart graficznych. Ich lista dostępna jest w pliku `Hardware HOWTO`. Jeśli plik taki nie został dołączony do dystrybucji, której używasz, możesz zdobyć go w większości węzłów FTP i BBS. Upewnij się, że plik dotyczy używanej przez Ciebie wersji Linuxa. Więcej informacji o programach `ConfigFX86` i `fx86config` znaleźć możesz w podrozdziale Używanie programów `ConfigFX86` i `fx86config`” (w niektórych przypadkach możesz również podać tym programom dość informacji, by skonfigurowały X do współpracy z kartą, która nie jest obsługiwana – ten temat omówiony jest dokładniej w następnym podrozdziale).

Jeśli Twoja karta nie jest wyszczególniona w pliku `Hardware HOWTO`, a nie chcesz używać sterowników ogólnych, musisz skonfigurować system XFree86 ręcznie. Nawet jeśli uda Ci się zaprzecząć do tego wymienione wyżej programy, możliwe, że i tak konieczne będzie wprowadzenie pewnych modyfikacji do plików konfiguracyjnych.

Dane konfiguracyjne systemu X w wersji 2.X zapisane są w większości w pliku `xconfig`, natomiast wersje 3.X używają do tego celu plików `xconfig` lub `XF86Config`. Niestety, dokumentacja dotycząca zawartych w tych plikach ustawień jest rozrzucona po kilku plikach. W większości przypadków powinieneś zajrzeć do plików `README`, `VideoModes.doc`, `README.Config` oraz `README.Linux`. Powinieneś również przeczytać strony `man` dotyczące programów `Xconfig`, `XF86Config`, `XFree86` i `XFree86kbd`. Na koniec powinieneś jeszcze przejrzeć strony `man` dotyczące serwera, którego zamierzasz użyć (wydrukowanie dokumentacji ułatwi Ci dostęp do niej).

Jeśli chcesz skompletować dane potrzebne do skonfigurowania plików `xconfig` i `XF86Config`, musisz znać następujące informacje:

- rodzaj serwera, którego zamierzasz używać;
- typ myszki przyłączonej do komputera i port, do którego jest ona podłączona;
- model karty graficznej i typ zastosowanych w niej układów (chipset); jeśli nie wiesz, jakie układy zainstalowane są na karcie graficznej, poszukaj odpowiednich informacji w dokumentacji karty lub uruchom jakiś program diagnostyczny (np. `SuperProbe` dla Linuxa lub `MSD` dla systemu DOS);
- marka i model monitora, którego używasz, oraz jego wielkość; dobrze jest również znać maksymalne dopuszczalne częstotliwości odświeżania poziomego i pionowego, które zwykle są podane w instrukcji obsługi;
- typ używanej klawiatury – jeżeli jest to klawiatura „inna niż wszystkie”; większość użytkowników posiada klawiatury w standardzie U.S, ale zdarzają się też inne modele.

Jeśli nie posiadasz którejś z tych informacji i nie potrafisz ich zdobyć, powinieneś przejrzeć dokumentację dostarczaną z `XFree86`, umieszoną zwykle w katalogu `/usr/X11/lib/X11/doc` (który może być dowiązaniem do katalogu `/usr/X386/lib/X11/doc` lub `/usr/X11R6/lib/X11/doc`). Zawiera ona informacje o wielu typach kart graficznych

i monitorów wraz z prawidłowymi danymi konfiguracyjnymi (takimi jak na przykład częstotliwości odświeżania dla monitorów, które zawsze trudno określić, ponieważ instrukcja gdzieś się zapodziała...).

Po zdobyciu tych informacji jesteś gotowy do rozpoczęcia konfiguracji plików `xconfig` i `XF86Config`.

Gdzie umieścić pliki Xconfig i XF86Config

Pliki `xconfig` i `XF86Config` mogą zostać umieszczone w kilku różnych miejscach w systemie plików. Zwykle znajdują się one w katalogu `/usr/X386/lib/X11`, w którym można również znaleźć pliki przykładowe. Jeśli masz dostęp do tego katalogu, jest to dobre miejsce dla Twoich plików konfiguracyjnych (ściślej rzecz ujmując pliki te odczytywane są z katalogu `/usr/X11R6/lib/X11` lub `/etc`, ale ponieważ zwykle katalog `/usr/X386` jest dowiązaniem do katalogu `/usr/X11R6`, obie ścieżki dostępu wskazują na to samo miejsce; w dokumentacji również można natknąć się na odniesienia do katalogu `X11R6`, ale dopóki utworzone jest dowiązanie `/usr/X386`, możesz używać obu wersji ścieżek dostępu).



Nie musisz martwić się o to, czy powinieneś używać pliku `Xconfig` czy `XF86Config`, o ile nie przeprowadzasz instalacji ręcznej. Skrypty instalacyjne same skonfigurują odpowiedni plik. Jeśli instalujesz X ręcznie, użyj pliku `Xconfig` w przypadku systemu w wersji 2.X i `XF86Config` w przypadku wersji 3.X.

Jeśli nie możesz używać katalogu `/usr/X386/lib/X11` (na przykład ponieważ nie masz prawa zapisu do tego katalogu albo jest on zainstalowany gdzieś na serwerze zdalnym) lub też nie chcesz go używać (gdy zamierzasz stworzyć konfigurację tylko dla siebie, niedostępna dla innych użytkowników), możesz umieścić pliki konfiguracyjne w katalogu `/etc` lub w swoim katalogu domowym. Umieszczenie ich w katalogu `/etc` oznacza, że będą one dostępne dla wszystkich użytkowników.

Mözesz również utworzyć osobne pliki konfiguracyjne dla różnych komputerów, zapisując je w katalogu `/usr/X386/lib/X11` z dołączoną do nazwy pliku nazwą komputera. Przykładowo, do danych konfiguracyjnych zapisanych w pliku o nazwie `Xconfig.merlin` dostęp mogą mieć tylko użytkownicy logujący się z komputera o nazwie `merlin`.

W systemie Linux plik `xconfig` zwykle umieszczany jest w katalogu `/etc`. Ponieważ pozostałe pliki systemu X umieszczane są w innych katalogach, należy stworzyć dowiązanie wskazujące na plik `/etc/xconfig` w katalogu `/usr/X386/lib/X11` lub w katalogu domowym. Dzięki temu system X będzie mógł prawidłowo odnaleźć plik konfiguracyjny.

SuperProbe

SuperProbe to program, który próbuje wykryć rodzaj zainstalowanej w systemie karty graficznej. Obsługuje architektury ISA, EISA, VLB, częściowo PCI, nie obsługuje natomiast architektury MCA (choć do czasu wydania tej książki mogą ukazać się już nowsze wersje tego programu). Jeśli wiesz, jaką kartę posiadasz, program SuperProbe nie jest Ci raczej potrzebny.

SuperProbe stara się zidentyfikować kartę graficzną, próbując zapisu do rejestrów dostępnych tylko w poszczególnych typach kart graficznych i obserwując rezultaty. Takie rozwiązanie ma niestety jedną wadę: może spowodować, że komputer przestanie odpowiadać. Choć mało prawdopodobne jest, by spowodowało to jakieś zniszczenia, system plików musi zostać sprawdzony, jeśli trzeba było zresetować komputer. Z tego powodu przed uruchomieniem tego programu powinieneś upewnić się, że jesteś aktualnie jedynym użytkownikiem systemu (żeby nie robić nikomu przykrych niespodzianek). Zalecane jest również utworzenie kopii zapasowej ważnych danych.



Uruchamianie programu SuperProbe bez żadnych parametrów to prawie pewny sposób na zawieszenie komputera. Używaj tego programu ostrożnie i przestrzegaj rad podanych w dalszej części tego podrozdziału, dotyczących podawania temu programowi choć szczegółowych informacji o tym, czego powinien szukać.

Program SuperProbe dołączony jest do większości dystrybucji zawierających XFree86. Może również zostać załadowany z sieci Internet. Nie jest to program działający wyłącznie pod Linuxem – można go uruchomić w niektórych innych systemach UNIX-owych na komputerach PC. Dostępna jest również strona [man](#) poświęcona programowi SuperProbe.

Aby dostosować zachowanie SuperProbe do konkretnego systemu, używa się szeregu opcji podawanych w wierszu poleceń. Choć bardziej zaawansowane opcje różnią się w zależności od wersji oprogramowania, najważniejsze z nich są we wszystkich wersjach takie same:

- bios** pozwala podać adres BIOS-owy karty graficznej, domyślnie ustawiony na C0000;
- info** wyświetla listę kart rozpoznawanych przez program SuperProbe;
- no_16** wyłącza testowanie 16-bitowe – opcja użyteczna w przypadku starych, ośmiobitowych kart graficznych;
- no_bios** wyłącza testowanie BIOS-u karty graficznej i zakłada, że jest to BIOS kompatybilny z EGA, VGA, SVGA albo jeszcze nowszy; opcja ta może się przydać w przypadku nowych modeli kart, jeśli testowanie kończy się zawieszeniem systemu;

- no_dac** wyłącza testowanie typu układu RAMDAC; można używać tej opcji w przypadku kart VGA i SVGA, aby zapobiec zawieszaniu się systemu;
- no_mem** pomija sprawdzanie ilości pamięci RAM dostępnej na karcie graficznej;
- order** pozwala podać kolejność, w jakiej należy testować chipset; opcja ta może być przydatna, gdy wiesz, jaki typ karty posiadasz, i chcesz się tylko upewnić;
- verbose** wyświetla na ekranie informacje o podejmowanych przez SuperProbe próbach; jest to bardzo przydatna opcja, pozwalająca na śledzenie procesu wykrywania karty graficznej i zidentyfikowanie przyczyny ewentualnych problemów.

Jednym z pierwszych kroków powinno być wyświetlenie rozpoznawanych przez SuperProbe typów kart graficznych poleceniem

```
SuperProbe -info
```

które pozwoli obejrzeć typy kart, chipsety i rodzaje układów RAMDAC, z którymi radzi sobie ten program. Zauważ, że nazwa programu SuperProbe składa się zarówno z małych, jak i z wielkich liter, co nie jest często spotykane w przypadku programów przeznaczonych dla systemów UNIX-owych.

Jeśli posiadasz starszą, ośmioróżową kartę graficzną, możesz spróbować zlecić jej rozpoznanie poleceniem

```
SuperProbe -no_16 -verbose
```

Jeśli posiadasz kartę 16-bitową i podejrzewasz, że jest to S3, Cirrus Logic lub Tseng, możesz użyć opcji **-order**, dzięki czemu skrócisz procedurę testowania i unikniesz potencjalnych problemów:

```
SuperProbe -order S3,Cirrus,Tseng -verbose
```

Pomiędzy typami chipsetów nie powinno być spacji. Ograniczenie zakresu poszukiwań jest zalecane w celu uniknięcia zawieszania się komputera. Nawet jeśli wiesz, jaki typ karty jest zainstalowany w systemie, nie możesz zakładać, że SuperProbe będzie działał prawidłowo. Ma on „brzydkiego” zwyczaj zawieszania komputera z powodu konfliktów sprzętowych. Używaj go ostrożnie.

Użycie programów ConfigFX86 i fx86config

Programy ConfigFX86 i fx86config posiadają prosty interfejs, pozwalający na wybór modelu karty graficznej i monitora. Jeśli Twoja karta i monitor są obsługiwane przez te programy (możesz to sprawdzić w pliku `Hardware HOWTO` i plikach `README` dostarczanych wraz z `XFree86`), proces konfiguracji masz już prawie za sobą. Jeśli programy

`ConfigFX86` i `fx86config` są dołączone do `XFree86`, znajdują się zwykle w katalogu `/usr/X386/bin`. Autorem programu `ConfigFX86` jest Stephen Zwaska.

Dokumentacja tych programów zwykle dostępna jest w katalogu `/usr/X386/bin`, wraz z plikami wykonywalnymi, w kilku różnych formatach. Nie jest ona jednak dostarczana ze wszystkimi wersjami Linuxa. Dokument w formacie ASCII nazywa się `ConfigFX86.txt`, natomiast jego wersja postscriptowa – `ConfigFX86.ps` (podobnie sprawa wygląda w przypadku dokumentacji programu `fx86config`).

Po uruchomieniu programu `ConfigFX86` lub `fx86config` wyświetlanych jest kilka ogólnych komunikatów, po czym należy podać informacje o konfiguracji sprzętowej systemu. W większości przypadków sprowadza się to do wybrania odpowiednich wartości z wyświetlanej przez program listy. Na podstawie dostarczonych informacji budowany jest plik `xconfig`.

Niezbędne jest również podanie innych danych konfiguracyjnych, np. program `fx86config` wymaga informacji dotyczących typu myszki, wykorzystania trzeciego klawisza myszki, poziomych i pionowych częstotliwości odświeżania monitora, modelu monitora, serwera, który należy uruchomić, ilości pamięci RAM dostępnej na karcie graficznej itd. Po zebraniu wszystkich informacji program umożliwia wygenerowanie odpowiednio skonfigurowanego pliku `XF86Config`.

Po stworzeniu pliku `XF86Config` czy `xconfig` powinieneś wstrzymać się jeszcze z uruchomieniem X i poświęcić chwilę na sprawdzenie poprawności informacji zawartych w tych plikach, aby uniknąć uszkodzenia sprzętu. Następny podrozdział, omawiający konfigurację ręczną, wyjaśni znaczenie poszczególnych wpisów. Po upewnieniu się, że wszystko jest w porządku, możesz uruchomić X polecienniem `startx`.

Pliki `Xconfig` i `XF86Config`

Jeśli tworzysz plik konfiguracyjny ręcznie, musisz znać jego format. Każda z wersji `XFree86` zawiera przynajmniej jeden plik przykładowy, zwykle o nazwie `xconfig.eg` lub `XF86Config.eg`, znajdujący się w katalogu `lib`. Powinieneś użyć go jako szablonu (po skopiowaniu do katalogu domowego i usunięciu rozszerzenia `.eg`) przy tworzeniu pliku uwzględniającego konfigurację sprzętową Twojego systemu.

Pliki konfiguracyjne `xconfig` i `XF86Config` nie są krótkie, ale większość tekstu w nich zawartego to komentarze. Podzielone są one na szereg sekcji, z których każda dotyczy ustawień innego rodzaju. Ogólny porządek tych sekcji jest następujący:

- υ ścieżki dostępu do plików wykonywalnych oraz czcionek;
- υ informacje dotyczące klawiatury;

- υ informacje dotyczące myszki;
- υ plik serwera;
- υ informacje o wyświetlaniu.

Przyjrzymy się bardziej szczegółowo każdej z tych sekcji. Jeżeli wygenerowałeś plik konfiguracyjny za pomocą programów ConfigFX86 albo fx86config, sprawdź, czy odpowiednie sekcje zawierają informacje adekwatne do konfiguracji Twojego systemu. Jeśli tworzysz plik ręcznie, rób to uważnie i powoli, a na pewno uda Ci się uniknąć błędów.



Fragmenty kodu podane w dalszej części tego podrozdziału pochodzą z pliku XF86Config wygenerowanego przez XFree86 w wersji 3.X, ponieważ jest to wersja najnowsza i jest ona dodawana do większości nowych dystrybucji. Format pliku Xconfig pochodzącego z wersji 2.X jest bardzo podobny, jeśli więc posiadasz właśnie tę wersję, również nie powinieneś mieć problemów ze zrozumieniem informacji zawartych w pliku konfiguracyjnym.

Zauważ, że każda z sekcji pliku konfiguracyjnego rozpoczyna się od słowa `section`, po którym następuje nazwa sekcji, natomiast kończy się słowem `EndSection`. Pozwala to na łatwiejsze odnalezienie potrzebnej informacji. Komentarze oznaczane są znakiem `#` na początku wiersza.

Ścieżki dostępu

W większości przypadków ścieżki dostępu zapisane w pliku konfiguracyjnym są prawidłowe i nie wymagają modyfikacji, chyba że instalowałeś X ręcznie do katalogów o nazwach innych niż domyślne. Ścieżki dostępu potrzebne do odnalezienia plików zawierających definicje czcionek i innych plików systemowych określone są w sekcji wyglądającej na przykład tak:

```
Section "Files"

# The location of the RGB database. Note, this is the name of the
# file minus the extension (like ".txt" or ".db"). There is normally
# no need to change the default.

RgbPath      "/usr/X11R6/lib/X11/rgb"

# Multiple FontPath entries are allowed (which are concatenated
# together),
# as well as specifying multiple comma-separated entries in one FontPath
# command (or a combination of both methods)

FontPath      "/usr/X11R6/lib/X11/fonts/misc/"
FontPath      "/usr/X11R6/lib/X11/fonts/75dpi/:unscaled"
FontPath      "/usr/X11R6/lib/X11/fonts/100dpi/:unscaled"
FontPath      "/usr/X11R6/lib/X11/fonts/Type1/"
```

```

FontPath      "/usr/X11R6/lib/X11/fonts/Speedo/"
FontPath      "/usr/X11R6/lib/X11/fonts/75dpi/"
FontPath      "/usr/X11R6/lib/X11/fonts/100dpi/"

# For OSs that support Dynamically loaded modules, ModulePath can be
# used to set a search path for the modules. This is currently supported
# for Linux ELF, FreeBSD 2.x and NetBSD 1.x. The default path is shown
# here.

#     ModulePath      "/usr/X11R6/lib/modules"

EndSection

```

Powyższy kod definiuje ścieżki dostępu do czcionek ekranowych i bazy danych RGB. Jeśli zainstalowałeś X w sposób standardowy, nie powinieneś tu nic zmieniać.

Zauważ, że odniesienia do katalogów są zgodne z konwencją X, czyli używają nazwy katalogu /usr/X11R6. Przeważnie katalog ten dołączony jest do /usr/X11 i /usr/X386. Upewnij się, że nazwy te faktycznie prowadzą do katalogów zawierających definicje czcionek wchodzące do każdego z nich i oglądając jego zawartość. Jeśli katalog nie istnieje lub jest pusty, system X nie będzie potrafił załadować czcionek i nie uruchomi się generując komunikaty o błędach.

Jeśli instalujesz nowe czcionki dla systemu XFree86, powinny one znaleźć się właśnie w jednym z katalogów określonych w pliku XF86Config.

Ustawienia klawiatury

W większości systemów domyślnie obsługiwana jest standardowa klawiatura U.S. posiadająca 101 klawiszy. Choć wprowadzanie modyfikacji w tej sekcji nie jest konieczne prawie we wszystkich systemach, kilka drobnych zmian może znacznie ułatwić pracę w systemie X, więc nie ignoruj jej. Oto przykładowy fragment pliku XF86Config:

```

Section "Keyboard"

    Protocol      "Standard"

    # when using XQUEUE, comment out the above line, and uncomment the
    # following line

    #     Protocol      "Xqueue"

    AutoRepeat 500 5

    # Let the server do the NumLock processing. This should only be required
    # when using pre-R6 clients
    #     ServerNumLock

    # Specifiy which keyboard LEDs can be user-controlled (eg, with xset(1))
    #     Xleds          1 2 3

    # To set the LeftAlt to Meta, RightAlt key to ModeShift,
    # RightCtl key to Compose, and ScrollLock key to ModeLock:

```

```
#      LeftAlt      Meta
#      RightAlt     ModeShift
#      RightCtl     Compose
#      ScrollLock   ModeLock

EndSection
```

Protokół powinien pozostać zdefiniowany jako standard. Wiersz dotyczący Xqueue jest zaznaczony jako komentarz i powinien tak pozostać, chyba że zainstalujesz w swoim systemie Xqueue. Ustawienie autorepeat decyduje o tym, jak długo system będzie czechał przed rozpoczęciem powtarzania naciśnięcia klawisza (przykładowo, jeśli wciśniesz klawisz x, po pewnej liczbie milisekund system rozpoczęcie generować kolejne znaki x).

Wartość ServNumLock decyduje o tym, czy znacznik NumLock powinien być załączony po uruchomieniu systemu X. W większości przypadków wiersz ten jest zaznaczony jako komentarz i może tak pozostać, chyba że chcesz to zmienić (jest to zalecane w wersji 2.X lub wcześniejszych).

Teoretycznie można zdefiniować wartość Xleds, aby pozwolić na programowe załączanie i wyłączanie diod (NumLock, CapsLock i ScrollLock) na klawiaturze. Nie jest to szczególnie przydatna opcja, można więc pozostawić ją zaznaczoną jako komentarz.

Pozostała część sekcji określa zachowanie klawiszy Alt, Control i Shift. Niektóre aplikacje oczekują specjalnych kombinacji klawiszy, nazywanych metaklawiszami, polegających na wciskaniu pewnych klawiszy w czasie, gdy wciśnięty jest inny klawisz (jak na przykład Control+C w systemach DOS i UNIX). Tu możesz określić, które z klawiszy będą interpretowane jako klawisze Alt, Meta, Control i ModeLock. Ponieważ tylko nieliczne aplikacje wymagają specjalnej obsługi klawiatury, w większości systemów tę część również można pozostawić zaznaczoną jako komentarz.

Möżesz również skonfigurować system XFree86 tak, by automatycznie tłumaczył pewne kombinacje klawiszy na znaki narodowe. W większości przypadków układ klawiatury odczytywany jest z jądra systemu, ale można zmienić to zachowanie. Standard X11 pozwala na definiowanie tylko czterech tablic klawiszy, czyli o wiele mniej, niż oferuje Linux.

Definiowanie myszki

W systemie XFree86 myszka jest wykorzystywana bardzo intensywnie, więc jeśli chcesz używać tego systemu musisz podać jej typ i port, do którego jest ona podłączona. Obsługiwane są najpopularniejsze typy myszek, a nietypowe modele mogą być używane dzięki emulacji jednego z popularnych typów takich jak Microsoft czy Logitech. Sekcja pliku XF86Config dotycząca myszki nazywa się Pointer. Poniżej przedstawiamy przykładową zawartość takiej sekcji.

```
Section "Pointer"
```

```

Protocol      "Microsoft"
Device        "/dev/mouse"

# When using XQUEUE, comment out the above two lines, and uncomment
# the following line.

#      Protocol "Xqueue"

# Baudrate and SampleRate are only for some Logitech mice

#      BaudRate    9600
#      SampleRate   150

# Emulate3Buttons is an option for 2-button Microsoft mice

#      Emulate3Buttons

# ChordMiddle is an option for some 3-button Logitech mice

#      ChordMiddle

EndSection

```

Po etykiecie `Protocol` następuje nazwa typu przyłączonej myszy (lub typu, który mysza potrafi emulować). Lista obsługiwanych typów myszek podana jest na stronach `man` dotyczących pliku `XF86Config`, jeśli więc posiadasz mysz inną niż Microsoft czy Logitech, sprawdź w dokumentacji, jakiej nazwy protokołu powinieneś użyć. Innym sposobem zidentyfikowania typu myszy jest obserwacja komunikatów generowanych podczas uruchamiania systemu – często pojawia się wśród nich informacja o wykrytym modelu myszy.

Myszki firmy Microsoft używają protokołu Microsoft. Większość urządzeń marki Logitech jest kompatybilna z protokołem Microsoft, a nowsze mogą również używać protokołu MouseMan. Większość innych modeli myszy używa również protokołu Microsoft.

Pozycja `Device` pozwala określić port, do którego podłączona jest myszka. W większości przypadków wartość `/dev/mouse` jest odpowiednim ustawieniem, ponieważ Linux podczas instalacji tworzy odpowiednie dowiązania. Jeśli używasz myszy podłączonej do portu PS/2, powinieneś podać tu odpowiednią nazwę pliku urządzenia. Wszystkie obsługiwane porty wypisane są w dokumentacji i na stronach `man`; oto niektóre z nich:

- υ **/dev/mouse** domyślny sterownik systemu Linux;
- υ **/dev/inportbm** sterownik wyłącznie dla myszek magistralowych Microsoft;
- υ **/dev/logibm** sterownik wyłącznie dla myszek magistralowych Logitech;
- υ **/dev/psaux** sterownik dla myszek PS/2.

Instalacja myszy magistralowej (ang. *bus mouse*) wymaga podania numeru używanego przerwania IRQ zarówno w jądrze systemu, jak i dla XFree86. Upewnij się, że podane wartości są jednakowe.

Podobnie jak w przypadku klawiatury, tu też można użyć protokołu `xqueue` – ponieważ większość instalacji XFree86 nie używa `xqueue`, powinieneś pozostawić odpowiedni wiersz zaznaczony jako komentarz. Parametry `BaudRate` i `SampleRate`, jak mówią komentarze, dotyczą tylko niektórych starszych myszek firmy Logitech. W większości przypadków nie trzeba ich używać. Jeśli musisz, spróbuj najpierw prędkości 9600 bodów, a następnie – w razie niepowodzenia – 1200 bodów. Niektóre wcześniejsze wersje XFree86 wolały, aby parametry te były podane, ale spróbuj najpierw uruchomić X nie podając tych parametrów.

Opcja `Emulate3Buttons` jest przydatna, gdy posiadasz mysz obsługującą tylko dwa klawisze. Kiedy jest ona aktywna, pozwala symulować naciśnięcie trzeciego klawiszów przez naciśnięcie obu klawiszy równocześnie. Wiele aplikacji linuxowych (i UNIX-owych) wykorzystuje trzeci klawisz myszy, jeśli więc posiadasz mysz Microsoft lub kompatybilną, warto załączyć tę opcję.

Opcja `ChordMiddle` używana jest tylko w przypadku niektórych myszy firmy Logitech. Jeśli korzystasz ze sterownika Logitech, spróbuj najpierw uruchomić X bez załączonej opcji `ChordMiddle`. Dopiero jeśli mysz nie działa poprawnie, załącz ją. Większość myszy firmy Logitech nie wymaga załączenia tej opcji.

Model monitora

Właściwe określenie modelu monitora jest bardzo istotne – podanie błędnej wartości może spowodować nawet uszkodzenie sprzętu! Cierpliwość i zdrowy rozsądek wystarczą, by zapobiec zniszczeniom, ale najlepszym źródłem informacji jest instrukcja obsługi monitora.

Jeśli nie jesteś pewny co do któregokolwiek z ustawień, użyj najbardziej podstawowych wartości. Jeśli nie wiesz, czy monitor obsługuje wyższe rozdzielczości, pozostań przy ustawieniach VGA lub SVGA przynajmniej do czasu, aż zdobędziesz dokładniejsze informacje co do jego możliwości.

Dla wygody użytkownika sekcja pliku `XF86Config` dotycząca monitora podzielona jest na kilka podsekcji. Przejrzymy się im z osobna. W pierwszej podsekcji zawarte są informacje dotyczące typu i modelu monitora:

```
Section "Monitor"  
    Identifier "Generic Monitor"  
    VendorName "Unknown"  
    ModelName "Unknown"
```

Wpisy te są tylko etykietami i nie mają żadnego znaczenia dla systemu X; są wyświetlane podczas uruchamiania serwera oraz przez programy diagnostyczne. Możesz je zmienić, by uczynić pracę z X nieco przyjemniejszą.

Następna podsekcja określa częstotliwości odchylania poziomego, z jakimi może pracować monitor. Są to ważne informacje i powinieneś poszukać właściwych danych w instrukcji obsługi monitora. Niektóre ustawienia dla poszczególnych modeli opisane są w dokumentach Monitors oraz VideoModes.doc rozprowadzanych wraz z systemem XFree86. Jeśli nie możesz znaleźć danych o możliwościach Twojego sprzętu, użyj najniższych dostępnych wartości.

```
# Bandwidth is in MHz unless units are specified
    BandWidth      25.2
# HorizSync is in kHz unless units are specified.
# HorizSync may be a comma separated list of discrete values, or a
# comma separated list of ranges of values.
# NOTE: THE VALUES HERE ARE EXAMPLES ONLY. REFER TO YOUR MONITOR'S
# USER MANUAL FOR THE CORRECT NUMBERS.

    HorizSync     31.5 # typical for a single frequency fixed-sync monitor

#      HorizSync 30-64          # multisync
#      HorizSync 31.5, 35.2    # multiple fixed sync frequencies
#      HorizSync 15-25, 30-50 # multiple ranges of sync frequencies
```

Jak widać, ustawienia te opatrzone są odpowiednim komentarzem, ułatwiającym konfigurację. Jeśli monitor obsługuje różne częstotliwości odchylania poziomego, możesz usunąć symbol komentarza z wiersza zawierającego odpowiedni zakres częstotliwości, na przykład 30 – 64 kHz, i zaznaczyć jako komentarz wiersz zawierający wartość 31.5 kHz.

Częstotliwości odchylania pionowego to następna bardzo newralgiczna sekcja. Tu również powinieneś zachować szczególny umiar i trzymać się wartości podanych przez producenta lub najniższych.

```
# VertRefresh is in Hz unless units are specified.
# VertRefresh may be a comma separated list of discrete values, or a
# comma separated list of ranges of values.
# NOTE: THE VALUES HERE ARE EXAMPLES ONLY. REFER TO YOUR MONITOR'S
# USER MANUAL FOR THE CORRECT NUMBERS.

    VertRefresh   60 # typical for a single frequency fixed-sync monitor

#      VertRefresh      50-100        # multisync
#      VertRefresh      60, 65       # multiple fixed sync frequencies
#      VertRefresh      40-50, 80-100 # multiple ranges of sync frequencies
```

Tu również komentarze ułatwiają wybranie odpowiednich wartości. Mogą stanowić wskazówkę, ale dokładnych informacji szukać należy w dokumentacji monitora.

Ustawienie właściwego trybu graficznego także jest bardzo ważne, ponieważ błędne ustawienie może spowodować śnieżenie, uzyskanie tylko czarnego ekranu albo – w niektórych

przypadkach – zawieszenie systemu. Opisany wcześniej program SuperProbe wyświetla informacje o dostępnych trybach graficznych, ale również w tym przypadku powinieneś trzymać się dokumentacji monitora. Dane o niektórych markach i modelach monitorów możesz znaleźć w pliku Monitors rozprowadzanym wraz z XFree86.

```
# Modes can be specified in two formats. A compact one-line format, or
# a multi-line format.

# A generic VGA 640x480 mode (hsync = 31.5kHz, refresh = 60Hz)
# These two are equivalent

#      ModeLine "640x480" 25.175 640 664 760 800 480 491 493 525

Mode "640x480"
    DotClock     25.175
    HTimings     640 664 760 800
    VTimings     480 491 493 525
EndMode

# These two are equivalent

#      ModeLine "1024x768i" 45 1024 1048 1208 1264 768 776 784 817
Interlace

Mode "1024x768i"
    DotClock     45
    HTimings     1024 1048 1208 1264
    VTimings     768 776 784 817
    Flags        "Interlace"
EndMode

EndSection
```

Powyższy przykład umożliwia wykorzystanie dwóch trybów: standardowego VGA (640x480) i wysokiej rozdzielczości (1024x768). Jak widać, potrzebna jest znajomość parametrów DotClock i czasów przelotu rastra specyficznych dla monitora i karty graficznej. Odpowiednie informacje znajdują się w dokumentacji sprzętu. Zauważ, że wszystkie dane dla jednego trybu pracy mogą zostać podane w jednym wierszu, ale nie są wtedy zbyt czytelne.

Karty graficzne

Następna sekcja określa parametry zainstalowanej karty graficznej. Możesz tu określić definicje kilku kart pracujących z różnymi rozdzielczościami bądź też zdefiniować jedną, której będziesz używał. Podany niżej przykładowy kod definiuje karty VGA i SVGA.

```
Section "Device"
    Identifier "Generic VGA"
    VendorName "Unknown"
    BoardName "Unknown"
    Chipset   "generic"
#    VideoRam 256
#    Clocks    25.2 28.3
```

```

EndSection
Section "Device"
    # SVGA server auto-detected chipset
    Identifier "Generic SVGA"
    VendorName "Unknown"
    BoardName "Unknown"
EndSection

```

Pola `Identifier`, `VendorName`, `BoardName` oraz opcjonalnie `Chipset` nie mają znaczenia dla systemu X i służą tylko identyfikacji urządzenia przez użytkownika. Pola `VideoRam` (ilość pamięci RAM na karcie graficznej) i `Clocks` określają ustawienia konkretnego egzemplarza karty. Powinny być ustawiane ostrożnie, bo błędne wartości w pewnych sytuacjach mogą spowodować uszkodzenie karty graficznej.

Jeśli posiadasz kartę graficzną o znanych Ci parametrach, możesz stworzyć dla niej specjalną konfigurację, na przykład dla karty Trident TVGA:

```

Section "Device"
    Identifier "Trident TVGA 9000"
    VendorName "Trident"
    BoardName "TVGA 9000"
    Chipset "tvga9000"
    VideoRam 512
    Clocks 25 28 45 36 57 65 50 40 25 28 0 45 72 77 80 75
EndSection

```

Dane dotyczące ustawień `VideoRam` i `Clocks` zostały zaczerpnięte z dokumentacji towarzyszącej XFree86, ale powinny również znajdować się w instrukcji obsługi karty.

Niektóre karty graficzne wymagają podania jeszcze innych danych. Na przykład karta Actix GE32+ z 2MB pamięci może być zdefiniowana w następujący sposób:

```

Section "Device"
    Identifier "Actix GE32+ 2MB"
    VendorName "Actix"
    BoardName "GE32+"
    Ramdac "ATT20C490"
    Dacspeed 110
    Option "dac_8_bit"
    Clocks 25.0 28.0 40.0 0.0 50.0 77.0 36.0 45.0
    Clocks 130.0 120.0 80.0 31.0 110.0 65.0 75.0 94.0
EndSection

```

Dodane zostały wpisy `Ramdac`, `Dacspeed` oraz `Option`. Parametry, które można definiować, zależą od wersji XFree86, więc jeśli chcesz wykorzystać w pełni możliwości swojej karty graficznej, powinieneś zatrzymać się na odpowiednie strony man.

Serwer XFree86

Problem wyboru serwera odpowiedniego dla danej karty graficznej omówiliśmy już wcześniej. W sekcji pliku `Xconfig` czy `XF86Config` dotyczącej serwera zawarte są informacje o tym, którego serwera i z jakimi ustawieniami należy użyć.

```
Section "Screen"
    Driver      "svga"
    Device      "Generic SVGA"
    Monitor     "Generic Monitor"
    DefaultColorDepth 8
    Subsection  "Display"
        Depth       8
        Modes      "640x480"
        ViewPort   0 0
        Virtual    800 600
    EndSubsection
EndSection
```

Powyższa podsekcja określa sterownik SVGA, obsługujący tryby 640x480 i 800x600. Jeśli posiadasz kartę i monitor o większych możliwościach, możesz użyć odpowiedniego serwera (na przykład przeznaczonego dla karty Actix GE32+ z 2 MB pamięci RAM), na przykład tak:

```
Section "Screen"
    Driver      "accel"
    Device      "Actix GE32+ 2MB"
    Monitor     "Generic Monitor"
    DefaultColorDepth 8
    Subsection  "Display"
        Depth       8
        Modes      "640x480"
        ViewPort   0 0
        Virtual    1280 1024
    EndSubsection
    SubSection "Display"
        Depth      16
        Weight     565
        Modes      "640x480"
        ViewPort   0 0
        Virtual    1024 768
    EndSubsection
EndSection
```

Zgodnie z powyższą definicją, używany będzie serwer obsługujący akcelerację sprzętową dla kart Actix, umożliwiający pracę w rozdzielcościach do 1280x1024. Sprawdź, czy istnieje serwer przeznaczony do współpracy z Twoją kartą graficzną. Jeśli nie jesteś pewien, użyj serwera ogólnego.

Opcje w tej sekcji nie dotyczą wszystkich kart graficznych, ale jeśli znasz odpowiednie wartości, możesz je podać. Najważniejsze i najczęściej używane z nich zebrano poniżej.

- υ **Depth:** ilość bitów koloru przypadających na jeden punkt (głębią koloru). Wartość ta zwykle wynosi 8, ale sterownik VGA16 obsługuje tylko 4, a karty monochromatyczne – 1 bit koloru. Nowsze karty obsługują 16, 24, 32 a nawet 64 bity koloru – czasem informacja ta podana jest jako część nazwy modelu karty (na przykład karta Diamond Stealth 24 obsługuje 24 – bitową głębię koloru, ale nie jest to reguła – sprawdź w dokumentacji, zanim przyjmiesz, że liczba w nazwie modelu oznacza właśnie głębię koloru).
- υ **Modes:** lista trybów graficznych określonych w polu `ModeLine` sekcji `Monitor`. Definiuje ona wszystkie tryby obsługiwane przez kartę, których zamierzasz używać. Pierwszy tryb z tej listy zostanie użyty przy uruchamianiu systemu X. Podczas działania serwera X można przełączać się pomiędzy tymi trybami.
- υ **Virtual:** rozmiar pulpitu wirtualnego. Jeśli posiadasz odpowiednią ilość pamięci graficznej, pulpit wirtualny systemu X może być większy niż rozmiar ekranu – wówczas można poruszać się po całym jego obszarze za pomocą myszy. Przykładowo, możesz mieć pulpit o wielkości 1024x768, ale wyświetlać go na ekranie o wielkości 800x600. Maksymalny rozmiar pulpitu zależy od ilości pamięci RAM na karcie graficznej, np. karta z 1 MB może obsłużyć wielkość 1024x768 przy 256 kolorach, natomiast 2 MB pamięci pozwalają na pracę z takim samym rozmiarem pulpitu przy 65 K kolorów lub z rozmiarem 1280x1024 przy 256 kolorach. Jeśli chcesz używać pulpitu wirtualnego, powinieneś zainstalować menedżer okienek `fvwm` (instalowany domyślnie).
- υ **ViewPort:** parametr używany z pulpitem wirtualnym do określenia położenia punktu o współrzędnych (0,0) przy uruchomieniu serwera.

Testowanie konfiguracji XFree86

Po skompletowaniu zawartości plików konfiguracyjnych czas uruchomić `XFree86`, na przykład wydając polecenie `startx`. `startx` uruchomi skrypt, który powinien załadować wszystkie potrzebne sterowniki, programy rezydentne, wyczyścić ekran i uruchomić odpowiedni menedżer okienek. Jeśli `XFree86` nie uruchamia się poprawnie, przyjrzyj się uważnie wyświetlonym komunikatom. Powinny one dać Ci wskazówkę co do przyczyny problemów. Zwykle są to problemy z obsługiwanyimi trybami graficznymi (dla przyzwyczajonych do używania UNIX-a: `startx` jest nakładką na program `xinit`, używaną do uruchamiania systemu X).

Jeśli masz problemy z uruchomieniem systemu X, najsukutczniejszym sposobem na znalezienie ich przyczyny jest ustawienie wszystkich opcji na najmniejsze wartości (z nimi system powinien się uruchomić), a następnie kolejne ich modyfikowanie. Jeśli nie działa nawet standardowy sterownik VGA, oznacza to, że problem tkwi gdzieś indziej (możliwe na przykład, że ścieżki dostępu do plików z definicjami czcionek są nieprawidłowe).

Plik .xinitrc

Plik `.xinitrc` jest plikiem konfiguracyjnym systemu X, spełniającym taką samą funkcję, jak plik `.profile` czy `.login` podczas uruchamiania interpretera poleceń. Zawiera zwykle wszystkie lokalne modyfikacje konfiguracji, takie jak polecenia automatycznie uruchamiające określone aplikacje itp. Jeśli używasz skryptu `startx` lub `runx`, plik `.xinitrc` zostanie przemianowany na `xinitrc`.

Plik zawierający dane o domyślnej konfiguracji systemowej nazywa się `/usr/lib/X11/xinit/xinitrc` lub `/etc/X11/xinit/xinitrc` (nazwa częściej spotykana w systemach linuxowych, ponieważ w niektórych systemach katalog `/usr` jest katalogiem tylko do odczytu, na przykład zapisanym na płycie CD-ROM).

Dane zapisane w domyślnym systemowym pliku konfiguracyjnym zostaną pominięte, jeśli w katalogu domowym znajduje się plik o nazwie `.xinitrc`. Dzięki temu możesz, po skopiowaniu systemowego pliku `xinitrc` do katalogu domowego i przemianowaniu go na `.xinitrc`, wprowadzać do niego własne modyfikacje. Strony `man` dotyczące skryptów `startx` i `runx` wyjaśniają szczegółowo.

Poniżej podano zawartość przykładowego pliku `.xinitrc`. Jest to domyślny plik pochodzący wprost z nowej instalacji `XFree86`. Pierwsza jego część definiuje ścieżki dostępu do poszczególnych typów plików:

```
userresources=$HOME/.Xresources
usermodmap=$HOME/.Xmodmap
sysresources=/usr/X11R6/lib/X11/xinit/.Xresources
sysmodmap=/usr/X11R6/lib/X11/xinit/.Xmodmap
```

Ścieżki te są zwykle ustalane przez system `XFree86`, warto jednak sprawdzić, czy są prawne. Wszystkie zdefiniowane powyżej zmienne są niezbędne do prawidłowego działania systemu X.

Następna część zawiera polecenia, które sprawdzają istnienie pewnych zasobów systemowych i w zależności od tego podejmują odpowiednie działania. Większość z nich nie musi być modyfikowana, chyba że masz bardzo nietypowe wymagania.

```
# merge in defaults and keymaps

if [ -f $sysresources ]; then
    xrdb -merge $sysresources
fi

if [ -f $sysmodmap ]; then
    xmodmap $sysmodmap
fi

if [ -f $userresources ]; then
    xrdb -merge $userresources
fi

if [ -f $usermodmap ]; then
```

```
        xmodmap $usermodmap
fi
```

Ostatnia sekcja pliku `.xinitrc` odpowiada za uruchomienie programów przy starcie systemu X. W przypadku, gdy menedżer `fvwm` jest zainstalowany, jest on uruchamiany, w przeciwnym – uruchamiany jest menedżer `twm`. Jeśli chcesz uruchamiać inne programy lub inny menedżer okienek, powinieneś dodać tu odpowiednie polecenia.

```
# start some nice programs

xsetroot -solid SteelBlue
if [ -f $HOME/.Xclients ]; then
    exec $HOME/.Xclients
elif [ -f /etc/X11/xinit/Xclients ]; then
    exec /etc/X11/xinit/Xclients
else
    xclock -geometry 50x50-1+1 &
    xterm -geometry 80x50+494+51 &
    xterm -geometry 80x20+494-0 &
    if [ -f /usr/X11R6/bin/arena -a -f /usr/doc/HTML/index.html ];
then
    arena /usr/doc/HTML/index.html &
fi
if [ -f /usr/X11R6/bin/fvwm ]; then
    exec fvwm
else
    exec twm
fi
fi
```

Polecenie `xsetroot` pozwala na ustawienie koloru pulpitu. Polecenie `xterm` powoduje uruchomienie programu emulacji terminalu w oknie systemu X, dzięki czemu możliwe jest późniejsze uruchamianie innych programów. W powyższym przykładzie uruchamiane są dwa takie terminale. Uruchamiany jest również program `xclock`, który wyświetla na ekranie zegar, a także (jeśli jest zainstalowany i istnieje odpowiedni plik dokumentacji) program `arena` – przeglądarka plików `html` – z dokumentacją dotyczącą Linuxa.

Każde polecenie w tym pliku powinno kończyć się znakiem `&`, który informuje powłokę, że nie należy czekać na zakończenie procesu, tylko uruchomić go w tle. Dzięki temu kilka programów może uruchamiać się jednocześnie. Co więcej – gdyby należało czekać na zakończenie np. programu `xclock`, system X nigdy by się nie uruchomił. Ważne jest również, by symbol `&` nie występował po poleceniu uruchamiającym menedżer okienek, ponieważ w przeciwnym przypadku uruchomi się on, po czym natychmiast zakończy działanie.

Podsumowanie

Po skonfigurowaniu systemu X zgodnie ze wskazówkami zawartymi w tym rozdziale, powinien on działać bezproblemowo. Omawianie szczegółów pracy z tym systemem

wykracza jednak poza zakres tej książki. Dobrym źródłem informacji są oczywiście strony `man`, inna dokumentacja oraz poświęcone temu tematowi książki.

Mimo tego, po skonfigurowaniu systemu `XFree86` możesz zacząć pracować pod jego kontrolą; wkrótce przekonasz się, że trudno powrócić do interfejsu tekstowego.

`Wabi`, program pozwalający na uruchamianie aplikacji Windows pod kontrolą systemu X, opisany jest w rozdziale 23. „`Wabi`”.

`GhostScript`, który pozwala na używanie języka PostScript i oglądanie plików postscriptowych na ekranie oraz ich drukowanie, omówiony jest w rozdziale 24. „`GhostScript` i `Ghostview`”.

O programowaniu w systemie Linux możesz dowiedzieć się więcej z części piątej, rozpoczynając od rozdziału 25. „`gawk`”.

Podstawy administrowania systemem omówione są w części szóstej, rozdział 32. „Podstawy administracji systemem”.

Rozdział 23.

Wabi

Tim Parker

W tym rozdziale:

- υ Co potrafi Wabi?
- υ Instalacja Wabi
- υ Uruchamianie aplikacji systemu Windows 3.X

Wabi (ang. *Windows Application Binary Interface*) to aplikacja napisana dla systemów Linux oraz UNIX i pozwalająca na uruchamianie programów przeznaczonych dla Windows 3.X pod kontrolą systemu X. Jest to swego rodzaju translator, tłumaczący zdarzenia systemu X na komunikaty Windows i odwrotnie. Dzięki Wabi możesz mieć dostęp do większości aplikacji systemu Windows, nie tracąc zalet środowiska UNIX-owego (jak choćby jego większa wydajność czy odporność na załamania systemu).

Nie wszystkie aplikacje Windows będą działać prawidłowo pod kontrolą programu Wabi. Aby program działał poprawnie, musi zachowywać się „jak należy”, tzn. spełniać wymogi stawiane programom Windows przez Microsoft. Większość standardowych programów, takich jak edytory tekstu czy arkusze kalkulacyjne, pracuje poprawnie. Niektóre jednak (szczególnie programy graficzne i gry) odbiegają od narzuconych standardów w celu uzyskania wydajności, na którą system Windows normalnie by nie pozwolił. Takie aplikacje nie będą działać poprawnie (o ile w ogóle). Poza tym za pomocą Wabi w wersji 2.1 nie można uruchamiać aplikacji przeznaczonych do pracy w systemie Windows 95.



Lista programów, które zostały przetestowane i działają poprawnie pod kontrolą Wabi, dostępna jest w większości węzłów FTP i na stronach WWW udostępniających oprogramowanie linuxowe, miedzy innymi pod adresem <http://wabiapps.psgroup.com>

Nie powinieneś jednak myśleć, że żaden z Twoich ulubionych programów nie będzie działać. Większość aplikacji przeznaczonych dla systemu Windows 3.X działa poprawnie, za wyjątkiem gier. Chcesz używać CorelDraw? Nie ma problemu. Microsoft Office? Jak najbar-

dziej.

Wszystkie programy przeznaczone do pracy biurowej najprawdopodobniej również będą działać (bo raczej nie mają one skłonności do niestandardowych zachowań).

Co potrafi Wabi?

Jak wspomniano wcześniej, *Wabi* jest jakby pośrednikiem pomiędzy aplikacją Windows a menedżerem okienek systemu X. Kiedy program wywoła jakieś polecenie systemu Windows (np. otwarcie nowego okna, zmiana czcionki itp.), jest ono przechwytywane przez *Wabi* i tłumaczone na odpowiednie polecenie systemu X. Kiedy X chce przesłać komunikat do okna programu Windows, jest on również przechwytywany przez *Wabi* i tłumaczony na odpowiednik systemu Windows. Dopóki używane są standardowe mechanizmy, wszystko działa bez zarzutu.

Tak naprawdę program *Wabi* nie robi wiele „sam z siebie”. Nie ma jakiejś aplikacji o nazwie *Wabi*, z którą mógłbyś pracować (nie wliczając kilku narzędzi konfiguracyjnych). *Wabi* udostępnia jednak aplikacjom Windows techniki takie jak:

- υ obsługa schowka (clipboard);
- υ obsługa trybu chronionego procesora;
- υ dostęp do dysków DOS-owych;
- υ OLE (ang. *Object Linking and Embedding*);
- υ DDE (ang. *Dynamic Data Exchange*);
- υ obsługa sieci;
- υ Windows sockets.

Co więcej, *Wabi* umożliwia na przykład kopiowanie danych pomiędzy aplikacjami X oraz Windows czy korzystanie z aplikacji sieciowych. Możliwe jest również, aby kilku użytkowników korzystało jednocześnie z tej samej aplikacji.

Prawdopodobnie ważniejsza dla użytkowników jest lista rzeczy, których *Wabi* nie potrafi zrobić. Nie jest ona dłuża, ale zawiera dość istotne ograniczenia. *Wabi* nie obsługuje:

- υ interfejsu MIDI,
- υ AVI (Audio-Visual Interface),
- υ protokołów IPX/SPX,

- υ sterowników graficznych VGA,
- υ formatowania dysków DOS-owych,
- υ wirtualnych sterowników urządzeń (VDx).

Większość nie obsługiwanych przez *Wabi* mechanizmów i rozwiązań może być dostępna w Linuxie dzięki odpowiednim narzędziom linuxowym (np. protokoły IPX/SPX czy obsługa MIDI). Program *Wabi* jest wciąż dopracowywany, możliwe więc, że nowe wersje zostaną wzbogacone o brakujące elementy.

Wabi nie zawiera w sobie Microsoft Windows ze względu na prawa autorskie, aby więc uruchamiać aplikacje systemu Windows, musisz posiadać jego kopię (większość użytkowników ją posiada, ponieważ jest ona często dołączana do nowych komputerów).

Instalacja Wabi

Wabi wymaga minimum 16 MB pamięci RAM, choć zalecane jest 32 MB. Konieczne jest również co najmniej 20 MB partycji lub pliku wymiany, a im więcej, tym lepiej. Na dysku twardym *Wabi* zajmuje ok. 25 MB, zależnie od wersji programu i obsługiwanych wersji Windows.

Do oprogramowania dołączony jest skrypt instalacyjny, który automatyzuje proces instalacji, biorąc na siebie wszystkie czynności, które musiałbyś normalnie wykonać. Instalacja programu *Wabi* za pomocą takiego skryptu sprowadza się do podania kilku informacji w odpowiedzi na monity. Jeśli otrzymałeś wersję instalacyjną w postaci archiwum, należy najpierw rozpakować ją do jakiegoś tymczasowego katalogu, a następnie uruchomić skrypt instalacyjny. Po zainstalowaniu będzie można wybrać wersję Windows, której chcesz używać. Obsługiwane są wersje 3.1 i 3.11.

Domyślnie *Wabi* instaluje się do katalogu `/opt/wabi`, a pliki wykonywalne znajdują się w katalogu `/opt/wabi/bin`. Można zmienić te ustawienia, ale najlepiej pozostawić je tak, jak są. Za każdym razem, gdy uruchamiany jest program *Wabi*, informacje konfiguracyjne są odczytywane z podkatalogu `wabi` w katalogu domowym. W podkatalogu `windows` katalogu `wabi` musi znajdować się również kopia systemu Microsoft Windows. Ponieważ każdy użytkownik posiada własny katalog domowy, pociąga to za sobą konieczność przechowywania jednej kopii systemu Windows dla każdego użytkownika *Wabi*. Choć niedogodność tę da się obejść, lepiej pozostawić tak, jak jest, ponieważ takie było założenie autorów tego programu. Można zauważyc dość daleko idące analogie pomiędzy katalogami `wabi` i `wabi/windows` oraz `c:\` i `c:\windows`.

Rozwiązywanie problemów

Jeśli używasz serwera Metro X 3.1.2 i spróbowajesz uruchomić *Wabi*, sesja Metro X zawiesi się. Problem leży po stronie serwera Metro: wersja 3.1.5 działa już poprawnie. Uaktualnienia wersji udostępniane są zwykle przez Metro Link, dystrybutora systemu Metro X.

Obejść ten problem (niestety, kosztem wydajności) można poprzez uruchomienie `Wabi` z opcją `-fs`.

Aktualna wersja `Wabi` obsługuje 256 kolorów, ale dostępne są również poprawki umożliwiające pracę w trybie True Color (16 milionów kolorów). Jeśli chcesz osiągnąć najlepszą wydajność, powinieneś skonfigurować swoją sesję X tak, by pracowała przy 256 kolorach.

Uruchamianie aplikacji systemu Windows 3.X

Jak wspomniano wcześniej, `Wabi` obsługuje tylko aplikacje przeznaczone dla systemów Windows 3.X. Są one uruchamiane w systemie linuxowym tak, jakby działały pod Windows. Po uruchomieniu `Wabi` znajdziesz się w środowisku do złudzenia przypominającym Windows 3.X. Od tej pory wszystko działa tak, jak się spodziewasz – żadnych nowości. `Wabi` troszczy się o prawidłowe mapowanie wszystkich urządzeń (na przykład dyskietek i odpowiednich katalogów).

Program `Wabi` może zostać uruchomiony z okna terminalu za pomocą polecenia

```
wabi &
```

Ampersand (`&`) na końcu polecenia, uruchamiający sesję `Wabi` w tle, nie jest konieczny, ale bez niego nie można używać otwartego okna terminalu aż do zakończenia pracy z `Wabi`. Po wydaniu tego polecenia pojawia się ekran powitalny z logiem programu `Wabi` (rys. 23.1). W dolnej części okna pojawia się pasek pokazujący postępy procesu ładowania czcionek, a po kilku sekundach uruchamiane jest główne okno programu.

Rysunek 23.1.
Ekran powitalny programu Wabi

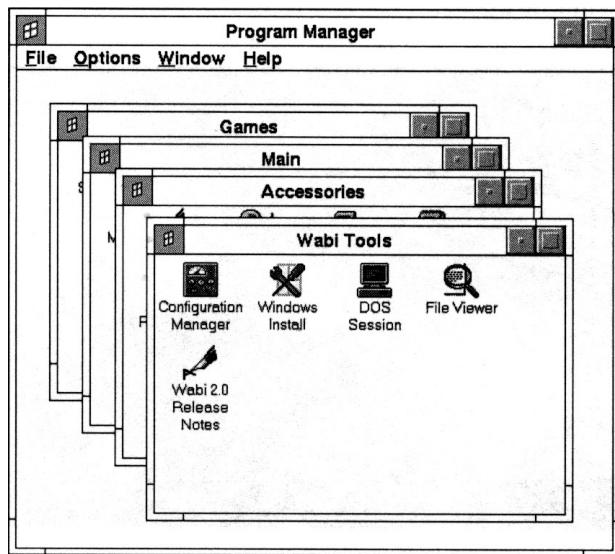


Jeśli pracujesz w systemie X11 (zarówno Metro X, jak i XFree86 są systemami X11), uruchamiany jest również oddzielny serwer czcionek – `wabif.s`.



Firma Sun Microsystems planowała w roku 1997 wydać wersję `Wabi` o numerze 3.0, obsługującą aplikacje systemu Windows 95. Niestety, zdecydowano, że projekt `Wabi` nie będzie kontynuowany. Jak dotąd żaden inny dostawca oprogramowania nie zainteresował się tym projektem. Obecnie nie istnieje wersja `Wabi` pozwalająca na uruchamianie programów dla Windows 95.

Rysunek 23.2.
Interfejs programu `Wabi` do złudzenia przypomina standardowy interfejs Windows



Instalowanie dodatkowego oprogramowania jest równie proste, jak w systemie Windows. Jeśli posiadasz wersję instalacyjną na dyskietce, wybierz z menu `File` polecenie `Run`, a następnie wpisz nazwę programu instalacyjnego, na przykład `a:\setup`. Jedyna różnica widoczna jest podczas korzystania z sieci. `Wabi` działa w oparciu o nieco inny model sieci niż Microsoft Windows.

Dodatkowe informacje można uzyskać z dwóch źródeł: tak jak w Windows, wybierając z menu polecenie `Help`, oraz ze stron `man`, które opisują opcje dostępne z wiersza poleceń, przykładowe tryby pracy i zmienne środowiskowe używane przez `Wabi`.

Podsumowanie

`Wabi` to program pozwalający na uruchamianie starszych aplikacji systemu Windows 3.X pod kontrolą Linuxa. Dzięki niemu możesz używać posiadanych programów, takich

jak edytory tekstów, arkusze kalkulacyjne itp. bez konieczności zakupu ich odpowiedników dla systemu Linux.

Jeśli chcesz dowiedzieć się więcej o programowaniu w języku `awk`, zajrzyj do rozdziału 25. „`gawk`”.

Programowanie w języku Perl, łatwym do opanowania i poręcznym do pisania krótkich programów, omawia rozdział 28. „Perl”.

O administrowaniu systemem linuxowym i o tym, co należy robić, by pracował on bezproblemowo, dowiesz się z rozdziału 32. „Podstawy administrowania systemem”.

O tym, jak użyć systemu Linux jako bramki internetowej, możesz przeczytać w rozdziale 47. „Konfiguracja węzła internetowego”.

Rozdział 24.

Ghostscript i Ghostview

Tim Parker

W tym rozdziale:

- υ Skąd można wziąć Ghostscript?
- υ Pakiet Ghostscript
- υ Ghostview

PostScript to popularny język opisu strony (ang. *Page Description Language*, PDL), używany przez wiele aplikacji. Jedną z jego podstawowych zalet jest fakt, że plik w formacie postscriptowym może zostać wydrukowany na każdej drukarce obsługującej go, bez względu na system operacyjny czy model drukarki. Pozwala to na stworzenie jednej wersji pliku zamiast tworzenia oddzielnej wersji dla drukarek HP LaserJet, drugiej dla urządzeń Epson itd.

Z punktu widzenia użytkownika Linuxa, głównym problemem jest fakt, że nazwa PostScript została zastrzeżona przez firmę Adobe Systems, Inc. Z tego powodu wersja tego języka dla Linuxa nazywa się *Ghostscript*. Jak można się domyślić, język ten jest prawie identyczny z językiem PostScript, ale jego dystrybucja nie jest ograniczona prawami autorskimi, związanymi z tym opłatami itd. W zasadzie Ghostscript również jest objęty prawem autorskim jego twórców, ale postanowili oni zezwolić na rozprowadzanie go na warunkach licencji GNU, dzięki czemu jest dostępny dla wszystkich użytkowników.



Właścicielem praw autorskich do języka Ghostscript jest firma Aladdin Enterprises. Jeśli interesuje Cię uzyskanie licencji na programy użytkowe, powinieneś się z nią skontaktować; podajemy więc adres:

Aladdin Enterprises, P.O. Box 60264, Palo Alto, CA 94306;
e-mail: ghost@aladdin.com.

Większość aplikacji potrafi generować pliki w formacie Ghostscript. Program o nazwie Ghostview pozwala je oglądać, dzięki czemu możesz ocenić efekty swojej pracy przed ich wydrukowaniem. Ghostscript to w zasadzie cały pakiet programów. W tym rozdziale zajmiemy się tylko interpreterem języka PostScript oraz funkcjami języka C, które mogą zostać dołączone do kodu tworzonego programu wzbogacając go o możliwość generowania plików postscriptowych.

Skąd można wziąć Ghostscript?

Ghostscript jest dostarczany z większością dystrybucji Linuxa. Jest częścią zestawu dysków AP (ang. *applications*). Jeśli zdecydowałeś się zainstalować ten zestaw (większość użytkowników instaluje go), Ghostscript jest gotowy do użycia w Twoim systemie. Jeśli nie instalowałeś oprogramowania z zestawu AP, możesz zrobić to w każdej chwili, wybierając z oprogramowania zawartego w tym zestawie tylko pakiet Ghostscript.

Jeśli nie wiesz, czy Ghostscript został zainstalowany, musisz poszukać odpowiednich plików wykonywalnych i bibliotek ręcznie, ponieważ różne wersje Linuxa instalują je do różnych katalogów. Najłatwiej poszukać pliku wykonywalnego o nazwie `gs` za pomocą polecenia `find`:

```
find / -name gs -print
```

Jeśli został on znaleziony (najprawdopodobniej w katalogu `/usr/bin`), oznacza to, że pakiet Ghostscript najprawdopodobniej został zainstalowany. Liczba plików instalowanych z pakietem Ghostscript różni się w zależności od wersji oprogramowania, ale działająca instalacja powinna na pewno zawierać pliki:

```
bdftops.ps  
decrypt.ps  
font2c.ps  
gs_dbt_e.ps  
gs_dpsl.ps  
gs_fonts.ps  
gs_init.ps  
gs_lev2.ps  
gs_statd.ps  
gs_sym_e.ps  
gs_type0.ps  
gslp.ps  
impath.ps  
landscap.ps  
level1.ps  
prfont.ps  
ps2ascii.ps  
ps2epsi.ps  
ps2image.ps  
pstoppm.ps  
quit.ps
```

```
showpage.ps  
type1ops.ps  
wrfont.ps  
uglyr.ps  
Fontmap
```

Zwykle pliki te przechowywane są w katalogu `/usr/lib/ghostscript` lub `/usr/share/ghostscript`. Czasem również do nazwy katalogu dodawany jest jeszcze numer wersji, co dodatkowo komplikuje problem. Również w tym przypadku najłatwiej odnaleźć potrzebne pliki polecienniem `find`, np.

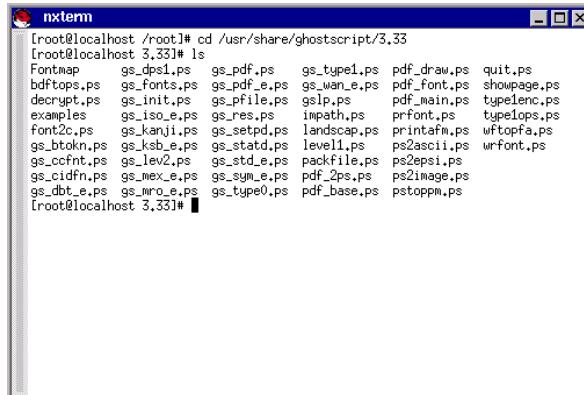
```
find / -name gs_statd.ps -print
```

W powyższym przykładzie polecenie `find` będzie próbowało odnaleźć plik `gs_statd.ps`, który powinien być częścią każdej instalacji Ghostscript. Znajduje się on przeważnie w katalogu `/usr/lib/ghostscript`, ale w niektórych systemach może być przechowywany w katalogu `/usr/share`. Oto wynik działania podanego wcześniej polecenia w systemie Caldera OpenLinux:

```
[root@linux /root]# find / -name gs_statd.ps -print  
/usr/share/ghostscript/3.33/gs_statd.ps
```

W tym przypadku pliki pakietu Ghostscript znajdują się w katalogu `/usr/share/ghostscript/3.33`. Aby sprawdzić, czy wszystkie potrzebne pliki są obecne w systemie, obejrzyj zawartość katalogu, w którym znajduje się plik odnaleziony przez polecenie `find`. Jeśli Ghostscript jest zainstalowany, powinieneś uzyskać efekt jak na rysunku 24.1

Rysunek 24.1.
Zawartość katalogu pakietu Ghostscript



```
nxtterm  
[root@localhost /root]# cd /usr/share/ghostscript/3.33  
[root@localhost 3.33]# ls  
Fontmap  gs_dps1.ps  gs_pdf.ps  gs_type1.ps  pdf_draw.ps  quit.ps  
bdftops.ps  gs_fonts.ps  gs_pdf_e.ps  gs_wan_e.ps  pdf_font.ps  showpage.ps  
decrypt.ps  gs_init.ps  gs_pfile.ps  gs_lps.ps  pdf_main.ps  type1enc.ps  
examples  gs_iso_e.ps  gs_res.ps  impath.ps  prfont.ps  type1ops.ps  
font2c.ps  gs_kanji.ps  gs_setpd.ps  landscape.ps  printafm.ps  wftopfa.ps  
gs_btfn.ps  gs_ksb_e.ps  gs_statd.ps  level1.ps  ps2ascii.ps  wrfont.ps  
gs_ccfnt.ps  gs_lev2.ps  gs_std_e.ps  packfile.ps  ps2epsi.ps  
gs_cdfn.ps  gs_mex_e.ps  gs_sunl_e.ps  pdf_2ps.ps  ps2image.ps  
gs_dbt_e.ps  gs_mro_e.ps  gs_type0.ps  pdf_base.ps  psstopm.ps  
[root@localhost 3.33]#
```

Wraz z pakietem Ghostscript dostarczana jest również spora ilość czcionek. Są one zwykle umieszczane w jednym z podkatalogów głównego katalogu pakietu, na przykład `/usr/share/ghostscript/fonts`.

Jeśli Twoja dystrybucja nie zawierała pakietu Ghostscript, możesz go załadować z każdej ze stron WWW i węzłów FTP udostępniających Linuxa. Ich adresy możesz

znaleźć m.in. w rozdziale 2. „Rodzaje Linuxa”. Pakiet `Ghostscript` rozprowadzany jest również przez firmę Aladdin Enterprises.

Pakiet Ghostscript

Głównym programem pakietu Ghostscript jest `gs`, interpreter języka `Ghostscript`. Wczytuje on zadane pliki i wyświetla w oknie systemu X efekt ich sformatowania. Uruchamianie programu `gs` w trybie tekstowym nie ma większego sensu, ponieważ do obejrzenia efektów i tak niezbędne jest okno graficzne.

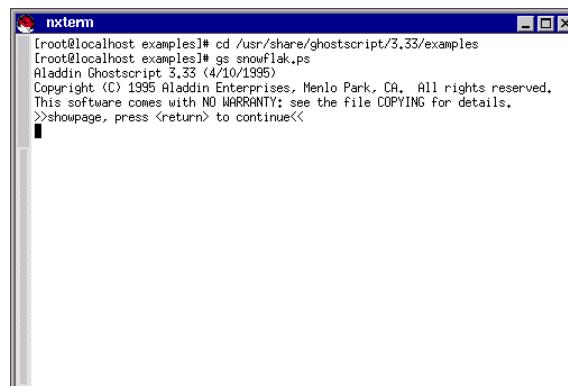
Aby wyświetlić w oknie X sformatowany plik `Ghostscript`, wydaj polecenie `gs`, podając jako parametr nazwę pliku, który chcesz obejrzeć (pliki `Ghostscript` mają zwykle rozszerzenie `.ps`):

```
gs plik1.ps
```

Możesz podać więcej niż jedną nazwę pliku, wówczas będą one wyświetlane kolejno, jeden po drugim. Po wydaniu takiego polecenia pojawi się informacja `Initializing...`, po której wyświetlona zostanie spora ilość komunikatów pochodzących od interpretera języka `Ghostscript`, a następnie otwarte zostanie okno zawierające wynik przetworzenia pliku postscriptowego. Na rysunku 24.2 przedstawione są komunikaty wyświetlane po wydaniu polecenia `gs`, natomiast na rysunku 24.3 – okno zawierające sformatowany dokument.

Rysunek 24.2.

Po wydaniu polecenia `gs` wyświetlanym jest kilka informacji o programie



The screenshot shows a terminal window titled "nxterm". The command entered was "gs snowflak.ps". The output includes copyright information from Aladdin Ghostscript 3.33 (4/10/1995), a note about no warranty, and a prompt to press return to continue. The window has standard window controls (minimize, maximize, close) at the top right.

Jeśli chcesz zapobiec wyświetlaniu komunikatów przez interpreter języka `Ghostscript` (jak na rysunku 24.2), możesz użyć opcji `-q` (ang. *quiet*) – wówczas wyświetlane będą tylko najważniejsze informacje.

Polecenie `gs` opisane jest na stronach `man`, które niestety są napisane w dość nieprzypadkowy sposób. Możesz również uzyskać odpowiednie informacje, wydając polecenie `gs`

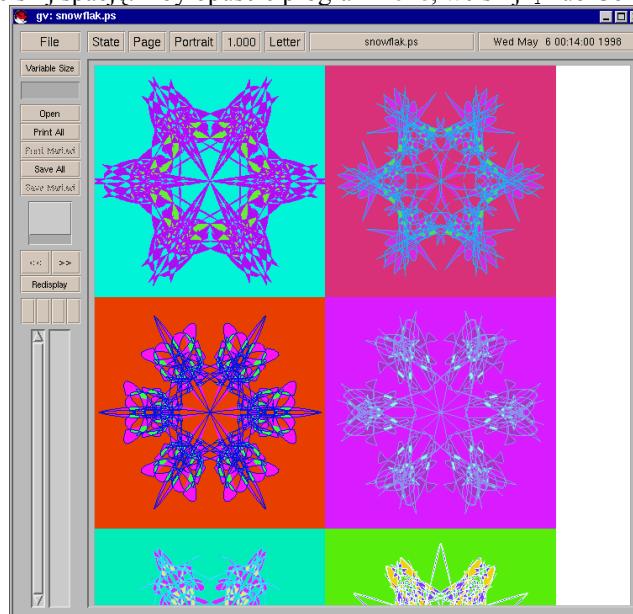
`-h` lub `gs -?`. Ponieważ wyświetlana jest dość spora ilość danych, skieruj wyjście tego polecenia do pliku lub użyj programu `more`, np.:

```
gs -h | more
```

Rezultat wydania tego polecenia przedstawiony jest rysunku 24.4. Jeśli chcesz obejrzeć następną stronę, wciśnij spację. Aby opuścić program `more`, wciśnij `q` lub Control+C.

Rysunek 24.3.

Ghostscript pozwala na wyświetlenie w oknie X sformatowanego dokumentu postscriptowego



Rysunek 24.4.

Do wyświetlenia informacji o programie gs warto wykorzystać polecenie more

```
nxterm
[root@localhost examples]# gs -h |more
Aladdin Ghostscript 3.33 (4/10/1995)
Copyright (C) 1995 Aladdin Enterprises, Menlo Park, CA. All rights reserved.
Usage: gs [switches] [file1.ps file2.ps ...]
Available devices:
  x11 x11alph x11cmjkl x11mono ap3250 imagen iwhi iwlw
  iwlq l350 l370 l375 l375plus lbp8 ln03 l3250
  l34dith lp2563 m8510 necp6 oce9050 paintjet pj pjetxl
  r4081 sj48 st800 stcolor t4693d2 t4693d4 t4693d8 tek4836
  xes deskjet djet500 djet500c dnj500 laserjet ljetplus ljet2p
  ljet3 ljet3d ljet4 cdeskjet cdjcolor cdjmono cdj500 cdj550
  pj pjxl pjx1300 bj10e bj200 bj600 epson eps3mid
  eps9high epsonic ibmpro jett3952 dfaxhigh dfaxlow faxg3 faxg32d
  faxg4 cp50 tiffg3 tiffg32d tiffg4 pcxmono pcxgray pcx16
  pcx256 pcx24b pbm pbmraw pgm pgmraw ppm ppmraw
  bit bitrgb bitcmjkl tifforcle tiffg3 tiffg32d tiffg4 tifflw
  tiffpack
Language interpreters:
  PostScript PostScriptLevel1 PostScriptLevel2 PIF
Search path:
  /usr/share/ghostscript/3.33:/usr/share/ghostscript/fonts
Most frequently used switches: (you can use # in place of =)
--More--
```

Konfigurowanie pakietu Ghostscript do współpracy z X

Niektóre (ale niestety nie wszystkie) wersje Linuxa są tak skonfigurowane, że system X potrafi od razu obsługiwać wyświetlanie okien zawierających efekty działania interpretera języka Ghostscript. Ghostscript używa zasobów systemu X zdefiniowanych w pliku `.Xdefaults`, w klasie `Ghostscript`. Plik `.Xdefaults` powinien zawierać co najmniej trzy wpisy dotyczące interpretera Ghostscript:

```
Ghostscript*geometry: -0+0
Ghostscript*xResolution      72
Ghostscript*yResolution      72
```

Jeśli ich tam nie ma, użyj edytora tekstu i wprowadź je, po czym zrestartuj serwer X albo przeładuj zasoby systemowe. Można to zrobić (jeśli bieżącym katalogiem jest katalog domowy) wydając polecenie:

```
xrdb -merge ./Xdefaults
```

`Ghostscript` może używać jeszcze wielu innych zasobów zdefiniowanych w pliku `.Xdefaults`, ale definiowanie ich nie jest konieczne, ponieważ wartości domyślne sprawują się całkiem dobrze. Zebrane je w tabeli 24.1.

Tabela 24.1. Zasoby systemu X używane przez Ghostscript

Nazwa zasobu	Klasa	Domyślna wartość
background	Background	white
foreground	Foreground	black
borderColor	BorderColor	black
borderWidth	BorderWidth	1
geometry	Geometry	NULL
xResolution	Resolution	calculated
yResolution	Resolution	calculated
useExternalFonts	UseExternalFonts	true
useScalableFonts	UseScalableFonts	true
logExternalFont	LogExternalFont	false
externalFontTolerance	ExternalFontTolerance	10.0
palette	Palette	Color
maxGrayRamp	MaxGrayRamp	128
maxRGBRamp	maxRGBRamp	5
useBackingPixmap	UseBackingPixmap	true
useXPutImage	UseXPutImage	true
useXSetTile	UseXSetTile	true
regularFonts	RegularFonts	none

symbolFonts	SymbolFonts	none
dingbatFonts	DingbatFonts	none

Przekierowanie wyjścia programu Ghostscript

Ghostscript jest bardzo elastyczny, jeśli chodzi o przekierowanie wyjścia do innych urządzeń. Używając argumentów wywołania, można skierować dane do urządzenia takiego jak drukarka za pomocą opcji `-sDEVICE`. Przykładowo, jeśli w Twoim systemie zainstalowane jest urządzenie o nazwie `HPLaser`, możesz skierować do niego dane wyjściowe poleciением:

```
gs -sDEVICE=HPLaser plik.ps
```

W takim przypadku nie zobaczysz efektów działania interpretera w oknie graficznym. Możesz oczywiście wydrukować jednocześnie kilka plików, np.:

```
gs -sDEVICE=HPLaser plik.ps plik1.ps plik2.ps
```

Przekierowanie obowiązuje do momentu zakończenia działania programu `gs`. Jeśli urządzenie, do którego próbujesz skierować dane, nie istnieje, `gs` zakończy działanie z komunikatem `unknown device`. Listę dostępnych urządzeń możesz obejrzeć, wydając polecenie

```
gs -h
```

Jak widać na rysunku 24.4, *Ghostscript* potrafi obsłużyć dość dużo typów urządzeń.

Kolejność parametrów ma znaczenie dla programu `gs`; przykładowo, polecenie

```
gs plik.ps plik1.ps -sDEVICE=HPLaser plik2.ps
```

powoduje wydrukowanie tylko pliku `plik2.ps`, wyświetlając pliki o nazwach `plik.ps` oraz `plik1.ps` na ekranie.

Możesz również wysyłać różne pliki do różnych urządzeń, na przykład tak:

```
gs -sDEVICE=CanonBJet plik.ps plik1.ps -sDEVICE=HPLaser plik2.ps
```

Powyższe polecenie spowoduje wydrukowanie pliku `plik.ps` na drukarce o nazwie `CanonBJet`, natomiast plik `plik2.ps` zostanie wydrukowany na drukarce `HPLaser`. W większości przypadków łatwiej jest jednak wydać dwa osobne polecenia, co pozwala również uniknąć pomyłek.

Ghostscript umożliwia modyfikowanie efektów swojego działania. Jeśli posiadasz jedną ze starszych drukarek igłowych pozwalających określić rozdzielcość drukowanego dokumentu (na przykład dla wydruków próbnych i końcowych), możesz za pomocą odpowiednich opcji wybrać jedną z rozdzielcości. Aby wydrukować dokument na 24-igłowej drukarce Epson, używając najwyższej możliwej rozdzielcości, wydaj polecenie:

```
gs -sDEVICE=Epson -r360x180
```

Jeśli z jakiegoś powodu chcesz odłożyć drukowanie dokumentu na później, możesz efekty działania interpretera zapisać do pliku:

```
gs -sOutputFile out1.ps plik1.ps plik2.ps
```

Choć w systemie Linux można zamiast podanych wyżej opcji użyć przekierowania wejścia i wyjścia, zostały one wprowadzone z myślą o systemach, które nie mają takich możliwości.

Zmiana rozmiaru papieru

Domyślnie Ghostscript używa rozmiarów papieru podanych w pliku `gs_statd.ps`. W pliku tym zdefiniowanych jest kilkadziesiąt różnych formatów, zgodnych z normami europejskimi i amerykańskimi. Aby użyć rozmiaru innego niż domyślny, należy podać opcję `-sPAPERSIZE`, np. aby użyć formatu A4, wydaj polecenie:

```
gs -sPAPERSIZE=a4 -sDEVICE=HPLaser plik1.ps
```

Jeśli rozmiar papieru, którego chcesz użyć, nie jest zdefiniowany w pliku `gs_stats.ps`, musisz zmodyfikować któryś z istniejących w nim wpisów lub utworzyć nowy.



Czasem zmiana rozmiaru papieru przez podanie opcji `-sPAPERSIZE` nie daje rezultatu, ponieważ język PostScript (a więc i Ghostscript) umożliwia osadzenie polecenia określającego rozmiar papieru w dokumencie. Polecenie takie ma pierwszeństwo przed parametrem podanym w wierszu poleceń. Jedynym sposobem rozwiązania problemu jest wówczas modyfikacja pliku postscriptowego.

Zmienne środowiskowe programu Ghostscript

Do modyfikowania zachowania programu Ghostscript można również wykorzystać zmienne środowiskowe. Zmienne używane przez ten interpreter wraz z opisem ich funkcji zebrano w tabeli 24.2.

Tabela 24.2. Zmienne używane przez Ghostscript

Zmienna	Znaczenie
<code>GS_DEVICE</code>	Domyślne urządzenie wyjściowe
<code>GS_FONTPATH</code>	Lista katalogów, w których należy szukać czcionek
<code>GS_LIB</code>	Ścieżka przeszukiwania dla plików inicjalizacyjnych i czcionek; poszczególne katalogi oddzielane są dwukropkami
<code>GS_OPTIONS</code>	Lista argumentów, która będzie przetwarzana przed argumentami podanymi w wierszu poleceń przy wywołaniu programu

Zmienna `GS_OPTIONS` może zostać użyta do modyfikacji działania interpretera `Ghostscript`. Można na przykład w jej definicji zatrzymać opcję `GS_DEVICE` z nazwą drukarki, dzięki czemu wyniki działania interpretera będą domyślnie kierowane na drukarkę. Zmienna `GS_OPTIONS` może zawierać zarówno argumenty, jak i opcje.

Zmienna `TEMP` wskazuje zwykle na katalog `/tmp`, ale można to oczywiście zmienić. Pliki tymczasowe tworzone przez `Ghostscript` mają nazwy rozpoczynające się od `gs_`. Niestety, pliki te nie zawsze są prawidłowo usuwane, więc powinieneś co jakiś czas usuwać je ręcznie, szczególnie jeśli intensywnie używasz programu `gs`.

Ghostview

`Ghostview` to program służący do wyświetlania plików postscriptowych na ekranie. Napisany został przez Tima Theisena i jest dostępny za darmo w Internecie i na płytach CD-ROM. Do opracowania danych używa interpretera `Ghostscript`, ale nie wymaga, by był on wcześniej zainstalowany w systemie.

`Ghostview` jest bardzo łatwy w obsłudze. Aby wyświetlić zawartość pliku `rozdz1.ps`, wydaj polecenie

```
ghostview rozdz1.ps
```

Większość użytkowników zmienia nazwę tego programu (albo tworzy odpowiedni alias) na `gv`, co jest o wiele wygodniejsze do wpisania niż jego pełna nazwa. Jeśli na przykład używasz powłoki `tcs`, możesz wydać polecenie:

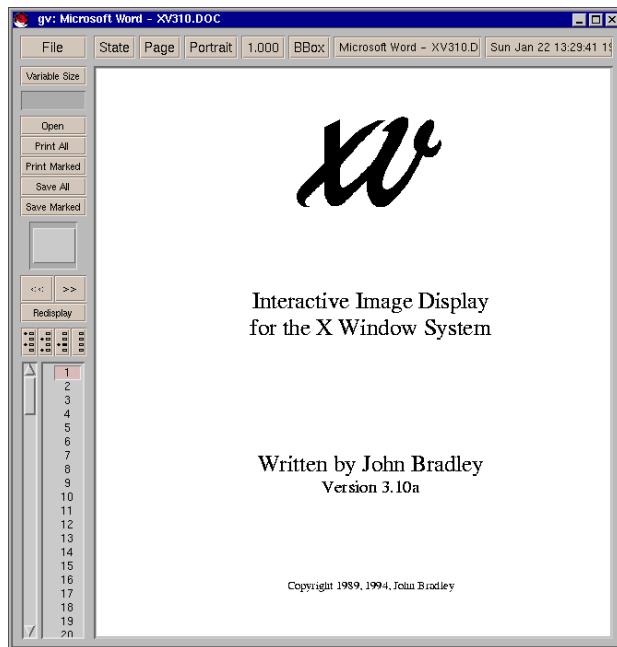
```
alias gv ghostview
```

które spowoduje utworzenie aliasu o nazwie `gv`, wskazującego na program `ghostview`.

Przykładowa zawartość okna głównego programu `gv` przedstawiona jest na rysunku 24.5.

Rysunek 24.5.

*Okno główne
programu ghostview*



Obsługa programu ghostview

Jak widać na rysunku 24.5, sformatowana zawartość pliku postscriptowego wyświetlana jest w taki sam sposób, jak miało to miejsce przy zastosowaniu interpretera Ghostscript (ponieważ program Ghostview wywołuje program Ghostscript, aby opracować dane zawarte w pliku wejściowym). Pasek po prawej pozwala na przesuwanie oglądanego dokumentu, można również kliknąć na odpowiednim numerze, by przejść do wybranej strony.

W oknie programu znajduje się również kilka przycisków. Oto funkcje niektórych z nich.

- υ Obsługa plików (**File**) – pozwala na odczytanie pliku, przeładowanie pliku z dysku, zakończenie działania programu i wyświetlenie informacji o prawach autorskich.
- υ Opcje strony (**Page**) – w menu rozwijanym po naciśnięciu tego przycisku znajdują się polecenia służące do poruszania się w obrębie dokumentu, odświeżania zawartości ekranu, zaznaczenia lub usunięcia zaznaczenia bieżącej strony itp.
- υ Skala (**1.000**) – po naciśnięciu tego przycisku można wybrać stopień powiększenia wyświetlanego dokumentu.
- υ Orientacja (**Portrait**) – pozwala zmienić orientację wyświetlanego dokumentu (**Portrait** – pionowa, **Landscape** – pozioma).

- v Rozmiar papieru (Box) – pozwala zmienić wielkość strony.

`ghostview` jest programem łatwym i przyjemnym w obsłudze. W górnej części okna wyświetlana jest informacja o tytule dokumentu (jeśli został on zdefiniowany) lub o nazwie wyświetlanyego pliku, informacja o wersji programu oraz data utworzenia dokumentu. Data określana jest na podstawie danych zapisanych w samym dokumencie lub na podstawie daty ostatniej modyfikacji wyświetlanyego pliku.



Jedną z cech programu `ghostview`, która może Cię nieco zdziwić, jest sposób, w jaki strony są wyświetlane po pomniejszeniu lub powiększeniu okna. Po zminimalizowaniu, `ghostview` przechowuje wszystkie dane w pamięci, ale przy ponownym powiększeniu okna najpierw sprawdza, czy plik nie został zmieniony. Może to wprowadzić pewne opóźnienie, szczególnie w przypadku dużych plików. Jest to wbrew pozorom efekt zamierzony (za wyjątkiem opóźnienia), ponieważ pozwala na wyświetlanie zawsze aktualnej wersji pliku.

Oprócz przycisków dostępnych w oknie programu `ghostview`, dostępnych jest również mnóstwo opcji podawanych w wierszu poleceń. Służą one głównie do zmiany konfiguracji zasobów systemu X Window lub modyfikacji zmiennych konfiguracyjnych i są omówione dokładniej w dokumentacji programu Ghostview.

Czcionki Ghostview

Wraz z programem `Ghostview` rozpowszechniane są czcionki PostScript typu 1 i 3, które powinny wystarczyć do wyświetlenia i wydrukowania każdego dokumentu, z którym się zetkniesz. Można również dodać inne czcionki, wzbogacając w ten sposób możliwości programu. Powinny one zostać umieszczone w katalogu, w którym znajdują się pozostałe czcionki, czyli np.

```
/usr/share/ghostscript/3.33/fonts.
```

Aby czcionki były prawidłowo rozpoznawane, należy również zmodyfikować zawartość pliku `Fontmap`. Format wpisów w tym pliku jest prosty, wymaga tylko podania nazwy czcionki i pliku, w którym można znaleźć jej definicję, np.:

```
smiesznaczcionka (fun_fnt.pfb);
```

Zasoby X używane przez Ghostview

Zasoby używane przez `Ghostview` różnią się od tych używanych przez `Ghostscript`. Zebrano je w tabeli 24.3.

Tabela 24.3. Zasoby systemu X używane przez `ghostview`

Nazwa zasobu i opis	Klasa	Domyślna
---------------------	-------	----------

		wartość
showTitle – wyświetlanie tytułu	Labels	true
showDate – wyświetlanie daty	Labels	true
showLocator – wyświetlanie informacji o lokalizacji	Labels	true
autoCenter – centrowanie strony w oknie za każdym razem, gdy zmienia się rozmiar okna	AutoCenter	true
horizontalMargin – ile punktów przeznaczyć na margines poziomy	Margin	20
verticalMargin – ile punktów przeznaczyć na margines pionowy	Margin	44
minimumMagstep – najmniejsze dostępne powiększenie	Magstep	-5
maximumMagstep – największe dostępne powiększenie	Magstep	5
magstep – powiększenie domyślne	Magstep	0
orientation – domyślna orientacja strony	Orientation	Portrait
page – numer strony, którą należy domyślnie wyświetlić	Page	
pageMedia – domyślny rodzaj papieru	PageMedia	Letter
forceOrientation – wymuszenie orientacji dokumentu	Force	false
forcePageMedia – wymuszenie rodzaju papieru	Force	false

Tabela 24.3. cd. Zasoby systemu X używane przez ghostview

Nazwa zasobu i opis	Klasa	Domyślna wartość
swapLandscape – zamiana znaczenia parametrów Landscape i Seascape	SwapLandscape	false
printCommand – polecenie używane do drukowania	PrintCommand	
printerVariable – nazwa zmiennej zawierającej informacje o drukarce	PrinterVariable	PRINTER
busyCursor – kurSOR pokazywany podczas odświeżania okna	Cursor	
cursor – domyślna postać kursora	Cursor	cross hair
safer – czy uruchamiać program w „bezpieczniejszym” trybie	Safer	true

Podsumowanie

W tym rozdziale zapoznaliśmy się z programami `Ghostscript` i `Ghostview`, narzędziami przeznaczonymi do obsługi plików postscriptowych. Możliwość obejrzenia zawartości pliku postscriptowego bez konieczności jego drukowania jest w praktyce bardzo przydatna, dlatego `Ghostview` jest jednym z częściej używanych programów w systemach UNIX-owych (czyli również w systemie Linux).

Proces konfigurowania drukarki w systemie linuxowym opisany jest w rozdziale 20. „Drukowanie”.

O programowaniu w systemie Linux mówi część piąta, począwszy od rozdziału 25. „gawk”.

Konfigurowanie różnych urządzeń, w tym drukarek, opisane jest w rozdziale 33. „Urządzenia”.

Część piąta

Linux dla programistów

W tej części:

- υ gawk
- υ Programowanie w języku C
- υ Język C++
- υ Perl
- υ Podstawy języków Tcl i Tk
- υ Inne kompilatory
- υ Smalltalk/X
- υ

Rozdział 25.

gawk

Tim Parker

W tym rozdziale:

- υ Ogólnie o języku gawk
- υ Pliki, rekordy i pola
- υ Kojarzenie wzorców i akcji
- υ Wywoływanie programów w języku gawk
- υ Instrukcje strukturalne

Język programowania o nazwie `awk` został opracowany przez trzech ludzi – byli to: Alfred Aho, Peter Weinberger i Brian Kernighan (nazwa tego języka pochodzi od pierwszych liter ich nazwisk). Program `gawk` jest implementacją języka `awk` rozprowadzaną zgodnie z warunkami licencji GNU.

`gawk` to coś więcej niż zwykły język programowania. Jest on praktycznie niezastąpionym narzędziem dla większości programistów i administratorów systemów. Sam język jest bardzo łatwy do opanowania, a jednocześnie zadziwiająco elastyczny. Gdy poznasz podstawowe zasady programowania w tym języku, będziesz zaskoczony, widząc, w jak wielu różnych sytuacjach może on znaleźć zastosowanie.

Aby pomóc Ci zrozumieć `gawk`, przedstawimy kolejno jego elementy, popierając teorię przykładami. Oczywiście najlepszym sposobem na zapoznanie się z językiem są samodzielne eksperymenty, do których gorąco zachęcamy. W tym rozdziale przedstawimy tylko podstawowe wiadomości dotyczące języka `gawk`, żywiąc nadzieję, że rozbudzi to Twoją ciekawość.

Ogólnie o języku gawk

`gawk` został zaprojektowany jako łatwy w użyciu język programowania, który umożliwia pracę z danymi zapisanymi w plikach (obejmuje to również dane przesypane z innych programów bezpośrednio, za pomocą mechanizmu przekierowania lub mechanizmu potoków, nazywanych po angielsku *pipe*). Podstawowe możliwości tego języka to:

- υ wyświetlanie fragmentu lub całej zawartości pliku, poszczególnych kolumn, wierszy czy pól,
- υ analizowanie tekstów ze względu na występowanie określonych ciągów znaków,
- υ przygotowanie sformatowanego raportu w oparciu o dane zawarte w pliku,
- υ filtrowanie tekstów w oparciu o różnorodne systemy filtrów,
- υ wykonywanie operacji arytmetycznych na danych numerycznych zapisanych w pliku.

Pod wieloma względami język ten idealnie nadaje się na „pierwszy” język programowania, głównie z powodu prostych reguł i wszechstronnego zastosowania w codziennej pracy. Doświadczeni programiści z pewnością również docenią jego prostotę.

Pliki, rekordy i pola

Zwykle program napisany w języku `gawk` pracuje na podstawie o danych zapisanych w pliku. Często są to dane numeryczne, ale `gawk` radzi sobie również z danymi tekstowymi. Jeśli dane nie są zapisane w pliku, można przesłać je do programu, używając mechanizmu potoków lub innego mechanizmu przekierowania danych. Tylko pliki tekstowe w standardzie ASCII mogą być przetwarzane prawidłowo. Choć `gawk` umożliwia pracę z plikami binarnymi, rezultaty często okazują się nieprzewidywalne. Ponieważ jednak w systemie Linux większość informacji zapisywana jest w standardzie ASCII, nie stanowi to większego problemu.

Jako prosty przykład pliku, który może być obsłużony przez program napisany w języku `gawk`, rozpatrzmy plik zawierający książkę telefoniczną. Książka taka składa się z dużej liczby wpisów, ale wszystkie wpisy mają taki sam format: nazwisko, imię, adres, numer telefonu. Cała książka telefoniczna jest posortowana (najczęściej alfabetycznie według nazwisk), ale brak w niej zaawansowanych metod wyszukiwania.

Każdy wiersz takiego pliku jest kompletnym zestawem danych dotyczących jednego numeru telefonicznego i nazywany jest *rekordem*; przykładowo, rekordem jest wiersz

zawierający dane na temat abonenta o nazwisku Jan Kowalski, włączając w to jego adres i numer telefonu.

Każda z informacji zawartych w rekordzie – na przykład imię, nazwisko czy numer telefonu – nazywana jest *polem*. W języku `gawk` polem jest każda pojedyncza informacja, natomiast rekordem – zestaw pól odpowiadający opisowi pojedynczej rzeczy. Zestaw rekordów nazywany jest *plikiem*.

W większości przypadków pola rozdzielane są jednym wybranym znakiem, na przykład spacją, średnikiem, przecinkiem czy znakiem tabulacji. Znak ten nazywany jest *separatorem pól*. Dobrym przykładem pliku o ścisłe określonym formacie rekordów jest plik `/etc/passwd`, który może wyglądać na przykład tak:

```
tparker:t36s54hsh:501:101:Tim Parker:/home/tparker:/bin/bash
etreijs:2ys639dj3h:502:101:Ed Trejis:/home/etreijs:/bin/tcsh
ychow:1h27sj:503:101:Yvonne Chow:/home/ychow:/bin/bash
```

Jeśli przyjrzeć się temu plikowi, można zauważyc, że jako separator pól używany jest dwukropka. Każdy z rekordów (wierszy) zawiera siedem pól: identyfikator użytkownika, hasło (zakodowane), numer identyfikacyjny użytkownika, numer identyfikacyjny grupy, pole komentarza, ścieżkę dostępu do katalogu domowego i do domyślnego interpretera poleceń. Dwukropki używane są wyłącznie do rozdzielania poszczególnych pól. Program, który ma wyszukać szóste pole, musi jedynie znaleźć piąty dwukropek (ponieważ przed pierwszym polem nie ma dwukropka) – po nim na pewno nastąpi szóste pole.

Tu napotykamy pierwszy problem. Wróćmy ponownie do przykładu książki telefonicznej. Założymy, że zawartość pliku jest następująca:

Smith, John	13 Wilson St.	555-1283
Smith, John	2735 Artside Dr, Apt 123	555-2738
Smith, John	125 Westmount Cr	555-1728

My wiemy, że każdy z rekordów zawiera cztery pola: nazwisko, imię, adres i numer telefonu. `gawk` widzi taki plik jednak nieco inaczej. Ponieważ separatorem pól jest spacja, w pierwszym wierszu rozpoznaje on ciąg `Smith` jako pierwsze pole, ciąg `John` – jako drugie pole, `13` jako pole trzecie, `Wilson` – czwarte itd. Z punktu widzenia języka `gawk` pierwszy wiersz składa się z sześciu pól. Drugi wiersz ma natomiast osiem pól. Dodatkowe znaki białe (spacje i znaki tabulacji) są ignorowane, chyba że zmienisz separator pól na znak spacji lub tabulacji.



Podczas pracy z dowolnym językiem programowania musisz niesięcić postrzegać dane właśnie w taki sposób, jaki wynika z zasad działania tego języka. Komputery biorą wszystko dosłownie.

Aby można było używać pliku zawierającego książkę telefoniczną zgodnie z naszą koncepcją, najłatwiej jest nieco zmienić jego format, na przykład wstawiając jako separator znak \:

```
Smith/John/13 Wilson St./555-1283
Smith/John/2735 Artside Dr, Apt 123/555-2738
Smith/John/125 Westmount Cr/555-1728
```

`gawk` domyślnie traktuje znaki białe jako separatory pól, chyba że zostanie poinstruowany, by używać innego znaku. Jeśli pozostaniesz przy ustawieniu domyślnym, to nie ma znaczenia, ile spacji czy tabulatorów będzie występowało obok siebie – zostaną one potraktowane jako pojedynczy separator. Oczywiście istnieje sposób, by zmienić to zachowanie.

Kojarzenie wzorców i akcji

W języku `gawk` istnieje jeden format określony prawie dla wszystkich poleceń. Poleceние składa się z dwóch części: wzorca i odpowiadającej mu akcji. Za każdym razem, gdy uda się dopasować wzorzec do danych wejściowych, wykonywana jest skojarzona z nim akcja.

Podejście to jest nieco podobne do języka naturalnego. Założmy, że chcesz wytlumaczyć komuś, jak dojść na pocztę. Móglbyś na przykład ująć to następująco: „na końcu ulicy skręć w prawo, potem idź aż do znaku stop, skręć w lewo, idź do końca ulicy i skręć w prawo”. Zapis takiej informacji zgodny z filozofią języka `gawk` mógłby wyglądać tak:

```
koniec ulicy: skręć w prawo  
znak "stop": skręć w lewo  
koniec ulicy: skręć w prawo
```

Po dopasowaniu wzorca podejmowane są odpowiednie akcje. Nie powinieneś skręcać w prawo, zanim nie dojdzieš do końca ulicy, nie powinieneś skręcać w lewo przed znakiem stop. Przykład jest może nieco uproszczony, ale oddaje ogólną ideę.

W języku `gawk` pary wzorzec-akcja podaje się w następujący sposób:

```
/wzorzec1/{akcja1}  
/wzorzec2/{akcja2}  
/wzorzec3/{akcja3}
```

Dzięki takiemu formatowi łatwo zorientować się, gdzie kończy się wzorzec, a zaczyna określenie skojarzonej z nim akcji. Każdy program w języku `gawk` jest zestawem takich par. Pamiętaj, że wzorce i akcje dotyczą danych tekstowych, więc wzorce są zwykle ciągami znaków, a akcje – poleceniami typu wyświetl czy usuń tekst.

W przypadku, gdy nie został podany żaden wzorzec, każdy tekst uważany jest za „pasujący” i odpowiednia akcja podejmowana jest za każdym razem. Jeśli nie podamy żadnej akcji, `gawk` skopiuje bez zmian wiersz pasujący do wzorca z wejścia na wyjście.

Spójrzmy na następujący przykład:

```
gawk '/tparker/' /etc/passwd
```

Polecenie to spowoduje wyszukanie wszystkich wierszy zawierających tekst `tparker` w pliku `/etc/passwd`, i, ponieważ nie określono żadnej akcji, wyświetlenie ich na ekranie. W tym przypadku `gawk` zachowuje się tak samo jak program `grep`.

Powyższy przykład pokazuje dwie ważne rzeczy: `gawk` może zostać uruchomiony z wiersza poleceń (jego parametrami są wtedy pary wzorzec-akcja i nazwa pliku z danymi wejściowymi), a pary wzorzec-akcja należy ująć w pojedynczy cudzysłów, aby mogły zostać odróżnione od nazwy pliku wejściowego.

Język `gawk` dopasowuje tekst do wzorca litera po literze, więc wzorzec „kot” zostanie odnaleziony zarówno w słowie „kot”, jak i w słowie „maskotka”. Jeśli chcesz wyszukać tylko wiersze zawierające cały wyraz, powinieneś otoczyć wzorzec spacjami („ kot ”). Ważna jest również wielkość liter. Na szczęście w języku `gawk` dostępnych jest wiele znaków specjalnych rozszerzających lub zawężających zakres poszukiwań; zostaną one omówione w sekcji „Symbole specjalne”.

Wybiegając nieco naprzód, przeanalizujmy następujące polecenie:

```
gawk '{print $3}' plik2.dat
```

Spowoduje ono wyświetlenie (akcja określona jest przez polecenie `print` – wyświetl) trzeciego pola (`$3`) każdego wiersza pliku `plik2.dat` (ponieważ nie podano żadnego wzorca). Domyślnym separatorem pól jest spacja, jeśli więc spróbujemy użyć tego polecenia, podając jako plik wejściowy plik `/etc/passwd`, najpewniej nie zostanie wyświetlony żaden tekst, ponieważ tam separatorem pól jest dwukropki.

Moglibyśmy połączyć dwa podane wcześniej przykłady:

```
gawk '/UNIX/{print $2}' plik2.dat
```

Powyższe polecenie spowoduje przeszukanie pliku `plik2.dat` i wyświetlenie drugiego pola każdego wiersza zawierającego wyraz `UNIX`.



Cudzysłów otaczający parę wzorzec-akcja jest bardzo ważny i nie powinien być pomijany. Bez niego polecenie może nie zostać prawidłowo wykonane. Upewnij się również, że używasz właściwych znaków cudzysłowu (a nie na przykład pojedynczego cudzysłowu na początku, a podwójnego na końcu).

W obrębie jednego polecenia można oczywiście zdefiniować więcej par wzorzec-akcja, np.:

```
gawk '/skandal/{print $1} /rozvod/{print $2}' plotki.txt
```

Powyższe polecenie spowoduje wyszukanie w pliku `plotki.txt` wszystkich wierszy zawierających słowo `skandal` i wyświetlenie ich pierwszego pola, a następnie ponowne przeszukanie tego pliku od początku, tym razem wyświetlając drugie pole wierszy zawierających słowo `rozvod`. Przeszukiwanie dla każdej pary wzorzec-akcja rozpoczyna się od początku pliku.

Proste wzorce

Jak już się pewno zorientowałeś, `gawk` numeruje poszczególne pola rekordu: pierwsze pole to `$1`, drugie – `$2` itd. Cały rekord nazywa się `$0`. Dla uproszczenia `gawk` pozwala opuścić argument `$0` w nieskomplikowanych instrukcjach, tak więc wszystkie poniższe polecenia dadzą ten sam rezultat:

```
gawk '/tparker/{print $0}' /etc/passwd  
gawk '/tparker/{print}' /etc/passwd  
gawk '/tparker/' /etc/passwd
```

Oczywiście `gawk` potrafi o wiele więcej niż tylko wyszukać i wydrukować fragment tekstu. Można na przykład porównać zawartość danego pola ze stałą:

```
gawk '$2 == "cos" {print $3}' plik_testowy
```

Powyższe polecenie nakazuje porównanie drugie pola każdego rekordu zapisanego w pliku `plik_testowy` ze stałą `"cos"`, a w przypadku, gdy wynik porównania będzie pozytywny, wydrukowanie trzeciego pola opracowywanego rekordu.

W tym przykładzie zauważycie można kilka ważnych szczegółów. Po pierwsze, wzorzec nie jest otoczony znakami `/`, ponieważ nie chcemy porównywać danych z konkretnym tekstem, ale sprawdzić pewien warunek. Po drugie, symbol `==` jest operatorem porównania. Należy używać podwójnego znaku równości, ponieważ pojedynczy znak `=`, jak się wkrótce przekonasz, służy do przypisywania wartości zmiennej. Podwójny cudzysłów otaczający tekst `cos` zapobiega jego interpretacji. Stałych numerycznych nie trzeba otaczać cudzysłowem.



Nie pomył cudzysłowu używanego dla wymuszenia "dosłownej" interpretacji ciągu znaków z tym otaczającym parę wzorcem-akcją w wierszu polecień. Jeśli w obu przypadkach użyjesz tego samego rodzaju cudzysłówów, `gawk` nie będzie w stanie prawidłowo zinterpretować polecenia.

Porównania i arytmetyka

Jedną z podstawowych cech każdego języka programowania jest możliwość porównania dwóch tekstów lub liczb i ustalenia, czy są one identyczne. W języku `gawk` dostępnych jest kilka operatorów pozwalających na porównywanie argumentów (między innymi wspomniany wcześniej operator `==`) – zostały one zebrane w tabeli 25.1.

Tabela 25.1. Operatory porównania

Symbol	Opis
<code>a==b</code>	a jest równe b

a != b	a nie jest równe b
a > b	a jest większe od b
a < b	a jest mniejsze od b
a >= b	a jest większe lub równe b
a <= b	a jest mniejsze lub równe b

Operatory te prawdopodobnie są Ci dobrze znane (identyczne lub bardzo podobne występują w każdym języku programowania). Jako przykład ich zastosowania niech posłuży poniższe polecenie, wyświetlające każdy wiersz pliku `test`, w którym wartość w czwartej kolumnie jest większa od 100:

```
gawk '$4 > 100' test
```

W języku `gawk` dostępne są również wszystkie podstawowe operatory arytmetyczne (dodawanie, odejmowanie, mnożenie i dzielenie), jak również kilka bardziej zaawansowanych (np. funkcje wykładnicze czy reszta z dzielenia). Przedstawia je tabela 25.2.

Tabela 25.2. Operatory arytmetyczne

Operator	Znaczenie	Przykład
+	Dodawanie	2 + 6
-	Odejmowanie	6 - 2
*	Mnożenie	2 * 5
/	Dzielenie	8 / 3
^	Potega	3 ^ 2 (=9)
%	Reszta z dzielenia	9 % 4 (=1)

Można również używać jednocześnie operatorów arytmetycznych i numerów pól, np. polecenie

```
{print $3/2}
```

powoduje wyświetlenie wartości zawartych w trzeciej kolumnie podzielonych przez dwa.

Do dyspozycji mamy też zestaw funkcji trygonometrycznych oraz generujących liczby przypadkowe; zebrano je w tabeli 25.3.

Tabela 25.3. Funkcje matematyczne

Funkcja	Opis
<code>sqrt(x)</code>	Pierwiastek kwadratowy z x
<code>sin(x)</code>	Sinus x (x w radianach)
<code>cos(x)</code>	Kosinus x (x w radianach)

<code>atan2 (x, y)</code>	Arkus tangens x/y
<code>log (x)</code>	Logarytm naturalny z^x
<code>exp (x)</code>	Stała e do potęgi x
<code>int (x)</code>	Część całkowita z^x
<code>rand ()</code>	Liczba przypadkowa z zakresu 0 – 1
<code>srand (x)</code>	Przypisanie zarodkowi generatora liczb losowych wartości x

Kolejność wykonywania operacji jest taka sama jak w normalnej arytmetyce: najpierw obliczane są wyrażenia w nawiasach, potem potęgowania, następnie mnożenia, dzielenia i obliczanie reszty, na koniec dodawania i odejmowania, na przykład:

```
{print $1+$2*$3}
```

powoduje wyświetlenie iloczynu wartości w kolumnach drugiej i trzeciej powiększonego o wartość z kolumny pierwszej. Dla wymuszenia innej kolejności wykonywania działań należy użyć nawiasów; polecenie

```
{print ($1+$2)*$3}
```

powoduje wyświetlenie iloczynu wartości w polu trzecim i sumy wartości z pól pierwszego i drugiego. Jeśli masz wątpliwości co do tego, czy działania zostaną wykonane w żądanej kolejności, zawsze powinieneś używać nawiasów.

Łańcuchy znaków i liczby

Jeśli znasz już jakiś język programowania, zagadnienia te nie są dla Ciebie niczym nowym. Jeśli nie, nie przejmuj się, ponieważ są one bardzo proste. Mimo tego jednak zadziwiająco wiele osób płacze się beznadziejnie przy programowaniu tylko dlatego, że użyli łańcucha znaków tam, gdzie powinni byli użyć liczby.

Łańcuch znaków powinien zawsze być otoczony cudzysłowem. Liczby nie są nim otaczane i traktowane są jako liczby rzeczywiste. Polecenie

```
gawk '$1!="Tim" {print}' test
```

spowoduje wyświetlenie wierszy pliku `test` nie zaczynających się od tekstu `Tim`. Gdybyśmy opuścili cudzysłów, polecenie nie zostałoby zinterpretowane poprawnie. Natomiast polecenie

```
gawk '$1=="50" {print}' test
```

spowoduje wyświetlenie wierszy, w których pierwsza kolumna zawiera tekst `50`; nie zostanie przeprowadzone porównanie wartości zapisanej w pierwszej kolumnie z liczbą `50`, ale porównanie znak po znaku. Tekst `"50"` nie oznacza w języku `gawk` tego samego, co liczba `50`.

Formatowanie wyjścia

Widziałeś już, w jaki sposób można wykonywać proste operacje. Można również podejmować akcje nieco bardziej skomplikowane, np.:

```
gawk '$1 != "Tim" {print $1, $5, $6, $2}' test
```

Powyższe polecenie wyświetla pierwszą, piątą, szóstą i drugą kolumnę każdego wiersza, w którego pierwszym polu znajduje się tekst inny niż Tim. Polecenie print może wyświetlić dowolną liczbę pól.

Można również za pomocą polecenia print wyświetlać stałe teksty, np.

```
gawk '$1 != "Tim" {print "Pole ", $1 " nie zawiera tekstu Tim"}' test
```

Poszczególne elementy polecenia print rozdzielane są przecinkami. Na końcach tekstów stałych, które mają zostać wydrukowane, wstawiono spacje, które będą rozdzielały tekst i wyświetlana po nim wartość.

Dostępne są także inne funkcje formatujące, zaczerpnięte z języka C, np. printf (ang. *print formatted*). Funkcja ta używa łańcucha formatującego wyświetlany tekst oraz ciągu zmiennych odpowiadających każdemu odwołaniu w łańcuchu, na przykład tak:

```
printf "%7s lubi ten język\n", $2
```

%7s to część łańcucha formatującego, powodująca wyświetlenie siedmiu znaków zmiennej, której identyfikator podano po tym łańcuchu (czyli drugiego pola). Symbol \n na końcu łańcucha formatującego powoduje przejście do następnego wiersza. Jeśli druga kolumna czterowierszowego pliku zawiera imiona, podane wyżej polecenie printf spowoduje, że tekst zostanie sformatowany w następujący sposób:

```
Lech lubi ten język
Jacek lubi ten język
Wojtek lubi ten język
Agniesz lubi ten język
```

Symbol formatu %7s powoduje wyrównanie wyświetlanych danych do prawej krawędzi pola, co zapobiega występowaniu nadmiarowych spacji w środku zdania.

Tabela 25.4 zawiera znaki specjalne, które mogą wystąpić po symbolu % w łańcuchu formatującym; znaki te decydują, jakiego typu zmienna ma zostać odczytana i podstawniona z listy zmiennych następującej po tym łańcuchu.

Tabela 25.4. Znaki specjalne łańcucha formatującego

Znak	Opis
c	Pierwszy znak łańcucha znaków lub znak o kodzie odpowiadającym wartości liczby całkowitej
d	Liczba całkowita
e	Liczba rzeczywista w notacji wykładniczej
f	Liczba rzeczywista w notacji tradycyjnej

g	Liczba rzeczywista w notacji wykładniczej lub tradycyjnej, w zależności od tego, która z nich jest krótsza
o	Liczba ósemkowa bez znaku
s	Łańcuch znaków
x	Liczba szesnastkowa bez znaku

Przed każdym z tych symboli można podać liczbę określającą szerokość pola, w którym dana zostanie wyświetlona, na przykład symbol %6d oznacza, że dana całkowita ma być wyświetlana w polu o szerokości sześciu znaków. Można również użyć znaku – (minus), który wymusi wyrównywanie do lewej strony, zamiast domyślnie do prawej. Analogicznie do poprzedniego przykładu wydanie polecenia

```
gawk {printf "%-7s lubi ten jezyk\n", $2} imiona
```

da następujący rezultat:

```
Lech    lubi ten jezyk
Jacek   lubi ten jezyk
Wojtek  lubi ten jezyk
Agnieszla lubi ten jezyk
```

Można użyć kilku rodzajów zmiennych jednocześnie, pamiętając o tym, że każdemu odwołaniu do zmiennej w łańcuchu formatującym musi odpowiadać dokładnie jedna zmienna:

```
{printf "%7s pracuje przez %2d godzin dziennie i zarabia %6fzl Σ
miesiecznie", $1, $2, $3}
```

Przy wyświetlanie liczb można określić ich precyzję używając kropki, po której następuje liczba cyfr po przecinku, które mają być wyświetlane:

```
{printf "%7s zarabia %.2fzl na godzine.", $1, $6}
```

Powyższe polecenie formatuje tekst w ten sposób, że dane zawarte w pierwszej kolumnie wyświetlane są w polu o szerokości siedmiu znaków, natomiast dane numeryczne z kolumny szóstej – z dokładnością do dwóch miejsc po przecinku, na przykład tak:

```
Piotr zarabia 9.12zl na godzine.
Michał zarabia 12.10zl na godzine.
```

Jeśli chcesz ograniczyć liczbę cyfr wyświetlanych na prawo od kropki, możesz to zrobić następująco:

```
{printf "%7s zarabia %6.2fzl na godzine.", $1, $6}
```

Powyższe polecenie spowoduje, że wyświetlonych zostanie sześć cyfr do kropki i dwie cyfry po kropce dziesiętnej.

Symboly rozpoczynające się od znaku lewego ukośnika \ (ang. *escape codes*) mają również specjalne znaczenie (jeden z nich już poznaleś – \n powoduje przejście do nowego wiersza). Zebrane je w tabeli 25.5.

Tabela 25.5. Symbole zaczynające się od znaku \

Symbol	Opis
\a	Sygnal dźwiękowy
\b	Usunięcie znak na lewo od kursora
\f	Wysunięcie papieru
\n	Nowy wiersz
\r	Powrót karetki
\t	Tabulator
\v	Tabulator pionowy
\ooo	Symbol o kodzie ósemkowym ooo
\xdd	Symbol o kodzie szesnastkowym dd
\znak	Dowolny znak, bez interpretowania przez gawk

Za pomocą tych symboli można uzyskać wydrukowanie znaku mającego dla języka gawk specjalne znaczenie – na przykład znaku " – przez poprzedzenie go lewym ukośnikiem:

```
{printf "Powiedziałem \"Czesc\" a on odpowiedzial \"Witaj\"."}
```

Takie użycie znaku lewego ukośnika wygląda może nieco dziwnie, ale jest konieczne dla uniknięcia problemów. W tym rozdziale pojawi się więcej przykładów użycia tego typu sekwencji znaków.

Zmiana separatora pól

Jak wspomniano wcześniej, domylnymi separatorami pól są znaki białe (spacje i znaki tabulacji). Nie zawsze jest to wygodne, o czym przekonaliśmy się, próbując przetwarzać plik /etc/passwd. Do zmiany separatora pól służy podawana z wiersza poleceń opcja -F, po której następuje nowy separator, np.

```
gawk -F":":' /tparker/{print}' /etc/passwd
```

Powyższe polecenie zmienia separator pól na dwukropki i szuka w pliku /etc/passwd wierszy zawierających tekst tparker. Nowy separator pól należy ująć w podwójny cudzysłów. Opcja -F (koniecznie pisana wielką literą) musi znaleźć się przed określeniem par wzorzec-akcja.

Symboli specjalne

W skład wzorców w języku `gawk` mogą również wchodzić symbole wieloznaczne i inne znaki specjalne. Z podobną sytuacją mieliśmy już do czynienia w przypadku interpretatorów poleceń. Przykładowo, wzorzec `kot` pasuje do każdego wiersza zawierającej te trzy litery, jedna po drugiej. Jeśli chcesz być bardziej precyzyjny i znaleźć tylko wystąpienia słowa `kot`, a nie na przykład maskotka `czy szkot`, możesz otoczyć to słowo spacjami:

```
/ kot /{print}
```

A co w sytuacji? jeśli chcesz znaleźć to słowo bez względu na to, czy jest pisane małymi, czy wielkimi literami? Można w takim przypadku użyć symbolu `|`, który oznacza „lub”:

```
/ kot | KOT /{print}
```

W ten sposób nadal jednak nie uda się odnaleźć słowa `kot`. Można użyć możliwości oferowanych przez operator `[]`, który pozwala na zdefiniowanie dowolnego zbioru akceptowanych znaków. Aby znaleźć słowo `kot` pisane małymi lub wielkimi literami w dowolnych kombinacjach, można wpisać:

```
/ [Kk][Oo][Tt] /{print}
```

Wygląda to dość dziwnie; na szczęście takie konstrukcje raczej nie przydają się w praktyce. Aby znaleźć tylko wyrazy `kot` i `Kot`, wystarczy wpisać

```
/ [Kk]ot /{print}
```

Przydatnym znakiem specjalnym jest tylda (~). Jest ona używana po to, by wskazać, w którym polu konkretnie ma być szukany zadany tekst, np. wzorzec:

```
$5 ~/tparker/
```

pasuje do każdego wiersza, w którym piąte pole zawiera tekst `tparker`. Działanie tego operatora jest podobne do operatora `==`. Operator `~` można również zaprzeczyć:

```
$5 !~/tparker/
```

Powyższy wzorzec pasuje tylko do tych wierszy, których piąte pole jest różne od tekstu `tparker`.

Istnieje jeszcze kilka innych znaków specjalnych języka `gawk`. W większości mają one funkcje podobne do tych używanych w powłokach, więc nie powinieneś mieć problemów z ich stosowaniem. Zebrano je w tabeli 25.6.

Tabela 25.6. Znaki specjalne języka `gawk`

Znak	Znaczenie	Przykład	Opis przykładu
<code>^</code>	Początek pola	<code>\$3~/^b/</code>	Pasuje, gdy trzecie pole zaczyna się od litery <code>b</code>

\$	Koniec pola	<code>\$3~/\\$b/</code>	Pasuje, gdy trzecie pole kończy się literą <code>b</code>
.	Dowolny znak	<code>\\$3~/i.m/</code>	Pasuje, gdy w trzecim polu występuje litera <code>i</code> , po niej dowolna jedna litera, a następnie <code>m</code>
	Lub	<code>/kot KOT/</code>	Pasuje o tekstu <code>kot</code> lub <code>KOT</code>
*	Zero lub więcej powtórzeń znaku	<code>/UNI*X/</code>	Pasuje do słów <code>UNIX</code> , <code>UNIX</code> , <code>UNIIX</code> , <code>UNIIIX</code> itd.
+	Jedno lub więcej powtórzenie znaku	<code>/UNI+X/</code>	Pasuje do słów <code>UNIX</code> , <code>UNIIX</code> itd., ale nie do <code>UNX</code>
\{a,b\}	Ilość powtórzeń pomiędzy <code>a</code> i <code>b</code>	<code>/UNI\{1,3\}X</code>	Pasuje tylko do tekstów <code>UNIX</code> , <code>UNIIX</code> i <code>UNIIIX</code>
?	Zero lub jedno powtóżenie	<code>/UNI?X/</code>	Pasuje tylko do <code>UNX</code> i <code>UNIX</code>
[]	Zestaw znaków	<code>/I[BDG]M/</code>	Pasuje do <code>IBM</code> , <code>IDM</code> i <code>IGM</code>
[^]	Znaki spoza zestawu	<code>/I[^DE]M/</code>	Pasuje do wszystkich ciągów trzyznakowych zaczynających się od <code>I</code> a kończących na <code>M</code> , za wyjątkiem <code>IDM</code> i <code>ITEM</code>

Niektóre z tych symboli używane są dość często. Przykłady ich użycia zawarte są w dalszej części tego rozdziału.

Wywoływanie programów w języku gawk

Wywoływanie programu `gawk` z pojedynczą parą wzorzec-polecenie nie jest w przypadku bardziej skomplikowanych zadań ani wygodne, ani szybkie. Warto wtedy zapisać polecenia języka `gawk` do pliku ASCII (zwanego dalej skryptem). Oto przykładowa zawartość skryptu `gawk`:

```
/tparker/{print $6}
$2!="cos"{print}
```

Pierwsze polecenie powoduje wyszukanie tekstu `tparker` i wyświetlenie szóstego pola zawierających go wierszy. Drugie polecenie skryptu powoduje wznowienie szukania od początku pliku, wyświetlając wszystkie wiersze, których drugie pole nie zawiera tekstu `cos`. W skrypcie nie trzeba otaczać par wzorzec-akcja pojedynczym cudzysłowem, tak jak miało to miejsce w przypadku podawania ich w wierszu polecień. Po zapisaniu takiego skryptu pod nazwą np. `skrypt1`, można uruchomić go, wydając polecenie

```
gawk -f skrypt1 plik_we
```

Polecenie to spowoduje przetworzenie wszystkich par wzorzec-akcja zapisanych w skrypcie, traktując jako dane wejściowe plik o nazwie `plik_we`. W ten sposób działa większość programów napisanych w języku `gawk`. Uważaj, by nie pomylić opcji `-f` i `-F`, ponieważ mają one zupełnie inne znaczenia.

Jeśli chcesz użyć innego separatora pól, opcja `-F` musi wystąpić po opcji `-f` (oczywiście można również zapisać odpowiednie polecenie w pliku skryptu, ale o tym za chwilę):

```
gawk -f skrypt1 -F ":" plik_we
```

Można też przetwarzać więcej niż jeden plik, dopisując na końcu polecenia kolejne ich nazwy:

```
gawk -f skrypt1 plik1 plik2 plik3 ...
```

Domyślnie dane wyjściowe programu `gawk` kierowane są na ekran. Możesz oczywiście przekierować je do pliku:

```
gawk -f skrypt1 plik_we > wynik
```

Można również podać nazwę pliku wyjściowego wewnętrz skryptu, o czym będzie mowa później.

BEGIN oraz END

Podczas pisania skryptów często przydatne okazują się dwa specjalne wzorce obsługiwane przez `gawk`: `BEGIN` oraz `END`. `BEGIN` używany jest do podejmowania pewnych działań przed rozpoczęciem przetwarzania pliku, zwykle do inicjalizacji jakichś wartości, ustawienia separatora pól itp. `END` pozwala na wykonanie działań po zakończeniu przetwarzania pliku, zwykle wyświetlenia podsumowania lub po prostu informacji o zakończeniu działania.

Instrukcje następujące po wzorcach `BEGIN` i `END` należy otoczyć nawiasami klamrowymi. Wzorce te muszą być pisane wielkimi literami. Poniżej przedstawiamy prosty przykład ich zastosowania:

```
BEGIN {print "Rozpoczynam przetwarzanie pliku"}  
$1 == "UNIX" {print}  
$2 >10 {printf "Ten wiersz zawiera wartosc %d", $2}  
END {print "Koniec przetwarzania pliku"}
```

W powyższym skrypcie najpierw wyświetlana jest wiadomość `Rozpoczynam przetwarzanie pliku`, a następnie wszystkie wiersze pliku wejściowego, które w pierwszej kolumnie zawierają wyraz `UNIX`. Następnie plik jest przetwarzany ponownie i jeśli drugie pole wiersza zawiera wartość liczbową większą od 10, wartość ta jest wyświetlana wraz z odpowiednim komunikatem. Na koniec wyświetlana jest informacja `"Koniec przetwarzania pliku"`.

Zmienne

Wiesz na pewno, że zmienna to miejsce, w którym zapisywane mogą być pewne wartości. Z każdą zmienną w języku `gawk` skojarzona jest odpowiadająca jej nazwa. Wartość zmiennej może oczywiście się zmieniać.

Do przypisania wartości zmiennej służy operator `=`, na przykład

```
zm1 = 10
```

powoduje przypisanie zmiennej `zm1` wartości 10 (jest to liczba, nie tekst). W języku `gawk`, w przeciwieństwie do wielu innych języków programowania nie deklaruje się zmiennych przed ich użyciem. Dzięki temu używanie zmiennych jest bardzo proste.



Operatory `=` (operator przypisania) i `==` (operator porównania) są często mylone. Trzeba uważać przy ich stosowaniu.

Zmiennych można używać wewnętrz nawiasów klamrowych określających akcje:

```
$1 == "Plastik" {licznik=licznik+1}
```

Ta para wzorców-akcja sprawdza, czy pierwsze pole rekordu zawiera tekst `Plastik`, jeśli tak, to zwiększa o jeden zmienną `licznik`. Gdzieś przed tym wierszem powinniśmy przypisać zmiennej `licznik` wartość początkową (zwykle w sekcji `BEGIN`), w przeciwnym przypadku może się okazać, że dodajemy jeden do jakiejś zupełnie przypadkowej wartości.



W zasadzie `gawk` inicjalizuje wszystkie zmienne wartością zero, więc ręczna inicjalizacja nie jest tak naprawdę niezbędna, ale taka praktyka jest dobrym zwyczajem i warto ją stosować, ponieważ jest wymagana w niektórych innych językach programowania.

Oto trochę bardziej praktyczny przykład:

```
BEGIN { licznik=0 }
$5=="UNIX" {licznik=licznik+1}
END {printf "Znaleziono %d wystapien slowa UNIX",licznik}
```

W sekcji `BEGIN` zmiennej `licznik` nadawana jest wartość 0. Następnie przetwarzany jest plik wejściowy, a każde wystąpienie tekstu `UNIX` w piątym polu opracowywanego rekordu powoduje zwiększenie o jeden wartości zmiennej `licznik`. Po zakończeniu przetwarzania pliku wyświetlona zostanie wartość tej zmiennej.

Zmienne mogą być używane zarówno w odniesieniu do pól, jak i wartości, dopuszczalne są więc konstrukcje takie jak:

```
licznik=licznik+$6
licznik=$5-8
licznik=$5+zm1
```

Zmienne mogą również być częścią wzorca:

```
$2 > max_wart {print "Wartosc maksymalna przekroczona o ", $2-max_wart}  
$4 - zml <min_wart {print "Nieprawidlowa wartosc :", $4}
```

Dostępne są również dwa specjalne operatory służące do zwiększenia (inkrementacji) i zmniejszania (dekrementacji) wartości zmiennej o 1. Zostały one zapożyczone z języka C:

licznik++ zwiększenie wartości zmiennej licznik o 1
licznik-- zmniejszenie wartości zmiennej licznik o 1.

Zmienne wewnętrzne

Język `gawk` posiada również pewną ilość zmiennych wewnętrznych (ang. *built-in*), zawierających takie informacje, jak liczba przetworzonych rekordów itp. Są one przydatne głównie przy wyświetlaniu różnego rodzaju podsumowań. Ważniejsze zmienne wewnętrzne zostały zebrane w tabeli 25.7.

Tabela 25.7. Ważniejsze zmienne wewnętrzne języka gawk

Zmienna	Opis
NR	Liczba przeczytanych rekordów
FNR	Liczba rekordów przeczytanych z bieżącego pliku
FILENAME	Nazwa aktualnego pliku wejściowego
FS	Separator pól (domyślnie znaki białe)
RS	Separator rekordów (domyślnie koniec wiersza)
OFMT	Format wyjściowy dla liczb (domyślnie %g)
OFS	Separator pól dla wyjścia
ORS	Separator rekordów dla wyjścia
NF	Liczba pól w bieżącym rekordzie

Wartości zmiennych `NR` i `FNR` są takie same, jeśli przetwarzany jest jeden plik, ale jeśli przetwarzasz ich więcej, `NR` zawiera wartość będącą sumą liczby rekordów przeczytyanych z wszystkich plików, a `FNR` – tylko z pliku bieżącego.

Aby zmienić separator pól wewnętrz skryptu, należy po prostu nadać nową wartość zmiennej `FS`; jeśli chcesz, by separatorem pól był dwukropki (jak w pliku `/etc/passwd`), użyj (na przykład w definicji akcji skojarzonej z wzorcem `BEGIN`) polecenia:

```
FS=":"
```

Zmiennych wewnętrznych można używać tak samo, jak wszystkich innych zmiennych. W poniższym przykładzie generowany jest komunikat o błędzie, jeśli rekord w pliku wejściowym ma zbyt mało pól:

```
NF <= 5 {print "Za malo pol w rekordzie"}
```

Instrukcje strukturalne

Wyjaśniliśmy już większość szczegółów niezbędnych do rozpoczęcia programowania w języku `gawk`. Teraz pora przyjrzeć się instrukcjom strukturalnym.

Jeśli masz już jakieś doświadczenie w programowaniu, instrukcje te na pewno wydadzą Ci się znajome. Nawet jeśli tak nie jest, powinieneś bez problemu zrozumieć zasady rządzące programowaniem języku `gawk`, ponieważ jest on bardzo prosty i nie zawiera żadnych udziwnionych reguł składniowych. Prześledź uważnie kilka podanych niżej przykładowych programów i spróbuj na własną rękę stworzyć programy pozwalające na wykonanie jakichś nieskomplikowanych zadań.

`gawk` pozwala na zamieszczanie w skryptach komentarzy, które oznaczane są za pomocą symbolu `#`. Powinieneś używać komentarzy w swoich skryptach, ponieważ pozwalają one osobie postronnej na szybkie zorientowanie się, o co chodzi w programie, a poza tym przysłużą się również Tobie, jeśli zajrzesz do swoich programów po dłuższym czasie.

Instrukcja if

Instrukcja `if` używana jest do testowania jakiegoś warunku, a następnie wykonywania pewnych operacji w zależności od wyniku takiego testu. Jej składnia jest następująca:

```
if (wyrażenie) {polecenia} else {polecenia}
```

wyrażenie może przyjmować dwie wartości: prawda lub fałsz. Oto prosty przykład:

```
# przyklad uzycia instrukcji if
(if ($1==0)

{
print "Komorka pierwsza ma wartosc 0"
}

else {
    printf " Wartosc: %d\n", $1
})
```

Oczywiście program ten można zapisać w jednym wierszu, ale odpowiednie sformatowanie czyni go o wiele bardziej czytelnym. Łatwiej jest również wylapać ewentualne błędy.

Powyższy skrypt sprawdza, czy w pierwszym polu rekordu znajduje się wartość 0 i jeśli tak jest, wyświetla komunikat `Komorka pierwsza ma wartosc 0`. W przeciwnym przypadku wyświetlana jest wartość zapisana w pierwszym polu.

Każda z sekcji instrukcji `if` może zawierać kilka poleceń, o ile są one otoczone nawiasami klamrowymi. Sekcja `else` nie jest obowiązkowa, może więc zostać pominięta:

```
(if ($1==0)
{
print "Ta komorka ma wartosc 0"
})
```

Język `gawk`, podobnie zresztą jak wiele innych języków programowania, posiada specjalną składnię dla prostych instrukcji `if`. Jest ona krótsza, ale i mniej przejrzysta, nie polecam jej więc nowicjuszom ani ludziom dbającym o porządek w swoim kodzie. Oto przykład instrukcji `if` w dwóch wariantach:

```
#wersja ladna
(if ($1>$2) {
    print "Pierwsza kolumna jest wieksza"
}
else {
    print "Druga kolumna jest wieksza"
})

#wersja krotka
$1>$2 {
    print "Pierwsza kolumna jest wieksza"
}
{print "Druga kolumna jest wieksza"}
```

Jak widać, można pominąć słowa `if` oraz `else`. Pozostałe części struktury pozostają bez zmian: warunek, polecenia do wykonania, jeśli warunek jest spełniony, polecenia do wykonania, jeśli warunek nie jest spełniony. Taka składnia jest o wiele mniej czytelna, szczególnie jeśli nie wiesz, że jest to instrukcja `if`. Nie wszystkie wersje `gawk` obsługują tę składnię. To jeszcze jeden powód, by jej nie używać.

Pętla while

Instrukcja `while` pozwala na powtarzanie zestawu poleceń tak długo, jak długo spełniony jest jakiś warunek. Jest on sprawdzany przy każdym przejściu pętli. Jej składnia jest następująca:

```
while (wyrażenie) {
    polecenia
}
```

Przykładowo, pętla `while` może zostać użyta do obliczenia, jaką kwotą będziesz dysponował po wpłaceniu na konto danej kwoty i oczekaniu pewnego czasu (odpowiedni wzór to `wartosc=wklad(1+oprocentowanie)^lata`):

```
#obliczanie procentu skladanego
#pobiera z pliku wklad, oprocentowanie i ilosc lat
{var =1
while (var <=$3) {
    printf ("%f\n", $1*(1+$2)^var)
    var++
```

```
    }  
}
```

Jak pewno zauważyleś, skrypt inicjalizuje zmienną `var` wartością 1 jeszcze przed wejściem do pętli `while`. Bez tego zostałaby jej przypisana wartość 0. Interesujące nas zmienne odczytywane są z pliku wejściowego. Do inkrementacji licznika pętli (zmiennej `var`) używany jest przyrostek `++`, powodujący zwiększenie wartości zmiennej o 1 przy każdym przebiegu pętli.

Pętla for

Przewagą pętli `for` nad pętlą `while` jest to, że pozwala ona na inicjalizację zmiennej sterującej wewnątrz pętli (zmienna ta jest inicjalizowana oczywiście tylko przy pierwszym przebiegu pętli). Oto jej składnia:

```
for (inicjalizacja; warunek; inkrementacja) {  
    polecenia  
}
```

inicjalizacja odbywa się tylko przy pierwszym przejściu pętli. inkrementacja odbywa się przy każdym przebiegu. Zwykle inkrementacja sprowadza się do zwiększenia jakiegoś rodzaju licznika pętli, ale można w to miejsce wstawić dowolne polecenie. Poniżej zamieszczono przykład działający tak samo, jak przykład z poprzedniego podrozdziału, ale oparty na pętli `for`.

```
#obliczanie procentu skladanego  
#pobiera z pliku wklad, oprocentowanie i ilosc lat  
{ for (var=1; var <=$3 ; var++) {  
    printf ("%f\n", $1*(1+$2)^var)  
}  
}
```

Zmienna sterująca `var` zostaje przed wejściem do pętli zainicjalizowana wartością 1. Pętla działa tak długo, jak długo prawdziwy jest warunek `var <= $3`. Po każdym przebiegu pętli wartość zmiennej `var` jest zwiększana o 1.

Składnia tej pętli może wyglądać na nieco zagmatwaną, szczególnie jeśli nie miałeś wcześniej do czynienia z innymi językami programowania, ale jest po prostu kopią składni używanej w języku C.

next oraz exit

Polecenie `next` wymusza przejście do następnego rekordu, bez względu na wykonywane instrukcje. Na przykład w skrypcie:

```
{      polecenie1  
      polecenie2  
      polecenie3  
      next  
      polecenie4  
}
```

po wykonaniu polecenia `polecenie3` zawsze nastąpi przejście do następnego rekordu (i wykonanie polecenia `polecenie1`), co oznacza, że instrukcja `polecenie4` nie zostanie nigdy wykonana.

Polecenie `next` zwykle używane jest wewnętrz pętli, aby wymusić jej opuszczenie po wystąpieniu jakiegoś warunku.

Polecenie `exit` powoduje, że `gawk` zachowuje się tak, jakby osiągnął koniec pliku, przehodząc do wykonania bloku `END`, jeśli taki istnieje, lub kończąc działanie. Jest ono użycieczne w przypadku, gdy wewnętrz skryptu wykryte zostaną nieprawidłowe dane i trzeba przerwać działanie programu.

Tablice

Jezyk `gawk` obsługuje również tablice. Ich tworzenie i używanie nie wymaga specjalnych zabiegów, traktowane są podobnie jak wszystkie inne zmienne. Dostęp do elementu tablicy możliwy jest za pomocą indeksowania, na przykład:

```
var[numer]=7
```

Za przykład zastosowania tablicy niech posłuży skrypt, który umożliwia stworzenie pliku wyjściowego, w którym wiersze występują w odwrotnym porządku niż w pliku wejściowym:

```
#odwracanie kolejnosci wierszy
{ wiersze[NR] = $0 } #zapamietaj kazdy wiersz
END {licznik=NR      #wypisanie w odwrotnym porzadku
      while (licznik > 0) {
          print wiersze[licznik]
          licznik--
      }
}
```

W tym prostym programie (spróbuj zrobić to samo w jakimkolwiek innym języku programowania, a przekonasz się, jak wydajny jest `gawk`) użyliśmy zmiennej wewnętrznej `NR`, oznaczającej liczbę przeczytanych wierszy. Zmiennej tej używamy do indeksowania tablicy `wiersze[]`, zawierającej poszczególne wiersze pliku wejściowego. Po wczytaniu pliku zawartość tablicy `wiersze[]` jest wyświetlana zaczynając od ostatniego elementu. Nie trzeba deklarować tablicy ani martwić się o przydzielenie pamięci. Jest to znaczące ułatwienie przy pisaniu programów zorientowanych na pracę z plikami.

Podsumowanie

W tym rozdziale poruszyliśmy bardzo niewiele z rozlicznych zastosowań języka `gawk`, ale na pewno przekonałeś się, że jest to język wyjątkowo prosty. Jest on idealny do pisania krótkich programów, na przykład skryptów służących do obliczania rozmiarów

plików i katalogów. W języku C wymagałoby to całkiem sporej ilości kodu, a w języku `gawk` wystarcza kilka wierszy.

Jeśli jesteś administratorem systemu, na pewno stwierdzisz wkrótce, że `gawk` jest doskonałym uzupełnieniem innych narzędzi, głównie ze względu na to, że potrafi obsłużyć mechanizmy przekierowania danych, oraz że może z powodzeniem wyręczyć Cię w większości codziennych obowiązków administratora systemu. Jeśli chcesz dowiedzieć się o nim czegoś więcej, zatrzymaj na strony `man` albo poszukaj dobrej książki na jego temat.

Dalszych informacji o programowaniu możesz szukać w następnych rozdziałach.

Język C dla systemu Linux opisany jest w rozdziale 26. „Programowanie w języku C”.

Perl, inny bardzo poręczny język programowania, który dostarczany jest razem z Linuxem, przedstawiony jest w rozdziale 28. „Perl”.

`Tcl` oraz `Tk`, języki programowania umożliwiające korzystanie z interfejsu X oraz Motif, omówione są w rozdziale 29. „Wstęp do Tcl i Tk”.

Rozdział 26.

Programowanie w języku C

Rick McMullin

W tym rozdziale:

- υ Język C
- υ Kompilator GNU C
- υ Wyszukiwanie błędów – debugger `gdb`
- υ Inne narzędzia dla programistów

Linux rozprowadzany jest wraz z wieloma programami narzędziowymi wspierającymi tworzenie oprogramowania. Większość z nich przeznaczona jest do współpracy z kompilatorami języka C i C++. Niektóre z tych programów – takie jak debugery (programy służące do wyszukiwania błędów), programy formatujące kod źródłowy czy programy do automatycznego generowania plików nagłówkowych – zostaną omówione w tym rozdziale. Jego celem nie jest nauka programowania w języku C, a tylko przedstawienie zasad używania kompilatora i innych narzędzi przeznaczonych dla programistów.

Język C

C to język programowania przeznaczony do zastosowań ogólnych, stworzony we wczesnej fazie rozwoju systemu UNIX. Pierwsze wersje tego systemu pisane były w assemblerze oraz języku nazywanym B. Język C zaprojektowany został w celu ominienia niektórych ograniczeń języka B. Od tamtej pory stał się najpopularniejszym językiem używanym w świecie komputerów.

Skąd wynika jego popularność? Oto kilka najważniejszych powodów.

- υ Jest to język przenośny. Praktycznie dla każdego komputera stworzony został przynajmniej jeden kompilator tego języka, a standaryzacja składni oraz bibliotek znacznie ułatwia przenoszenie programów pomiędzy różnymi platformami sprzętowymi, co jest bardzo ważne dla programistów.

- υ Programy pisane w języku C są szybkie.
- υ C jest językiem systemowym we wszystkich wersjach UNIX-a.

Język C ewoluował dość znacząco w ciągu ostatnich 20 lat. W drugiej połowie lat osiemdziesiątych Amerykański Instytut Standardów (American National Standard Institute) opracował normy obowiązujące w tym języku; standard ten jest dziś znany jako ANSI C. To jeszcze bardziej poprawiło przenośność C. Również w tym okresie do języka C wprowadzono obsługę obiektów, co zaowocowało powstaniem języka C++ (który zostanie omówiony w następnym rozdziale).

Kompilator języka C dla systemu Linux (ang. *GNU C compiler*, w skrócie GCC) został stworzony w ramach Free Software Foundation i rozpowszechniany jest zgodnie z zasadami licencji GNU. Znajdziesz go na dysku CD-ROM dołączonym do tej książki.

Kompilator GNU C

Kompilator GNU C (GCC) jest w pełni funkcjonalnym kompilatorem, spełniającym normy standardu ANSI C. Jeśli potrafisz obsługiwać jakiś inny kompilator języka C, będziesz w stanie nauczyć się obsługi kompilatora `gcc` w bardzo krótkim czasie. Ten podrozdział opisuje pokrótkę używanie tego kompilatora oraz przedstawia najczęściej wykorzystywane opcje.

Uruchamianie GCC

Kompilator GCC wywołuje się podając w wierszu poleceń cały szereg opcji oraz nazw plików. Najprościej składnię jego można zdefiniować następująco:

```
gcc [opcje] [nowy_plików]
```

Każdy z plików poddawany jest przetwarzaniu zgodnie z informacjami podanymi za pomocą opcji. Najważniejsze z nich opisane są w następnym podrozdziale.

Opcje kompilatora GCC

Opcji, które mogą zostać przesłane do GCC, jest ponad setka. Większości z nich prawdopodobnie nigdy nie użyjesz, ale niektóre są wręcz niezbędne nawet przy podstawowych zastosowaniach. Wiele opcji może składać się z więcej niż jednego znaku. Z tego powodu każda z nich musi zaczynać się od własnego myślnika, nie można grupować opcji po kilka za jednym myślnikiem, jak ma to miejsce w większości poleceń systemu Linux. Przykładowo, dwa poniższe polecenia nie są równoważne:

```
gcc -p -g test.c  
gcc -pg test.c
```

Pierwsze mówi kompilatorowi, by skompilował plik `test.c`, zapisując informacje służące do późniejszego profilowania za pomocą programu `prof` oraz informacje dla debugera. Drugie polecenie nakazuje skompilowanie tego pliku i zapisanie informacji do profilowania za pomocą programu `gprof`, nie powoduje natomiast wygenerowania informacji dla debugera.

Kiedy kompilujesz program, nie używając żadnych opcji, w bieżącym katalogu tworzony jest plik wykonywalny (o ile komplikacja zakończy się bez błędów) o nazwie `a.out`. Jeśli chcesz, by nazwa tworzonego pliku wykonywalnego była inna, powinieneś użyć opcji `-o`. Przykładowo, aby skompilować program zapisany w pliku `licznik.c` do pliku wykonywalnego o nazwie `licznik`, należy wydać polecenie:

```
gcc -o licznik licznik.c
```



Nazwa pliku wyjściowego musi pojawić się bezpośrednio po opcji `-o`.

Inne opcje kompilatora decydują o tym, na jakim etapie należy zakończyć proces kompilacji. Opcja `-c` powoduje zakończenie procesu po utworzeniu pliku pośredniego (domyślnie z rozszerzeniem `.o`). Jest ona używana dość często, ponieważ umożliwia przyśpieszenie komplikacji złożonych, wieloplikowych programów.

Opcja `-s` powoduje zakończenie komplikacji po wygenerowaniu plików asemblera (domyślnie z rozszerzeniem `.s`). Opcja `-E` powoduje, że kompilator tylko wstępnie przetworzy pliki wejściowe, wykonując zawarte w nich dyrektywy preprocesora. W takim przypadku dane wyjściowe wysyłane są na ekran, a nie do pliku.

Opcje dotyczące optymalizacji

Kompilator GCC stara się utworzyć kod wynikowy w możliwie najkrótszym czasie w taki sposób, by łatwo było znaleźć w nim ewentualne błędy. Oznacza to, że kolejność operacji pozostaje taka sama jak w pliku źródłowym i nie jest dokonywana żadna optymalizacja. Istnieje wiele opcji, dzięki którym możesz nakazać tworzenie mniejszych czy szybszych wersji programów, kosztem czasu ich komplikacji oraz łatwości wyszukiwania błędów. Najczęściej używa się opcji `-O` oraz `-O2`.

Opcja `-O` powoduje zastosowanie podstawowych technik optymalizacyjnych. Owocuje to zwykle powstaniem szybciej działających wersji programów. Opcja `-O2` powoduje, że wygenerowany zostanie możliwie krótki oraz szybki kod. Czas komplikacji w tym przypadku jest dłuższy, ale za to program wynikowy działa szybciej.

Poza tymi opcjami istnieje jeszcze wiele opcji niskiego poziomu, których można użyć do dalszego przyspieszania działania programu. Są one jednak bardzo specyficzne i powinieneś używać ich tylko wtedy, gdy dokładnie zdajesz sobie sprawę z tego, co powodują, i jakie mogą być ich konsekwencje. Bardziej szczegółowe informacje o tych opcjach znajdziesz na stronach `man` (wydaj polecenie `man gcc`).

Opcje współpracy z debugerem i programem profilującym

Spośród kilku opcji dotyczących wstawiania dodatkowego kodu służącego do określania szybkości wykonywania programu i ułatwiającego uruchamianie i testowanie, najczęściej używane są dwie: `-g` oraz `-pg`.

Opcja `-g` powoduje dołączenie do pliku wykonywalnego informacji dla debugera `gdb`, który często okazuje się niezbędny w procesie wyszukiwania błędów. GCC oferuje Ci coś, czego nie daje większość innych kompilatorów: możliwość łącznego użycia opcji `-g` oraz `-O` (która powoduje wygenerowanie zoptymalizowanej wersji programu). Jest to bardzo przydatne, szczególnie jeśli chcesz testować produkt jak najbardziej zbliżony do wersji końcowej. Powinieneś jednak zdawać sobie sprawę, że część kodu zostanie przez kompilator nieco zmodyfikowana.Więcej informacji na ten temat znajdziesz w podrozdziale „Wyszukiwanie błędów – debugger `gdb`”.

Opcja `-pg` pozwala na dołączenie do programu wykonywalnego dodatkowego kodu, który, po uruchomieniu programu, wygeneruje informacje o czasie wykonania poszczególnych sekcji programu. Informacje te mogą być przeglądane za pomocą programu `gprof`. Więcej informacji na jego temat znajdziesz w podrozdziale „`gprof`”.

Wyszukiwanie błędów - debugger `gdb`

Wraz z Linuxem rozprowadzany jest program `gdb`, który jest bardzo potężnym debugerem, służącym do wyszukiwania błędów w programach napisanych w językach C i C++. Umożliwia dostęp do struktur danych i pamięci wykorzystywanej przez program podczas jego działania. Oto jego podstawowe zalety:

- υ pozwala śledzić wartości zmiennych podczas wykonywania programu,
- υ pozwala ustawić pułapki, które zatrzymują program po osiągnięciu danego wiersza kodu,
- υ pozwala wykonywać program krokowo, wiersz po wierszu.

Program `gdb` można uruchomić, wydając polecenie `gdb`. Jeśli Twój system jest prawidłowo skonfigurowany, przywita Cię on informacją podobną do tej:

```
GDB is free software and you are welcome to distribute copies of it
under certain conditions; type "show copying" to see the conditions.
There is absolutely no warranty for GDB; type "show warranty" for details.
GDB 4.12 (i486-unknown-linux), Copyright 1994 Free Software Foundation, Inc.
(gdb)
```

Przy uruchamianiu tego programu można również podać różne parametry w wierszu poleceń. Zwykle program ten uruchamia się, podając nazwę pliku wykonywalnego, w którym chcemy szukać błędów:

```
gdb <nazwa_pliku>
```

W takim przypadku plik wykonywalny zostanie automatycznie załadowany. Można również uruchomić `gdb` w ten sposób, by możliwe było oglądanie zawartości pliku ze zrzutem pamięci (ang. *core dump*) wygenerowanego przez program; można też dołączyć `gdb` do działającego już procesu. Aby obejrzeć listę dostępnych opcji z ich krótkim opisem, zajrzyj na stronę `man` lub uruchom `gdb` z opcją `-h`.

Kompilowanie kodu przeznaczonego do debugowania

Aby `gdb` mógł działać poprawnie, do pliku wykonywalnego muszą zostać dołączone przez kompilator informacje o typach i nazwach zmiennych, o mapowaniu kodu na linie programu źródłowego itd., które pozwolą na powiązanie kodu źródłowego i skompilowanego kodu programu.

Aby do tworzonego programu wykonywalnego dołączyć informacje pozwalające na współpracę z debugerem, należy uruchomić kompilator z opcją `-g`.

Podstawowe polecenia `gdb`

`gdb` obsługuje wiele różnych poleceń, od prostych, służących do ładowania pliku itp., aż do bardzo zaawansowanych, pozwalających np. obejrzeć zawartość stosu. Tabela 26.1 zawiera polecenia niezbędne do rozpoczęcia pracy z `gdb`. Opis pozostałych poleceń znajdziesz na stronach `man`.

Tabela 26.1. Podstawowe polecenia `gdb`

Polecenie	Opis
<code>file</code>	Ładuje plik wykonywalny, w którym można będzie szukać błędów.
<code>kill</code>	Kończy działanie aktualnie wykonywanego programu.
<code>list</code>	Wyświetla listę sekcji kodu źródłowego użytego do utworzenia pliku wykonywalnego.
<code>next</code>	Przechodzi do kolejnego wiersza kodu w bieżącej funkcji, bez wchodzenia do funkcji podrzędnych.
<code>step</code>	Przechodzi do kolejnego wiersza kodu w bieżącej funkcji, wchodząc również do funkcji podrzędnych.
<code>run</code>	Powoduje wykonanie bieżącego programu.
<code>quit</code>	Kończy pracę debugera <code>gdb</code> .
<code>watch</code>	Pozwala na sprawdzenie wartości zmiennej po każdej jej zmianie.
<code>break</code>	Ustawia pułapkę w kodzie źródłowym. Pułapka powoduje wstrzymanie wykonywania programu po dojściu do zadanego punktu.
<code>make</code>	Pozwala na przekompilowanie programu bez konieczności wychodzenia z <code>gdb</code> .
<code>shell</code>	Umożliwia wykonanie polecenia powłoki.

Środowisko `gdb` obsługuje większość udogodnień znanych z interpreterów poleceń. Można na przykład używać dokończania polecen za pomocą klawisza Tab, a jeśli polecenie nie jest jednoznaczne, po ponownym naciśnięciu tego klawisza wyświetlane zostaną wszystkie możliwości. Można również poruszać się po liście wydanych wcześniej poleceń za pomocą klawiszy kurSORA.

Przykładowa sesja `gdb`

Ten podrozdział poprowadzi Cię krok po kroku przez przykładową sesję `gdb`. Program, którym się zajmiemy, będzie nazywał się `pozdr`, a jego jedynym celem będzie wyświetlenie tekstu prostego pozdrowienia w przód i wstecz.

```
#include <stdio.h>

main()
{
    char moj_tekst[]="Witam Cie";

    druk_1(moj_tekst);
    druk_2(moj_tekst);
}

void druk_1 (char *tekst)
{
    printf ("Tekst normalnie: %s\n",tekst);
}

void druk_2 (char *tekst)
{
    char *tekst2;
    int rozm, i;
    rozm=strlen(tekst);
    tekst2=(char *)malloc(rozm+1);
    for (i=0; i<rozm; i++)
        tekst2[rozm-i]=tekst[i];
    tekst2[rozm+1]='\0';
    printf ("Tekst wstecz: %s\n",tekst2);
}
```

Aby skompilować ten program, użyj programu `gcc`, podając jako parametr nazwę pliku źródłowego. Aby plik wyjściowy nie nazywał się `a.out`, użyj opcji `-o`, na przykład tak:

```
gcc -g -o pozdr pozdr.c
```

Po uruchomieniu program wyświetla następujące informację:

```
Tekst normalnie: Witam Cie
Tekst wstecz:
```

Pierwszy wiersz jest w porządku, ale z drugim zdecydowanie jest coś nie tak. Powinien on oczywiście wyglądać następująco:

```
Tekst wstecz: eic matiW
```

Z jakiegoś powodu jednak funkcja `druk_2` nie działa prawidłowo. Spróbujmy rozwiązać ten problem za pomocą `gdb`. Najpierw trzeba go uruchomić:

```
(gdb) pozdr
```



Pamiętaj o użyciu opcji `-g` kompilatora, która spowoduje dodanie do wynikowego programu informacji pozwalających na współpracę z debugerem.

Jeśli nazwa pliku, który należy załadować, nie zostanie podana, można podać ją później, wydając polecenie

```
(gdb) file pozdr
```

Spowoduje ono załadowanie pliku wykonywalnego, tak jakby jego nazwa została podana jako argument wywołania.

Spróbujmy teraz uruchomić załadowany program polecienniem `run`. Na ekranie powinny pojawić się następujące komunikaty:

```
(gdb) run
Starting program: /root/pozdr
Tekst normalnie: Witam Cie
Tekst wstecz:
Program exited with code 041
```

Wyniki działania programu nie zmieniły się. Pozostaje pytanie: dlaczego nie działa funkcja `druk_2`? Aby na nie odpowiedzieć, spróbujmy ustawić pułapkę w wierszu następującym po instrukcji `for` w funkcji `druk_2`. W tym celu obejrzyjmy najpierw kod źródłowy, wydając trzykrotnie polecenie `list`:

```
(gdb) list
(gdb) list
```



Wciśnięcie klawisza Enter w pustym wierszu powoduje powtórzenie ostatniego polecenia.

```
(gdb) list
```

Po pierwszym wprowadzeniu polecenia `list` na ekranie pojawi się coś takiego:

```
1      #include <stdio.h>
2
3      main()
4      {
5          char moj_tekst[]="Witam Cie";
6
7          druk_1(moj_tekst);
8          druk_2(moj_tekst);
9      }
10
```

Wciśnij Enter, a `gdb` ponownie wykona polecenie `list`:

```

11     void druk_1 (char *tekst)
12     {
13         printf ("Tekst normalnie: %s\n",tekst);
14     }
15
16     void druk_2 (char *tekst)
17     {
18         char *tekst2;
19         int rozm, i;
20

```

A po ponownym wciśnięciu klawisza Enter:

```

21         rozm=strlen(tekst);
22         tekst2=(char *)malloc(rozm+1);
23         for (i=0; i<rozm; i++)
24             tekst2[rozm-i]=tekst[i];
25         tekst2[rozm+1]='\0';
26         printf ("Tekst wstecz: %s\n",tekst2);
27     }

```

Po obejrzeniu kodu źródłowego możemy stwierdzić, że pułapkę należy założyć w wierszu o numerze 24. W tym celu trzeba wydać polecenie:

```
(gdb) break 24
```

`gdb` powinien odpowiedzieć w następujący sposób:

```
Breakpoint 1 at 0x139: file pozdr.c, line 24
(gdb)
```

Spróbujmy ponownie uruchomić program za pomocą polecenia `run`. Wygenerowane zostaną następujące komunikaty:

```
Starting program: /root/pozdr
Tekst normalnie: Witam Cie
```

```
Breakpoint 1, druk_2 (string = 0xbffffdc4 "Witam Cie") at pozdr.c : 24
24         tekst2[rozm-i]=tekst[i];
```

W którym miejscu tkwi błąd, możemy stwierdzić po sprawdzeniu, jaką wartość przyjmuje wyrażenie `tekst2[rozm-i]`. Można to zrobić za pomocą polecenia `watch`:

```
(gdb) watch tekst2[rozm-i]
```

`gdb` potwierdzi przyjęcie polecenia krótkim komunikatem:

```
Watchpoint 2: tekst2[rozm-i]
```

Wersja debugera dostarczona na dołączonej do tej książki płycie CD-ROM wyświetla nieco inny komunikat – zamiast terminu `Watchpoint` używane jest określenie `Hardware Watchpoint`, ale oba te komunikaty mają dokładnie takie samo znaczenie. Możemy teraz przeledzić wykonanie pętli `for` krok po kroku, używając polecenia

```
(gdb) next
```

Po pierwszym przebiegu pętli `gdb` poinformuje nas, że aktualną wartością obserwowanego wyrażenia `tekst2[rozm-i]` jest '`w`', wyświetlając informacje:

```
Watchpoint 2, tekst2[rozм-i]
Old value = 0 '\000'
New value = 087 'w'
druk_2(string = 0xbffffdc4 "Witam Cie") at pozdr.c:23
23 for (i=0;i<rozм;i++)
```

Jest to wartość, której się spodziewaliśmy. Kolejne przejścia pętli również działają bez zarzutu. Po dotarciu do punktu, gdy zmienna `i` przyjmuje wartość 10, wyrażenie `tekst2[rozм-i]` przyjmuje wartość '`e`', a wartość `rozм-i` jest równa 1; program doszedł do ostatniego znaku, który należy skopiować do nowej tablicy.

Po kolejnym przebiegu pętli jasne staje się, że nigdy nie następuje przypisanie do zmiennej `tekst2[0]`, będącej pierwszym znakiem tekstu. Ponieważ domyślnie funkcja `malloc` inicjalizuje pamięć, którą przydziela, wartością `NULL`, pierwszy znak w przydzielonym obszarze ma kod 0. Kod ten oznacza koniec tekstu, więc nic nie jest wyświetlane. Problem został znaleziony.

Teraz już łatwo jest go naprawić. Trzeba zmienić algorytm tak, aby pierwszy znak kopowany do łańcucha `tekst2` był zapisywany z przesunięciem `rozм-1`, a nie `rozм`, ponieważ chociaż długość tekstu `Witam Cie` wynosi 9, numerowanie rozpoczyna się od 0.

Powyzszy program może zostać poprawiony na wiele sposobów. Można na przykład utworzyć oddzielną zmienną określającą długość tekstu, która będzie miała wartość o 1 mniejszą od rzeczywistej długości. Poniżej przedstawiamy program oparty na takim rozwiązaniu:

```
#include <stdio.h>

main()
{
    char moj_tekst[]="Witam Cie";

    druk_1(moj_tekst);
    druk_2(moj_tekst);
}

void druk_1 (char *tekst)
{
    printf ("Tekst normalnie: %s\n",tekst);
}

void druk_2 (char *tekst)
{
    char *tekst2;
    int rozм, rozм2, i;

    rozм=strlen(tekst);
    rozм2=rozм-1;
    tekst2=(char *)malloc(rozм+1);
    for (i=0; i<rozм; i++)
        tekst2[rozм2-i]=tekst[i];
    tekst2[rozм]='\0';
    printf ("Tekst wstecz: %s\n",tekst2);
}
```

Inne narzędzia dla programistów

W skład dystrybucji Linuxa wchodzą również inne narzędzia mające służyć programistom, które nie zostały jeszcze opisane. Poniżej znaleźć możesz krótkie opisy najpopularniejszych z nich.

xxgdb

`xxgdb` to interfejs graficzny programu `gdb`. Dziedziczy więc wszystkie cechy wersji obsługiwanej z wiersza poleceń. Pozwala wykonywać większość podstawowych operacji poprzez wybór odpowiednich przycisków zamiast wydawania poleceń. Graficznie zaznacza punkty, w których zastawione są pułapki.

Uruchomić ten program można, wpisując w oknie terminalu polecenie `xxgdb`. Można również w wierszu poleceń przekazać wszystkie parametry, które są obsługiwane przez program `gdb`. Dostępna jest również pewna liczba opcji charakterystycznych tylko dla `xxgdb` – zostały one zebrane w tabeli 26.2.

Tabela 26.2. Opcje programu `xxgdb` dostępne w wierszu poleceń

Opcja	Opis
<code>db_name</code>	Pozwala podać nazwę debugera, którego należy użyć; domyślnie jest to <code>gdb</code>
<code>db_prompt</code>	Pozwala określić znak zachęty używany przez debugera; domyślnie – <code>gdb</code>
<code>gdbinit</code>	Określa nazwę pliku inicjalizacyjnego, zawierającego polecenia, które należy wykonywać przy uruchomieniu; domyślnie – <code>.gdbinit</code>
<code>nx</code>	Powoduje, że plik inicjalizacyjny nie będzie przetwarzany
<code>bigicon</code>	Powoduje użycie dużych ikon

Po uruchomieniu `xxgdb` na ekranie pojawia się jego okno główne – tak jak na rysunku 26.1.

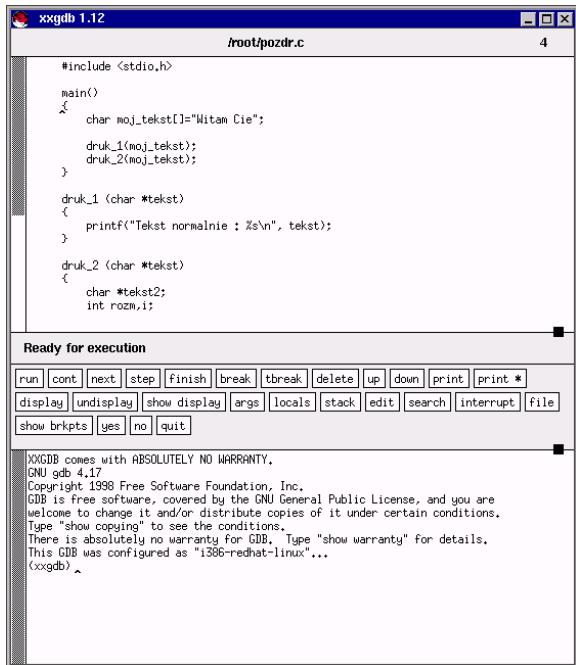
Zawiera ono komunikaty podobne do tych, które wyświetlane są w wersji tekstuowej. W środkowej części okna znajdują się przyciski, których wybranie powoduje wywołanie odpowiednich poleceń `gdb`. Góra okna zawiera kod źródłowy programu, a dolna – komunikaty generowane przez wykonywany program oraz `gdb`. Można polecić `gdb` wyświetlenie wartości jakieś zmiennej czy wyrażenia przez podświetlenie go w kodzie źródłowym i wcisnięcie przycisku `Display` w środkowej części okna.



W niektórych wersjach `xxgdb` każda z opisanych wyżej części okna otwiera się w osobnym oknie.

Więcej informacji o programie `xxgdb` możesz znaleźć na stronach `man` poświęconych programom `gdb` i `xxgdb`.

Rysunek 26.1.
Główne okno programu xxgdb



The screenshot shows the xxgdb 1.12 interface. At the top, there's a title bar with the application name and a file number (4). Below it is a code editor window containing a C program:

```
#include <stdio.h>
main()
{
    char moj_tekst[]="Witam Cie";
    druk_1(moj_tekst);
    druk_2(moj_tekst);
}
druk_1 (char *tekst)
{
    printf("Tekst normalnie : %s\n", tekst);
}
druk_2 (char *tekst)
{
    char *tekst2;
    int rozm,i;
```

Below the code editor is a toolbar labeled "Ready for execution" with various buttons like run, cont, next, step, finish, break, tbreak, delete, up, down, print, etc. At the bottom of the window is a text area displaying the GDB license and copyright information:

```
XXGDB comes with ABSOLUTELY NO WARRANTY.
GNU gdb 4.17
Copyright 1998 Free Software Foundation, Inc.
GDB is free software; delivered by the GNU General Public License, and you are
welcome to redistribute it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB. Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux"...
(gdb)
```

calls

Ten program nie jest zamieszczony na dołączonym do książki CD-ROM-ie. Możesz go znaleźć w archiwum FTP pod adresem <ftp://sunsite.berkeley.edu/pub/Linux/devel/lang/c/calls.tar.Z>. Niektóre starsze dystrybucje Linuxa rozprowadzane na płytach CD-ROM zawierają ten plik. Jeśli uważasz, że program calls będzie Ci potrzebny, powinieneś załadować go z któregoś z węzłów BBS lub FTP albo skopiować z innego dysku CD-ROM. calls uruchamia preprocesor GCC, przetwarzając pliki, których nazwy zostały podane jako argumenty wywołania, a następnie wyświetla drzewo wywołań funkcji w tych plikach.



Aby zainstalować program calls w swoim systemie, zaloguj się jako root, a następnie:

- υ rozpakuj otrzymane archiwum w katalogu /tmp;
- υ wejdź do katalogu calls utworzonego podczas rozpakowywania archiwum;
- υ przenieś plik o nazwie calls do katalogu /usr/bin;
- υ przenieś plik calls.1 do katalogu /usr/man/man1;
- υ usuń katalog /tmp/calls.

Powyższe kroki spowodują zainstalowanie programu calls i poświę-

conej mu strony man.

Oprócz drzewa wywołań funkcji program `calls` podaje też (w nawiasach kwadratowych) informacje o pliku, w którym funkcja ta się znajduje, np.:

```
main [test.c]
```

Jeśli funkcja nie znajduje się w żadnym z plików wyszczególnionych w wierszu polecień, wyświetlana jest tylko jej nazwa:

```
printf
```

`calls` informuje również o wywołaniach rekurencyjnych:

```
sil <<< recursive in silnia.c >>>
```

oraz funkcjach statycznych:

```
total [static in calculate.c]
```

Dla przykładu założmy, że za pomocą programu `calls` chcemy uzyskać informacje o pliku zawierającym następujący program:

```
#include <stdio.h>

main()
{
    char moj_tekst[]="Witam Cie";

    druk_1(moj_tekst);
    druk_2(moj_tekst);
}

void druk_1 (char *tekst)
{
    printf ("Tekst normalnie: %s\n",tekst);
}

void druk_2 (char *tekst)
{
    char *tekst2;
    int rozm, rozm2, i;

    rozm=strlen(tekst);
    rozm2=rozm-1;
    tekst2=(char *)malloc(rozm+1);
    for (i=0; i<rozm; i++)
        tekst2[rozm2-i]=tekst[i];
    tekst2[rozm]='\0';
    printf ("Tekst wstecz: %s\n",tekst2);
}
```

Po przetworzeniu tego pliku program `calls` wyświetli następujące informacje:

```
1 main [pozdr.c]
2     druk_1 [pozdr.c]
3         printf
```

```
4     druk_2 [pozdr.c]
5         strlen
6         malloc
7         printf
```

`calls` rozpoznaje również wiele opcji podawanych w wierszu poleceń, które pozwalają modyfikować postać informacji wyjściowych oraz rodzaj funkcji, które mają być wyświetlane. Więcej informacji o opcjach programu `calls` możesz otrzymać, wydając polecenie `calls -h` i – oczywiście – zaglądając na strony `man`.

cproto

Program `cproto` również nie jest zamieszczony na CD-ROM-ie dołączonym do książki. Możesz go otrzymać w każdym archiwum FTP udostępniającym Linuxa. Pozwala on na automatyczne generowanie plików nagłówkowych zawierających deklaracje wszystkich zdefiniowanych w programie funkcji. Dzięki temu można zaoszczędzić sobie ręcznego ich wpisywania.



Aby zainstalować program `cproto` w swoim systemie, zaloguj się jako `root`, a następnie:

- υ rozpakuj otrzymane archiwum w katalogu `/tmp`;
- υ wejdź do katalogu `cproto` utworzonego podczas rozpakowywania plików;
- υ przenieś plik o nazwie `cproto` do katalogu `/usr/bin`;
- υ przenieś plik `cproto.1` do katalogu `/usr/man/man1`;
- υ usuń katalog `/tmp/cproto`.

Powyższe kroki spowodują zainstalowanie programu `cproto` i poświęconej mu strony `man`.

Spróbujmy zastosować program `cproto` do wygenerowania nagłówków funkcji zdefiniowanych w przykładowym pliku:

```
#include <stdio.h>

main()
{
    char moj_tekst[]="Witam Cie";

    druk_1(moj_tekst);
    druk_2(moj_tekst);
}

void druk_1 (char *tekst)
{
    printf ("Tekst normalnie: %s\n",tekst);
}
```

```

void druk_2 (char *tekst)
{
    char *tekst2;
    int rozm, rozm2, i;

    rozm=strlen(tekst);
    rozm2=rozm-1;
    tekst2=(char *)malloc(rozm+1);
    for (i=0; i<rozm; i++)
        tekst2[rozm2-i]=tekst[i];
    tekst2[rozm]='\0';
    printf ("Tekst wstecz: %s\n",tekst2);
}
Program cproto wygeneruje następujące informacje:
/* test.c */
int main(void);
int druk_1(char *tekst);
int druk_2(char *tekst);

```

Mogą one zostać wykorzystane w pliku nagłówkowym, zawierającym prototypy wszystkich funkcji.

indent

Program `indent` znajduje się na dołączonej do książki płycie CD-ROM. Program ten formatuje albo drukuje w przyjemnym dla oka układzie kod źródłowy w języku C, zachowując konsekwentny układ wcięć, stały układ nawiasów, nawiasów klamrowych itd. Jego zachowanie można modyfikować, podając odpowiednie opcje w wierszu poleceń – ich opis możesz znaleźć na stronach `man` lub też po wydaniu polecenia `indent -h`.

Poniższy przykład pozwoli Ci zorientować się w działaniu tego programu.

Oto kod źródłowy przed sformatowaniem go za pomocą programu `indent`:

```

#include <stdio.h>

main() {
    char moj_tekst[]="Witam Cie";
    druk_1(moj_tekst);
    druk_2(moj_tekst);}

void druk_1 (char *tekst)
{
    printf ("Tekst normalnie: %s\n",tekst);

void druk_2          (char *tekst) {
    char *tekst2;
    int rozm, rozm2, i;

    rozm=strlen(tekst);
    rozm2=rozm-1;
    tekst2=(char *)malloc(rozm+1);
    for (i=0; i<rozm; i++)
        tekst2[rozm2-i]=tekst[i];
    tekst2[rozm]='\0';

```

```
        printf ("Tekst wstecz: %s\n",tekst2);
    }
```

A to ten sam kod po sformatowaniu:

```
#include <stdio.h>

main ()
{
    char moj_tekst[]="Witam Cie";
    druk_1(moj_tekst);
    druk_2(moj_tekst);
}

void druk_1 (char *tekst)
{
    printf ("Tekst normalnie: %s\n",tekst);
}

void druk_2 (char *tekst)
{
    char *tekst2;
    int rozm, rozm2, i;

    rozm=strlen(tekst);
    rozm2=rozm-1;
    tekst2=(char *)malloc(rozm+1);
    for (i=0; i<rozm; i++)
        tekst2[rozm2-i]=tekst[i];
    tekst2[rozm]='\0';
    printf ("Tekst wstecz: %s\n",tekst2);
}
```

Program indent nie zmienia kodu w sensie składniowym – kompilator zrozumie go dokładnie w ten sam sposób. Zmienia tylko wygląd kodu źródłowego, dzięki czemu staje się on bardziej czytelny, co zawsze jest godne uwagi.

gprof

Program `gprof` instalowany jest w katalogu `/usr/bin`. Pozwala on na profilowanie tworzonych programów, tzn. ustalanie, na czym spędzają one najwięcej czasu.

`gprof` podaje informacje o tym, ile razy wywołana została każda funkcja, oraz ile procent ogólnego czasu wykonania programu zajęła. Dzięki temu można łatwo stwierdzić, która część kodu programu wymaga usprawnienia.

Aby można było używać programu `gprof`, tworzony program musi zostać skompilowany z załączoną opcją `-pg` kompilatora. Dołączony do programu kod spowoduje utworzenie pliku o nazwie `gmon.out`. Ten plik jest następnie przetwarzany przez program `gprof`, który przedstawia zawarte w nim informacje w czytelny sposób.

Po uruchomieniu testowanego programu (i utworzeniu pliku `gmon.out`) należy uruchomić program `gprof` podając nazwę pliku programu jako parametr:

```
gprof <nazwa_programu>
```



Dane wyświetlane przez program `gprof` są dość sporych rozmiarów, dlatego warto skierować je do pliku.

f2c oraz p2c

Programy `f2c` i `p2c` służą do konwersji kodu źródłowego – `f2c` zamienia kod źródłowy w języku FORTRAN na kod w języku C, natomiast `p2c` – kod napisany w Pascalu na C. Oba te programy instalowane są automatycznie przy instalacji kompilatora GCC.

Jeśli masz kod źródłowy programu napisanego w języku FORTRAN albo Pascal, a chcesz przepisać go w języku C, programy te będą Ci bardzo pomocne. Potrafią one wygenerować kod w języku C, który może być zwykle skompilowany za pomocą kompilatora `gcc` bez interwencji programisty.

Przy konwertowaniu niewielkich, prostych programów, kod wynikowy powinien nadawać się bezpośrednio do skompilowania. W przypadku programów bardziej skomplikowanych, składających się z kilku plików, nie obejdzie się bez podania dodatkowych opcji.

Aby przetłumaczyć na C program w języku FORTRAN, powinieneś wydać polecenie

```
f2c pr_fortran.f
```



`f2c` wymaga, aby plik zawierający kod źródłowy konwertowanego programu posiadał rozszerzenie `.f` albo `.F`.

Konwersji programu w Pascalu dokonać możesz poprzez wydanie polecenia:

```
p2c pr_pascal.pas
```

Programy `p2c` i `f2c` tworzą plik zawierający kod źródłowy w języku C, o takiej samej nazwie, jak plik zawierający konwertowany program, ale z rozszerzeniem `.c` zamiast `.f` czy `.pas`.

Więcej informacji o specyficznych opcjach dostępnych w tych programach znajdziesz na stronach `man`.

Podsumowanie

W rozdziale tym omówiliśmy pokrótkę zasady użycia kompilatora GNU C, wraz z częściej używanymi opcjami. Przedstawiliśmy również podstawowe zasady posługiwania się programami `gdb` i `xxgdb` oraz wspomnialiśmy o kilku innych narzędziach ułatwiających pracę programisty.



Jeśli będziesz pisał programy w języku C, czas, który przeznaczysz na naukę obsługi `gdb` i innych programów narzędziowych zwróci się wielokrotnie podczas wyszukiwania usterek w programach.

W następnym rozdziale omówimy podobne tematy, skupiając się jednak na języku C++. Jeśli chcesz, możesz przejść do innych rozdziałów.

Rozdział 28. „Perl” dotyczy języka Perl, doskonale nadającego się do tworzenia skryptów.

`Tcl` i `Tk`, języki używane głównie do tworzenia makropoleceń, przedstawione są w rozdziale 29. „Podstawy języków Tcl i Tk”.

Inne kompilatory dostępne w systemie Linux opisane są w rozdziale 30. „Inne kompilatory”.

O języku Smalltalk przeczytać możesz w rozdziale 31. „Smalltalk/X”.

Rozdział 27.

Programowanie w C++

Rick McMullin

W tym rozdziale:

- υ Język C++
- υ Klasy i metody
- υ Opcje kompilatora GCC
- υ Opcje współpracy z debuggerem i programem profilującym
- υ Wyszukiwanie błędów w aplikacjach C++
- υ Wyszukiwanie błędów w funkcjach wirtualnych
- υ Wyszukiwanie błędów w funkcjach obsługi wyjątków
- υ Polecenia `gdb` specyficzne dla C++
- υ Biblioteki klas GNU C++

Język C++

C++ to wersja języka C rozszerzona o obsługę obiektów. Język ten opracowany został w Bell Labs na początku lat osiemdziesiątych i szybko zyskał sobie uznanie i popularność wśród programistów. Na rynku istnieje obecnie kilkanaście różnych kompilatorów tego języka. Najpopularniejsze z kompilatorów przeznaczonych dla platformy PC to Borland C++, Microsoft Visual C++, Zortech C++ i Watcom C++. Przeznaczone są one do tworzenia programów działających w systemach DOS i Windows, choć niektóre z nich potrafią również tworzyć programy pracujące pod kontrolą Windows NT i OS/2. Oczywiście istnieje również wiele kompilatorów języka C++ przeznaczonych dla innych platform sprzętowych.

W przypadku większości systemów UNIX-owych kompilatory języka C++ rozprowadzane są przez dystrybutorów systemu. Podobnie jest w przypadku Linuxa. Kompilator C++, który kiedyś nazywał się `g++`, jest bardzo blisko spokrewniony z GCC. W wersji 2.0 oba te kompilatory zostały scalone w jeden program.

Obecnie GCC to połączenie kompilatorów języków C, C++ oraz C z obiektami. W systemie nadal znajduje się plik o nazwie `g++`, ale jest on tylko skryptem wywołującym kompilator GCC z odpowiednimi opcjami.

Dlaczego C++?

C++ i programowanie obiektowe (ang. *object-oriented programming*, OOP) nie powstały od razu. Wiele lat temu, gdy na świecie panowały komputery ośmiobitowe, programiści zamiast języka maszynowego zaczęli używać asemblera, wykorzystując nieco większe możliwości obliczeniowe następców komputerów czterobitowych. Umożliwiło to przekształcenie części pracy związanej z tworzeniem programu na komputer.

Z upływem czasu moc obliczeniowa komputerów rosła i możliwe stało się pisanie coraz bardziej skomplikowanych programów. Było to jednak coraz trudniejsze. Powstawać zaczęły kompilatory języków wyższego poziomu, które brały ogrom „czarnej roboty” na siebie. Pierwsze języki były językami strukturalnymi – na przykład FORTRAN, COBOL, Pascal czy C. Strukturalizacja programów pozwalała na ich uproszczenie, zwiększenie przejrzyści, a przede wszystkim zmniejszenie liczby błędów dzięki podzieleniu zadania na mniejsze problemy (funkcje i procedury), łatwiejsze do rozwiązywania.

Programowanie strukturalne dobrze spełniało swoją funkcję, ale znów tylko do pewnego czasu. W miarę wzrostu rozmiarów aplikacji zaczęło zawodzić. Programowanie zorientowane obiektowo powstało jako odpowiedź na problemy stwarzane przez programowanie strukturalne.

Główne nowe pojęcia wprowadzane przez OOP to:

- υ hermetyzacja danych,
- υ dziedziczenie,
- υ polimorfizm.

Hermetyzacja danych

Przy programowaniu strukturalnym problemy pojawiały się najczęściej wtedy, gdy różne funkcje czy moduły programu korzystały ze wspólnych danych. Dowolny fragment programu mógł odwoływać się do danych bez poinformowania o tym innego fragmentu.

Hermetyzacja danych ma na celu zabezpieczenie przed tego typu problemami. Polega ona na zgrupowaniu wspólnych danych, zapisaniu ich w odpowiednim typie i zapewnieniu spójnego interfejsu dostępu do nich. Dzięki temu nikt nie może skorzystać z danych, pomijając ten interfejs.

Najważniejszą zaletą takiego rozwiązania jest zabezpieczenie danych przed ich nieautoryzowaną bezpośrednią modyfikacją. Często bowiem zmiana danych musi wiązać się z jakimś scisłe określonymi czynnościami. Poza tym zmiana struktury danych nie jest widziana na zewnątrz samej struktury, o ile tylko interfejs dostępu do danych pozostanie niezmieniony. Znaczco upraszcza to tworzenie i poprawianie bardziej złożonych programów.

W języku C++ hermetyzację danych zapewnia mechanizm klas.

Dziedziczenie

Dziedziczenie pozwala na ponowne użycie części kodu. Zwykle stosowane jest w przypadku, gdy jakaś część programu posiada zasadniczo wszystkie cechy posiadane przez inną część, plus kilka innych, na przykład gdy jeden obiekt jest szczególnym przypadkiem drugiego.

Dziedziczenie jest w języku C++ zaimplementowane jako dziedziczenie klas.

Polimorfizm

Polimorfizm umożliwia zdefiniowanie funkcji wykonujących różne działania w zależności od typu danych, na których operują. Moc tego mechanizmu ujawnia się, gdy informacje przesyłane są poprzez klasę bazową do klas pochodnych, i dla każdej z nich mogą one oznaczać coś innego.

Polimorfizm w języku C++ zaimplementowany jest poprzez funkcje wirtualne.

Klasy i metody

Klasy w języku C++ można w zasadzie wyobrażać sobie podobnie jak struktury w języku C, z tym, że oprócz danych mogą one zawierać również funkcje, które na tych danych mogą być wykonywane. Rozpatrzmy na przykład typ danych reprezentujący bryłę geometryczną. Bryły mogą mieć najróżniejsze kształty, ale wszystkie mają kilka wspólnych atrybutów, na przykład pole powierzchni czy objętość. Można zatem zdefiniować w języku C strukturę o nazwie `bryla`:

```
struct bryla {  
    float pole;  
    float obj;  
}
```

Jeśli do tej definicji dołączymy jeszcze kilka funkcji, otrzymamy odpowiednik klasy:

```
struct bryla {  
    float pole;  
    float obj;  
    float licz_pole();
```

```
float licz_obj();  
};
```

W ten sposób zadeklarowaliśmy klasę w języku C++. Funkcje wchodzące w skład klasy nazywa się *metodami*. Zmienną typu `bryla` nazywa się *obiektem*. Można ją zadeklarować w następujący sposób:

```
bryla kula1;
```

Obiekt jest fizyczną realizacją klasy, podobnie jak zmienna była realizacją typu.

Opcje kompilatora GCC

W tym podrozdziale opiszemy najczęściej używane opcje kompilatora GCC. Najpierw przyjrzymy się opcjom wspólnym dla języków C i C++, a następnie przejdziemy do omówienia opcji specyficznych tylko dla języka C++. Każda opcja dostępna dla języka C może być użyta dla języka C++ (choć nie zawsze ma to sens – w takim przypadku zostanie ona zignorowana).



Do kompilowania programów napisanych w języku C++ najłatwiej użyć skryptu `g++`, ponieważ ustawia on automatycznie wszystkie wymagane opcje.

Do kompilatora GCC przekazać można bardzo dużo różnego rodzaju opcji. Większość z nich uzależniona jest od konkretnej platformy sprzętowej albo służy do dokładnego „dostrajania” wygenerowanego kodu i prawdopodobnie nigdy nie będziesz musiał ich użyć. Poniżej omówimy opcje, które przydadzą Ci się w codziennej pracy.

Wiele opcji programu GCC wymaga podania więcej niż jednej litery. Z tego powodu nie jest możliwe grupowanie ich po wspólnym myślniku, jak ma to miejsce w przypadku większości programów systemu Linux. Przed każdą opcją znaleźć się musi osobny myślnik.

Skompilowanie programu bez podania żadnych opcji powoduje utworzenie pliku wykonywalnego o nazwie `a.out`. Jeśli chcesz, aby program wynikowy miał inną nazwę, powinieneś użyć opcji `-o`, na przykład jeśli chcesz, aby program wygenerowany na podstawie kodu źródłowego zapisanego w pliku o nazwie `licznik.C` (rozszerzenie `.C` oznacza, że plik zawiera program w języku C++, w przeciwieństwie do rozszerzenia `.c`, oznaczającego kod w języku C) miał nazwę `licznik`, powinieneś wydać polecenie:

```
gcc -o licznik licznik.C
```



Do opcji `-o` w wierszu poleceń nie powinno być spacji! Nazwa pliku wykonywalnego musi pojawić się bezpośrednio po opcji `-o`.

Inne opcje pozwalają decydować, na jakim etapie proces komplikacji ma zostać zakończony. Opcja `-c` powoduje pominięcie konsolidacji (konwersji plików pośrednich na plik wykonywalny), tworząc tylko skompilowane pliki pośrednie (z rozszerzeniem `.o`). Jest ona szczególnie przydatna przy tworzeniu większych programów, ponieważ pozwala uniknąć wielokrotnego komplikowania plików, które nie zostały zmodyfikowane.

Opcja `-S` powoduje zatrzymanie komplikacji po wygenerowaniu plików asemblera (z rozszerzeniem `.s`). Opcja `-E` powoduje tylko wstępne przetworzenie plików źródłowych i wykonanie dyrektyw preprocesora, wyniki przesyłając do standardowego urządzenia wyjściowego.

Opcje współpracy z debugerem i programem profilującym

Program GCC posiada również kilka opcji umożliwiających współpracę z debugerem i programem profilującym. Najczęściej używane z nich to `-gstabs+` oraz `-pg`.

Opcja `-gstabs+` powoduje dołączenie do kodu programu wykonywalnego dodatkowych informacji dla debugera `gdb`, co znacznie ułatwia wyszukiwanie usterek w programach. Proces wyszukiwania błędów w programach napisanych w języku C++ omówiony jest dokładniej w podrozdziale „Wyszukiwanie błędów w programach C++” w dalszej części tego rozdziału.

Opcja `-pg` powoduje dołączenie kodu generującego informacje o ilości czasu poświęconego na wywołanie każdej z funkcji programu, które mogą być następnie przetworzone przez program `gprof`. Jeśli chcesz dowiedzieć się czegoś więcej o tym programie, zajrzyj do podrozdziału „`gprof`” w rozdziale 26. „Programowanie w języku C”.

Opcje specyficzne dla języka C++

Opcje pozwalające na modyfikację sposobu komplikacji programów w języku C++ zebrane zostały w tabeli 27.1.

Tabela 27.1. Opcje komplikacji specyficzne dla C++

Opcja	Znaczenie
<code>-fall-virtual</code>	Traktowanie wszystkich możliwych funkcji składowych jako wirtualnych (wszystkich prócz konstruktorów oraz przeciążonych operatorów <code>new</code> i <code>delete</code>)
<code>-fdollars-in-identifiers</code>	Zezwolenie na używanie znaku \$ w identyfikatorach; opcja zabraniająca jego użycia: <code>-fno-dollars-in-identifiers</code>
<code>-felide-constructors</code>	Pomijanie konstruktorów tam, gdzie to możliwe
<code>-fenum-int-equiv</code>	Zezwolenie na niejawną konwersję z typu wyliczeniowego do <code>int</code>
<code>-fexternal-templates</code>	Generowanie mniejszego kodu dla szablonów – kompilator

generuje tylko jedną kopię żądanej funkcji tam, gdzie jest ona zdefiniowana

cd. na następnej stronie

Tabela 27.1. cd. Opcje kompilacji specyficzne dla C++

Opcja	Znaczenie
-fmemorize-lookups	Użycie technik heurystycznych dla przyspieszenia kompilacji; opcja ta jest domyślnie wyłączona, ale różnice są widoczne tylko przy określonym typie danych wejściowych
-fno-script-prototype	Traktowanie deklaracji funkcji bez argumentów w ten sam sposób, jak w języku C (czyli akceptowanie dowolnej liczby argumentów)
-fno-null-objects	Założenie, że obiekty, do których odnoszą się referencje, są zainicjalizowane
-fsave-memorized	To samo co -fmemorize-lookups
-fthis-is-variable	Zezwolenie na przypisanie do wskaźnika <code>this</code>
-nostdinc++	Pominiecie wyszukiwania plików nagłówkowych w standardowych katalogach C++
-traditional	To samo co -fthis-is-variable
-fno-default-inline	Funkcje zdefiniowane w ciele klasy nie są traktowane jako <code>inline</code>
-wenum-clash	Generowanie ostrzeżeń przy konwersji pomiędzy typami wyliczeniowymi
-woverloaded-virtual	Generowanie ostrzeżeń, gdy klasa pochodna może być zdefiniowana błędnie przez zadeklarowanie funkcji wirtualnych; funkcja wirtualna w klasie pochodnej musi być wirtualna również w klasie bazowej
-wtemplate-debugging	Generowanie ostrzeżeń w przypadku, gdy niemożliwe będzie użycie debugera z powodu stosowania szablonów
+eN	Kontrola sposobu, w jaki używane będą funkcje wirtualne
-gstabs+	Dodanie do kodu programu dodatkowych informacji, zrozumiałych tylko dla debugera GNU; dzięki nim może on prawidłowo obsługiwać programy w języku C++

Wyszukiwanie błędów w aplikacjach C++

Możliwość wyszukiwania błędów powstających podczas pisania programu to bardzo ważna część procesu tworzenia aplikacji. Debugger opisany w rozdziale 26. może również zostać użyty dla programów napisanych w języku C++. W tym podrozdziale opisane są najważniejsze różnice pomiędzy wyszukiwaniem błędów w programach napisanych w językach C i C++.

Podstawowe polecenia debugera `gdb`, przedstawione już raz w rozdziale 26., dla wygody podane są również w tabeli 27.2.

Tabela 27.2. Podstawowe polecenia `gdb`

Polecenie	Opis
<code>file</code>	Laduje plik wykonywalny, w którym można będzie szukać błędów.
<code>kill</code>	Kończy działanie aktualnie wykonywanego programu.
<code>list</code>	Wyświetla listę sekcji kodu źródłowego użytego do utworzenia pliku wykonywalnego.
<code>next</code>	Przechodzi do kolejnego wiersza kodu w bieżącej funkcji, bez wchodzenia do funkcji podrzędnych.
<code>step</code>	Przechodzi do kolejnego wiersza kodu w bieżącej funkcji, wchodząc również do funkcji podrzędnych.
<code>run</code>	Powoduje wykonanie załadowanego programu.
<code>quit</code>	Kończy pracę debugera <code>gdb</code> .
<code>watch</code>	Pozwala na sprawdzenie wartości zmiennej po każdej jej zmianie.
<code>break</code>	Ustawia pułapkę w kodzie źródłowym. Pułapka powoduje wstrzymanie wykonywania programu po dojściu do zadanego punktu.
<code>make</code>	Pozwala na przekompilowanie programu bez konieczności wychodzenia z <code>gdb</code> .
<code>shell</code>	Umożliwia wykonanie polecenia powłoki.

Z punktu widzenia programisty, proces wyszukiwania usterek w programach napisanych w języku C++ jest bardziej złożony, niż w przypadku programów w języku C. Główną przyczyną takiego stanu rzeczy jest obsługa funkcji wirtualnych oraz wyjątków. Debugger `gdb` potrafi poradzić sobie z tymi mechanizmami.

Wyszukiwanie błędów w funkcjach wirtualnych

Język C++ implementuje polimorfizm za pomocą funkcji wirtualnych. Dzięki nim istnieć może więcej niż jedna funkcja o określonej nazwie. Jednym sposobem na ich rozróżnienie jest określenie typów argumentów, na których funkcje te operują. Określenie typów stanowi sygnaturę funkcji. Przykładowo, funkcja o prototypie

```
void func(int x, real f)
```

posiada sygnaturę `int, real`. Mechanizm funkcji wirtualnych stwarza pewien problem: jeśli na przykład zdefiniowano kilka funkcji o nazwie `licz`, jak ustawić pułapkę wywoływaną przy odwołaniu do jednej z tych funkcji? Przy wyszukiwaniu błędów w programach

napisanych w języku C można było to zrobić przez podanie po poleceniu `break` nazwy funkcji, np.:

```
(gdb) break licz
```

Ta metoda nie działa w przypadku funkcji wirtualnych, ponieważ nie wiadomo, o którą funkcję `licz` chodzi. Debugger `gdb` potrafi poradzić sobie z takim problemem na kilka sposobów. Pierwszym z nich jest podanie całego nagłówka funkcji, na przykład tak:

```
(gdb) break 'licz (float)'
```

Teraz `gdb` ma już dość informacji, by wiedzieć, którą funkcję miałeś na myśli. Drugie rozwiązanie to wybranie interesującej nas funkcji z menu wyświetlanego przez `gdb`, jeśli ma on wątpliwości co do tego, o którą funkcję chodzi. Wybranie pierwszej pozycji menu powoduje anulowanie polecenia. Druga pozycja umożliwia ustawienie pułapki we wszystkich funkcjach o danej nazwie. Pozostałe pozycje odnoszą się kolejno do wszystkich funkcji o zadanej nazwie. Spójrzmy na przykład:

```
(gdb) break bryla::licz
[0] cancel
[1] all
[2] file: bryla.C line number: 153
[3] file: bryla.C line number: 207
[4] file: bryla.C line number: 247
>2 3
Breakpoint 1 at 0xb234: file: bryla.C line 153
Breakpoint 2 at 0xa435: file: bryla.C line 207
Multiple breakpoints were set
Use the "delete" command to delete unwanted breakpoints
(gdb)
```

Wyszukiwanie błędów w funkcjach obsługi wyjątków

Wyjątki (ang. *exceptions*) to błędy występujące podczas działania programu. W C++ można tworzyć własne funkcje służące do ich obsługi. Przykładowo, jeśli pisałeś program w C i używałeś funkcji `malloc` do przydzielenia bloku pamięci, za każdym razem musiałeś sprawdzić zwracaną przez nią wartość, aby upewnić się, że wszystko przebiegło pomyślnie. Gdyby język C posiadał mechanizm obsługi wyjątków, wystarczyłoby napisać jedną funkcję obsługi wyjątku zgłoszanego przez funkcję `malloc`.

W `gdb` dodano dwa polecenia pozwalające na wyszukiwania błędów w funkcjach obsługi wyjątków: polecenie `catch`, które ustawia pułapkę w aktywnej funkcji obsługi wyjątku, oraz `catch info`, które wyświetla informacje o wszystkich funkcjach obsługi wyjątków. Składnia polecenia `catch` jest następująca:

```
catch wyjątki
```

gdzie `wyjątki` to lista wyjątków, które należy przechwycić.

Polecenia gdb specyficzne dla C++

Oprócz poleceń `catch` i `catch info`, do obsługi programów napisanych w języku C++ dodano również nowe opcje polecień `set` i `show`. Zostały one zebrane w tabeli 27.3.

Tabela 27.3. Opcje polecień `set` i `show` specyficzne dla języka C++

Polecenie	Opis
<code>set print demangle</code>	Wyświetlanie nazw funkcji tak, jak pojawiają się one w kodzie źródłowym, a nie tak, jak przekazywane są do asemblera
<code>show print demangle</code>	Wyświetlanie informacji o tym, czy opcja <code>print demangle</code> jest aktywna
<code>set demangle-style</code>	Definicja trybu wyświetlania nazw; dostępne wartości: <code>auto</code> , <code>gnu</code> , <code>lucid</code> i <code>arm</code>
<code>show demangle-style</code>	Wyświetlanie informacji o aktualnym trybie wyświetlania nazw
<code>set print object</code>	Wyświetlanie aktualnego typu obiektu podczas wyświetlania wskaźnika do niego
<code>show print object</code>	Wyświetlanie informacji o tym, czy opcja <code>print object</code> jest aktywna
<code>set print vtbl</code>	Wyświetlanie tablicy funkcji wirtualnych
<code>show print vtbl</code>	Wyświetlanie informacji o tym, czy opcja <code>print vtbl</code> jest aktywna

Biblioteki klas GNU C++

Pakiet GNU C++ zawiera również całkiem pokaźną bibliotekę klas. Jest to zestaw dość uniwersalnych klas, które mogą zostać użyte w różnych programach. Przykładami mogą być klasy do obsługi baz danych, graficznego interfejsu użytkownika czy implementujące abstrakcyjne struktury danych.

Na rynku istnieją również inne biblioteki klas obsługujących graficzny interfejs użytkownika, np. Microsoft Foundation Classes czy Borland's Object Windows Library, ale przeznaczone są one tylko dla platformy Windows.

W tym podrozdziale opiszemy niektóre możliwości oferowane przez biblioteki klas GNU C++.

Strumienie

Biblioteka wejścia / wyjścia `libio` jest implementacją strumieni obsługujących urządzenia standardowego wejścia i wyjścia dla programów opartych na interfejsie tekstowym. Jest ona funkcjonalnie niemal identyczna z bibliotekami dołączanymi do innych kompi-

latorów. Jej główną częścią jest obsługa strumieni wejścia, wyjścia oraz błędów. Odpowiadają one strumieniom znanym z języka C i nazywają się odpowiednio `cin`, `cout` i `cerr`. Dane można do nich wysyłać za pomocą operatora `<<`, a odczytywać operatorem `>>`.

Poniższy program pokazuje, jak można wykorzystać bibliotekę wejścia / wyjścia.

```
#include <iostream.h>
int main ()
{
    char name[10];
    cout << "Podaj imię.\n";
    cin >> name;
    cout << "Witaj " << name << ", jak leci?\n";
}
```

Łańcuchy znaków

Klasa `String` zastępuje znane z języka C tablice znaków zakończone zerem. Daje programistom nowe możliwości, na przykład pozwala na używanie operatorów i funkcji. W tabeli 27.4 zebrane są operatory zdefiniowane dla obiektów klasy `String`.

Tabela 27.4. Operatory zdefiniowane dla obiektów klasy `String`

Operator	Znaczenie
<code>str1==str2</code>	Zwraca logiczną prawdę, jeśli tekst reprezentowany przez obiekt <code>str1</code> jest taki sam, jak reprezentowany przez <code>str2</code>
<code>str1!=str2</code>	Zwraca logiczną prawdę, jeśli tekst <code>str1</code> jest różny od <code>str2</code>
<code>str1<str2</code>	Zwraca logiczną prawdę, jeśli tekst <code>str1</code> jest mniejszy (alfabetycznie) od <code>str2</code>
<code>str1<=str2</code>	Zwraca logiczną prawdę, jeśli tekst <code>str1</code> jest mniejszy lub równy <code>str2</code>
<code>str1>str2</code>	Zwraca logiczną prawdę, jeśli tekst <code>str1</code> jest większy od <code>str2</code>
<code>str1>=str2</code>	Zwraca logiczną prawdę, jeśli tekst <code>str1</code> jest większy lub równy <code>str2</code>
<code>compare(str1,str2)</code>	Porównuje teksty <code>str1</code> i <code>str2</code> bez zwracania uwagi na wielkość liter
<code>str3=str1+str2</code>	Zapisuje rezultat połączenia tekstów <code>str1</code> i <code>str2</code> do <code>str3</code>

Dostępne są również inne funkcje, pozwalające na różnego typu porównania, sklejenia czy wyszukiwanie i wycinanie fragmentów tekstów.

Liczby przypadkowe

W skład biblioteki klas GNU C++ wchodzą również klasy pozwalające na generowanie różnego typu liczb przypadkowych. Są to klasy `RNG` oraz `Random`.

Analiza danych statystycznych

Biblioteka klas zawiera też dwie klasy służące do zbierania i przetwarzania danych statystycznych. Są to klasy `SampleStatistic` oraz `SampleHistogram`. Klasa `SampleStatistic` umożliwia zbieranie danych oraz obliczanie podstawowych parametrów takiego zbioru, np. wartości średniej, maksimum i minimum, wariancji, odchylenia standardego itd.

Klasa `SampleHistogram` jest pochodną klasy `SampleStatistic` i służy do wyświetlanego histogramów.

Listy

Biblioteka klas GNU C++ pozwala na użycie dwóch rodzajów list: jednokierunkowych (klasa `SList`) i dwukierunkowych (`DList`). Dla obu zaimplementowane są wszystkie podstawowe metody. Najważniejsze z nich zebrane są w tabeli 27.5.

Tabela 27.5. Metody klas obsługujących listy

Metoda	Opis
<code>lista.empty()</code>	Zwraca logiczną prawdę, jeśli lista nie zawiera żadnych elementów
<code>lista.length()</code>	Zwraca liczbę elementów listy
<code>lista.prepend(a)</code>	Umieszcza element <code>a</code> na początku listy
<code>lista.append(a)</code>	Umieszcza element <code>a</code> na końcu listy
<code>lista.join(lista2)</code>	Dołącza listę <code>list2</code> do listy <code>lista</code> , usuwając listę <code>list2</code>
<code>a=lista.front()</code>	Zwraca wskaźnik do pierwszego elementu listy
<code>a=lista.rear()</code>	Zwraca wskaźnik do ostatniego elementu listy
<code>a=lista.remove_front()</code>	Usuwa z listy pierwszy element, zwracając wskaźnik do niego
<code>lista.del_front()</code>	Usuwa i niszczy pierwszy element listy
<code>lista.clear()</code>	Usuwa całą zawartość listy
<code>lista.ins_after(i,a)</code>	Wstawia element <code>a</code> do listy po pozycji <code>i</code>
<code>lista.del_after(i)</code>	Usuwa element następny po pozycji <code>i</code>

Dodatkowe metody obsługiwane przez listę dwukierunkową zebrane są w tabeli 27.6.

Tabela 27.6. Dodatkowe metody klasy `DList`

Metoda	Opis
<code>a=list.remove_rear()</code>	Usuwa z listy ostatni element, zwracając wskaźnik do niego
<code>list.del_rear()</code>	Usuwa i niszczy ostatni element listy
<code>list.ins_before(i,a)</code>	Wstawia element <code>a</code> przed <code>i</code>
<code>list.del(i,dir)</code>	Usuwa element na bieżącej pozycji, a następnie przesuwa się do przodu o jedną pozycję, jeśli <code>dir</code> jest wartością dodatnią.

w przeciwnym przypadku – o jedną pozycję do tyłu

Klasy Plex

Obiekty klasy `Plex` zachowują się podobne jak tablice, ale mają znacznie większe możliwości. Oto niektóre z ich własności:

- υ posiadają ustaloną górną i dolną granicę indeksowania;
- υ mogą dynamicznie rozrastać się w lewo i prawo;
- υ ich elementy mogą być indeksowane; inaczej niż w przypadku tablic, indeksy są sprawdzane podczas wykonania programu;
- υ dodawane do tablicy mogą być tylko elementy wcześniej zainicjalizowane, po-dobnie tylko elementy zainicjalizowane można z niej odczytać.

Zdefiniowane zostały cztery typy struktur `Flex`: `Flex`, która może rozrastać się i kurczyć w zdefiniowanych wcześniej granicach, `XFlex`, mogąca zmieniać rozmiary bez żadnych ograniczeń, `RFlex`, podobna do `XFlex`, ale posiadająca nowe możliwości indeksowania oraz `MFlex`, która jest pochodną `RFlex`, pozwalającą na logiczne usuwanie i odzyskiwanie elementów.

Tabela 27.7 zawiera niektóre metody i operatory tych klas.

Tabela 27.7. Metody i operatory klas *Plex*

Metoda	Opis
<code>Plex b(a)</code>	Przypisuje kopię struktury <code>a</code> do <code>b</code>
<code>b=a</code>	Kopiuje <code>Plex a</code> do <code>b</code>
<code>a.length()</code>	Zwraca liczbę elementów struktury <code>a</code>
<code>a.empty()</code>	Zwraca logiczną prawdę, jeśli <code>a</code> posiada 0 elementów
<code>a.full()</code>	Zwraca logiczną prawdę, jeśli struktura <code>a</code> jest pełna
<code>a.clear()</code>	Usuwa wszystkie elementy struktury <code>a</code>
<code>a.append(b)</code>	Dołącza <code>Plex b</code> do górnej części <code>a</code>
<code>a.prepend(b)</code>	Dołącza <code>Plex b</code> do dolnej części <code>a</code>
<code>a.fill(z)</code>	Wstawia do struktury elementy równe <code>z</code> na wszystkich pozycjach
<code>a.valid(i)</code>	Zwraca logiczną prawdę, jeśli indeks <code>i</code> jest prawidłowy
<code>a.low_element()</code>	Zwraca wskaźnik do elementu o najniższej pozycji
<code>a.high_element()</code>	Zwraca wskaźnik do elementu o najwyższej pozycji

Klasy `Plex` są bardzo przydatne. W oparciu o nie opracowano wiele innych klas z biblioteki GNU C++, na przykład stosy, kolejki, listy itp.

Stosy

Klasa `Stacks` jest implementacją stosu (ang. LIFO, *last in, first out*). Dostępne są trzy odmiany tej struktury: `vStack` – stos o ustalonym rozmiarze, podawanym podczas inicjalizacji, oraz `xPStack` i `SLStack` – stosy o dynamicznie zmienianych rozmiarach, różniące się nieco szczegółami implementacji.

Tabela 27.8 zawiera metody wspólne dla tych klas.

Tabela 27.8. Metody i operatory klas `Stacks`

Metoda	Opis
<code>Stack st</code>	Konstruktor
<code>Stack st(rozm)</code>	Konstruktor stosu o rozmiarze <code>rozm</code>
<code>st.empty()</code>	Zwraca logiczną prawdę, jeśli stos jest pusty
<code>st.full()</code>	Zwraca logiczną prawdę, jeśli stos jest pełny
<code>st.length()</code>	Zwraca liczbę elementów przechowywanych na stosie
<code>st.push(x)</code>	Odkłada element <code>x</code> na stos
<code>x=st.pop()</code>	Zdejmuje element ze stosu i przypisuje go zmiennej <code>x</code>
<code>st.top()</code>	Zwraca wskaźnik do najwyższej położonego elementu na stosie
<code>st.del_top()</code>	Usuwa ze stosu najwyższy położony element
<code>st.clear()</code>	Usuwa ze stosu wszystkie elementy

Kolejki

Kolejki (ang. FIFO, *first in, first out*) implementowane są za pomocą klas `Queue`. Dostępne są trzy odmiany tych struktur: `vQueue` – kolejka o ustalonym rozmiarze (podawanym podczas inicjalizacji), oraz `xPQueue` i `SLQueue` – kolejki o rozmiarze zmienianym dynamicznie (podanie rozmiaru nie jest wymagane). W tabeli 27.9 zebrane zostały metody zaimplementowane w tych klasach.

Tabela 27.9. Metody i operatory klas `Queue`

Metoda	Opis
<code>Queue q</code>	Konstruktor
<code>Queue q(rozm)</code>	Konstruktor kolejki o rozmiarze <code>rozm</code>
<code>q.empty()</code>	Zwraca logiczną prawdę, jeśli kolejka jest pusta
<code>q.full()</code>	Zwraca logiczną prawdę, jeśli kolejka jest pełna

<code>q.length()</code>	Zwraca liczbę elementów przechowywanych w kolejce
<code>q.enq(x)</code>	Dopisuje element <code>x</code> do kolejki

cd. na następnej stronie

Tabela 27.9. cd. Metody i operatory klas Queue

Metoda	Opis
<code>x=q.deq()</code>	Pobiera element z kolejki i przypisuje go do zmiennej <code>x</code>
<code>q.front()</code>	Zwraca wskaźnik do pierwszego elementu w kolejce
<code>q.del_front()</code>	Usuwa z kolejki pierwszy element
<code>q.clear()</code>	Usuwa z kolejki wszystkie elementy

Oprócz wymienionych wyżej typów kolejek, w skład biblioteki klas GNU C++ wchodzą również klasy obsługujące kolejki dwukierunkowe i priorytetowe. Kolejka dwukierunkowa zawiera dodatkowe operatory pozwalające na dostęp do elementu przechowywanego na jej końcu. Kolejka priorytetowa pozwala na szybki dostęp do najmniejszego elementu i posiada dodatkowe operatory służące do wyszukiwania elementów.

Zbiory

Klasy `set` używane są do przechowywania zbiorów elementów. Jedynym ograniczeniem nałożonym na elementy jest fakt, że nie mogą się one powtarzać. Dostępnych jest kilka implementacji tej klasy, ale wszystkie one mają taki sam interfejs. Metody tych klas zebrane w tabeli 27.10.

Tabela 27.10. Metody i operatory klas Set

Metoda	Opis
<code>Set s</code>	Konstruktor
<code>Set s(rozm)</code>	Konstruktor zbioru o maksymalnym rozmiarze <code>rozm</code>
<code>s.empty()</code>	Zwraca logiczną prawdę, jeśli zbiór jest pusty
<code>s.length()</code>	Zwraca liczbę elementów zbioru
<code>i=s.add(z)</code>	Dodaje element <code>z</code> do zbioru <code>s</code> , zwracając numer indeksu
<code>s.del(z)</code>	Usuwa ze zbioru <code>s</code> element <code>z</code>
<code>s.clear()</code>	Usuwa wszystkie elementy ze zbioru <code>s</code>
<code>s.contains(z)</code>	Zwraca logiczną prawdę, jeśli element <code>z</code> należy do zbioru <code>s</code>
<code>s.(i)</code>	Zwraca wskaźnik do elementu o indeksie <code>i</code>
<code>i=s.first()</code>	Zwraca indeks pierwszego elementu zbioru
<code>s.next(i)</code>	Przypisuje zmiennej <code>i</code> indeks następnego elementu zbioru <code>s</code>
<code>i=s.seek(z)</code>	Przypisuje zmiennej <code>i</code> indeks elementu <code>z</code> , jeśli należy on do zbioru <code>s</code> ,

	w przeciwnym przypadku – zero
<code>set1==set2</code>	Zwraca logiczną prawdę, jeśli <code>set1</code> zawiera te same elementy, co <code>set2</code>
<code>set1!=set2</code>	Zwraca logiczną prawdę, jeśli zbiór <code>set1</code> nie zawiera tych samych elementów, co <code>set2</code>

Tabela 27.10. cd. Metody i operatory klas Set

Metoda	Opis
<code>set1<=set2</code>	Zwraca TRUE, jeśli zbiór <code>set1</code> jest podzbiorem <code>set2</code>
<code>set1 =set2</code>	Dodaje elementy zbioru <code>set2</code> do zbioru <code>set1</code>
<code>set1-=set2</code>	Usuwa ze zbioru <code>set1</code> wszystkie elementy, które występują w zbiorze <code>set2</code>
<code>set1&=set2</code>	Usuwa ze zbioru <code>set1</code> elementy, które nie występują w <code>set2</code>

Oprócz zbiorów dostępne są również inne klasy o podobnej funkcji – są to klasy `bag` (ang. *torba*). Struktury tego typu mogą, podobnie jak zbiory, zawierać elementy w dowolnym porządku, ale różnią się od zbiorów tym, że mogą również zawierać elementy powtarzające się. Nie obsługują w związku z tym operatorów `==`, `!=`, `|=`, `<=`, `-=` i `&=`. Nowe metody implementowane w klasie `bag` zebrane w tabeli 27.11.

Tabela 27.11. Metody klasy bag nie występujące w klasie Set

Metoda	Opis
<code>b.remove(z)</code>	Usuwa wszystkie wystąpienia elementu <code>z</code>
<code>b.nof(z)</code>	Zwraca liczbę wystąpień elementu <code>z</code>

Poza opisanymi wyżej, biblioteka klas GNU C++ zawiera jeszcze wiele innych użytecznych klas. Oprócz biblioteki dostarczanej wraz z kompilatorem, dostępne są także inne, darmowe biblioteki, mogące również okazać się bardzo przydatne.

Podsumowanie

C++ oferuje programistom wiele więcej niż język C. Najbardziej oczywistą przewagą jest możliwość programowania obiektowego oraz dostępność elastycznych bibliotek klas. W tym rozdziale przyjrzyliśmy się kompilatorowi i debugerowi, a także omówiliśmy pokróćce najczęściej używane klasy wchodzące w skład biblioteki GNU C++.

Jednym z problemów, na które narzekali programiści używający języka C++, był brak darmowych programów narzędziowych. Jak zauważyleś, o wiele więcej jest programów użytkowych wspomagających programowanie w języku C. Z upływem czasu sytuacja jednak się odwraca. Rosnąca ilość użytkowników języka C++ powoduje powstawanie nowych programów narzędziowych oraz ewolucję już istniejących. Nie należy więc obawiać się „przesiadki” na C++.

Kompilator i programy narzędziowe dla języka C omówione są w rozdziale 26. „Programowanie w języku C”.

Perl, język nadający się świetnie do pisania krótkich i wydajnych programów, przedstawiony jest w rozdziale 28. „Perl”.

Kompilatory innych języków dla systemu Linux opisane są w rozdziale 30. „Inne kompilatory”.

Rozdział 28.

Perl

Tim Parker

W tym rozdziale:

- υ Język Perl
- υ Tworzenie i uruchamianie programów w języku Perl
- υ Dane
- υ Tablice
- υ Elementy strukturalne
- υ Funkcje
- υ Operatory
- υ Konwertowanie programów do języka Perl

Język Perl (ang. *Practical Extraction and Report Language*) jest językiem interpretowanym, opracowanym przez Larry'ego Walla. Jego pierwotnym przeznaczeniem było zastąpienie takich, języków jak `awk`, służących do przetwarzania danych tekstowych. W tym rozdziale omówimy następujące tematy:

- υ koncepcja języka Perl,
- υ tworzenie i uruchamianie programów w języku Perl,
- υ dane, zmienne i tablice,
- υ elementy strukturalne,
- υ funkcje.

Po przeczytaniu tego rozdziału powinieneś orientować się w przewagach języka Perl nad innymi językami programowania. Będziesz również potrafił pisać proste programy, ułatwiające codzienną pracę z systemem Linux.

Język Perl

Jak zaznaczono już wcześniej, język Perl zaprojektowany został do obsługi plików tekstowych. Jeśli więc znasz choć trochę język `awk` (opisany w rozdziale 25. „gawk”), nie powinno Cię dziwić, że przejął z niego wiele konstrukcji. Perl łączy w sobie najlepsze cechy języków C, `sed`, `awk` i języków dostępnych w interpreterach poleceń powłoki, takich jak `bash` czy `tcsh`.

Perl jest podobny do języków powłoki zarówno pod względem działania, jak i składni. Istnieje jednak kilka różnic, o których nie sposób nie wspomnieć. Jedną z największych jest fakt, że nie jest to język czysto interpretowany. Programy w języku Perl są najpierw wczytywane w całości, następnie zamieniane na postać pośrednią, a dopiero potem wykonywane, natomiast programy powłoki są wczytywane i wykonywane wiersz po wierszu. Dzięki temu duże programy pisane w języku Perl wykonywane są znacznie szybciej niż skrypty powłoki. Przed wykonaniem sprawdzana jest również składnia poleceń w całym programie, co pozwala uniknąć sytuacji, w których program zatrzymuje się z powodu błędu składniowego, po przetworzeniu części danych – co bywa sytuacją nadzwyczaj kłopotliwą.

Fakt, że Perl nie jest językiem czysto interpretowanym, jest również w pewnym sensie wadą, ponieważ nawet niewielkie programy muszą zostać zamienione na postać pośrednią, przez co ich wykonanie staje się wolniejsze niż w przypadku skryptów powłoki. Nie jest to jednak szczególnie poważny problem i mało kto się nim przejmuje.

Tworzenie i uruchamianie programów w języku Perl

Proces tworzenia programu w języku Perl jest prawie identyczny, jak w przypadku skryptów powłoki. Program składa się z jednego lub więcej poleceń, umieszczonych w pliku tekstowym. Najpierw więc należy stworzyć taki plik, na przykład używając dowolnego edytora tekstów. Zaczniemy od bardzo prostego programu, wyświetlającego komunikat `Witaj!:`

```
#!/usr/bin/perl  
print "Witaj!\n";
```

Pierwszy, wyglądający nieco dziwnie wiersz, informuje interpreter poleceń, jakiego programu należy użyć do przetworzenia pozostały części pliku. Drugi wiersz powoduje

wydrukowanie napisu `Witaj!` i przejście do następnego wiersza. Można tu zauważyć duże podobieństwo do języka C.

Aby uruchomić program, musisz zrobić jeszcze jedną rzecz: przypisać mu prawo do wykonywania. Jeśli plik nazywa się `hello`, można zrobić to w następujący sposób:

```
chmod +x hello
```

Teraz można już uruchomić program, wpisując polecenie

```
hello
```

Spowoduje ono wywołanie interpretera języka Perl, który przetworzy cały program, a następnie uruchomi skompilowany kod.

Mожно również uruchamiać interpreter języka Perl, przekazując nazwę programu, który ma zostać wykonany, jako argument wywołania, na przykład tak:

```
perl hello
```

Dane

Na najniższym poziomie Perl rozróżnia dwa typy danych: numeryczne i tekstowe. W tym podrozdziale dowiesz się, jak typy te są obsługiwane i jak ich używać w programach. Na początek zajmijmy się zmiennymi i ich użyciem.

Zmienne

Zmienne w języku Perl obsługiwane są podobnie jak w skryptach powłoki. Podstawowa różnica polega na tym, że nazwę zmiennej należy w Perlu poprzedzać znakiem `$` zarówno przy przypisywaniu wartości, jak i przy odczycie (w skryptach powłoki podczas przypisywania wartości nie używa się znaku `$`). Poniższe polecenie przypisuje zmiennej `pozdr` wartość `Witaj!`:

```
$pozdr="Witaj!";
```

W skryptach powłoki analogiczne polecenie miałyby postać:

```
pozdr="Witaj!"  
lub  
set pozdr = "Witaj!" (w interpreterze tcsh).
```

W odróżnieniu od większości skryptów powłoki, Perl ignoruje spacje po obu stronach znaku równości, pozwalając programiście na dowolność w tym zakresie.

Inna różnica polega na tym, że operacje na zmiennych wykonuje się bezpośrednio, nie używając dodatkowo polecenia `expr`. Przykładowo, polecenie

```
$a=1+2;
```

spowoduje przypisanie do zmiennej `$a` wartości 3. Jeśli wprowadzisz polecenia

```
$a=1+2;
$b=3*$a;
```

zmienna `$b` będzie miała wartość 9. Tego właśnie oczekiwaliśmy, ale skąd Perl wie, czy dana jest tekstowa, czy numeryczna? Otóż nie wie tego. Za każdym razem, gdy zmieniona (lub stała) jest używana jako argument operatora arytmetycznego, jest ona zamieniana na postać numeryczną. W naszym przykładzie wszystko jest w porządku, ponieważ zmienna `$a` zawiera wartość 3. Co jednak byłoby, gdyby zawierała na przykład literę u (która nie reprezentuje żadnej wartości liczbowej)? Perl radzi sobie z tym w niezbyt elegancki sposób – jeśli zmiennej nie można przekształcić na liczbę, jest ona traktowana jako zero. Domyślnie Perl nawet nie informuje o takiej konwersji, możesz jednak tego zażądać, wywołując interpreter `perl` z opcją `-w`.

Liczby

Wszystkie liczby, również całkowite, zapisywane są w Perlu w formacie zmiennoprzecinkowym. Nie oznacza to, że nie możesz używać wartości całkowitych, ale i tak będą one traktowane jako liczby rzeczywiste.

Perl udostępnia zestaw operatorów, których można używać do przeprowadzania porównań i podstawowych operacji arytmetycznych. Tabela 28.1 zawiera niektóre z nich.

Tabela 28.1. Operatory arytmetyczne

Operator	Opis
<code>op1 + op2</code>	Suma
<code>op1 - op2</code>	Różnica
<code>op1 * op2</code>	Iloczyn
<code>op1 / op2</code>	Iloraz
<code>op1 ** op2</code>	<code>op1</code> do potęgi <code>op2</code>
<code>op1 % op2</code>	<code>op1</code> modulo <code>op2</code>
<code>op1==op2</code>	Zwraca wartość logiczną „prawda”, jeśli liczby są równe.
<code>op1!=op2</code>	Zwraca wartość logiczną „prawda”, jeśli liczby są różne.
<code>op1<op2</code>	Zwraca wartość logiczną „prawda”, jeśli <code>op1</code> jest mniejszy niż <code>op2</code> .
<code>op1>op2</code>	Zwraca wartość logiczną „prawda”, jeśli <code>op1</code> jest większy niż <code>op2</code> .
<code>op1<=op2</code>	Zwraca wartość logiczną „prawda”, jeśli <code>op1</code> jest mniejszy lub równy <code>op2</code> .
<code>op1>=op2</code>	Zwraca wartość logiczną „prawda”, jeśli <code>op1</code> jest większy lub równy <code>op2</code> .

Łańcuchy znaków

Łańcuchy znaków zwykle zawierają znaki alfanumeryczne (wielkie i małe litery, cyfry i znaki przestankowe). Perl nie odróżnia tych znaków od znaków niedrukowalnych, co oznacza, że można używać tego języka do przetwarzania plików binarnych.

Łańcuch znaków może być zapisany w Perlu na dwa sposoby – przez otoczenie go znakami pojedynczymi lub podwójnymi cudzysłówów. Łańcuch otoczony podwójnym cudzysłowem zachowuje się mniej więcej tak samo, jak w języku C. Można w nim użyć symboli specjalnych, poprzedzonych znakiem lewego ukośnika (ang. *backslash*). Ich lista zamieszczona jest w tabeli 28.2.

Tabela 28.2. Znaki specjalne dostępne w języku Perl

Symbol	Opis
\a	Powoduje wygenerowanie krótkiego dźwięku.
\b	Usuwa znaku przed kursorem (Backspace).
\cD	Pozwala umieścić w tekście dowolny znak niedrukowalny (w tym przypadku Control+D).
\f	Przejście do nowej strony.
\e	Znak Escape (kod 27).
\E	Zamyka obszar działania znaków \L i \U.
\l	Powoduje, że litera następująca po tym symbolu będzie traktowana jako mała litera.
\L	Powoduje, że wszystkie litery następujące po tym symbolu, aż do napotkania symbolu \E, będą traktowane jako małe litery.
\n	Nowy wiersz.
\r	Powrót karetki.
\t	Znak tabulacji.
\\"	Lewy ukośnik (\).
\"	Cudzysłów.
\u	Powoduje, że następna litera następująca po tym symbolu będzie traktowana jako wielka litera.
\U	Powoduje, że wszystkie litery następujące po tym symbolu, aż do napotkania symbolu \E, będą traktowane jako wielkie litery.

Wewnątrz tekstu otoczonego podwójnym cudzysłowem możliwe jest również podstawienie wartości zmiennej. Przykładowy fragment programu

```
$imie="Wojtek";
print "Czesc $imie, jak sie masz?";
```

spowoduje wyświetlenie napisu

```
Czesc Wojtek, jak sie masz?
```

Łańcuchy znaków otoczone pojedynczymi cudzysłowami różnią się od opisanych poprzednio tym, że znaki specjalne nie są w nich interpretowane, za wyjątkiem znaku ' oraz \. Po napotkaniu pierwszego z nich Perl uznaje, że jest to koniec łańcucha. Oznacza to, że jeśli chcesz wstawić pojedynczy cudzysłów do łańcucha znaków ujętego w pojedyncze cudzysłowy, musisz poprzedzić go znakiem \. Ta sama reguła dotyczy umieszczania w łańcuchach znaku \ (lewy ukośnik).

Przykładowo, jeśli chcesz wypisać tekst `Don't do that!` i przejść do następnego wiersza, powinieneś wprowadzić następujące polecenie:

```
print 'Don\'t do that!', "\n";
```



Pamiętaj, że jeśli umieścisz symbol `\n` wewnątrz tekstu otoczonego pojedynczym cudzysłowem, zostanie on wydrukowany (na ekranie pojawią się znaki `\n`) i nie nastąpi przejście do nowego wiersza.

Podobnie jak w przypadku liczb, również dla łańcuchów znaków dostępnych jest w Perlusie wiele operatorów – najważniejsze zebrane w tabeli 28.3.

Tabela 28.3. Operatory działające na łańcuchach znaków

Operator	Opis
<code>op1 . op2</code>	Łączy łańcuchy <code>op1</code> i <code>op2</code> .
<code>op1 x op2</code>	Powtarza <code>op1</code> <code>op2</code> razy.
<code>op1 eq op2</code>	Zwraca wartość logiczną „prawda”, jeśli teksty są równe.
<code>op1 ne op2</code>	Zwraca wartość logiczną „prawda”, jeśli teksty są różne.
<code>op1 lt op2</code>	Zwraca wartość logiczną „prawda”, jeśli <code>op1</code> jest mniejszy niż <code>op2</code> .
<code>op1 gt op2</code>	Zwraca wartość logiczną „prawda”, jeśli <code>op1</code> jest większy niż <code>op2</code> .
<code>op1 le op2</code>	Zwraca wartość logiczną „prawda”, jeśli <code>op1</code> jest mniejszy lub równy <code>op2</code> .
<code>op1 ge op2</code>	Zwraca wartość logiczną „prawda”, jeśli <code>op1</code> jest większy lub równy <code>op2</code> .

Operatory plikowe

Podobnie jak języki powłoki, Perl udostępnia operatory działające na plikach i katalogach, pozwalające sprawdzić ich właściwości. Najbardziej przydatne z nich zebrane zostały w tabeli 28.4.

Tabela 28.4. Operatory plikowe

Operator	Opis
<code>-B</code>	Sprawdza, czy plik jest plikiem binarnym.
<code>-d</code>	Sprawdza, czy plik jest katalogiem.

-e	Sprawdza, czy plik istnieje.
-f	Sprawdza, czy plik jest zwykłym plikiem.
-r	Sprawdza, czy plik da się odczytać.
-s	Sprawdza, czy rozmiar pliku jest różny od zera.
-T	Sprawdza, czy plik jest plikiem tekstowym.
-w	Sprawdza, czy można pisać do pliku.
-x	Sprawdza, czy plik jest wykonywalny.
-z	Sprawdza, czy plik ma zerową długość.

Operatory te mogą być używane w dowolnych wyrażeniach. W poniższym przykładzie użyliśmy instrukcji `if` do sprawdzenia, czy w bieżącym katalogu znajduje się plik o nazwie `.profile`:

```
if (-e ".profile") {
    print "Plik .profile znaleziony\n";
}
```

Tablice

Tablica to indeksowana lista elementów. W języku Perl dostępne są dwa typy tablic. Pierwszy z nich nie różni się od tablic znanych z innych języków programowania. Pozwala on na zapisanie elementu (tekstu lub liczby) do zmiennej mogącej przechować dowolną ilość takich elementów. Dostęp do nich jest możliwy przez indeksowanie tej zmiennej. Przykładowo, następująca instrukcja zapisuje łańcuchy znaków `1, luty i 1999` do tablicy o nazwie `data`:

```
@data=("1", "luty", "1999");
```



Liczba elementów przechowywanych w tablicy jest ograniczona wielkością dostępnej pamięci.

Jeśli chcesz uzyskać dostęp do danych zapisanych w tablicy `data`, musisz podać nazwę tablicy i odpowiedni indeks (indeksowanie zaczyna się od zera). Nazwę tablicy należy wówczas poprzedzić znakiem `$`, a nie `@`, ponieważ odwołujesz się nie do całej tablicy, tylko do jej elementu, który jest zwykłą zmienną. Aby na przykład przypisać wartość `luty` (przechowywaną jako drugi element tablicy `data`) zmiennej `miesiac`, powinieneś wydać polecenie:

```
$miesiac=$data[1];
```



Podobnie jak ma to miejsce w językach C i C++, w języku Perl indeksowanie tablicy zaczyna się od zera. Oznacza to, że na przykład

tablica pięcioelementowa zawiera elementy o indeksach [0], [1], [2], [3] i [4], nie zawiera natomiast elementu o indeksie [5].

Tablice drugiego typu to tablice asocjacyjne (zwane również haszującymi). Są one podobne do zwykłych tablic, z tym, że indeksować je można wybranym przez użytkownika łańcuchem znaków lub liczbą.

Ponieważ dla tablicy asocjacyjnej możesz sam wybrać klucz, według którego dane będą indeksowane, przypisywanie i odczytywanie zapisanych wartości odbywa się nieco inaczej niż w przypadku zwykłych tablic. Aby przypisać wartość elementowi tablicy asocjacyjnej, musisz oprócz samej wartości podać również klucz, który będzie z nią związany. Na przykład, jeśli chcesz elementowi tablicy `słowa` o indeksie `pozdr` przypisać wartość `Witaj!`, powinieneś wydać polecenie:

```
$słowa{"pozdr"}="Witaj!";
```

Przed nazwą tablicy znajduje się znak `$`, ponieważ odnosi się tylko do jednego elementu – czyli do zwykłej zmiennej. Operacje na całych tablicach wykonuje się przez poprzedzenie ich nazwy znakiem `:`:

```
%słowa2=%słowa;
```

Powyższe polecenie powoduje skopiowanie całej zawartości tablicy `słowa` do tablicy `słowa2`. Dostęp do elementu tablicy możliwy jest poprzez zdefiniowany wcześniej indeks:

```
$wartosc=$słowa{"pozdr"};
```

Oprócz operatorów, które mogą być używane również w odniesieniu do normalnych tablic, dla tablic asocjacyjnych dostępne są cztery nowe: `keys`, `values`, `delete` oraz `each`.

Operator `keys` zwraca tablicę słów indeksujących elementy tablicy asocjacyjnej. Poniższe polecenia spowodują, że w tablicy `indeksy` znajdą się łańcuchy `pozdr` i `pozdr1`:

```
$słowa{"pozdr"}="Witaj!";
$słowa{"pozdr1"}="Witam Cie";
@indeksy=keys(%słowa);
```

Pierwsze dwa polecenia przypisują wartości poszczególnym elementom tablicy asocjacyjnej `słowa`. Pierwsze powoduje przypisanie tekstu `Witaj!` do elementu o indeksie `pozdr`, drugie – tekstu `Witam Cie` do elementu o indeksie `pozdr1`.

Trzecie polecenie przypisuje wszystkie łańcuchy znaków będące indeksami elementów tablicy asocjacyjnej `słowa` do tablicy (zwykłej, nie asocjacyjnej) `indeksy`. Zauważ, że przed nazwą tablicy `słowa` znajduje się znak `%`, który informuje interpreter, że chodzi o całą tablicę, a nie o któryś z jej elementów.



Kolejność elementów w tablicy `indeksy` nie jest zdefiniowana. Może ona zawierać teksty (`pozdr`, `pozdr1`), ale równie dobrze może zawierać teksty (`pozdr1`, `pozdr`).

Operator `values` zwraca tablicę wszystkich wartości przechowywanych w tablicy asocjacyjnej. Poniższy program spowoduje, że tablica `tablwart` zawierać będzie elementy 123.5 i 105 (w dowolnej kolejności):

```
$ceny{"normalny"}=123.5;
$ceny{"wyprzedaz"}=105;
@tablwart=values(%ceny);
```

Pierwsze dwa polecenia przypisują odpowiednie wartości elementom tablicy asocjacyjnej `ceny`. Trzecie polecenie powoduje przypisanie elementom zwykłej tablicy `tablwart` wszystkich wartości przechowywanych w tablicy asocjacyjnej `ceny`.

Operator `delete` pozwala usuwać elementy tablicy asocjacyjnej. Usuwany jest zarówno klucz – indeks, jak i odpowiadająca mu wartość:

```
%kawa=( "rozpuszczalna", 5.20, "mielona", 6.99, "capuccino", 2.15 );
delete $kawa{"mielona"};
```

Pierwsze polecenie zapisuje trzy elementy do tablicy asocjacyjnej `kawa`. Drugie usuwa z niej element o indeksie `mielona`.

Operator `each` pozwala w łatwy sposób oddziaływać na wszystkie elementy tablicy asocjacyjnej. Jego pierwsze wywołanie zwraca pierwszy element tablicy, kolejne wywołania – następne elementy, aż do osiągnięcia ostatniego elementu. Poniższy fragment kodu powoduje wyświetlenie cen wszystkich gatunków kawy z poprzedniego przykładu.

```
while (( $typ, $scena ) = each (%kawa) ) {
    print "Kawa $typ kosztuje $scena zł\n";
}
```

Elementy strukturalne

Instrukcje strukturalne dostępne w języku Perl są podobne do znanych z innych języków programowania wysokiego poziomu. Opiszemy je skrótnie w tym podrozdziale.

Instrukcja blokowa

Instrukcje blokowe składają się z jednej lub więcej instrukcji otoczonych nawiasami klamrowymi. Ogólnie ich składnia jest następująca:

```
{
    instr_1;
    instr_2;
    ...
}
```

Instrukcji blokowych można użyć wszędzie tam, gdzie można użyć pojedynczych instrukcji. Najczęściej są one wykorzystywane do grupowania poleceń mających wykonywać się w pętlach (na przykład `while` albo `for`) lub warunkowo (w instrukcji `if`).

Instrukcja if

Instrukcja `if` używana jest do warunkowego wykonywania fragmentu programu. Jej składnia jest podobna do składni znanej z języka C:

```
if (wyrażenie) {
    instrukcje;
}
```

Instrukcja `if` powoduje obliczenie wartości wyrażenia; jeśli jego wartością jest logiczna prawda, wykonywane są polecenia w nawiasach klamrowych. W przeciwnym przypadku nie są one wykonywane.

Opcjonalnie po instrukcji `if` podać można jedną lub więcej instrukcji `elsif` i jedną instrukcję `else`:

```
if (wyrażenie1) {
    instrukcje;
} elsif (wyrażenie2) {
    instrukcje;
} elsif (wyrażenie3) {
    instrukcje;
} else {
    instrukcje;
}
```

W tej formie najpierw obliczona zostanie wartość wyrażenia `wyrażenie1`; jeśli jest ono prawdziwe, wykonany zostanie pierwszy blok instrukcji. Jeśli `wyrażenie1` nie jest prawdziwe, zostanie sprawdzona wartość wyrażenia `wyrażenie2`. Jeśli `wyrażenie2` jest prawdziwe, zostaną wykonane instrukcje zapisane w drugim bloku. W przeciwnym przypadku zostanie sprawdzony warunek trzeci (`wyrażenie3`); jeśli jest on prawdziwy, zostanie wykonany trzeci blok instrukcji. Wreszcie jeśli żaden z warunków nie jest spełniony, zostanie wykonany blok instrukcji po słowie kluczowym `else`. W przypadku takiej „drabinki” pamiętać należy o tym, że wykonany zostanie tylko jeden z bloków instrukcji.



W języku Perl wyrażenie jest prawdziwe, jeśli ma wartość inną niż zero, i fałszywe w przeciwnym przypadku - czyli tak jak w języku C i dokładnie odwrotnie niż w programach użytkowych, takich jak np. test.

Poniższy przykład ilustruje zastosowanie instrukcji `if`:

```
print "Wprowadz imie lub wpisz \"pomoc\" aby uzyskac \n";
print "objasnienie dzialania programu \n";
$imie=<STDIN>;
chop($imie);
if ($imie eq "") {
    print "Musisz wprowadzic imie\n";
    exit;
} elsif ($imie eq "pomoc") {
    print "Program wczytuje imie uzytkownika\n";
    print "a nastepnie drukuje na ekranie krotkie pozdrowienie\n";
    exit;
} else {
```

```
    print "Witaj $imie, to właśnie jest Perl";
}
```

Po uruchomieniu powyższy program wyświetli krótki komunikat i będzie oczekwał na wprowadzenie jakiejś wartości. Wprowadzona wartość zostanie przypisana zmiennej `$imie` (trzeci wiersz programu). Instrukcja `if` przekazuje sterowanie do odpowiedniego bloku programu, w zależności od wprowadzonego tekstu. Jeśli wciśniesz Enter nie podając żadnego tekstu, zostanie wyświetlony komunikat `Musisz wprowadzić imię.`. Jeżeli wprowadzisz tekst `pomoc` i wciśniesz Enter, wyświetcone zostanie objaśnienie dotyczące działania programu. W innych przypadkach wprowadzony tekst zostanie potraktowany jako imię i zostanie wyświetlony odpowiedni komunikat.

Polecenie `chop` usuwa ostatni znak zmiennej będącej jego argumentem. W powyższym programie jest ono potrzebne, ponieważ ostatnim znakiem wprowadzonej z klawiatury zmiennej jest symbol nowego wiersza. Gdybyśmy go nie usunęli, polecenie `print` wydrukowałby go, dając na wyjściu na przykład

```
Witaj Michal
, to właśnie jest Perl
```

Instrukcja unless

Instrukcja `unless` jest przeciwnieństwem instrukcji `if`. Oblicza wartość wyrażenia warunkowego, i jeśli jest ono fałszywe – wykonuje dany ciąg instrukcji. Jego składnia jest następująca:

```
unless (wyrażenie) {
    instrukcje;
}
```

Oczywiście w każdym miejscu, gdzie można zastosować instrukcję `unless`, można równie dobrze użyć instrukcji `if`, negując warunek – funkcje obu tych instrukcji pokrywają się. Z tego powodu instrukcja `unless` jest używana dość rzadko. Poniżej podano przykład jej zastosowania:

```
print "Wprowadź numer rachunku\n";
$rach=<STDIN>
unless ($rach<1000) {
    print "Jako klient uprzywilejowany otrzymujesz 10% zniżki";
}
```

Powyższy program pyta użytkownika o numer rachunku. Jeśli numer ten jest większy lub równy 1000, wypisywana jest informacja `Jako klient uprzywilejowany otrzymujesz 10% zniżki`.

Instrukcja for

Jak dotąd omówiliśmy tylko instrukcje warunkowe. Teraz omówimy instrukcję `for`, która jest jedną z instrukcji iteracyjnych dostępnych w języku Perl. Instrukcje iteracyjne służą do wykonywania bloków programu pewną liczbę razy. Składnia instrukcji `for` jest następująca:

```
for (wyrażenie_inicjalizujace; warunek; wyrażenie) {
    instrukcje;
}
```

Zwykle w wyrażeniu `wyrażenie_inicjalizujace` następuje przypisanie wartości początkowej jakiejś zmiennej spełniającej funkcję licznika pętli. Wyrażenie to jest obliczane tylko raz, przed rozpoczęciem iteracji. Przed każdą iteracją obliczana jest wartość wyrażenia `warunek`. Pętla jest powtarzana, dopóki jest ono prawdziwe. Trzecie wyrażenie zwykle używane jest do zwiększania licznika pętli, ale może być wykorzystane również w inny sposób. Jest ono wykonywane po każdym przebiegu pętli. Poniżej podano przykład użycia tej instrukcji.

```
for ($i=0; $i<20; $i=$i+2) {
    print $i*2, " ";
}
print "\n Koniec\n";
```

Powyższy program wyświetla w każdym przebiegu pętli wartość `$i*2`. Przed pierwszym jej wykonaniem zmiennej `$i` przypisywana jest wartość zero. Następnie wartość zmiennej `$i` porównywana jest z liczbą 20; ponieważ wartość `$i` jest mniejsza niż 20, wykonywane jest ciało pętli. Pierwszy przebieg pętli powoduje więc wyświetlenie liczby 0. Po każdym przebiegu pętli wartość zmiennej `$i` jest zwiększana o dwa. Pętla powtarzana jest do momentu, aż wartość `$i` jest większa lub równa 20 (warunek wykonania pętli przestaje być spełniony). Po zakończeniu pętli następuje przejście do następnych instrukcji. W wyniku działania tego programu otrzymamy więc:

```
0 4 8 12 16 20 24 32 36
Koniec
```

Pętle `for` są często używane do obsługi tablic o stałych rozmiarach. Poniższy program wyświetla wszystkie wartości zapisane w tablicy pięćdziesięcioelementowej:

```
for ($i=0; $i<50; $i=$i+1) {
    print $tablica[$i], "\n";
}
```

W przypadku, gdy rozmiar tablicy nie jest znany (co często zdarza się w programach pisanych w języku Perl), bardziej przydatne są instrukcje `foreach` lub `while`.

Instrukcja foreach

`foreach` to kolejna instrukcja iteracyjna, podobna do instrukcji `for` znanej z języków powłok `bash` i `pdksh` oraz instrukcji `foreach` języka powłoki `tcsh`. Wykonuje ona blok instrukcji dla każdego elementu listy podanej jako jej drugi argument. Składnia instrukcji `foreach` jest następująca:

```
foreach $i (@lista) {
    instrukcje;
}
```

W miejscu tablicy `lista` może również pojawić się bezpośrednio podana lista elementów. Oto przykład użycia instrukcji `foreach`:

```
@tab1=(5,10,15,20,25);
foreach $i (@tab1) {
    print $i*3, " ";
}
print "\n";
```

Powyższy program wyświetla kolejno wszystkie wartości zapisane w tablicy `tab1` pomnożone przez trzy, czyli

```
15 30 45 60 75
```

Ciekawa jest również możliwość przypisania wartości zmiennej `$i` wewnętrz pętli. Dzięki temu można np. zmienić wartość wszystkich elementów tablicy o nieznanym rozmiarze – w poniższym przykładzie pomnożono je przez trzy:

```
@tab1=(5,10,15,20,25);
foreach $i (@tab1) {
    $i=$i*3;
}
```

Po wykonaniu powyższego kodu w tablicy `tab1` znajdą się wartości `(15, 30, 45, 60, 75)`.

Instrukcja while

Instrukcja `while` pozwala na wykonywanie bloku programu tak długo, jak długo zadane wyrażenie warunkowe jest prawdziwe. Składnia instrukcji `while` jest następująca:

```
while (wyrażenie) {
    instrukcje;
}
```

Wyrażenie warunkowe obliczane jest przed każdym przebiegiem pętli; jeśli jest ono prawdziwe, wykonywane są instrukcje stanowiące ciało pętli. W przykładowym programie pętlę `while` zastosowano do wyświetlenia wszystkich elementów tablicy:

```
$i=0;
@tablica=(1,2,3,4,5);
while ($tablica[$i]) {
    print $tablica[i];
    $i=$i+1;
}
```

Pętla będzie powtarzana dopóki wartość zmiennej `$tablica[i]` będzie zdefiniowana (czyli dla wszystkich elementów tablicy). Zauważ, że dzięki takiemu rozwiązaniu nie musisz wiedzieć, ile elementów znajduje się w tablicy.

Instrukcja until

Instrukcja `until` jest bardzo podobna do instrukcji `while`. Ciało pętli jest jednak wykonywane w przypadku, gdy wyrażenie warunkowe jest fałszywe. Każdą pętlę `until` można zastąpić pętlą `while` negując wyrażenie warunkowe, dlatego nie jest ona używana szczególnie często. Oto przykład zastosowania pętli `until`:

```
$i=0;
until ($i>10) {
    print "$i ";
    $i=$i+1;
}
```

Polecenia wewnętrz pętli wykonywane będą tak długo, aż wartość zmiennej `$i` przekroczy 10. Powyższy fragment programu jest równoważny następującemu:

```
$i=0;
while ($i<=10) {
    print "$i";
    $i=$i+1;
}
```

Po uruchomieniu dowolnego z powyższych przykładów na ekranie otrzymamy następujący wynik:

```
0 1 2 3 4 5 6 7 8 9 10
```

Funkcje

Poznałeś już kilka funkcji wewnętrznych języka Perl, takich jak na przykład funkcja `print`. Pora dowiedzieć się, w jaki sposób można tworzyć własne funkcje.

Składnia definicji funkcji (podprogramu) w języku Perl jest następująca:

```
sub nazwa_podprogramu {
    instrukcje;
}
```

Instrukcje zawarte w podprogramie są wykonywane przy każdym jego wywołaniu.

Spójrzmy na przykładową definicję funkcji wyświetlającej iloczyn dwóch liczb będących wartościami zmiennych `$num1` i `$num2`:

```
sub mult {
    print "$num1 * $num2 =", $num1*$num2, "\n";
}
```

Aby wywołać funkcję, musisz poprzedzić jej nazwę symbolem `&`, tak jak w poniższym przykładzie:

```
$num1=5;  
$num2=10;  
&mult;
```

Po wywołaniu takiego programu na ekranie zostaną wyświetcone następujące dane:

```
5*10=50
```

Jednak funkcje nie potrafiące pobrać parametrów i zwrócić wartości nie są szczególnie przydatne.

Przekazywanie argumentów do funkcji

Przekazywanie argumentów do funkcji zwiększa w znacznym stopniu ich użyteczność, ponieważ przekazując jakieś dane jako argumenty nie trzeba znać nazw zmiennych używanych przez funkcję. Przepiszmy funkcję `mult` tak, aby działała nie na zmiennych globalnych (w poprzednim przykładzie operowała ona na zmiennych `num1` i `num2`), ale pobierała dwa argumenty:

```
sub mult2 {  
    print "$_[0] * $_[1]=", $_[0]*$_[1], "\n";  
}
```

Zmienne `$_[0]` i `$_[1]`, będące elementami tablicy `$_` (znak podkreślenia), zawierają właśnie dwa pierwsze argumenty wywołania funkcji. Wywołanie takiej funkcji może mieć następującą postać:

```
&mult2(5,10);
```

Efekty działania funkcji są takie same jak w poprzednim przykładzie:

```
5*10=50
```

Argumenty funkcji dostępne są w jej wnętrzu jako elementy tablicy o nazwie `$_`. Pierwszy z nich jest więc zapisany pod nazwą `$_[0]`, drugi – `$_[1]`, i tak dalej.

Zwracanie wartości

Jeśli nie chcesz, by wynik działania funkcji był wyświetlany na ekranie, tylko zwracany do programu, musisz umieć zmusić funkcję do zwrócenia określonej wartości. W języku Perl każda funkcja zwraca jakąś wartość. Domyslnie jest to wartość zwrócona przez ostatnio wykonaną operację. Tak więc w poprzednich przykładach funkcje zwracały to, co zostało zwrócone przez funkcję `print` (która zwraca wartość `1`, jeśli wszystko przebiegło pomyślnie). Aby iloczyn dwóch liczb był zwracany do programu, funkcja powinna wyglądać tak:

```
sub mult3 {  
    $_[0]*$_[1];  
}
```

Jej wywołanie w programie może mieć następującą postać:

```
$wynik=&mult3(2,3);  
print "Wynikiem mnożenia jest $wynik";
```

Po wykonaniu takiego programu wynik mnożenia liczb 2 i 3 staje się wartością zmiennej `$wynik`, a następnie jest on wyświetlany na ekranie.

Operatory

W przykładach pojawiających się w tym rozdziale znalazło się całkiem sporo operatorów. Były nimi na przykład polecenia takie jak `print`, czy też służące do obsługi tablic asocjacyjnych, jak `keys`, `values`, `each`, `delete` itd. W języku Perl dostępna jest ogromna liczba innych operatorów. Aby zorientować się, jakiego typu operacje pozwalają one wykonać, albo obejrzeć ich pełny spis, przejrzyj dokumentację dostępną na stronach `man`.

Konwertowanie programów do języka Perl

Perl ma podobne możliwości jak wiele innych języków programowania dostępnych dla Linuxa. Ponieważ jest on nadzbiorem języków `sed` i `gawk`, każdy program napisany w tych językach może być z łatwością przetłumaczony na język Perl. Służą do tego programy narzędziowe `a2p` (konwersja z języka `gawk`) i `s2p` (konwersja z języka `sed`).



Translator języka `gawk` na Perl nazywa się nie `g2p`, ale `a2p`, ponieważ `gawk` to wersja GNU języka `awk`.

Aby uruchomić program `a2p` (lub `s2p`), wydaj po prostu polecenie:

```
a2p program_gawk > program_perl
```

lub odpowiednio

```
s2p program_sed > program_perl
```

Polecenia te spowodują przetłumaczenie programu w języku `gawk` (lub `sed`) na Perl.

Podsumowanie

W tym rozdziale przedstawiliśmy w skrócie podstawowe reguły programowania w języku Perl. Jest to chyba najpotężniejszy z języków dostępnych dla systemu Linux. Zawiera operatory pozwalające zrobić wszystko to, na co pozwala system operacyjny. Ogromna

liczba dostępnych funkcji powoduje, że aby zostać ekspertem potrzebnych jest co najmniej kilka miesięcy intensywnej praktyki, ale nauczenie się go w podstawowym zakresie jest bardzo łatwe.

Choć w rozdziale tym poruszyliśmy najważniejsze koncepcje języka Perl, nawet nie zsygnalizowaliśmy ogromnej liczby jego zastosowań. Pewne pojęcie o potędze tego języka może dać dokumentacja dostępna w postaci stron [man](#).

`awk`, język przeznaczony do obsługi plików tekstowych, opisany jest w rozdziale 25. „*gawk*”.

Tcl i Tk, języki programowania przydatne głównie do pisania niewielkich skryptów, działających również w środowisku X, omówione są w rozdziale 29. „Podstawy języków Tcl i Tk”.

Tworzenie skryptów CGI, obecnie chyba najważniejsze z zastosowań języka Perl, przedstawione jest w rozdziale 52. „*Skrypty CGI*”.

Rozdział 29.

Podstawy języków Tcl i Tk

Tim Parker

W tym rozdziale:

- υ Czym jest Tcl?
- υ Czym jest Tk?
- υ Język Tcl
- υ Tk – nowe możliwości Tcl

Czym jest Tcl?

Tcl (nazwa ta czyta się „tikl”, a pochodzi od angielskiej nazwy Tool Command Language) jest językiem przeznaczonym do pisania skryptów, podobnym nieco do języków interpreterów poleceń powłoki, omówionych w rozdziale 14. Używa się go głównie do szybkiego tworzenia programów działających w trybie tekstowym.

Język Tcl został zaprojektowany przez Johna Ousterhouta, późniejszego pracownika Uniwersytetu Kalifornijskiego w Berkeley. Jest on językiem interpretowanym, posiada więc typowe wady i zalety tego rodzaju języków. Największą wadą języków interpretowanych jest niewielka szybkość działania: program napisany w języku kompilowanym może działać wielokrotnie szybciej. Podstawową zaletą jest krótki czas tworzenia aplikacji i możliwość natychmiastowego obejrzenia efektów wprowadzonych zmian, co wynika z braku etapu komplikacji programu.

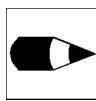
Podstawowe możliwości funkcjonalne języka Tcl realizowane są za pomocą bazowego zestawu funkcji wewnętrznych, jednak jego prawdziwa potęga tkwi w nieograniczonych wręcz możliwościach rozbudowy. Programista może bez trudu tworzyć własne biblioteki, co więcej – może osadzać biblioteki Tcl we własnych programach. Dzięki temu możliwe jest wbudowanie do tworzonej aplikacji elementów języka skryptowego bez konieczności projektowania takiego języka od podstaw. W konsekwencji użytkownicy

aplikacji uzyskują dostęp zarówno do wszystkich wewnętrznych poleceń Tcl, jak i do funkcji utworzonych przez programistę i umieszczonych w bibliotece języka.

Polecenia języka Tcl mogą być wywoływanie po uruchomieniu jego interpretera. W większości systemów nazywa się on `tclsh` (w niektórych nosi nazwę `tcl`). Po uruchomieniu interpretera można już bezpośrednio wpisywać polecenia, na przykład:

```
puts stdout "Ahoj, przygodo!"
```

Polecenie to składa się z trzech elementów. Właściwe polecenie to `puts`; nazwa ta jest skrótem od angielskiego „*put string*”, zaś samo polecenie zapisuje dane do urządzenia określonego przez drugi element. W naszym przypadku urządzeniem tym jest standardowe urządzenie wyjściowe, czyli na ogół ekran monitora. Trzecim elementem polecenia jest wyprowadzany tekst. Cudzysłowy zapewniają, że zostanie on zinterpretowany jako pojedynczy wyraz.



`stdout` (standardowe urządzenie wyjścia) jest domyślnym urządzeniem wyjściowym dla polecenia `puts`, jeśli więc chcesz wydrukować coś na ekranie, drugi argument może zostać pominięty.

Nawet na tak nieskomplikowanym przykładzie widać, jaką postać ma składnia prostych poleceń języka Tcl (poleceń jest oczywiście znacznie więcej, jednak składnia pozostaje we wszystkich przypadkach identyczna):

```
polecenie parametr1 parametr2 ...
```

`polecenie` może być dowolną funkcją wewnętrzną języka Tcl bądź też wywołaniem zdefiniowanej przez programistę procedury rozszerzającej zestaw poleceń wbudowanych. W obu przypadkach obowiązująca składnia pozostaje taka sama.

Czym jest Tk?

Tk, również opracowany przez Johna Ousterhouta, jest rozszerzeniem języka Tcl o interfejs graficzny (oparty na systemie X Window). Pozwala on na tworzenie programów dla X Window w sposób znacznie łatwiejszy, niż przy użyciu konkurencyjnych bibliotek X, takich jak Motif czy Open Look. Pod wieloma względami przypomina on rodzinę języków Visual dla Windows (np. Visual Basic).

Podobnie jak Tcl, Tk również pozwala na wpisywanie poleceń, które następnie są interpretowane. Interpreter Tk jest nadziobrem Tcl, toteż każdy program napisany w Tcl będzie działał w języku Tk. Tk obsługuje jednak wiele nowych funkcji, pozwalających łatwo i szybko dodać do programu interfejs X.

Interpreter poleceń Tk nazywa się `wish` (ang. *windowing shell*) i musi być uruchamiany w środowisku X (po uruchomieniu wyświetla on okno prezentujące wyniki wykonania interpretowanych poleceń trybu graficznego).

Po tym krótkim wprowadzeniu spróbujmy przepisać poprzedni program przykładowy tak, aby napis `Ahoj, przygodo` pojawił się na przycisku wyświetlonym w oknie interpretera `wish`. W tym celu należy najpierw uruchomić ten ostatni, czego dokonuje się po przez wpisanie w oknie `xterm` polecenia:

```
wish
```

Spowoduje to wyświetlenie okna `wish` i uruchomienie interpretera Tk w oknie `xterm`. Polecenia języka Tcl (lub Tk) wpisuje się bezpośrednio w oknie `xterm`. Aby wyświetlić w oknie `wish` nasz przycisk z napisem, musimy wprowadzić następujące polecenia:

```
button .b -text "Ahoj, przygodo!" -command exit  
pack .b
```

Jak widać, składnia jest taka sama jak poprzednio: po właściwym poleceniu następuje pewna liczba argumentów. Pierwszy z nich to nazwa, którą zamierzamy przypisać przyciskowi. Postać kolejnych argumentów różni się od użytych w Tcl-owej wersji programu – składają się one z dwóch części: nazwy argumentu i jego wartości.

Drugi argument nazywa się `text`, a jego wartością jest tekst, który ma pojawić się na przycisku. Trzeci argument, o nazwie `command`, zawiera nazwę polecenia, które ma zostać wykonane po naciśnięciu przycisku. W naszym przypadku program nie powinien nic zrobić, toteż nakazujemy interpreterowi zakończenie przetwarzania skryptu.

Przycisk, który utworzyliśmy, nazywa się `.b`. Do jego wyświetlenia w oknie `wish` służy polecenie `pack`.

W naszym przykładzie polecenie to posiada tylko jeden argument: nazwę przycisku utworzonego w pierwszym wierszu programu. Wynikiem jego wykonania będzie więc wyświetlenie w oknie `wish` przycisku zawierającego napis `Ahoj, przygodo`.

W powyższym przykładzie warterem omówienia są jeszcze dwie rzeczy. Po pierwsze, wyjaśnijmy, dlaczego nasz przycisk nosi nazwę `.b`, a nie np. `b` czy `przycisk..` Sama nazwa jest tu mało istotna (można by oczywiście użyć nazwy `przycisk`), ważna jest natomiast rozpoczynająca ją kropka. Symbol kropki służy tu do reprezentowania hierarchii elementów sterujących (ang. `widget`). Każdy z nich zawarty jest w innym elemencie. Główny, nadzędny element hierarchii zawarty jest bezpośrednio w oknie `wish`, a jego nazwą jest `.` (kropka; można tu zauważyć pewną analogię z linuxowym systemem plików, w którym poszczególne katalogi są wzajemnie zagnieżdżone, zaś katalog główny (najwyższego poziomu) nosi nazwę `/`). Za każdym razem, gdy tworzony jest nowy element, musimy poinformować interpreter Tk, w którym elemencie hierarchii będzie on zawarty. W naszym przypadku „pojemnikiem” na nowo tworzony przycisk jest element `..`, czyli główny element hierarchii.

Po drugie, zauważ, że po wykonaniu polecenia `pack` okno `wish` zmienia rozmiar, kurcząc się do rozmiarów utworzonego przycisku. Do zmiany tego zachowania i modyfikacji sposobu wyświetlania obiektów na ekranie służą liczne polecenia języka, którymi zajmiemy się nieco później.

Język Tcl

Po prezentacji przykładów wróćmy do podstaw i przyjrzymy się bliżej językowi Tcl. Zawiera on wiele poleceń, które pozwalają na tworzenie wszystkich konstrukcji spotykanych w innych językach programowania. Polecenia te zostaną omówione poniżej.

Zmienne i przypisywanie wartości

Podobnie jak interpretery poleceń powłoki UNIX-a, Tcl pozwala na korzystanie ze zmiennych, będących tymczasowymi pojemnikami na dane przetwarzane w programie. Nazwy zmiennych w języku Tcl składać się mogą z dowolnych znaków drukowalnych i na ogół związane są w jakiś sposób z przechowywaną w zmiennych informacją. Poprawnymi nazwami zmiennej przechowującej miesięczne zarobki pracownika są na przykład:

```
Miesieczne_zarobki  
"Miesieczne zarobki"
```



Użycie cudzysłowu powoduje, że Tcl traktuje spacje i znaki tabulacyjne (czyli tzw. białe znaki) nie jako znaki rozdzielające wyrazy, tylko jako część nazwy. Dokładniejsze omówienie tego tematu znajdziesz w punkcie „Użycie cudzysłówów”.

Również wartość zmiennej może składać się z dowolnych znaków, na przykład:

```
"15,000"  
15000  
"patrz tabela 4"
```

Do przypisywania wartości do zmiennych służy w języku Tcl polecenie `set`. Może ono posiadać jeden lub dwa argumenty. W formie dwuargumentowej pierwszym argumentem jest nazwa zmiennej, zaś drugim – przypisywana do niej wartość. W przypadku użycia tylko jednego argumentu, jest on interpretowany jako nazwa zmiennej, której wartość zostanie zwrócona przez polecenie `set`. Poniższe polecenie powoduje przypisanie zmiennej `Miesieczne_zarobki` wartości `15000`, a następnie wyświetlenie jej na ekranie:

```
set Miesieczne_zarobki 15000
```

Jeśli chcesz wyświetlić wartość zmiennej `Miesieczne_zarobki`, wpisz polecenie:

```
set Miesieczne_zarobki
```

Wszystkie zmienne (również numeryczne) przechowywane są jako łańcuchy znaków. Przykładowo, po przypisaniu:

```
set num 40
```

zmienna `num` będzie zawierała nie wartość liczbową, lecz ciąg znaków `40` (4 i 0).

Wiesz już, jak nadać zmiennej wartość i jak wyświetlić ją na ekranie. Aby wartości zmiennej użyć w poleceniu innym niż `set`, musisz jej nazwę poprzedzić znakiem `$`, nakazującym interpreterowi Tcl zwrócenie wartości następującej po nim zmiennej, np.:

```
set Miesieczne_zarobki 15000  
expr $Miesieczne_zarobki*12
```

Pierwsze polecenie powoduje przypisanie zmiennej `Miesieczne_zarobki` wartości `15000`. Polecenie `expr` służy do obliczania wartości wyrażeń – w naszym przypadku oblicza ono wartość zmiennej `Miesieczne_zarobki` pomnożoną przez `12`.

Podstawienie polecenia

Podstawienie polecenia umożliwia użycie wyniku wykonania polecenia jako argumentu innego polecenia. Jest to możliwe dzięki otoczeniu nawiasami kwadratowymi polecenia, którego wartość ma zostać użyta w innym poleceniu, na przykład:

```
set Miesieczne_zarobki 15000  
set Roczne_zarobki [ expr $Miesieczne_zarobki*12 ]
```

Pierwsze polecenie przypisuje zmiennej `Miesieczne_zarobki` wartość `15000`. Drugie powoduje natomiast przypisanie zmiennej `Roczne_zarobki` wartości zwróconej przez polecenie `expr` zawarte w nawiasach kwadratowych (czyli wartości zmiennej `Miesieczne_zarobki` pomnożonej przez `12`).

W powyższym przykładzie w nawiasie kwadratowym występuje tylko jedno polecenie. Tcl pozwala jednak na umieszczenie tam dowolnego prawidłowego skryptu, czyli dowolnej liczby prawidłowych poleceń.

Użycie cudzysłowów

Zdarza się, że w programach trzeba użyć znaków mających specjalne znaczenie dla języka Tcl, ale nie chcesz, aby były one interpretowane. Możesz je ukryć przedinterpretorem na trzy sposoby.

Pierwszy z nich to użycie podwójnego cudzysłowu (""). Zapobiega on interpretacji znaków białych. Kiedy chcesz, aby Tcl potraktował kilka wyrazów jako całość, powinieneś otoczyć je podwójnym cudzysłowem. Przykładem może być nazwa zmiennej składająca się z dwóch wyrazów: `Miesieczne_zarobki`. Aby przypisać zmiennej o takiej nazwie wartość `1500`, powinieneś wydać polecenie:

```
set "Miesieczne_zarobki" 1500
```

Podwójne cudzysłowy przydają się również w przypadku, gdy chcesz przypisać zmiennej wartość składającą się z więcej niż jednego wyrazu, np.:

```
set Nag11 "Zarobki w firmie"
```

Drugim sposobem ukrywania znaków przed interpreterem języka Tcl jest użycie symbolu lewego ukośnika (\). Powoduje on ukrycie jednego, następującego bezpośrednio po nim znaku. Najczęściej używa się go przed znakami takimi jak \$, na przykład w taki sposób:

```
set Nagl1 "Miesięczne zarobki wynosiły średnio \$1200"
```

W powyższym przykładzie zmiennej Nagl1 zostanie przypisana wartość Miesięczne zarobki wynosiły średnio \$1200. Dzięki zastosowaniu znaku \ interpreter języka Tcl nie próbuje wykonywać podstawienia zmiennej o nazwie \$1200, co spowodowałoby błąd (ponieważ taka zmienna nie została zdefiniowana). Znak \\$ traktowany jest w tym przypadku tak jak każdy inny znak.

Trzeci sposób, najbardziej „skuteczny”, to użycie nawiasów klamrowych ({}). Zapobiegają one interpretacji wszystkich znaków specjalnych, w tym również znaków białych. Poniższe polecenie ma taki sam skutek jak polecenie podane w poprzednim przykładzie:

```
set Nagl1 {Miesięczne zarobki wynosiły średnio $1200}
```

Bardzo ważny jest fakt, że znaki specjalne występujące w argumentach poleceń nie są interpretowane od razu przez Tcl, ale dopiero po przesłaniu ich do odpowiedniego polecenia. Oto przykład takiej opóźnionej interpretacji:

```
set liczn 0
while {$liczn < 3} {
    puts "Wartosc zmiennej liczn wynosi $liczn"
    set liczn [ expr $liczn + 1]
}
```

W powyższym przykładzie instrukcja while musi przetworzyć oba argumenty w każdym przebiegu pętli. Z tego powodu konieczne jest, by interpreter zignorował wszystkie symbole specjalne występujące w tych argumentach, pozostawiając ich interpretację poleceniu while.

Zapoznaliśmy się już z podstawami języka Tcl, pora więc przyjrzeć się kilku nieco bardziej zaawansowanym poleceniom.

Instrukcja if

Instrukcja if działa tak, jak w innych językach programowania: powoduje wykonanie fragmentu kodu pod warunkiem, że jakieś wyrażenie jest prawdziwe. Jej składnia jest następująca:

```
if {wyrażenie} {polecenie}
```

Polecenie if wymaga podania dwóch argumentów. Pierwszy z nich to warunek, w zależności od którego wykonywane są polecenia podane jako drugi argument. Warunek jest zwykle wyrażeniem logicznym, przyjmującym wartości prawda lub fałsz, np.:

```
$i < 10
$num = 2
```



Tcl traktuje wszystkie wyrażenia o wartości zero jako fałszywe, zaś pozostałe - jako prawdziwe. Podobną sytuację mamy w takich językach programowania, jak C czy Perl, ale dokładnie odwrotnie jest na przykład w przypadku programu użytkowego `test`. Musisz niestety przyzwyczaić się do tej niedogodności.

Wyrażenia takie jak:

```
$i + $b  
10*3
```

mogą również zostać użyte jako warunek w instrukcji `if`, ale trudno znaleźć dla nich jakieś praktyczne zastosowanie.

Drugim argumentem polecenia `if` jest skrypt Tcl, mogący zawierać dowolną liczbę poleceń. Jest on wykonywany tylko wtedy, kiedy wartością pierwszego wyrażenia jest prawda.

Po poleceniu `if` może wystąpić jedna lub więcej instrukcja `elseif` i jedna instrukcja `else`; składnia tego polecenia jest wówczas następująca:

```
if {warunek} {  
    polecenia }  
elseif {warunek} {  
    polecenia }  
elseif {warunek} {  
    polecenia }  
else {  
    polecenia }
```

Wykonane zostaną polecenia zawarte w nawiasach klamrowych po pierwszym warunku, który jest spełniony. Jeśli żaden warunek nie okaże się prawdziwy, wykonane zostaną polecenia występujące po słowie kluczowym `else`.



Nawias klamrowy musi zostać otwarty w tym samym wierszu, w którym znajduje się słowo `go` poprzedzające, ponieważ znak nowego wiersza traktowany jest jako separator poleceń. Jeśli pierwszą część polecenia `if` (`if {warunek}`) umieścisz w jednym wierszu, natomiast resztę polecenia (w nawiasach klamrowych) w wierszu następnym, obie części potraktowane zostaną jako dwa oddzielne polecenia.

Instrukcja for

Polecenie `for` w języku Tcl służy do tworzenia pętli. Pętla `for` nie różni się zasadniczo od pętli tego rodzaju znanych z innych języków programowania, na przykład C. Instrukcja `for` wymaga podania czterech argumentów. Pierwszym z nich jest skrypt, służący zwykle do inicjalizowania licznika pętli. Drugi to wyrażenie, którego wartość jest obliczana

przy każdym przebiegu pętli; na tej podstawie podejmowana jest decyzja, czy pętla ma być kontynuowana. Trzeci argument najczęściej używany jest do zwiększania wartości licznika pętli, czwarty zaś – to skrypt Tcl, który będzie wykonywany w każdym przebiegu pętli. Oto przykładowa pętla `for`:

```
for {set i 0 } {$i<10} {incr $i 1} {  
    puts [expr 2*$i]  
}
```

Powyższa pętla wykonana zostanie 10 razy. Licznik `i` inicjalizowany jest wartością 0. Ciało pętli (polecenie `puts` wyświetlające wartość zmiennej `i` pomnożoną przez 2) wykonywane jest tak długo, jak długo wartość zmiennej `i` jest mniejsza od 10. Po każdym przebiegu pętli wartość zmiennej `i` zwiększana jest o 1. Po wykonaniu powyższego fragmentu programu na ekranie zostaną wyświetlane wartości:

```
0  
2  
4  
6  
8  
10  
12  
14  
16  
18
```

Polecenie while

Polecenie `while` jest również poleceniem służącym do tworzenia pętli, bardzo podobnym do polecenia `for`. Jedyna ważna różnica polega na tym, że polecenie `for` pozwala na stosowanie bardziej wyszukanych warunków wejścia do i wyjścia z pętli. Składnię polecenia `while` pokażemy na przykładzie:

```
set i 0  
while {$i < 10} {  
    puts [expr 2*$i]  
    set i [expr $i+1]  
}
```

Działanie tego przykładu jest identyczne jak programu w poprzednim podrozdziale, opisującym pętlę `for`. Program oblicza wartość $\$i^2$ w każdym przebiegu pętli i wyświetla ją na ekranie. Zauważ, że teraz o zwiększeniu licznika zadbać trzeba w skrypcie, który jest wykonywany w pętli. W przypadku instrukcji `for` można było zrobić to podając odpowiednie polecenie jako trzeci parametr.

Polecenie switch

Polecenie `switch` zastępuje polecenie `if` w przypadku, gdy trzeba sprawdzić wiele warunków – wtedy po poleceniu `if` musiałaby wystąpić duża liczba polecień `elseif`. Porównuje ono zadaną wartość (zwykle przechowywaną jako wartość zmiennej) z dowolną

ilością wzorców; jeśli napotka wzorzec identyczny z wartością zmiennej – wykonuje powiązany z nią skrypt.

```
switch $rzecz {  
    auto {puts "Zmienna rzecz ma wartosc auto"}  
    rower {puts "Zmienna rzecz ma wartosc rower"}  
    pociag {puts "Zmienna rzecz ma wartosc pociag"}  
    default {puts "Nie rozpoznano wartosci zmiennej rzecz"}  
}
```

Polecenie `switch` jest odpowiednikiem instrukcji `case` znanej z Pascal'a i niektórych innych języków programowania. Porównuje ono wartość przechowywaną w zmiennej `rzecz` (wartość tę należy oczywiście odpowiednio ustalić przed wywołaniem polecenia `switch`) z łańcuchem znaków `auto`. Jeśli obie wartości są równe, wypisany zostanie tekst `Zmienna rzecz ma wartosc auto`. W przeciwnym przypadku wartość zmiennej `rzecz` porównana zostanie z tekstem `rower`, jeśli porównanie wypadnie pomyślnie, wyświetlony zostanie tekst `Zmienna rzecz ma wartosc rower`, i tak dalej. Wartość `default` pozwala na zdefiniowanie akcji podejmowanej w przypadku, gdy żaden inny warunek nie jest prawdziwy. W naszym przykładzie jest to wyświetlenie tekstu `Nie rozpoznano wartosci zmiennej rzecz`.



Gdy chcesz sprawdzić, czy wartość zmiennej jest równa jednej z wielu wartości, zamiast `if` użyj instrukcji `switch` - powoduje to znaczne zwiększenie czytelności kodu.

Komentarze

Dołączenie do kodu programu napisanego w języku Tcl (i w każdym innym języku programowania) komentarzy jest zawsze dobrym pomysłem. Komentarze stają się szczególnie potrzebne, gdy:

- υ ktoś inny musi przeglądać napisany przez Ciebie program;
- υ Twój program staje się duży;
- υ nie będziesz przeglądał kodu przez dłuższy czas po jego napisaniu.

Komentarzy nie można wstawiać w środku poleceń – muszą one występować pomiędzy poleceniami. Do oznaczenia komentarza używa się symbolu `#`.

```
#To jest prawidlowy komentarz  
set a 1 ; #To tez jest dobrze  
set a 1 #To nie jest poprawny komentarz;
```

Trzeci wiersz powyższego programu nie jest poprawny, ponieważ komentarz zaczyna się w środku polecenia. Pamiętaj, że Tcl interpretuje tekst aż do znaku nowego wiersza lub średnika jako jedno polecenie.

Tk - nowe możliwości Tcl

Wcześniej w tym rozdziale przedstawiliśmy krótkutki program napisany w języku Tk, wyświetlający przycisk z napisem „Ahoj, przygodo!”. Tk ma jednak o wiele większe możliwości. Poza przyciskami stworzyć można systemy menu, paski przewijania, rozwijane listy itp. W tym podrozdziale przyjrzymy się niektórym z elementów interfejsu graficznego, podamy również przykłady ich użycia.

Ramki

Ramki służą do przechowywania innych elementów. Nie mają, w przeciwieństwie do innych elementów sterujących, żadnych interesujących właściwości. Jedynymi widocznymi ich atrybutami są kolor tła i wygląd krawędzi. Krawędzie mogą być płaskie (`flat` – wartość domyślna), wypukłe (`raised`) i wklęsłe (`sunken`). Możesz poeksperymentować z tymi wartościami by przekonać się, jak poszczególne typy ramek prezentują się na ekranie.

Ramka płaska nie prezentuje się zbyt ciekawie. Wygląda dokładnie tak, jak domyślne okno `wish` (ponieważ domyślnie ramki są płaskie).

Przyciski

Przyciski służą do pobierania specyficznego typu informacji od użytkownika, który może je aktywować przez umieszczenie nad nimi kurSORA i wcisnięcie lewego przycisku myszy. Tk udostępnia trzy podstawowe typy przycisków:

- przycisk poleceń (ang. *button*),
- przycisk wyboru (ang. *check button*; przyciski tego typu mogą być załączone lub wyłączone),
- przycisk opcji (ang. *radio button*; jest to grupa przycisków, z których tylko jeden może być załączony).

Przyciski poleceń używane są do rozpoczęcia jakichś działań (na przykład ładowania pliku itp.). Przeważnie znajduje się na nich tekst odpowiadający czynności podejmowanej po naciśnięciu (na przykład `Load file`).

Przyciski wyboru pozwalają na zdecydowanie, czy jakaś opcja ma być załączona, czy nie. Przycisk zacieniony odpowiada załączeniu opcji, niezacieniony – wyłączeniu.

Przyciski opcji są podobne do przycisków wyboru, z tym że używane są w grupach. W danej chwili aktywny może być tylko jeden z nich. Przycisk zacieniony jest włączony. Przyciski takie używane są do ustawiania opcji wykluczających się wzajemnie.

Menu i przyciski menu

Przyciski menu używane są do implementowania menu rozwijanych, kaskadowych i podręcznych. Menu to obiekt wyższego poziomu, zawierający przyciski menu, z którymi skojarzone są odpowiednie wartości lub polecenia. Poniżej zebrano elementy, które mogą wchodzić w skład menu.

- **Elementy kaskadowe** – powodują wyświetlenie podmenu w momencie, gdy znajdzie się nad nimi kursor. Mają one funkcję podobną jak przycisk menu.
- **Przyciski polecenia** – powodują wykonanie skojarzonego z nimi polecenia po ich uaktywnieniu. Ich funkcja odpowiada funkcji niezależnych przycisków polecenia.
- **Przyciski wyboru** – przyciski te umieszczone w menu pozwalają na załączanie i wyłączanie pewnych opcji. Przycisk zacieniony odpowiada załączeniu opcji – podobną sytuację mieliśmy w przypadku niezależnych przycisków wyboru.
- **Przyciski opcji** – podobne do przycisków wyboru, z tym że używane w grupach. Wybranie jednego z nich powoduje wyłączenie innych.
- **Separatory** – służą do poprawienia wyglądu menu przez wstawienie poziomej linii. Nie jest z nimi związana żadna akcja.

Podstawowa różnica pomiędzy elementami menu a przyciskami niezależnymi jest taka, że elementy menu nie mogą istnieć samodzielnie, mogą występować tylko w kontekście obiektu menu.

Przycisk menu jest przyciskiem podobnym do przycisków polecenia. Naciśnięcie go nie powoduje jednak wykonania skryptu, ale otwarcie skojarzonego z nim menu. Zwykle na przycisku menu znajduje się tekst opisujący w jakiś sposób menu, które pojawi się po jego uaktywnieniu. Na przykład na przycisku, z którym skojarzone jest menu pozwalające na przeprowadzenie operacji typu zapisywanie czy wczytywanie plików, zwykle znajduje się tekst `File`.

Aktywować menu można umieszczając nad przyciskiem menu kursor i wciskając lewy klawisz myszy. Spowoduje to otwarcie skojarzonego z przyciskiem menu i wyświetlenie jego elementów na ekranie. Teraz można przesunąć wskaźnik myszy w dół i wybrać jeden z nich.

Menu `File` może przykładowo zawierać dwa przyciski polecenia (`Open`, służący do otwierania pliku i `Quit` – do zamknięcia programu), separator i element kaskadowy o etykiecie `Save As`. Menu pojawiające się po wybraniu elementu kaskadowego `Save As` może zawierać trzy przyciski polecień, oznaczone na przykład `Text`, `Ver1` i `Ver2`.

Listy rozwijane

Lista rozwijana pozwala użytkownikowi aplikacji na wybranie z listy jednej z wielu dostępnych pozycji. Jeśli pozycji jest więcej, niż można naraz wyświetlić, z listą należy skojarzyć paski przewijania, które umożliwiają dostęp do pozycji niewidocznych na ekranie.

Paski przewijania

Paski przewijania używane są do kontrolowania tego, co pojawia się w innych obiektach. Należy skojarzyć je z obiektem, który do wyświetlania informacji dysponuje ograniczoną ilością miejsca, a wyświetlane dane mogą nie mieścić się w tym obszarze. Przykładem takiego obiektu może być rozwijana lista zawierająca dużo elementów.

Podsumowanie

Ten rozdział przedstawia podstawowe informacje dotyczące języków Tk i Tcl. Choć porusza dość sporo zagadnień, pokazuje tylko niewielką część możliwości tych języków. Nie zostały opisane nawet tak podstawowe problemy, jak posługiwanie się tablicami, tworzenie funkcji i procedur itp. Również większość elementów interfejsu X została z konieczności pominięta. Jeśli chcesz poznać lepiej te języki, przeczytaj jedną z książek poświęconych im w całości.

Tcl to tylko jedno z wielu rozszerzeń możliwości języka C. Dostępnych jest również wiele innych pakietów wzbogacających ten język, np.:

- Ak** rozszerzenie pozwalające na nagrywanie i odtwarzanie dźwięku;
- XF** interaktywny system projektowania interfejsu;
- Tcl-DP** rozszerzenie ułatwiające dystrybucję oprogramowania.

Jeśli chcesz dowiedzieć się więcej o innych kompilatorach dostępnych z systemem Linux, przejdź do rozdziału 30. „Inne kompilatory”.

Tworzenie skryptów CGI, do czego znakomicie nadaje się Perl, opisane jest w rozdziale 52. „Skrypty CGI”

Rozdział 30.

Inne kompilatory

Tim Parker

W tym rozdziale:

- υ Ada
- υ COBOL
- υ DSP
- υ Eiffel
- υ FORTRAN
- υ LISP
- υ Modula-3
- υ OGI
- υ Scheme
- υ Scilab

Do tej pory przyjrzaliśmy się najpopularniejszym i najczęściej używanym językom programowania dla Linuxa. Jeśli jednak nie jesteś entuzjastą C, C++, Perla, awk czy Tcl, ale chciałbyś programować w języku Ada czy FORTRAN, nie trać nadziei. Dla systemu Linux dostępnych jest również wiele innych kompilatorów i narzędzi wspomagających tworzenie aplikacji w innych językach programowania. W tym rozdziale wymienimy dostępne obecnie języki programowania dla platformy Linux, podamy też wskazówki co do tego, gdzie można je znaleźć.

Na pewno nie uda się wymienić wszystkich dostępnych kompilatorów, ponieważ nowe wersje i ulepszenia pojawiają się właściwie co tydzień. Postaramy się jednak przedstawić najpopularniejsze z nich.

Ada

Język Ada jest językiem powszechnie używanym w zastosowaniach militarnych. W związku z jego rozpowszechnieniem wśród programistów tworzących tego typu aplikacje, powstało również kilka wersji tego języka dla systemu Linux. Najpopularniejszą z nich jest GNAT (Gnu Ada Translator), pakiet opracowany i pierwotnie wspierany przez New York University. Jeśli chcesz dowiedzieć się czegoś więcej na temat projektu GNAT lub sprawdzić, jaka jest jego najnowsza wersja, wyślij pocztą elektroniczną wiadomość pod adres gnat-request@cs.nyu.edu. Aktualna wersja kompilatora języka Ada dostępna jest w wielu węzłach FTP i musi zostać skompilowana za pomocą kompilatora GNU C.



Archiwum GNAT dostępne jest na serwerach uniwersytetu w Nowym Jorku, jak również w innych węzłach FTP. Najświeższą wersję znajdziesz zawsze pod adresem

tsx-11.mit.edu/pub/linux/packages/ada

GNAT obsługuje standardy Ada83 i Ada90 i jest kompatybilny z niektórymi kompilatorami komercyjnymi.

COBOL

COBOL to język programowania istniejący od bardzo dawna; prawdopodobnie w języku COBOL napisano więcej kodu niż w jakimkolwiek innym języku programowania. Istnieje kilka komercyjnych kompilatorów tego języka przeznaczonych dla systemu Linux, z których najważniejszy jest chyba COBOL-85 firmy Acucobol (szczegółowe informacje o nim można znaleźć pod adresem <http://www.acucobol.com>). Obecnie, choć w węzłach FTP można znaleźć różne narzędzia wspomagające pisanie programów w tym języku, nie jest dostępny żaden darmowy kompilator.

DSP

Pakiet do cyfrowej obróbki sygnałów (ang. *Digital Signal Processing*) dostępny dla Linuxa nazywa się ObjectProDSP. Jest to narzędzie obiektowe, posiadające interfejs graficzny oparty na systemie X, używane w wielu zastosowaniach inżynierskich i naukowych.



Kilka wersji pakietu ObjectProDSP dostępnych jest w węzłach FTP, na przykład pod adresem tsx-11.mit.edu/pub/linux/packages/dsp. Jeśli używasz przeglądarki HTML, powinieneś podać adres

<ftp://tsx-11.mit.edu/pub/linux/packages/dsp>

Z autorem pakietu ObjectProDSP można skontaktować się pod adresem
mtnmath@mtnmath.com.

Eiffel

Eiffel to obiektowy język programowania, mający szczególnie duże możliwości, jeśli chodzi o ponowne wykorzystanie kodu. Od czasu opracowania jego pierwszej wersji przez Bertranda Meyera w późnych latach osiemdziesiątych język ten bardzo się rozwinął. Dla wszystkich udostępniła go organizacja Nonprofit International Consortium for Eiffel (NICE). Posiada wszystkie najlepsze cechy, których mógłbyś spodziewać się po języku zorientowanym obiektowo.



Kompilator języka Eiffel dostępny jest w węzłach FTP i pod adresem
<http://www.cm.cf.ac.uk/Tower>

FORTRAN

FORTRAN (ang. *Formula Translator*) był językiem powszechnie używanym w zastosowaniach inżynierskich w latach sześćdziesiątych i siedemdziesiątych. Dla Linuxa dostępnych jest kilka implementacji standardu FORTRAN77, pojawiają się również niekomercyjne wersje implementujące standard FORTRAN90.



Kilka wersji kompilatorów języka FORTRAN dostępnych jest pod adresem tsx-11.mit.edu/pub/linux/packages/fortran. Również tam dostępne są programy narzędziowe dla programujących w tym języku.

Oprócz kompilatorów języka FORTRAN, dostępnych jest również wiele programów narzędziowych mających ułatwić pracę z tym językiem, nie wyłączając translatorów na język C.

Programiści powinni również zwrócić uwagę na zestaw programów narzędziowych i bibliotek mających rozliczne zastosowania, na przykład ułatwiających drukowanie czy formatowanie danych wyjściowych. Pakiet `mpfun` umożliwia obliczenia ze zmienną precyzją, nawet z dokładnością do 16 milionów cyfr znaczących (!). Istnieją również narzędzia pozwalające na konwersję pomiędzy standardami FORTRAN77 i FORTRAN90.

LISP

LISP jest językiem zaprojektowanym pod koniec lat pięćdziesiątych do programowania sztucznej inteligencji. Kompilator o nazwie `clisp` dostępny jest praktycznie z każdą

dystrybucją Linuxa, można go również znaleźć w każdym z węzłów FTP oferujących oprogramowanie dla Linuxa.

Modula-3

Modula-3 to poprawiona wersja języka Modula-2 (który z kolei wywodzi się z Pascala), opracowana przez Digital Equipment Corporation. Jest to język zorientowany obiektowo, zaprojektowany do tworzenia programów wieloprocesowych. Język Modula-3 oparty jest na interfejsie X.



Wersje języka Modula-3 dla systemu Linux dostępne są między innymi pod adresem gatekeeper.dec.com/pub/DEC/Modula-3 oraz ftp.vlsi.polymtl.ca/pub/m3/binaries/LINUX.m3.

Stronę WWW poświęconą temu językowi można znaleźć pod adresem

<http://www.research.digital.com/SRC/>

Co jakiś czas pojawiają się nowe wersje języka Modula-3, więc jeśli chcesz być „na bieżąco”, powinieneś co jakiś czas zaglądać pod wspomniane wyżej adresy i uaktualniać wersje kompilatora i narzędzi.

OGI

OGI Speech Tools to język przeznaczony do obsługi mowy – zarówno do jej syntez, jak i analizy. Zaprojektowany został w Center for Spoken Language Understanding. Używając go, możesz budować interpretery języka mówionego oraz bazy danych zawierające wymowę poszczególnych słów, skąd już tylko krok do interfejsu głosowego. Do pracy z tym językiem niezbędna jest karta dźwiękowa.



Najnowsze wersje języka OGI dostępne są pod adresem sunsite.unc.edu/pub/Linux/apps/sound oraz tsx-11.mit.edu/pub/linux/packages/ogi.

Dokumentacja pakietu OGI (który jest dość skomplikowany) również dostępna jest w węzłach FTP.

Scheme

Scheme to język programowania sztucznej inteligencji, będący połączeniem języków LISP oraz C. Jest bardzo elastyczny i zaskakująco łatwy do opanowania. Oprócz różnych wersji kompilatorów, w węzłach FTP dostępne są również translatory na inne języki programowania.

Scilab

Scilab to pakiet matematyczny pozwalający na obliczenia macierzowe, rysowanie wykresów i projektowanie funkcji, przeznaczony głównie dla naukowców i inżynierów. Nie jest tak potężny jak konkurujące z nim aplikacje komercyjne, ale jest bezpłatny. Został zaprojektowany przez francuski Institut de Recherche Informatique et Automatique (INRIA).



Scilab dostępny jest pod adresem

[ftp.inria.fr/INRIA/ Projects/Meta2/Scilab.](ftp://ftp.inria.fr/INRIA/Projects/Meta2/Scilab)

Zwykle można znaleźć tam kilka wersji pakietu Scilab - wybierz tę najnowszą.

Scilab jest językiem łatwym do opanowania nawet dla tych, którzy nigdy wcześniej nie korzystali z żadnego interaktywnego języka matematycznego. Pracuje w oknie systemu X i jest zadziwiająco elastyczny.

Podsumowanie

Powyżej przedstawiliśmy tylko kilka z wielu języków dostępnych dla systemu Linux. O wiele więcej możesz znaleźć, przeglądając archiwa FTP, których adresy podaliśmy w dodatku A „Węzły FTP i grupy dyskusyjne poświęcone Linuxowi”. Jeśli szukasz konkretnego języka, polecamy użycie jednej z wyszukiwarek internetowych.

Język Smalltalk – oraz jego wersja przeznaczona do pracy w systemie X – przedstawiony jest w rozdziale 31. „Smalltalk”.

Jak skonfigurować obsługę poczty elektronicznej, która może oddać nieocenione usługi gdy natkniesz się na jakieś kłopoty podczas używania kompilatorów, dowiesz się z rozdziału 40. „Konfigurowanie poczty”.

Ladowanie i przeglądanie grup dyskusyjnych, pozwalających być na bieżąco z wydarzeniami dotyczącymi każdego z języków programowania, omówione jest w rozdziale 41. „Konfigurowanie grup dyskusyjnych”.

Rozdział 31.

Smalltalk/X

Rick McMullin

W tym rozdziale:

- υ Co to jest Smalltalk/X
- υ Instalacja Smalltalk/X
- υ Uruchamianie systemu Smalltalk/X
- υ Środowisko Smalltalk/X
- υ Przeglądarki
- υ Edycja w przeglądarce obiektów
- υ Inspector
- υ Debugger

W tym rozdziale opiszemy aplikację Smalltalk/X, która jest kompletną implementacją środowiska programistycznego Smalltalk-80. Jeśli używalesz kiedyś jakiejś innej wersji języka Smalltalk, będziesz zaskoczony tym, jak wiele może Ci zaoferować jego darmowa wersja. Po przeczytaniu tego rozdziału będziesz orientował się w możliwościach Smalltalk/X, będziesz również potrafił pracować z interfejsem udostępnianym przez to środowisko.

Co to jest Smalltalk/X

Zanim powiemy, co to jest Smalltalk/X, warto może opisać w kilku zdaniach sam język Smalltalk. Smalltalk to obiektowy język programowania, powstały we wczesnych latach siedemdziesiątych. Nie był to pierwszy język obiektowy, ale był pierwszym, który stał się powszechnie używany w przemyśle.

Język ten jest popularny dopiero od kilku lat. Wykładowany jest na wielu uczelniach technicznych, używany w niejednej firmie ze względu na łatwość i szybkość projektowania programów.

Smalltalk/X opracowany został przez Clausa Gittingera. Pierwsza wersja tego systemu ujrzała światło dzienne w roku 1988. Dziś jest to kompletne środowisko programistyczne, zawierające takie, narzędzia jak przeglądarki klas i potężny debugger. Bardzo przydatna bywa możliwość przetłumaczenia programu w języku Smalltalk na C. Dzięki temu można osiągnąć większą szybkość działania programów.

Instalacja Smalltalk/X

Smalltalk/X nie jest instalowany automatycznie przez program instalacyjny, ale znajduje się na dołączonym do tej książki dysku CD-ROM. Jego pliki znajdziesz w katalogu `devel/smalltalkx`. Aby go zainstalować, zaloguj się jako `root`, a następnie:

1. utwórz katalog `/usr/local/lib/smalltalk`;
2. skopiuj do utworzonego katalogu następujące pliki:

```
bitmaps.tar.Z  
doc.tar.Z  
exe.tar.Z  
goodies.tar.Z  
source.tar.Z
```

3. rozpakuj skopowane pliki wydając kolejno polecenia:

```
uncompress *.Z  
tar -xf bitmaps.tar  
tar -xf doc.tar  
tar -xf exe.tar  
tar -xf goodies.tar  
tar -xf source.tar
```

4. usuń niepotrzebne już pliki `.tar`:

```
rm *.tar
```

Smalltalk/X powinien być gotowy do uruchomienia. Jeśli nie masz prawa zapisu do katalogu `/usr/local/lib`, możesz zainstalować go gdzieś indziej, ale wówczas musisz odpowiednio zmodyfikować wartość zmiennej `SMLLTALK_LIBDIR`.

Uruchamianie systemu Smalltalk/X

Aby uruchomić system Smalltalk/X (ST/X), wystarczy w oknie `xterm` wydać polecenie `smalltalk`

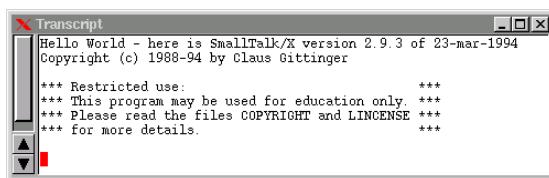
Po uruchomieniu, ST/X ładuje plik konfiguracyjny, nazywany plikiem obrazu. Jeśli go nie znajdzie, na podstawie danych zapisanych w pliku `smalltalk.rc` ustawia domyślny wygląd środowiska. Domyślnie ładowany jest plik o nazwie `st.img`, zawierający informacje o stanie środowiska w chwili, gdy zostało ono ostatnio zamknięte. Można również użyć pliku o innej nazwie, podając ją po opcji `-i`:

```
smalltalk -i nazwapliku.img
```

Środowisko Smalltalk/X

Po uruchomieniu ST/X pojawiają się dwa okna (nazywane również widokami): Transcript oraz Launcher. Rysunek 31.1 przedstawia okno Transcript.

Rysunek 31.1.
Okno Transcript



Okno Transcript pełni funkcję konsoli, do której kierowane są informacje systemowe. Okno Launcher przedstawione jest na rysunku 31.2.

Rysunek 31.2.
Okno Launcher



Okno Launcher zawiera menu pozwalające na szybkie uruchomienie wszystkich dostępnych w systemie ST/X narzędzi. Elementy tego menu opisane są skrótnie w tabeli 31.1.

Tabela 31.1. Elementy menu Launcher

Element	Opis
Browsers	Uruchamia podmenu dające dostęp do różnego rodzaju przeglądarek obiektów.
Workspace	Wywołuje widok obszaru roboczego.
FileBrowser	Pozwala manipulować plikami i katalogami.

cd. na następnej stronie

Tabela 31.1.cd. Elementy menu Launcher

Element	Opis
Projects	Pozwala na utworzenie nowego lub załadowanie istniejącego projektu.
Utilities	Pozwala na uruchomienie narzędzi programistycznych.
Goodies	Udostępnia narzędzia nie związane bezpośrednio z programowaniem.
Games & Demos	Pozwala na uruchomienie kilku przykładowych programów i gier.
Info & Help	Zawiera opis środowiska i języka programowania.
Snapshot	Powoduje utworzenie pliku konfiguracyjnego i zapisanie go pod podaną przez użytkownika nazwą.
Exit	Kończy pracę środowiska ST/X.

Poniżej opiszemy nieco dokładniej poszczególne elementy menu Launcher.

Przeglądarki (Browsers)

Podmenu **Browsers** daje użytkownikowi dostęp do przeglądarek i edytorów, pozwalających poruszać się w hierarchii klas, oglądać kod źródłowy metod itd. W podmenu tym dostępne są następujące opcje:

- υ przeglądarka systemowa (System Browser)
- υ przeglądarka hierarchii klas (Class Hierarchy Browser)
- υ klasy implementujące daną metodę (Implementors)
- υ klasy wysyłające dany komunikat (Senders)
- υ przeglądarka wprowadzonych zmian (Changes Browser)
- υ przeglądarka katalogów (Directory Browser)

Każda z tych opcji omówimy dokładniej.

Przeglądarka systemowa (System Browser)

Okno przeglądarki systemowej przedstawione jest na rysunku 31.3.

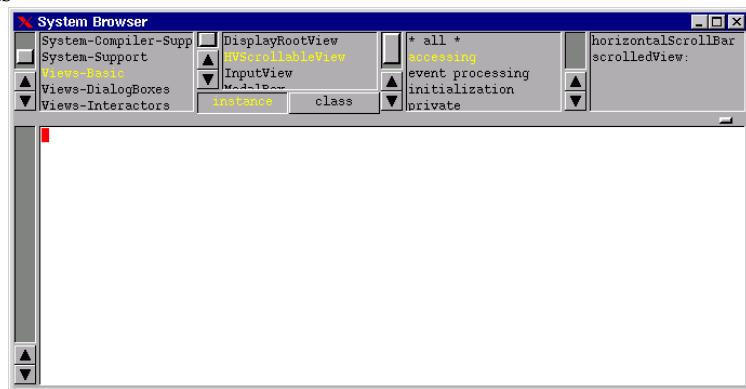
Zawiera ono pięć podwidoków:

- υ Lista kategorii klas

- Lista klas

Rysunek 31.3.

Okno System
Browser



- Lista kategorii metod

- Lista metod

- Przeglądarka kodu źródłowego

W systemie ST/X klasy podzielone są na kategorie. Kategoria to po prostu jeden z atrybutów każdej klasy, dzięki którym można łączyć je w logiczne grupy, co znacznie ułatwia zarządzanie nimi. Aby wybrać kategorię klas, wybierz jej nazwę z listy znajdującej się z lewej strony górnej części okna. Spowoduje to wyświetlenie w podwidoku listy klas wszystkich klas należących do danej kategorii. Możesz również wybrać wszystkie kategorie (*all*), wtedy w podwidoku listy klas wyświetlane zostaną wszystkie klasy uporządkowane alfabetycznie, lub widok hierarchiczny (*hierarchy*), co spowoduje wyświetlenie klas w formie drzewa obrazującego dziedziczenie.

Po wybraniu klasy z listy klas, w następnym podwidoku wyświetlane zostaną wszystkie kategorie metod należących do tej klasy. Podobnie jak w przypadku kategorii klas, kategorie metod używane są do logicznego grupowania metod w zależności od spełnianych przez nie funkcji. Po wybraniu kategorii, w następnej części okna wyświetlane zostaną wszystkie metody należące do danej kategorii. W końcu po wybraniu którejś z metod, w dolnej części okna wyświetlony zostanie jej kod źródłowy.

W każdym z podwidoków dostępne jest również menu podrzczne wywoływanie średkowym klawiszem myszki. Menu dostępne w podwidoku kategorii klas przedstawione jest na rysunku 31.4 i opisane w tabeli 31.2.

Rysunek 31.4.
Menu dostępne
w podwidoku listy
kategorii klas

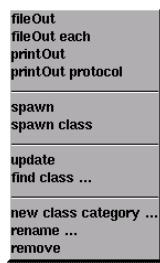


Tabela 31.2. Elementy menu podręcznego dostępnego w podwidoku listy kategorii klas

Element	Opis
FileOut	Zapisuje wszystkie klasy bieżącej kategorii do jednego pliku źródłowego o nazwie postaci kategoriaKlas.st.
fileOut each	Zapisuje wszystkie klasy, umieszczając każdą z nich w osobnym pliku o nazwie nazwaKlasy.st.
PrintOut	Drukuje kod źródłowy klasy, włącznie z implementacją metod.
printOut protocol	Drukuje specyfikację protokołu klasy, z pominięciem implementacji metod.
Spawn	Uruchamia na aktualnie wybranej kategorii przeglądarkę klas bez listy kategorii.
spawn class	Uruchamia pełną przeglądarkę klas, która pozwala edytować cały kod klasy w jednym oknie.
update	Ponownie odczytuje wszystkie dane o klasach.
find class	Uruchamia okno dialogowe, w którym można wpisać nazwę klasy, która ma być przeglądana.
rename	Pozwala zmienić nazwę kategorii klas.
remove	Usuwa kategorię wraz ze wszystkimi należącymi do niej klasami.

Menu podręczne dostępne w podwidoku listy klas pokazane jest na rysunku 31.5 i opisane w tabeli 31.3.

Rysunek 31.5.
Menu podręczne
dostępne
w podwidoku
listy klas

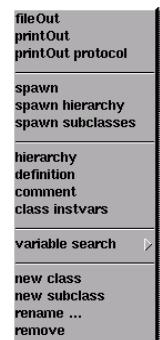


Tabela 31.3. Elementy menu podręcznego dostępnego w podwidoku listy klas

Element	Opis
fileOut	Zapisuje kod źródłowy aktualnie wybranej klasy do pliku o nazwie postaci nazwaKlasy.st.
printOut	Drukuje pełny kod źródłowy klasy.
printOut protocol	Drukuje specyfikację protokołu klasy (pomijając implementację metod).

Tabela 31.3. cd. Elementy menu podręcznego dostępnego w podwidoku listy klas

Element	Opis
spawn hierarchy	Uruchamia przeglądarkę klas z wybraną klasą i wszystkimi klasami potomnymi.
hierarchy	Pokazuje hierarchię aktualnie wybranej klasy w podwidoku przeglądarki kodu źródłowego.
definition	Pokazuje definicję klasy, pozwalając również na jej edycję.
comment	Wyświetla komentarz klasy, pozwalając również go edytować.
class instvars	Wyświetla zmienne wystąpienia danej klasy, pozwalając na ich edycję.
variable search	Pozwala szukać odniesień do zmiennych oraz metod odnoszących się do danej zmiennej.
new class	Pozwala utworzyć nową klasę, przy użyciu wybranej klasy jako szablonu.
new subclass	Tworzy klasę potomną w stosunku do aktualnie wybranej klasy.
Rename	Zmienia nazwę wybranej klasy.
Remove	Usuwa klasę i wszystkie klasy potomne.

Menu podręczne dostępne w podwidoku listy kategorii metod pokazane jest na rysunku 31.6 i opisane w tabeli 31.4.

Rysunek 31.6.

Menu podręczne
dostępne
w podwidoku listy
kategorii metod



Tabela 31.4. Elementy menu podręcznego dostępnego w podwidoku listy kategorii metod

Element	Opis
fileOut	Zapisuje kod źródłowy aktualnie wybranej kategorii metod do pliku o nazwie postaci nazwaKlasy-kategoria.st.

printOut	Drukuję kod źródłowy aktualnie wybranej kategorii metod.
spawn	Uruchamia przeglądarkę kategorii metod na wybranej kategorii wybranej klasy.
spawn category	Uruchamia przeglądarkę na wszystkich metodach danej kategorii we wszystkich klasach.
find method here	Wyszukuje implementację wybranego selektora.

cd. na następnej stronie

Tabela 31.4. cd. Elementy menu podręcznego dostępnego w podwidoku listy kategorii metod

Element	Opis
find method	Wyszukuje w hierarchii klas pierwszą klasę implementującą selektor wybrany w oknie dialogowym.
new cathegory	Pozwala dodać do listy nową kategorię metod.
copy category	Pozwala skopiować wszystkie metody wchodzące w skład kategorii klas do aktualnie wybranej klasy.
create access methods	Tworzy metody dostępu do zmiennych wystąpienia.
rename	Pozwala zmienić nazwę wybranej kategorii metod.
remove	Usuwa wybraną kategorię metod wraz ze wszystkimi metodami wchodzącymi w jej skład.

Menu dostępne w podwidoku listy metod pokazane jest na rysunku 31.7 i opisane w tabeli 31.5.

Rysunek 31.7.
*Menu dostępne
w podwidoku
listy metod*



Tabela 31.5. Elementy menu dostępnego w podwidoku listy metod

Element	Opis
fileOut	Zapisuje kod źródłowy aktualnie wybranej metody do pliku o nazwie postaci nazwaKlasy-selektor.st.
printOut	Drukuję kod źródłowy aktualnie wybranej metody.
spawn	Uruchamia przeglądarkę pozwalającą edytować wybraną metodę.
senders	Uruchamia przeglądarkę pozwalającą przeglądać metody wysyłające

	określony komunikat.
local sender	Podobnie jak <code>senders</code> , ale tylko w obrębie danej klasy i klas potomnych.
implementors	Uruchamia przeglądarkę pozwalającą przeglądać wszystkie metody implementujące dany komunikat.
globals	Uruchamia przeglądarkę pozwalającą przeglądać wszystkie metody korzystające z danego obiektu globalnego i metody wysyłające odpowiadający mu komunikat.

Tabela 31.5. cd. Elementy menu dostępnego w podwidoku listy metod

Element	Opis
<code>new method</code>	Tworzy nową metodę, używając aktualnie wyświetlanego kodu jako szablonu.
<code>change category</code>	Pozwala zmienić kategorię wybranej metody.
<code>remove</code>	Pozwala usunąć metodę.

Kiedy dodajesz lub usuwasz zmienne wystąpienia wchodzące w skład klas systemowych i zapisujesz wprowadzone zmiany, system tworzy nową klasę zamiast modyfikować istniejącą. Oryginalna postać klasy nadal istnieje, dzięki czemu istniejące wystąpienia mają nadal prawidłową klasę, ale sama klasa nie jest dostępna poprzez nazwę. Po zapisaniu zmian nie można edytować pierwotnej postaci klasy.



Modyfikowanie klas systemowych nie jest zalecane. Bezpieczniej jest skopiować klasę i modyfikować utworzoną kopię. Klasy systemowe wykorzystywane są przez środowisko ST/X i ich zmiana może spowodować jego niepoprawne funkcjonowanie.

W najniższej części okna wyświetlany jest kod źródłowy, który można edytować. Menu dostępne po naciśnięciu środkowego klawisza myszy zawiera polecenia typowe dla edytora tekstów. Są one opisane w podrozdziale „Edycja kodu za pomocą przeglądarki”.

Przeglądarka hierarchii klas (Class Hierarchy Browser)

Po uruchomieniu przeglądarki hierarchii klas wyświetlane jest okno dialogowe pozwalające wybrać nazwę klasy, od której należy rozpocząć przeglądanie. Okno przeglądarki jest podobne do okna System Browser, tyle że nie pojawia się podwidok listy kategorii klas. Dostępne w poszczególnych podwidokach menu podręczne są również takie same.

Klasy implementujące daną metodę (Implementors)

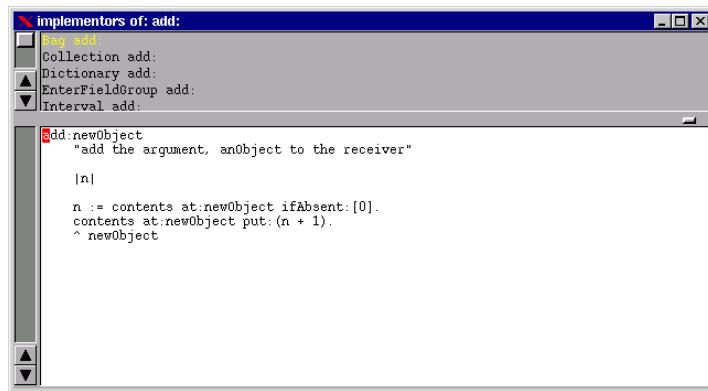
Po wybraniu opcji Implementors wyświetlane jest okienko dialogowe, w którym należy podać nazwę selektora. Selektor to nazwa typu operacji, która jest wywoływana przez komunikat u adresata tego komunikatu.

Jeśli podany zostanie prawidłowy selektor, w okienku Implementors wyświetcone zostaną implementacje metody określonej przez wybrany selektor we wszystkich klasach. Przykładowa zawartość takiego okienka pokazana jest na rysunku 31.8.

Menu podręczne dostępne w górnej części tego okienka nie różni się niczym od menu dostępnego w podwidoku listy metod przeglądarki System Browser, opisanego w podrozdziale „System Browser”.

Rysunek 31.8.

Okienko
Implementors

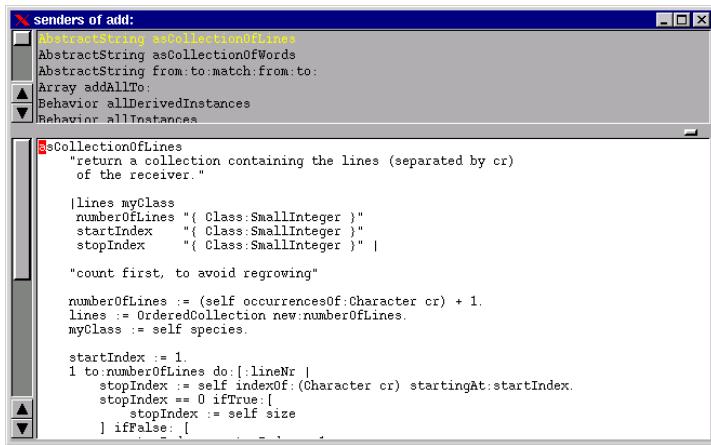


Klasy wysyłające dany komunikat (Senders)

Po wybraniu opcji Senders wyświetlane jest okienko dialogowe, w którym należy podać nazwę selektora.

Jeśli podany zostanie prawidłowy selektor, w okienku Senders wyświetcone zostaną implementacje metody wysyłającej wybrany komunikat. Przykładowa zawartość takiego okienka przedstawiona jest na rysunku 31.9.

Rysunek 31.9.
Okienko Senders



The screenshot shows the 'Senders' browser window with the title bar 'senders of add:'. The left pane lists methods: 'AbstractString asCollectionOfLines', 'AbstractString asCollectionOfWords', 'AbstractString from:to:match:from:to:', 'Array addAllTo:', 'Behavior allDerivedInstances', and 'Behavior allInstances'. The right pane displays the source code for the 'asCollectionOfLines' method:

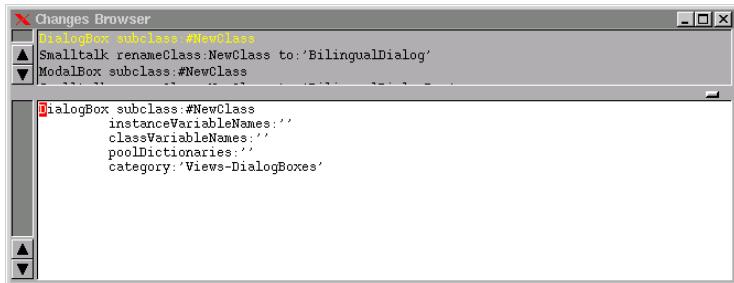
```
asCollectionOfLines
    "return a collection containing the lines (separated by cr)
     of the receiver."
    |lines myClass|
    numberofLines := (Class:SmallInteger) "
    startIndex := (Class:SmallInteger) "
    stopIndex := (Class:SmallInteger) |
    "count first, to avoid regrowing"
    numberofLines := (self occurrencesOf:Character cr) + 1.
    lines := OrderedCollection new: numberofLines.
    myClass := self species.
    startIndex := 1.
    1 to:numberofLines do: [:lineNr |
        stopIndex := self indexOf:(Character cr) startingAt:startIndex.
        stopIndex = 0 ifTrue:[
            stopIndex := self size
        ] ifFalse:[
            .
        ].
    ].
    ^ lines.
```

Menu podręczne dostępne w górnej części tego okienka nie różni się niczym od menu dostępnego w podwidoku listy metod przeglądarki System Browser, opisanego w podrozdziale „System Browser”.

Przeglądarka wprowadzonych zmian (Changes Browser)

Za każdym razem, kiedy dokonujesz jakichś zmian w hierarchii klas lub w kodzie źródłowym metody, ST/X zapisuje informacje o nich w specjalnym pliku. Okno Changes Browser pozwala na przeglądanie i modyfikowanie zawartości tego pliku. Dostępne są w nim dwa podwidoki: lista zmian w porządku chronologicznym i treść zmian. Przykładowa zawartość takiego okna pokazana jest na rysunku 31.10.

Rysunek 31.10.
*Okienko Changes
Browser*



The screenshot shows the 'Changes Browser' window with the title bar 'Changes Browser'. The left pane lists two changes: 'DialogBox subclass:#NewClass' and 'ModalBox subclass:#NewClass'. The right pane displays the details for the first change:

```
DialogBox subclass:#NewClass
    Smalltalk renameClass: NewClass to:'BilingualDialog'
    ModalBox subclass:#NewClass

DialogBox subclass:#NewClass
    instanceVariableNames: ''
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Views-DialogBoxes'
```

Aby obejrzeć treść zmiany, należy wybrać ją z listy. Zostanie ona przedstawiona w dolnej części okna. Menu podręczne dostępne w obszarze listy zmian opisane jest w tabeli 31.6.

Tabela 31.6. Elementy menu dostępnego w widoku listy zmian

Element	Opis
---------	------

apply change	Uaktywnia wybraną zmianę.
apply to end	Uaktywnia zmiany od wybranej do końca listy.
apply all changes	Uaktywnia wszystkie zmiany zapisane w pliku.
Delete	Usuwa wybraną zmianę z listy.
delete to end	Usuwa wybraną zmianę i wszystkie następne.
delete changes for	Usuwa wszystkie zmiany dotyczące tej samej klasy.
delete all changes	Usuwa wszystkie zmiany.
Update	Ponownie odczytuje plik zmian.
Compress	Zmniejsza rozmiar pliku zmian, usuwając wielokrotne zmiany tej samej klasy.
Version	Porównuje kod źródłowy metody z bieżącą wersją, wyświetlając informacje o różnicach w oknie Transcript.
make a change patch	Dołącza zmianę na koniec pliku poprawek przetwarzanego automatycznie przy uruchamianiu systemu ST/X.

cd. na następnej stronie

Tabela 31.6. cd. Elementy menu dostępnego w widoku listy zmian

Element	Opis
update sourcefile from change	Funkcja nie zaimplementowana.
Writeback	Zapisuje listę zmian z powrotem do pliku changefile. Wszystkie operacje usuwania pozycji listy i kompresowania listy nie dotykają tego pliku aż do momentu zapisania zmian.

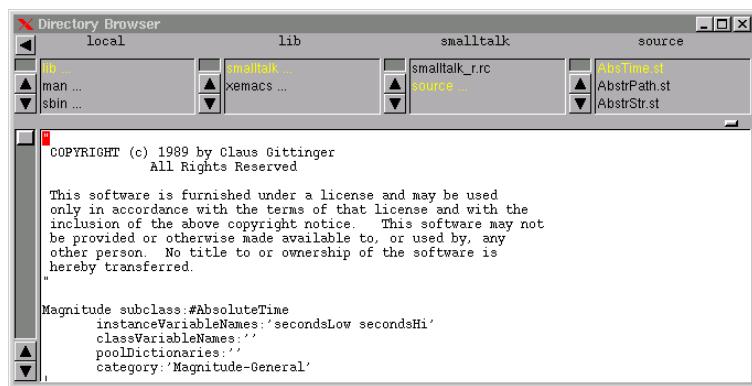
Przeglądarka wprowadzonych zmian jest szczególnie przydatna w momencie, gdy system okaże się niestabilny po wprowadzeniu jakichś nie do końca przemyślanych zmian – można wówczas wrócić do poprzedniego stanu. Aby panować nad rozmiarem pliku zmian, dobrze jest co jakiś czas go kompresować. Spowoduje to usunięcie informacji o zmianach wprowadzonych dawno, pozostawiając jednak dane o najświeższych modyfikacjach.

Przeglądarka katalogów (Directory Browser)

Po wybraniu opcji Directory Browser wyświetlane jest okno zawierające pięć podwidoków. W górnej części okna wyświetlany jest katalog bieżący i wszystkie jego podkatalogi oraz pliki w nim zapisane. Po wybraniu katalogu jego zawartość jest wyświetlana w podwidoku znajdującym się na prawo od niego. Po wybraniu pliku w dolnej części okna wyświetlana jest jego zawartość. Menu podręczne dostępne w podwidokach katalogów zawiera tylko dwa polecenia: `up` – które powoduje przejście do katalogu nadzawanego, i `goto directory` – pozwalające na przejście do wybranego katalogu. Menu podręczne

dostępne w dolnej części okna jest takie samo, jak we wszystkich okienkach służących do edycji tekstu i zostało omówione w podrozdziale „Edycja za pomocą przeglądarki”. Typowy wygląd okienka Directory Browser przedstawiony jest na rysunku 31.11.

Rysunek 31.11.
*Okienko Directory
Browser*



Workspace (obszar roboczy)

Wybranie opcji Workspace powoduje wyświetlenie okienka zawierającego obszar roboczy. Obszar roboczy to miejsce, z którego można skompilować kod napisany w języku Smalltalk. Jest on zwykle używany do testowania programów przed włączeniem ich do bibliotek.

File Browser (przeglądarka plików)

Okno File Browser pozwala na poruszanie się w systemie plików i manipulowanie nimi. Typowy wygląd tego okna przedstawiony jest na rysunku 31.12.

Rysunek 31.12.
Przeglądarka plików



Okno przeglądarki plików składa się z czterech podwidoków, opisanych w tabeli 31.7.

Tabela 31.7. Podwidoki okna FileBrowser

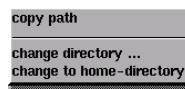
Podwidok	Opis
ścieżka	Wyświetla nazwę bieżącego katalogu.
typ plików	Pozwala na podanie wzorca nazwy plików, które mają być wyświetlane.
lista plików	Zawiera nazwy plików i podkatalogów bieżącego katalogu.
zawartość	Wyświetla zawartość wybranego pliku.

Jeśli chcesz obejrzeć zawartość pliku, kliknij dwukrotnie na jego nazwie. Podobnie jeśli chcesz zmienić katalog bieżący, kliknij dwukrotnie nazwę wybranego katalogu. Nazwy katalogów są wyświetlane bez względu na wybrany typ plików.

Domyślnym wzorcem nazwy plików jest gwiazdka (*), powodująca wyświetlanie wszystkich plików. Wzorzec ten można zmodyfikować przenosząc kurSOR do pola definiującego typ plików, wprowadzając odpowiednie symbole i wciskając klawisz Enter lub wybierając z menu podręcznego pozycję accept.

Podobnie jak w innych omówionych przeglądarkach, do każdego z podwidoków przypisane jest menu podręczne. W podwidoku ścieżki dostępne jest menu pokazane na rysunku 31.13.

Rysunek 31.13.
Menu dostępne
w podwidoku ścieżki



Funkcje dostępne w tym menu opisane są w tabeli 31.8.

Tabela 31.8. Elementy menu dostępnego w podwidoku ścieżki

Element	Opis
copy path	Kopiuje bieżącą ścieżkę do bufora.

change directory	Otwiera okno dialogowe pozwalające wprowadzić nazwę katalogu, do którego chcesz przejść.
change to home dir	Powoduje przejście do katalogu domowego.

Menu dostępne w podwidoku listy plików pokazane jest na rysunku 31.14 i opisane w tabeli 31.9.

Rysunek 31.14.

Menu dostępne w podwidoku listy plików



Tabela 31.9. Elementy menu dostępnego w podwidoku listy plików

Element	Opis
Spawn	Uruchamia kolejną przeglądarkę plików w bieżącym katalogu lub katalogu wybranym z listy plików.
get contents	Wyświetla zawartość aktualnie wybranego pliku.
show info	Wyświetla informacje o typie, rozmiarze i atrybutach wybranego pliku.
show full info	Wyświetla bardziej szczegółowe informacje o pliku, na przykład dane o czasie ostatniej modyfikacji, ostatniego otwarcia pliku itd.
FileIn	Laduje wybrany plik do systemu, odczytując i przetwarzając wszystkie zapisane w nim instrukcje.

Tabela 31.9. cd. Elementy menu dostępnego w podwidoku listy plików

Element	Opis
Update	Odczytuje ponownie zawartość katalogu, aktualizując listę plików.
execute UNIX command	Pozwala na wykonanie dowolnego polecenia systemu UNIX.
Remove	Usuwa wybrane pliki lub katalogi.
Rename	Pozwala zmienić nazwę wybranego pliku.
display long list	Wyświetla informacje o plikach w podwidoku listy plików. Po wybraniu tej opcji w menu podrzędnym wyświetlana jest opcja display short list, pozwalająca wrócić do poprzedniego stanu.
show all files	Powoduje wyświetlanie wszystkich plików, włącznie z plikami ukryтыimi. Po wybraniu tej opcji w menu podrzędnym wyświetlana jest opcja

	hide hidden files, pozwalająca wrócić do poprzedniego stanu.
create directory	Tworzy nowy katalog.
create file	Tworzy nowy plik.

Menu podręczne wyświetlane w podwidoku zawartości plików jest takie samo, jak we wszystkich okienkach służących do edycji tekstów i zostało omówione w podrozdziale „Edycja za pomocą przeglądarki”.

Projekty (Projects)

Opcja Projects menu Launcher pozwala utworzyć nowy projekt lub otworzyć projekt utworzony wcześniej. Po wybraniu funkcji new project automatycznie tworzony jest nowy projekt, a na ekranie wyświetlany jest obiekt nowego projektu. Funkcja select object powoduje wyświetlenie okna dialogowego pozwalającego wybrać jeden z utworzonych wcześniej projektów. Po wybraniu któregoś z projektów jest on ładowany do środowiska.

Narzędzia (Utilities)

Opcja Utilities udostępnia 13 narzędzi, które mogą być przydatne przy programowaniu w środowisku ST/X. Tabela 31.10 zawiera krótkie opisy tych narzędzi.

Tabela 31.10. Opcja Utilities

Narzędzie	Opis
Transcript	Otwiera okienko Transcript.
Window tree	Wyświetla drzewo reprezentujące strukturę hierarchii okien aktywnych lub będących w stanie oczekiwania w chwili wydania tego polecenia.

cd. na następnej stronie

Tabela 31.10. cd. Opcja Utilities

Narzędzie	Opis
Class tree	Wyświetla drzewo reprezentujące strukturę hierarchii klas wchodzących w skład systemu.
Event monitor	Wyświetla okienko pozwalające na monitorowanie wszystkich zdarzeń.
Process monitor	Wyświetla okienko zawierające informacje o wszystkich aktywnych i czekających procesach. Informacja jest uaktualniana w przypadku zmian na liście procesów.
Memory monitor	Wyświetla graf zawierający informacje o aktualnym zużyciu pamięci. Informacja jest uaktualniana przy zmianach zużycia

	pamięci.
collect Garbage	Uruchamia algorytm Generation Scavenge (wymiatanie), który wyszukuje obiekty krótkoterminowe i usuwa je. Jeśli obiekt przetrwa dość długo, zostaje skopiowany w oddzielnego obszar pamięci, skąd może być usunięty na żądanie użytkownika.
collect Garbage & compress	Ta opcja ma takie samo znaczenie, jak opcja collect Garbage, z tym że oprócz usuwania obiektów stosuje również kompresję, co pozwala na odzyskanie dodatkowego miejsca.
fullscreen hardcopy	Wykonuje zrzut ekranu, pozwalając następnie wybrać nazwę pliku, w którym zostanie zapisany obraz w formacie .tiff.
screen area hardcopy	Podobnie jak fullscreen hardcopy, z tym że pozwala wybrać fragment ekranu.
view hardcopy	Podobnie jak fullscreen hardcopy, ale tylko w odniesieniu do konkretnego widoku.
ScreenSaver	Pozwala wybrać jeden spośród trzech dostępnych w systemie ST/X wygaszacz ekranu.

Goodies

Wybranie opcji Goodies menu Launcher powoduje otwarcie podmenu, umożliwiającego dostęp do sześciu programów narzędziowych mogących przydać się w różnych sytuacjach, nie tylko podczas pracy z systemem Smalltalk/X. Programy te zostały opisane w tabeli 31.11

Tabela 31.11. Opcja Goodies

Program	Opis
Clock	Wyświetla prostokątny, analogowy zegar, dający możliwość wyłączania i włączania sekundnika.
Round Clock	Podobnie jak Clock, ale zegar jest okrągły i pozostaje widoczny kiedy jest zminimalizowany.

Tabela 31.11. cd. Opcja Goodies

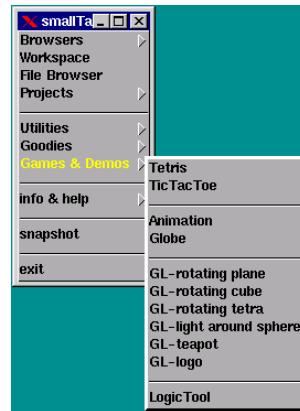
Program	Opis
Directory View	Wyświetla pliki i katalogi w postaci pictogramów. Teczka to odpowiednik katalogu, natomiast dokument symbolizuje plik.
Mail Tool	Narzędzie pozwalające na dostęp do poczty elektronicznej.
News Tool	Składnica nowości, informacji i dokumentów.
Draw Tool	Proste w obsłudze narzędzie służące do rysowania diagramów, wykresów, obrazków itp.

Gry i programy demonstracyjne (Games & Demos)

Po wybraniu opcji Games & Demos wyświetlane jest podmenu (pokazane na rysunku 31.15), pozwalające na dostęp do przykładowych aplikacji i gier.

Rysunek 31.15.

Menu dostępne po wybraniu opcji Games & Demos



Edycja za pomocą przeglądarki

Wszystkie widoki, w których wyświetlany jest tekst, pozwalają na użycie podstawowych funkcji edycyjnych dostępnych za pomocą menu podręcznego. Funkcje te zebrane zostały w tabeli 31.12.

Tabela 31.12. Funkcje edycyjne

Funkcja	Opis
again	Powtarza ostatnią operację edycyjną.
copy	Kopiuje wybrany tekst.
Cut	Wycina wybrany tekst z pliku.
paste	Wkleja tekst, który został wcześniej skopiowany lub wycięty w miejsce wskazywane przez kursor.
accept	Po zakończeniu edycji należy użyć tej opcji do zapisania wprowadzonych zmian – w przeciwnym przypadku zmiany nie zostaną zapisane.
doIt	Powoduje przetworzenie zaznaczonego tekstu.
printIt	Wyświetla reprezentację wyniku przetworzenia tekstu określonego przez

	aktualną pozycję kurSORA.
inspectIt	Wywołuje okno Inspector na wyniku opracowanego wyrażenia.
search...	Pozwala na wyszukiwanie określonego tekstu.
goto...	Pozwala przejść do określonej pozycji w pliku.
font...	Pozwala zmienić czcionkę używaną do wyświetlania zawartości pliku.
indent...	Pozwala zmodyfikować wcięcia.
save as...	Pozwala zapisać plik pod dowolną nazwą.
print	Drukujesz zawartość pliku.

Aby zaznaczyć (podświetlić) tekst, należy wcisnąć lewy przycisk myszy nad pierwszym znakiem wybranego tekstu, a następnie przeciągnąć wskaźnik myszy na koniec tekstu i zwolnić przycisk. Po ponownym wcisnięciu lewego przycisku zaznaczenie jest usuwane i można wybrać inny fragment tekstu.

Do przewijania wyświetlanego tekstu służą paski przewijania z lewej strony widoku. Po kliknięciu powyżej lub poniżej znacznika w obszarze paska przewijania, tekst przewijany jest jedną stroną do przodu lub do tyłu. Jeśli w tym samym czasie wcisnięty jest klawisz Shift, tekst przewijany jest od razu do pozycji wskaźnika w polu paska przewijania. Taka możliwość jest wykorzystywana głównie do szybkiego przemieszczania się w obrębie większych plików.

Okno Inspector

Inspector pozwala przyjrzeć się bliżej wybranemu obiektemu. Okno to składa się z dwóch podwidoków – w jednym z nich wyświetlane są nazwy zmiennych wystąpienia (atrybutów) obiektu, natomiast w drugim – wartość wybranej zmiennej wystąpienia. Okno Inspector można wywołać przez wybranie opcji `inspectIt` z menu edycyjnego lub też przez wysłanie do obiektu następującego komunikatu:

```
jakisObiekt inspect
lub
jakisObiekt basicInspect
```

Polecenie `basicInspect` powoduje otwarcie okna pokazującego zmienne wystąpienia faktycznie wchodzące w skład obiektu. Polecenie `inspect` jest dla niektórych klas przeddefiniowane w taki sposób, że wyświetlane są dane o logicznej zawartości obiektu.

Okno Debugger

Okno Debugger otwierane jest za każdym razem, gdy w kodzie źródłowym wystąpi błąd. Pokazuje ono miejsce, w którym wystąpił błąd, oraz pozwala zorientować się, w jaki sposób system do niego dotarł. Debugger może działać w trzech trybach: `normal`, `modal` i `inspecting`.

Jeśli podczas pracy w trybie `normal` wystąpi błąd w procesie nie będącym procesem obsługi zdarzenia, debugger jest uruchamiany nad błędnym procesem. Powoduje to zablokowanie dostępu do tego procesu i wszystkich jego widoków. Pozostałe widoki pozostają aktywne i odpowiadają w zwykły sposób.

Jeśli błąd wystąpi w procesie obsługi zdarzenia, debugger uruchamiany jest w trybie `modal`. W tym trybie nie można komunikować się z żadnym z widoków.

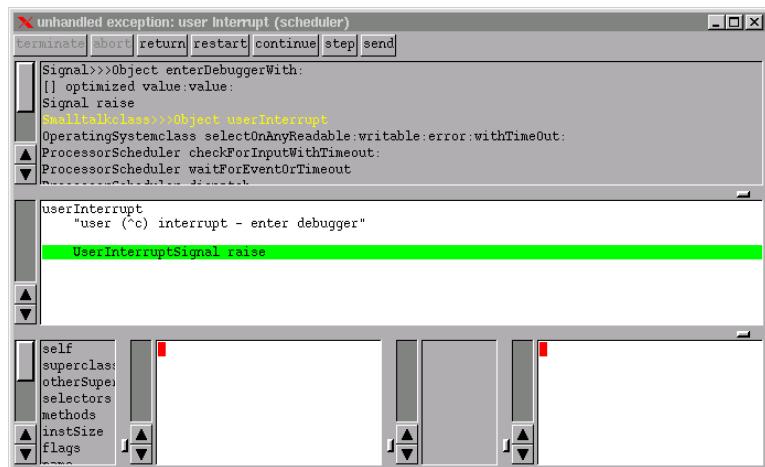
Tryb `inspecting` może zostać uruchomiony z okna ProcessMonitor za pomocą menu podręcznego i pozwala na obserwowanie stanu innych procesów. Ponieważ jednak obserwowany proces może kontynuować działanie, wyświetlane dane pochodzą z chwili, w której zostało wydane polecenie.

Okno Debugger zawiera cztery podwidoki:

- υ Context Walkback List – ślad programu, pozwalający prześledzić drogę, która doprowadziła do wystąpienia błędu,
- υ Method Source View – kod źródłowy metody, w której wystąpił błąd,
- υ Receiver Inspector – pozwala na obserwację obiektu, który jest odbiorcą wybranego komunikatu,
- υ Context Inspector – dostarcza informacji o argumentach i zmiennych lokalnych kontekstu.

Okno Debugger przedstawione jest na rysunku 31.16.

Rysunek 31.16.
Debugger



Funkcje wspólne dla wszystkich podwidoków dostępne są poprzez przyciski znajdujące się powyżej podwidoku śladu programu. Funkcje te opisane są w tabeli 31.13.

Tabela 31.13. Przyciski funkcyjne debugera

Przycisk	Opis
continue	Powoduje kontynuowanie wykonania programu.
terminate	Kończy działanie procesu powodującego błąd.
abort	Przerywa, jeśli to możliwe, wykonywane aktualnie czynności.
step	Pozwala na kontynuowanie procesu do czasu, aż w wybranym kontekście wykonań zostanie następne polecenie send.
send	Pozwala na kontynuowanie procesu aż do czasu wysłania pierwszej wiadomości.
return	Powoduje kontynuację działania tak, jakby wybrany kontekst zakończył działanie.
restart	Powoduje kontynuację działania uruchamiając ponownie wybrany kontekst.

W podwidoku śladu programu dostępne jest menu podręczne, którego funkcje zebrane w tabeli 31.14.

Tabela 31.14. Menu podręczne podwidoku śladu programu

Funkcja	Opis
exit smalltalk	Powoduje wyjście z systemu ST/X bez zapisywania pliku obrazu.
show more	Wyświetla następnych 50 kontekstów określających ślad wykonania programu.
breakpoints	Funkcja niedostępna w tej wersji systemu.
trace on/off	Funkcja niedostępna w tej wersji systemu.
trace step	Funkcja niedostępna w tej wersji systemu.

Jeśli przy uruchamianiu debugera wystąpi błąd, wyświetlane jest okno miniDebugger. Wyświetla on kolejne wiersze kodu źródłowego i nie pozwala korzystać z graficznego interfejsu użytkownika. Można go obsługiwać wpisując polecenia w oknie `xterm`, z którego został uruchomiony system ST/X. Po wcisnięciu klawisza `?` i klawisza Enter w odpowiedzi na znak zachęty, wyświetlona zostanie lista dostępnych w tym okrojonym debuggerze.

Podsumowanie

W tym rozdziale omówiliśmy skrótnie środowisko Smalltalk/X, pozwalające na programowanie w języku Smalltalk. Jeśli interesuje Cię ten język, a nie masz dostępu do komercyjnej wersji, wersja Smalltalk/X będzie dla Ciebie idealna. Oprócz omówionych w tym rozdziale programów narzędziowych i drobiazgów ułatwiających tworzenie programów zawiera ona również wiele programów przykładowych oraz sporo dokumentacji, mogącej znacznie ułatwić naukę programowania w języku Smalltalk.

Warto również wspomnieć o tym, że wraz z Linuxem rozprowadzana jest wersja GNU języka Smalltalk, o nazwie `mst`. W tym rozdziale zdecydowaliśmy się jednak na omówienie wersji Smalltalk/X, ponieważ jest ona o wiele bardziej kompletna.

Język Perl, stosowany głównie do szybkiego tworzenia prostych i użytecznych programów, omówiony jest w rozdziale 28. „Perl”.

Czytając rozdział 45. „Kopie zapasowe”, możesz dowiedzieć się, w jaki sposób należy wykonywać kopie zapasowe, dzięki którym programy w języku Smalltalk będą bezpieczne.

Jeśli chcesz skonfigurować swój system tak, by inni użytkownicy Internetu mieli do niego dostęp, przejdź do rozdziału 47. „Konfiguracja węzła internetowego”.

Część szósta

Linux dla administratorów

W tej części:

- υ Podstawy administrowania systemem
- υ Urządzenia
- υ Procesy
- υ Użytkownicy i konta
- υ Obsługa urządzeń SCSI
- υ Praca w sieci
- υ SLIP i PPP
- υ UUCP
- υ Konfiguracja poczty
- υ Konfiguracja grup dyskusyjnych
- υ Bezpieczeństwo w sieci
- υ NFS
- υ NIS i YP
- υ Kopie zapasowe
- υ cron oraz at

Rozdział 32.

Podstawy administrowania systemem

Tim Parker

W tym rozdziale:

- υ Konto `root`
- υ Uruchamianie i zamykanie systemu
- υ Montowanie systemów plików
- υ Kompresja danych – programy `gzip` i `compress`
- υ Program `tar`
- υ Kopie zapasowe
- υ Konfiguracja systemu

Do tej pory miałeś już okazję zobaczyć, jak Linux radzi sobie z najróżniejszymi zadaniami. Jednak nie omówiliśmy jeszcze wielu istotnych, choć rzadziej spotykanych problemów, występujących głównie w systemach, w których pracuje wielu użytkowników – z tego typu problemami radzić sobie musi tylko ich administrator. W tym rozdziale przyjrzymy się najważniejszym zadaniom administratora, a ponieważ jesteś prawdopodobnie administratorem swojego systemu, powinieneś zapoznać się z zawartymi w nim wskaźnikami. Omówimy między innymi:

- υ uruchamianie i prawidłowe zamykanie systemu;
- υ zarządzanie partycjami dyskowymi;
- υ tworzenie kopii zapasowych;
- υ obsługę programów `gzip`, `compress` i `tar`;

- υ wyświetlanie informacji dnia;
- υ używanie dyskietek startowych.

Oczywiście nie sposób omówić szczegółowo wszystkich zagadnień dotyczących wydajnego administrowania systemem – zamiast tego podamy tylko najważniejsze informacje, pozostawiając Ci pole do eksperymentowania. Dokładniejsze dane znajdziesz w dokumentacji rozprowadzanej wraz z Linuxem. Na rynku dostępnych jest również wiele książek dotyczących administrowania systemami UNIX-owymi, których treść w zasadzie w całości stosuje się do Linuxa.

Konto root

Jak już na pewno wiesz, na użytkownika `root` nie są nałożone żadne ograniczenia. Ma on pełny dostęp do wszystkich plików i katalogów, ma również kontrolę nad wszystkimi procesami. Taka „potęga” ma niestety również swoją drugą stronę: każda pomyłka może mieć opłakane skutki, włącznie ze zniszczeniem całego systemu.

Nieograniczone panowanie nad systemem jest nieodpartą pokusą dla wielu użytkowników systemów UNIX-owych. Należy jednak pamiętać, że zwykłe polecenie `rm` wyданie przez pomyłkę w niewłaściwym miejscu może kosztować Cię całe godziny pracy nad przywracaniem systemu do stanu używalności.

Z tych powodów konto `root` powinno być używane tylko w ograniczonym zakresie – do prac, które wymagają specjalnych uprawnień (jak na przykład komplilowanie nowego jądra, instalowanie oprogramowania czy konfigurowanie nowych systemów plików). Jako zasadę należy przyjąć, że nie używa się tego konta do codziennej pracy.

Oczywiście większość nowych użytkowników Linuxa i tak używa konta `root` na co dzień, ignorując wszystkie rady, ponieważ wydaje im się, że nie popełniają błędów. Niestety, od czasu do czasu każdy się myli. Zapytaj jakiegoś administratora, a na pewno opowie Ci, co przydarzyło mu się podczas używania konta `root`. Choć większość administratorów na początku nie stosuje się do tych rad, w końcu przekonują się na własnej skórze, dlaczego są one tak istotne.

Uruchamianie i zamykanie systemu

Istnieje kilka metod uruchamiania systemu linuxowego, podobnie jak istnieje kilka metod jego prawidłowego zamykania. Niektóre z nich zostały omówione już wcześniej. Ponieważ Linux może być zainstalowany na wiele sposobów, nie ma jednej „właściwej” metody uruchamiania go. W dalszej części tego podrozdziału przyjrzymy się bliżej procesowi uruchamiania systemu Linux z dysku twardego i dyskietki.

Uruchamianie systemu z dyskietki

Dyskietka startowa zawiera jądro systemu. Na niej również (zamiast na dysku twardym) założona jest partycja z głównym systemem plików (ang. *root partition*). Bez takiej partycji Linux nie potrafiłby znaleźć dysków twardych, na których zapisana jest reszta systemu operacyjnego.

Dyskietkę startową można utworzyć w trakcie instalacji systemu – w większości dystrybucji Linuxa program instalacyjny oferuje taką możliwość. Wiele wersji Linuxa pozwala również na utworzenie dyskietki startowej podczas rekompilacji jądra systemu, co może zaoszczędzić kłopotów w przypadku nieprawidłowego działania nowego jądra.

W zasadzie dyskietka startowa niezbędna jest tylko wtedy, gdy z jakichś powodów system nie chce uruchomić się normalnie (tzn. z dysku twardego). Pozwala ona na uruchomienie systemu i zamontowanie dysków twardych, a następnie sprawdzenie systemów plików sprawiających problemy i ewentualne usunięcie przyczyny problemów – na szczęście zdarza się to dość rzadko. Jeśli zdecydowałeś się nie używać programu LILO do ładowania Linuxa, możesz również potrzebować dyskietki, by uruchomić system. W takim przypadku dyskietka startowa działa tak samo jak DOS-owa dyskietka systemowa.

Można również samodzielnie stworzyć dyskietkę startową, kopując na nią jądro systemu z dysku twardego. Jądro systemu zwykle przechowywane jest w pliku o nazwie `vmlinuz`, `vmlinux`, `Image` lub `/etc/Image`, zależnie od dystrybucji. Jądro o nazwie `vmlinuz` jest skompresowane (rozpakowuje się automatycznie podczas ładowania do pamięci), dzięki czemu zajmuje mniej miejsca na dysku.

Po odnalezieniu pliku zawierającego jądro systemu należy zmienić partycję montowaną przez jądro jako partycja główna (ang. *root partition*). W naszym przypadku jako partycja główna ma zostać użytą partycja zapisana na dyskietce. Partycja główna ustawiana jest poleceniem `rdev`, którego składnia jest następująca:

```
rdev nazwa_jądra urządzenie
```

`nazwa_jądra` to nazwa pliku zawierającego jądro systemu. `Urządzenie` to linuxowa nazwa partycji, która ma zostać zamontowana jako partycja główna. Jeśli partycją główną używaną przez jądro `vmlinuz` ma być partycja zapisana na dyskietce w pierwszej stacji dysków, należy wydać następujące polecenie:

```
rdev vmlinuz /dev/fd0
```

Aby zmodyfikować proces uruchamiania systemu, można również wywołać polecenie `rdev` z innymi argumentami – dokładniejsze informacje na ten temat znajdziesz na stronach `man`.

Ostatnim krokiem procesu tworzenia dyskietki startowej jest skopiowanie na nią jądra systemu. Należy użyć sformatowanej wcześniej (na przykład w systemie DOS) dyskietki, dzięki czemu Linux będzie mógł rozpoznać typ i gęstość nośnika. Aby skopiować plik zawierający kernel na dyskietkę, wydaj polecenie:

```
cp vmlinuz /dev/fd0
```

Teraz dyskietka jest już gotowa do użycia. Niestety, w tej chwili nie ma możliwości uruchomienia systemu z dysku twardego – w tym celu należy ponownie zmienić partycję montowaną jako partycja główna (na przykład wydając polecenie `rdev vmlinuz /dev/hda6`). Po ponowym skonfigurowaniu partycji głównej można uruchomić system zarówno z dyskietki, jak i z dysku twardego, co umożliwia stworzenie kilku konfiguracji. Dyskietkę startową zwykle również utworzyć używając programu `setup`.

Uruchamianie systemu za pomocą programu LILO

LILO to program znajdujący się w sektorze startowym (ang. *boot sector*) dysku twardego i pozwalający na życzenie uruchamiać Linuxa bądź też, po odczekaniu zadanej liczby sekund, uruchamiać system operacyjny skonfigurowany jako domyślny.

Program LILO może być używany z systemami innymi niż Linux, na przykład takimi jak OS/2 czy DOS. Jeśli jego konfiguracja przewiduje automatyczne uruchamianie systemu Linux, to aby przerwać proces uruchamiania, należy wcisnąć klawisz Alt, Control lub Shift. Umożliwi to wybranie systemu operacyjnego, który ma zostać uruchomiony. Jeśli żaden system operacyjny nie jest skonfigurowany jako domyślny, za każdym razem trzeba będzie zdecydować, który system należy uruchomić.

W niektórych dystrybucjach Linuxa do konfigurowania procesu uruchamiania systemu służy plik `/etc/lilo` (lub `/etc/lilo.conf`); w innych proces ten jest konfigurowany przez program instalacyjny – w takim przypadku właśnie tego programu należy użyć do wprowadzania zmian konfiguracyjnych.

Zamykanie systemu

Wyłączenie zasilania na pewno nie jest dobrym sposobem na zakończenie pracy systemu. Taka sytuacja może spowodować uszkodzenie systemu plików, czasem nawet nieodwracalne. Ponieważ w systemie Linux przez cały czas otwartych jest wiele plików i działają różne procesy, wszystkie one muszą być prawidłowo zamknięte przed wyłączeniem zasilania.

Istnieje kilka metod zamykania systemu, ale najczęściej używa się w tym celu polecenia `shutdown`. Jego składnia jest następująca:

```
shutdown [minuty] [informacja]
```

`minuty` to liczba minut, jaką należy odczekać przed zamknięciem systemu. Tekst przekazany jako parametr `informacja` jest wyświetlany na konsolach wszystkich aktualnie zalogowanych użytkowników. Niektóre wersje polecenia `shutdown` pozwalają podać parametr `now` zamiast liczby minut, co powoduje natychmiastowe zamknięcie systemu; inne wersje w tym celu należy wywołać bez argumentów, jeszcze innym należy podać `0` jako liczbę minut. Można również zażądać, by po zamknięciu system został ponownie uruchomiony (za pomocą opcji `-r` – ang. *reboot*).

Użycie polecenia `shutdown` jest jak najbardziej na miejscu, jeśli z systemu korzysta ktoś oprócz Ciebie. Dzięki temu pozostały użytkownicy mają czas na zapisanie wyników swojej pracy i wylogowanie się z systemu. Polecenie to może również być używane do automatycznego przeładowywania systemu (na przykład codziennie o trzeciej nad ranem). Zalogowani użytkownicy zostaną uprzedzeni stosownym komunikatem.

Jeśli chcesz natychmiast zamknąć system, możesz użyć kombinacji klawiszy Alt+Control+Delete lub polecenia `halt`. Spowoduje to natychmiastowe zakończenie działania wszystkich procesów i wstrzymanie działania systemu, dzięki czemu można wyłączyć komputer.



Niektóre wersje Linuksa nie obsługują kombinacji klawiszy Alt+Control+Delete, co gorsza - starsze dystrybucje po ich naciśnięciu zatrzymywały system bez prawidłowego zakończenia działających procesów, co może powodować problemy. Powinieneś to sprawdzić w dokumentacji.

Montowanie systemów plików

Systemy plików nie są dostępne do momentu, aż zostaną zamontowane w głównym systemie plików. Nawet dyski twarde muszą być montowane (podczas uruchamiania systemu montowany jest tylko główny system plików). Do montowania systemów plików służy polecenie `mount`.

Podczas procesu uruchamiania systemu polecenie `mount` używane jest w czasie przetwarzania plików konfiguracyjnych (takich jak `/etc/rc` lub znajdujących się w katalogu `/etc/rc.d`) do zamontowania wszystkich systemów plików wyszczególnionych w pliku `/etc/fstab`. Każdy z nich posiada tam własny wpis, składający się z nazwy urządzenia, punktu zamontowania, typu systemu plików i opcji.

Za pomocą polecenia `mount` można również w każdej chwili dodać nowy system plików zapisany na dysku twardym, dysku CD-ROM, dyskietce czy dowolnym innym nośniku obsługiwany przez system. Składnia polecenia `mount` jest następująca:

```
mount system_plików punkt_zamontowania
```

`system_plików` to nazwa urządzenia, na którym zapisany jest system plików. Przykładowo, jeśli chcesz zamontować CD-ROM SCSI w katalogu `/usr/cdrom`, powinieneś wydać polecenie

```
mount /dev/cd0 /usr/cdrom
```

Katalog `/usr/cdrom` musi zostać utworzony wcześniej, w przeciwnym przypadku program `mount` generuje trudne do interpretacji komunikaty o błędach. W miejsce `/usr/cd0` należy oczywiście podstawić nazwę zainstalowanego w systemie napędu CD-ROM. Po prawidłowym zamontowaniu systemu plików będzie on dostępny w katalogu `/usr/cdrom`, tak jakby stanowił integralną część głównego systemu plików.

Jeśli w pliku `/etc/fstab` nie znajdują się żadne wpisy, do montowania systemów plików należy użyć nieco innej składni polecenia `mount`:

```
mount -t typ_systemu_plikow system_plikow punkt_zamontowania
```

Parametr `typ_systemu_plikow` może przyjmować jedną z kilku wartości obsługiwanych przez jądro systemu (zwykle `iso9660`, `msdos` itd.). Pozostałe parametry mają takie samo znaczenie jak w poprzednim przykładzie. Opcja `-t` używana jest, gdy w pliku `/etc/fstab` nie ma wpisu charakteryzującego dane urządzenie.

Montowanie dyskietki

System plików zapisany na dyskietce zamontować można poleciением podobnym jak w przypadku dysku CD-ROM. Aby zamontować w katalogu `/mnt` system plików zapisany na dyskietce włożonej do pierwszej stacji dysków, należy wydać polecenie

```
mount /dev/fd0 /mnt
```

W większości przypadków nazwy urządzeń stacji dyskietek zaczynają się od liter `fd`, podobnie jak w przypadku większości dysków twardych IDE – od liter `hd`. Jeśli system plików zapisany na dyskietce jest innego typu niż przyjmowany domyślnie, należy podać jego typ explicit, używając opcji `-t`, np. by zamontować system typu `ext2` należy wydać polecenie:

```
mount -t ext2, /dev/fd0 /mnt
```

Tworzenie nowego systemu plików

Aby utworzyć nowy system plików na dyskietce (który będzie można później zamontować w głównym systemie plików), powinieneś posłużyć się programem użytkowym `mke2fs` albo użyć polecenia `mkdev fs`, zależnie od wersji Linuxa. Przykładowo, jeśli chcesz za pomocą programu `mke2fs` utworzyć nowy system plików na 3.5-calowym dysku o pojemności 1,44 MB, powinieneś wydać polecenie:

```
mke2fs /def/fd0 1440
```

Odmontowywanie systemów plików

Aby odłączyć system plików od głównego systemu plików Linuxa, należy użyć polecenia `umount`. Aby np. odmontować system plików zapisany na dyskietce w stacji `/dev/fd0`, możesz wydać polecenie

```
umount /dev/fd0
```

System plików zapisany na dyskietce zostanie odłączony od głównego systemu plików. Zwróć uwagę, że nazwa polecenia brzmi `umount`, a nie `unmount`.

Jeśli chcesz wymienić dyskietkę znajdującą się w stacji dysków, musisz najpierw odmontować i wyjąć pierwszą dyskietkę, następnie włożyć drugą i zamontować ją. Sama wymiana dyskietki może spowodować nieprawidłowe działanie systemu (najczęściej objawiające się błędymi wydrukami zawartości katalogów, ale mogące prowadzić również do uszkodzenia danych).

Sprawdzanie systemów plików

Mimo wszystkich zabezpieczeń może zdarzyć się, że jakiś plik ulegnie uszkodzeniu lub tablica I-node nie będzie zawierała informacji zgodnych z bieżącą zawartością dysku. Sytuacje takie zdarzają się w zasadzie tylko w przypadku nieprawidłowego zamknięcia systemu, ale dobrym pomysłem jest sprawdzanie co jakiś czas systemu plików na okoliczność występowania błędów. Dostępne są różne programy potrafiące sprawdzać systemy plików, zależnie od wersji Linuxa. W niektórych systemach dostępny jest program `fsck`, w innych – `e2fsck`. Do sprawdzania systemów plików innego niż `ext2` typu niektóre wersje Linuxa udostępniają programy `xfsck` i `efsfsck`.

Jeśli chcesz użyć programu `e2fsck` do sprawdzenia systemu plików zapisanego na pierwszej partycji podstawowej pierwszego dysku twardego IDE, powinieneś wydać polecenie

```
e2fsck -av /dev/hda1
```

Opcja `-a` powoduje, że znalezione błędy są automatycznie naprawiane, natomiast `-v` – że wyświetlane są informacje diagnostyczne. Jeśli niezbędne okaże się wprowadzenie zmian w tablicy partycji, powinieneś jak najszybciej zrestartować system, pozwalając mu dostosować się do nowych ustawień.

Jeśli to tylko możliwe, zawsze dobrze jest odmontować system plików przed sprawdzeniem go. Dzięki temu można uniknąć problemów z otwartymi plikami. Nie można oczywiście odmontować głównego systemu plików, w takim przypadku należy uruchomić system z dyskietki, a następnie z niej uruchomić program sprawdzający.

Plik wymiany

Podczas instalowania Linuxa prawdopodobnie założyłeś partycję wymiany. Po zakończeniu instalacji poza partycją można również skonfigurować plik wymiany, zwalniając miejsce zajmowane na dysku przez partycję wymiany.

Generalnie plik wymiany działa nieco wolniej niż partycja wymiany, ponieważ w przesyłaniu danych udział bierze system plików, ale efekt ten nie ma większego znaczenia, jeśli w systemie działa w miarę szybki dysk i procesor. Plik wymiany jest użyteczny, gdy okazuje się, że przewidziana wielkość partycji wymiany nie wystarcza, a utworzenie większej partycji raczej nie wchodzi w grę, albo gdy trzeba uruchomić jakąś aplikację o bardzo dużych wymaganiach pamięciowych, na przykład kompilator.

Aby utworzyć plik wymiany, wydaj następujące polecenie:

```
dd if=/dev/zero of=/swap bs=1024 count=16416
```

Spowoduje ono utworzenie pliku wymiany o nazwie `swap`, o rozmiarze ok. 16 MB (16416 bloków – można oczywiście podać inny rozmiar i nazwę pliku).

Następnie za pomocą polecenia `mkswap` należy nadać plikowi odpowiednią strukturę:

```
mkswap /swap 16416
```

Liczba bloków musi zgadzać się z podaną w poprzednim poleceniu. Teraz pozostaje załączyć plik wymiany:

```
swapon /swap
```

Do wyłączenia pliku wymiany służy polecenie `swapoff`:

```
swapoff /swap
```

W niektórych wersjach Linuxa plik wymiany nie może być większy niż 16 MB, ale można używać do ośmiu plików i partycji wymiany jednocześnie.

Kompresja danych - programy gzip i compress

W każdym systemie w pewnym momencie na dysku zaczyna brakować wolnego miejsca. Oprócz usuwania plików jest jeszcze jedno rozwiązanie, pozwalające odzyskać nieco wolnej przestrzeni: kompresja danych. W systemach UNIX-owych i w Linuxie dostępnych jest kilka programów kompresujących, z których najczęściej używanymi są program `compress` i – dostarczany z nowszymi wersjami Linuxa – program `gzip`.

Po uruchomieniu, program `compress` zmniejsza rozmiar pliku z danymi i do jego nazwy dodaje rozszerzenie `.z`, dzięki czemu łatwo jest rozpoznać skompresowany plik. Składnia tego polecenia jest następująca:

```
compress nazwa_pliku
```

Aby skompresować kilka plików, można wykorzystać możliwości oferowane przez znaki specjalne interpretera polecenia powłoki. Program `compress` pozwala na użycie wielu opcji, ale rzadko jest to konieczne. Domyslnie podezas kompresowania oryginalny (nie skompresowany) plik jest usuwany; można zmienić to zachowanie za pomocą jednej z opcji podawanych w wierszu polecenia.

Do rozpakowania skompresowanego pliku służy polecenie `uncompress`:

```
uncompress nazwa_pliku.Z
```

Tu również można używać znaków specjalnych, na przykład by rozpakować wszystkie skompresowane pliki zapisane w bieżącym katalogu wydaj polecenie:

```
uncompress *.Z
```

Program `gzip` jest nowszy od programu `compress`, jest bardziej elastyczny i używa nieco innych algorytmów. Możliwe jest na przykład wybranie stopnia kompresji (im wyższy stopień kompresji, tym więcej czasu zabiera). W najprostszym przypadku składnia programu `gzip` jest taka sama, jak programu `compress`:

```
gzip nazwa_pliku
```

Można również podać opcję `-9`, która zamiast domyślnego załączy najwyższy stopień kompresji. Plik spakowany tym programem będzie miał rozszerzenie `.gz`, a oryginalny plik zostanie usunięty. Do dekompresji pliku spakowanego programem `gzip` należy użyć programu `gunzip` (można również użyć programu `gzip` z opcją `-d` – przyp. tłum.).

Program tar

Program `tar` (ang. *tape archiver*) używany jest w systemach UNIX-owych już od wiele lat. Niestety, nie jest zbyt łatwy w obsłudze, głównie dlatego, że nawet do wykonania najprostszych czynności konieczne jest podanie całego szeregu opcji.

Program `tar` przeznaczony jest do tworzenia pojedynczego pliku – archiwum – z wielu plików. Dzięki temu łatwiej jest przenosić czy tworzyć kopie zapasowe plików. Ogólna składnia polecenia `tar` jest następująca:

```
tar [opcje] [plik]
```

Lista opcji w niektórych przypadkach musi być – niestety – dość dłuża. Nazwy plików mogą zawierać symbole wieloznaczne. Oto prosty przykład użycia programu `tar`:

```
tar cvf arch1.tar /home/reksio
```

Powyższe polecenie spowoduje utworzenie archiwum `arch1.tar`, w którym znajdą się wszystkie pliki zapisane w katalogu `/home/reksio`. Opcja `c` powoduje utworzenie archiwum, `v` – wyświetlanie na ekranie informacji diagnostycznych, natomiast opcja `f` pozwala podać nazwę pliku wyjściowego – w tym przypadku `arch1.tar`.

Rozszerzenie `.tar` nie jest dodawane automatycznie do nazw archiwów; dodawanie go jest jednak dość powszechnie przyjętą konwencją dla oznaczania plików zawierających archiva utworzone programem `tar`, dzięki czemu łatwiej je potem zidentyfikować.

Opcja `c` powoduje utworzenie nowego archiwum (jeśli plik o danej nazwie już istnieje, zostanie usunięty). Opcja `u` (ang. *update*) pozwala dołączyć pliki do istniejącego archiwum lub utworzyć nowe, jeśli plik nie istnieje. Opcja `x` pozwala na wydobycie plików z archi-

wum. Aby wydobyć wszystkie pliki zarchiwizowane w poprzednim przykładzie, należy wydać polecenie:

```
tar xvf arch1.tar
```

Nie ma konieczności podawania nazw plików i katalogów, ponieważ zostaną one przywrócone podczas rozpakowywania. Trzeba pamiętać, że ścieżki dostępu są przechowywane w archiwum, jeśli więc powyższe polecenie wydane zostanie w katalogu `/home/reksio`, rozpakowane pliki znajdą się w katalogu `/home/reksio/home/reksio`. Za pomocą odpowiedniej opcji możliwe jest również wymuszenie rozpakowania plików do innego katalogu.

Program `tar`, w przeciwieństwie do programów kompresujących, nie usuwa archiwizowanych plików, trzeba więc zrobić to ręcznie.

Programu `tar` można również użyć do kopирования plików na dyskietkę czy taśmę, podając opcję `f` i nazwę urządzenia jako nazwę archiwum. Aby zarchiwizować pliki z katalogu `/home/reksio` na dyskietce, można wydać polecenie:

```
tar cvf /dev/fd0 /home/reksio
```

Problemy mogą pojawić się w momencie, gdy dyskietka ma zbyt małą pojemność. W takiej sytuacji należy użyć opcji `k`, dzięki której można podać pojemność nośnika:

```
tar cvfk /dev/fd0 1440 /home/reksio
```

Jeśli na dyskietce zabraknie miejsca przed zakończeniem archiwizacji, zostaniesz poproszony o włożenie następnej. Ważna jest kolejność parametrów: opcja `k` następuje po opcji `f`, więc najpierw należy podać pojemność nośnika, a potem nazwę pliku wyjściowego. Program `tar` wymaga, by wszystkie opcje były zgrupowane razem, co nie upraszcza jego użycia.

Ostatnim problemem związanym z archiwizacją na dyskietkach jest rozmiar bloków danych, który czasem również trzeba podać programowi `tar`. Stacje dyskietek zwykle wymagają, aby wielkość ta (reprezentująca liczbę bloków danych, które mogą być jednorazowo przesłane do urządzenia) wynosiła 4; polecenie przyjmuje więc postać:

```
tar cvfkb /dev/fd0 1440 4 /home/reksio
```

Czasem również zdarza się, że podanie ogólnej nazwy urządzenia (`/dev/fd0`) nie wystarcza i trzeba dokładnie podać typ dysku.

Bardziej kompletne informacje o tym programie znajdziesz na stronach `man` oraz w książkach poświęconych administrowaniu systemem.

Archiwa `tar` często poddawane są kompresji, dzięki czemu zajmują mniej miejsca (oczywiście równie dobrze można archiwizować skompresowane pliki). Nazwy plików zawierających takie archiwa mają postać:

```
nazwa_pliku.tar.gz
```

Aby odzyskać dane zapisane w takim pliku, trzeba go najpierw zdekompresować, a następnie wydobyć pliki za pomocą programu `tar`. Można to zrobić wyając polecenie:

```
gunzip nazwa_pliku.tar.gz | tar xvf -
```

Myśnik informuje program `tar`, że dane pobierać należy ze standardowego urządzenia wejściowego.

Programowi `tar` przyjrzymy się dokładniej w rozdziale 45. „Kopie zapasowe”.

Kopie zapasowe

Trzy podstawowe rzeczy, jakie należy robić, administrując systemem, to kopie zapasowe, kopie zapasowe i kopie zapasowe. Może zabrzmi to jak banał, ale kopia zapasowa może naprawdę Cię uratować, ilekroć zrobisz coś, zanim się zastanowisz, albo w przypadku awarii systemu. W systemach UNIX-owych większość kopii zapasowych trafia za pośrednictwem programu `tar` na taśmy, ale wielu użytkowników Linuxa nie posiada napędów taśmowych, dlatego też muszą wykonywać kopie zapasowe na dyskietkach.

Jak wspomniano wcześniej, kopie zapasowe wykonuje się za pomocą programu `tar`. Odpowiednia procedura została omówiona w poprzednim podrozdziale. Aby utworzyć kopię całego systemu plików na dyskietkach, należy wydać polecenie:

```
tar cvfbk /def/fd0 4 1440 /
```

Aby wykonać kopię zapasową na taśmie (zakładając, że napęd taśmowy nazywa się `/dev/rct0`) o pojemności większej niż rozmiar systemu plików (co eliminuje konieczność podania rozmiaru nośnika), można wydać polecenie:

```
tar cvfk /dev/rct0 20 /
```

W większości przypadków nie warto robić kopii całego systemu plików, ponieważ w razie awarii łatwiej jest go przeinstalować z dysku CD-ROM. Powinieneś jednak posiadać kopie plików użytkowników, zapisanych w katalogach `/usr` czy `/home`; warto również mieć kopię plików konfiguracyjnych.

Aby odtworzyć pliki z kopii zapasowej, należy ponownie użyć programu `tar`:

```
tar xvf /dev/rct0
```

Polecenie powyższe spowoduje odtworzenie wszystkich plików zapisanych na urządzeniu `/dev/rct0`. Można również zażądać odtworzenia tylko poszczególnych plików.

Na rynku dostępnych jest kilka komercyjnych programów narzędziowych, pozwalających na zautomatyzowanie procesu tworzenia kopii zapasowych, ale z równym powodzeniem można używać dostępnego za darmo programu `cron`.

Konfiguracja systemu

Warto poświęcić kilka chwil na to, by „dostroić” system tak, aby pracowało się z nim wygodniej i wydajniej. Polecenia służące temu celowi zależne są jednak od wersji systemu i używanych aplikacji. Przyjrzyjmy się kilku z nich.

Zmiana nazwy systemu

Nazwa systemu (ang. *host name*) zapisana jest w pliku `/etc/HOSTNAME`. Staje się ona istotna dopiero wtedy, gdy komputer jest podłączony do sieci. Na podstawie tej nazwy komputer jest identyfikowany przez inne systemy. Powinna więc to być nazwa niepowtarzalna (przynajmniej w obrębie sieci lokalnej) i mówiąca coś o komputerze. W praktyce jednak można nadać komputerowi dowolną nazwę.

Aby zmienić nazwę systemu, można zmodyfikować zawartość plików systemowych (po czym należy uruchomić system ponownie, aby zmiany przyniosły efekt), albo użyć polecenia `hostname`. Poniższe polecenie zmienia nazwę systemu na `hellfire`:

```
hostname hellfire
```

Dyskietka startowa

Posiadanie dyskietki startowej jest obowiązkowe w każdym poważniejszym systemie. Z jej pomocą można sprawdzić główny system plików, naprawić niektóre problemy związane z dyskiem twardym czy rozwiązać problem typu „zapomniałem hasła użytkownika `root`”. Dyskietkę startową można zwykle utworzyć za pomocą programów instalacyjnych dostarczonych z dystrybucją Linuxa. Odpowiednie pliki dostępne są również w węzłach FTP.

Po uruchomieniu systemu z dyskietki startowej, można zamontować potrzebne systemy plików za pomocą polecenia `mount`.

Zapomniałeś hasła użytkownika `root`?

Jest to dość kłopotliwe, ale rozwiązanie tego problemu w systemach linuxowych jest proste: uruchom system z dyskietki, zamontuj partycję montowaną normalnie jako partycję główną, edytuj plik `/etc/passwd` i usuń z niego hasło dla użytkownika `root`. Po ponownym uruchomieniu komputera możesz ponownie zmienić hasło.



Znaleźliśmy właśnie jedną z najważniejszych luk w bezpieczeństwie systemów linuxowych: każdy, kto posiada dyskietkę startową i może uruchomić z niej system, ma nieograniczony dostęp do sys-

temu.

Informacja dnia

Jeśli w Twoim systemie pracuje więcej niż jeden użytkownik, nie od rzeczy będzie informowanie o stanie systemu, nowym oprogramowaniu itp. Można to zrobić za pomocą wiadomości wyświetlanej przy logowaniu, zapisanej w pliku `/etc/motd` (ang. *message of the day*).

Do zmiany zawartości tego pliku można użyć dowolnego edytora tekstów. Plik może być dowolnie długi, ale użytkownicy wolą zwięzłe i treściwe informacje. Plik ten jest szczególnie przydatny do informowania o godzinach zamknięcia systemu, porach tworzenia kopii zapasowych czy o instalacji nowych wersji programów.

Podsumowanie

Administrowanie systemem linuxowym nie jest szczególnie skomplikowanym zadaniem, chyba że chcesz zagłębiać się w szczegóły konfiguracji systemu. Dla większości użytkowników wskazówki podane w tym rozdziale będą wystarczające. Jeśli chcesz dowiedzieć się czegoś więcej, powinieneś poszukać dobrej książki o administrowaniu systemami UNIX-owymi.

O tym, jak dodać do systemu urządzenia obsługujące interfejs SCSI (ang. *Small Computer Systems Interface*) przeczytać możesz w rozdziale 36. „Obsługa urządzeń SCSI”.

Konfigurowanie poczty elektronicznej omówione jest w rozdziale 40. „Konfigurowanie poczty”.

Program `tar` i problemy związane z tworzeniem kopii zapasowych przedstawione są dokładniej w rozdziale 45. „Kopie zapasowe”.

Rozdział 33.

Urządzenia

Tim Parker

W tym rozdziale:

- υ Urządzenia znakowe i blokowe
- υ Drukarki
- υ Terminale
- υ Modemy

Ten rozdział poświęcony jest urządzeniom, które podłączyć można do systemu linuxowego – takim jak terminal, modemy i drukarki. Omawia problemy dotyczące dodawania poszczególnych urządzeń i zarządzania nimi, jak również polecenia służące do tego celu.

Informacje przedstawione w tym rozdziale będą niezbędne w przypadku, jeśli chcesz, by system działał bez zarzutu. Nawet jeśli nie zamierzasz podłączać terminalu czy modemu, powinieneś wiedzieć co nieco o procesie uruchamiania systemu i o tym, jak obsługiwane są pliki konfiguracyjne.

Urządzenia znakowe i blokowe

Wszystko, co podłączone jest do komputera, traktowane jest przez Linuxa jednakowo – jako urządzenie. Nie ma znaczenia, czy jest to terminal, dysk twardy, drukarka czy modem. Wszystko, co potrafi wysyłać albo odbierać dane od systemu operacyjnego, jest urządzeniem.

Koncepcja traktowania wszystkich urządzeń jednakowo przeniesiona została z systemów UNIX-owych. Każdy typ urządzenia posiada w jądrze systemu przeznaczony do jego obsługi kod, nazywany sterownikiem. Zawiera on wszystkie, potrzebne do komunikowania się z urządzeniem. Nowo opracowane urządzenie może być używane w systemie linuxowym dopiero po napisaniu odpowiedniego sterownika.

Dzięki temu, że sterowniki urządzeń są zapisywane w osobnych plikach, mogą być one ładowane w razie potrzeby (w przypadku urządzeń używanych rzadko) lub przechowywane w pamięci przez cały czas pracy systemu. Jeśli urządzenia zostaną w jakiś sposób zmodyfikowane, możliwe jest również dołączenie do jądra odpowiednio dostosowanego sterownika, pozwalającego poprawić wydajność czy udostępniającego nowe możliwości.

Kiedy aplikacja odwołuje się do urządzenia, jądro systemu nie musi martwić się szczegółami jego obsługi. Po prostu przekazuje żądanie dalej, do sterownika, pozwalając mu obsłużyć komunikację. Jeśli na przykład wpisujesz coś z klawiatury terminalu, jego sterownik przyjmuje informacje o naciśnięciach klawiszy, a następnie przekazuje je do powłoki czy innej aplikacji, filtrując po drodze wszystkie kody specjalne, z którymi jądro systemu nie umiałoby sobie poradzić i zastępując je kodami zrozumiałymi dla kernela.

Pliki urządzeń przechowywane są w katalogu `/dev`. Wiąże się to z ogólnie przyjętą konwencją, ale można je przechowywać w dowolnym miejscu.

Urządzenia mogą komunikować się z systemem na dwa sposoby: przesyłając informacje znak po znaku lub blokami. Terminale, drukarki i modemy asynchroniczne są *urządzeniami znakowymi*, tzn. wykorzystują pierwszy mechanizm komunikacji, przesyłając jednorazowo tylko jeden znak, który jest odbierany przez urządzenie na drugim końcu połączenia. Dyski twarde i większość napędów taśmowych używa komunikacji blokowej, ponieważ pozwala ona na osiągnięcie większych prędkości przesyłania danych – są one nazywane *urządzeniami blokowymi*.



Urządzenia znakowe można również odróżnić od blokowych rozpatrując sposób, w jaki buforowana jest transmisja danych. Urządzenia znakowe zwykle same dokonują buforowania. Urządzenia blokowe, które przeważnie przesyłają i odbierają dane w blokach po 512 lub 1024 bajty, wymagają buforowania przez jądro systemu.

Niektoře urządzenia mogą działać zarówno jako urządzenia blokowe, jak i znakowe - dotyczy to np. niektórych napędów taśmowych. Posiadają one wtedy dwa osobne sterowniki, a wybór trybu komunikacji zależy od użytkownika.

Urządzenia blokowe i znakowe można rozróżnić wydając polecenie `ls -l` w katalogu `/dev`. W kolumnie zawierającej prawa dostępu pierwsza litera oznacza typ urządzenia: `c` – urządzenie znakowe, `b` – blokowe.

Pliki urządzeń zwykle mają nazwy pozwalające na ich identyfikację (na przykład nazwy terminali zaczynają się od liter `tty`, ang. *teletype*) i zawierające numer czy litery identyfikujące konkretne urządzenie (np. `tty1`, `tty1A` czy `tty04`). W połączeniu ze ścieżką dostępu `/dev` daje to pełne nazwy urządzeń, takie jak `/dev/tty01` itp.

Główne i poboczne numery urządzeń

W systemie linuxowym może działać więcej niż jedno urządzenie danego typu – można na przykład zainstalować kartę multiport, do której podłączonych jest 10 terminali. Dla

wszystkich tych terminali Linux może używać tego samego sterownika (zakładając, że są to terminale tego samego typu).

Musi jednak istnieć metoda pozwalająca systemowi zorientować się, który z terminali chcesz zaadresować. Służą do tego numery urządzeń. Numer główny identyfikuje sterownik, którego należy użyć, natomiast numer poboczny jest numerem samego urządzenia. Wszystkie terminale z powyższego przykładu będą więc posiadały taki sam numer główny, ale różne numery poboczne, dzięki czemu system będzie mógł je rozróżnić.

Każde urządzenie w systemie ma przypisaną unikalną parę numer główny-numer poboczny. Jeśli dwa urządzenia miałyby takie same numery, Linux nie potrafiłby komunikować się z nimi poprawnie.

Niektóre sterowniki urządzeń wykorzystują jednak te numery w dość dziwaczny sposób. Przykładowo, sterowniki napędów taśmowych czasem używają numeru pobocznego, by zidentyfikować gęstość zapisu danych na taśmie.

Pliki urządzeń tworzy się polecieniem `mknod` (ang. *make node*), a usuwa tak, jak normalne pliki – polecieniem `rm`.

Polecenie `mknod`

Polecenie `mknod` spełnia w systemie Linux kilka funkcji. Potrafi utworzyć kolejkę (ang. FIFO – *first in, first out*) oraz pliki urządzeń blokowych i znakowych. Jego składnia jest następująca:

```
mknod [opcje] urządzenie b|c|p|u nr_główny nr_poboczny
```

Opcje mogą przybierać następujące wartości:

- help:** wyświetlenie krótkiej instrukcji obsługi i zakończenie programu,
- m:** ustalenie praw dostępu do tworzonego pliku
(domyślnie 0666 – dozwolona jest tylko notacja symboliczna),
- version:** wyświetlenie informacji o wersji i zakończenie programu.

Argument po nazwie urządzenia określa jego typ; dostępne wartości to: `b` – urządzenie blokowe, `c` – urządzenie znakowe, `p` – kolejka (FIFO) i `u` – niebuforowane urządzenie znakowe. Trzeba użyć dokładnie jednej z tych wartości.

Następnie podać należy dwie liczby, określające główny i poboczny numer urządzenia. Oba te numery są wymagane w przypadku urządzeń znakowych, blokowych i niebuforowanych. Nie trzeba ich podawać w przypadku tworzenia kolejki.

Przykłady użycia tego polecenia pojawią się w dalszej części tego rozdziału, gdy omawiać będziemy dodawanie do systemu nowych urządzeń.

Drukarki

Drukarki są bardzo często używanymi urządzeniami i raczej nie sprawiają kłopotów administratorom. Łatwo je skonfigurować, o ile dostępne są dane techniczne sprzętu. Zarządzanie kolejkami drukowania również nie jest trudne, ale jak to często bywa w Linuksie, żeby system sprawował się bez zarzutu, trzeba znać kilka drobnych „sztuczek”.

Linux oparty jest na wersji BSD systemu UNIX, która – niestety – nie była najlepiej rozwiązana, jeśli chodzi o obsługę drukowania. Mimo tego, ponieważ Linux rzadko używany jest w bardzo dużych sieciach posiadających wiele drukarek, administrowanie nimi można sprowadzić do niezbędnego minimum. Trzeba jednak zdawać sobie sprawę z faktu, że polecenia służące do administrowania drukarkami w systemie BSD UNIX uważane są za niespójne i generalnie niezbyt przyjemne.

lpd - rezydentny program drukujący

Drukowanie w systemie Linux obsługiwane jest przez program rezydentny `lpd`. Podczas uruchamiania systemu `lpd` odczytuje zawartość pliku `/etc/printcap`, szukając sekcji dotyczących drukarek podłączonych do systemu. Używa on dwóch innych procesów, `listen` i `accept`, które przyjmują zlecenia drukowania kopując je do katalogu kolejki (ang. *spool directory*).

W większości przypadków modyfikacja zachowania programu `lpd` nie jest konieczna. Może się jednak zdarzyć, że trzeba będzie go zrestartować. Należy wtedy zakończyć odpowiedni proces (za pomocą polecenia `kill`), a następnie uruchomić go ponownie poleceniem:

```
lpd [-l] [port]
```

Opcja `-l` załącza rejestrowanie wszystkich zleceń drukowania, co czasem przydaje się przy szukaniu błędów w systemie drukowania. Parametr `port` jest internetowym numerem portu, który należy podać, jeśli nie chcemy używać domyślnych ustawień systemowych. Najprawdopodobniej nigdy nie będziesz musiał użyć tych opcji.

Rozmiar kolejek drukowania (każda drukarka posiada własny katalog, zawierający pliki do wydrukowania, nazywany po angielsku *spool directory*) jest określony w plikach o nazwie `minfree` zawartych w każdym z katalogów kolejek. Plik taki zawiera liczbę bloków dysku, które należy zarezerwować, dzięki czemu duże zadania drukowania nie zapełniają dysku twardego. Plik `minfree` może być modyfikowany za pomocą dowolnego edytora tekstów.

Nie każdy użytkownik ma jednak prawo drukować. Program `lpd` sprawdza zawartość plików `/etc/hosts.equiv` i `/etc/host.lpd`; jeśli nazwa systemu, z którego loguje się użytkownik, nie jest wymieniona w żadnym z tych plików, żądanie drukowania zostanie zignorowane. Ponieważ nazwa systemu lokalnego zawsze znajduje się w pliku `hosts.equiv`, jego użytkownicy zawsze mogą drukować.

Proces drukowania

Prześledźmy proces drukowania, idąc tropem zleceń wydruku. Jest ono generowane przez program `lpr`. Zbiera on razem dane przeznaczone do wydrukowania i kopiuje je do katalogu kolejki, w którym oczekują na obsłużenie przez program `lpd`.



Program `lpr` jest jedynym programem, który może wstawać pliki do kolejki drukowania. Wszystkie inne programy pozwalające na drukowanie działają w oparciu o ten program.

Sposób, w jaki plik ma zostać wydrukowany, można określić w wierszu poleceń (podając odpowiednie opcje polecenia `lpr`) lub ustalając odpowiednie wartości zmiennych środowiskowych – jeśli nie zostaną przekazane żadne argumenty i wartości odpowiednich zmiennych nie są zdefiniowane, użyte zostaną domyślne wartości systemowe.

Program `lpr` wie, do której kolejki ma dołączyć plik do wydrukowania, dzięki oznaczeniu drukarki docelowej. Nazwę drukarki docelowej można podać w wierszu poleceń lub ustalając wartość odpowiedniej zmiennej środowiskowej. Po określeniu, do której drukarki skierowane jest żądanie wydruku, `lpr` szuka wpisu w pliku `/etc/printcap` zawierającego informacje o tejże drukarce, między innymi o ścieżce dostępu do jej katalogu kolejki (zwykle `/usr/spool/nazwa_drukarki`, na przykład `/usr/spool/lp1`).

Następnie program `lpr` tworzy w katalogu kolejki dwa pliki. Nazwa pierwszego z nich zaczyna się od liter `cf` (ang. *control file*), po których następuje numer identyfikacyjny zlecenia drukowania. W tym pliku zawarte są informacje dotyczące zadania drukowania, między innymi identyfikator użytkownika, który wydał zlecenie drukowania. Drugi plik ma nazwę zaczynającą się od liter `df` (ang. *data file*) i zawiera dane, które mają zostać wydrukowane. Po zakończeniu tworzenia pliku z danymi program `lpr` informuje program `lpd` o fakcie, że w kolejce czeka na obsłużenie nowe zadanie.

Kiedy program `lpd` otrzymuje sygnał od `lpr`, sprawdza w pliku `/etc/printcap`, czy drukarka docelowa jest drukarką lokalną, czy sieciową. Jeśli jest to drukarka sieciowa, otwarte zostaje połączenie z systemem zdalnym, a następnie plik kontrolny i plik danych zostają przeniesione do tego systemu.

Jeśli drukarka jest drukarką lokalną, `lpd` sprawdza, czy jest ona dostępna i aktywna, a następnie wysyła zlecenie drukowania do programu rezydentnego obsługującego daną kolejkę drukowania.

Plik `/etc/printcap` i katalogi kolejki

Zarówno program `lpr`, jak i `lpd` korzysta z danych zapisanych w pliku `/etc/printcap`. Plik ten zawiera informacje o wszystkich drukarkach dostępnych w systemie linuxowym.

Format tego pliku jest prosty (i podobny do formatu pliku `/etc/termcap`, zawierającego informacje o terminalach). Oto przykładowy wpis:

```
# HP Laserjet
lp|hplj|laserjet-michal|HP LaserJet 4M w pokoju 425:\ 
:lp=/dev/lp0:\ 
:sd=/usr/spool/lp0:\ 
 lf=/usr/spool/errorlog:\ 
mx#0:\ 
of=/usr/spool/lp0/hpjlp:\
```

Pierwsze pole każdego wpisu to lista nazw danej drukarki (nazwy te mogą być używane zamiennie). Za pomocą tych nazw identyfikowana jest drukarka, która ma wydrukować określone dane – nazwa taka może być użyta przy ustalaniu wartości jednej ze zmiennych środowiskowych lub jako jeden z argumentów polecenia `lp`. Dostępne nazwy rozdzielane są kreską pionową.

Zwykle każda drukarka posiada co najmniej trzy nazwy: krótką, składającą się z czterech lub mniej znaków (np. `hplj`), nieco dłuższą, zawierającą np. identyfikator jej właściciela (np. `laserjet-michal`), oraz pełną, opisową nazwę, zawierającą informacje niezbędne do zidentyfikowania urządzenia (np. `HP LaserJet 4M w pokoju 425`).



Jeśli zlecenie drukowania nie zawiera informacji o drukarce docelowej, a informacja taka nie jest również dostępna poprzez zmienne środowiskowe, zadanie zostaje skierowane do drukarki o nazwie `lp`. Z tego powodu jedna z drukarek powinna posiadać nazwę (jedną z kilku) `lp`.

W pliku `/etc/printcap` komentarze oznacza się znakiem `#` na początku wiersza. Po nazwie drukarki następuje zestaw dwuznakowych nazw parametrów i ich wartości. Format tych wpisów jest następujący:

- NN** wartość logiczna
- NN=tekst** wartość tekstowa `tekst;`
- NN#liczba** wartość numeryczna różna od liczby `liczba`.

Kiedy używana jest wartość logiczna (czyli po nazwie parametru nie występuje ani tekst, ani liczba), parametr przyjmuje wartość True (czyli zero). Jeśli wymagana jest wartość False (różna od zera), po prostu nie należy wyszczególniać danego parametru.

Większość parametrów i ich wartości otoczonych jest dwukropkami, głównie dla zwiększenia przejrzystości pliku i ułatwienia przetwarzania go programom obsługi drukowania. Dozwolone są również przypisania puste, czyli dwa dwukropki obok siebie.

Kilka parametrów wartych jest odnotowania ze względu na ich przydatność podczas administrowania systemem. Nie wszystkie mogą pojawić się przy definicji każdej drukarki. Oto niektóre z nich.

sd Określenie ścieżki dostępu do katalogu kolejki.

- lf** Katalog, w którym rejestrowane będą komunikaty o błędach.
- mx** Określenie typów plików, które mogą być drukowane.
- af** Nazwa pliku, w którym mają być rejestrowane zadania drukowania.
- of** Filtr wyjściowy, który ma być używany przy drukowaniu.

Każda drukarka powinna posiadać własny katalog kolejki. Katalog taki jest niezbędny zarówno dla drukarki lokalnej, jak i sieciowej. Jeśli dodajesz do systemu nową drukarkę, to jest prawdopodobne, że o stworzenie takiego katalogu (poleciem `mkdir`) będziesz musiał zadbać sam. Prawa dostępu do takiego katalogu powinny mieć wartość 755. Właścicielem katalogu musi być użytkownik `root` lub `daemon`. Identyfikator grupy posiadającej ten katalog powinien również mieć wartość `root` lub `daemon`. W większości przypadków `daemon` jest lepszym rozwiązaniem (ze względów bezpieczeństwa), choć ustawienie `root` również jest poprawne.

Plik, w którym rejestrowane będą informacje o błędach, umieścić można w dowolnym miejscu. Może on być wspólny dla wszystkich drukarek, ponieważ każdy wpis i tak zawiera nazwę drukarki.

Plik, w którym rejestrowane są zlecenia drukowania, jest szczególnie przydatny w systemach, w których użytkownicy płacą za wydruki. Może być również użyty do celów statystycznych. Nowy wpis dodawany jest po zakończeniu drukowania. Wpisy mogą być wyświetlane za pomocą polecenia `pac` (więcej informacji na temat tego polecenia znajdziesz na stronach `man`).

Parametr `mx` pozwala określić, jakiego typu pliki mogą być drukowane. Zwykle jego definicja ma postać `mx#0`, co oznacza zezwolenie na drukowanie wszystkich typów plików.

Filtr wyjściowy modyfikuje format drukowanych danych, dostosowując je do wymagań drukarki. Przykładowo, niektóre drukarki laserowe nie mogą zmieścić na stronie 66 wierszy tekstu, więc filtr dzieli dokument na strony zawierające po 60 wierszy (wartość tę można zmienić). Czasem filtr jest również potrzebny, aby dodawać kody kontrolne, wymagane na przykład przy zmianie czcionki czy rodzaju papieru.

W każdym katalogu kolejki mogą znajdować się jeszcze dwa pliki: `status` i `lock`. Oba zawierają tylko jeden wiersz i mogą być modyfikowane za pomocą edytora tekstów. W plikach tych zapisana jest informacja o aktualnym stanie drukarki. Są one tworzone przez program `lpd` i używane przez niektóre programy do informowania o statusie drukarki.

Dodawanie drukarki za pomocą polecenia `mknod`

Linux obsługuje zarówno drukarki szeregowe, jak i równoległe. Drukarki obu typów są urządzeniami znakowymi. Niestety, większość wersji Linuxa nie zawiera programów pozwalających na łatwą konfigurację drukarki (programy takie są dostępne w większości

ści wersji UNIX-a), co pociąga za sobą konieczność ręcznego modyfikowania plików systemowych.

Drukarki równolegle zwykle nazywają się `lp0`, `lp1`, `lp2` itd., zależnie od adresu portu, do którego są podłączone (najczęściej spotyka się pojedynczy port równoległy, podłączona do niego drukarka nazywa się wtedy `lp0`). Oto dostępne nazwy urządzeń, odpowiadające im adresy portów i nazwy używane w systemie DOS:

<code>/dev/lp0</code>	<code>0x03bc</code>	<code>LPT1</code>
<code>/dev/lp1</code>	<code>0x0378</code>	<code>LPT2</code>
<code>/dev/lp2</code>	<code>0x0278</code>	<code>LPT3</code>



Aby określić adres portu równoległego, możesz posłużyć się jakimś programem diagnostycznym (np. DOS-owym `MSD.EXE`). Niektóre wersje BIOS-u wyświetlają adresy portów równoległych podczas uruchamiania komputera. Jeśli nie jesteś pewny, próbuj po kolei, zaczynając od `lp0` i sprawdzając, czy drukowanie jest możliwe. Pierwszy port równoległy w systemach PC ma zwykle adres `0x03bc`.

W systemie Linux do utworzenia pliku drukarki należy użyć polecenia `mknod`. Właścicielem takiego pliku musi być użytkownik `root` lub `daemon`.

Poniższe polecenia tworzą plik drukarki dołączonej do pierwszego portu równoległego (`/dev/lp0`):

```
mknod -m 620 /dev/lp0 c 6 0  
chown root.daemon /dev/lp0
```

Prawa dostępu do nowo utworzonego pliku `/dev/lp0` ustawiono na 620; plik ten jest plikiem urządzenia znakowego o numerze głównym 6 i pobocznym 0. Numery poboczne urządzeń zwykle rozpoczynają się od zera i są zwiększane o jeden dla każdego nowego urządzenia. Ponieważ dodawaliśmy pierwszą drukarkę w systemie, przypisaliśmy jej numer poboczny 0.



Właściciel `root.daemon` to specjalna składnia używana w systemie Linux dla programów rezydentnych uruchamianych przez użytkownika `root`. Wpis `root.daemon` nie pojawia się w pliku `/etc/passwd`. Pierwsza część identyfikatora (do kropki) oznacza użytkownika, natomiast druga - grupę.

Jeśli konfigurujesz jeszcze inne urządzenia, muszą one mieć oczywiście różne nazwy, np.:

```
mknod -m 620 /dev/lp0 c 6 0  
mknod -m 620 /dev/lp1 c 6 1  
mknod -m 620 /dev/lp2 c 6 2
```

W powyższych poleceniach numer poboczny urządzenia zwiększany jest tak, by odpowiadał numerowi portu. Nie jest to wymagane, ale może pomóc w identyfikacji urządzenia.

Po wydaniu powyższych poleceń dobrze jest jeszcze raz sprawdzić, czy właściciel pliku został ustawiony prawidłowo i czy utworzony został katalog kolejki. Jeśli katalog kolejki nie istnieje, trzeba założyć go ręcznie. Zagadnienia dotyczące właściciela i praw do dostępu do tego katalogu omówiliśmy w podrozdziale „Plik /etc/printcap i katalogi kolejki”.

Zarządzanie drukarkami - program lpc

Drukarki kontrolowane są za pomocą programu `lpc`. Pozwala on między innymi na:

- υ wyświetlanie informacji o statusie drukarki,
- υ włączanie i wyłączanie drukarki,
- υ włączanie i wyłączanie obsługi kolejki drukowania,
- υ usuwanie wszystkich oczekujących zadań drukowania,
- υ przesuwanie zadań na początek kolejki,
- υ wprowadzanie zmian w konfiguracji programu `lpd`.

Program `lpc` nie może być używany do zarządzania drukarkami sieciowymi, a tylko do zmiany ustawień drukarek podłączonych bezpośrednio do Twojego komputera.



Musisz wiedzieć o tym, że `lpc` to jeden z najmniej przewidywalnych i stabilnych programów dostarczanych z systemem Linux. Potrafi zawiesić się bez żadnej widocznej przyczyny, zdarza mu się również wyświetlać błędne informacje o statusie drukarki. Czasem jedyną drogą naprawienia zepsutego w jakiś sposób systemu drukowania jest zresetowanie komputera.

Program `lpc` wywołany bez żadnych argumentów oczekuje na wydanie polecenia. Poniżej przedstawiamy kilka poleceń tego programu wraz z ich argumentami.

- υ **abort nazwa_drukarki | all** Powoduje natychmiastowe wstrzymanie drukowania przez drukarkę o określonej nazwie (lub wszystkie drukarki, jeśli jego parametrem jest słowo `all`). Przerwane zadania drukowania są wstawiane ponownie do kolejki po uruchomieniu drukarki. Poza tym polecenie to działa podobnie do polecenia `stop`.
- υ **clean nazwa_drukarki | all** Usuwa wszystkie zadania drukowania, włącznie z aktywnymi. Często zdarza się jednak, że dokument, którego drukowanie się już rozpoczęło, drukowany jest do końca, ponieważ został już przesłany do programu `lpd` lub bufora drukarki. Jeśli użyty zostanie parametr `all`, wy czyszczone zostaną kolejki wszystkich drukarek.

- υ **disable nazwa_drukarki | all** Wyłącza kolejkowanie zadań. Wszystkie zadania oczekujące na wydruk zostaną obsłużone, ale nowe zadania nie będą przyjmowane – użytkownik próbujący wydrukować coś za pomocą takiej drukarki otrzyma komunikat informujący o tym, że drukarka jest wyłączona i zlecenie nie zostało przyjęte. Wyłączanie i włączanie kolejkowania sprowadza się do zmiany zawartości pliku `lock`.
- υ **down nazwa_drukarki wiadomość** To polecenie używane jest do całkowitego odłączenia drukarki. Jeśli dołączona jest wiadomość (o dowolnej długości), zostanie ona zapisana w pliku statusu drukarki (o nazwie `status`), dzięki czemu będzie wyświetlana przy próbach drukowania. Polecenia `down` używa się zwykle wtedy, gdy występują jakieś poważniejsze problemy z drukarką i będzie ona odłączona przez więcej niż jeden dzień.
- υ **enable nazwa_drukarki | all** Załącza kolejkowanie zadań dla danej drukarki lub wszystkich drukarek.
- υ **exit** Zakończenie pracy programu (polecenie to działa tak samo, jak polecenie `quit`).
- υ **help lub ?** Wyświetlenie listy wszystkich dostępnych poleceń. Jeśli jako argument tego polecenia podana zostanie nazwa innego polecenia, wyświetlona zostanie krótka informacja na jego temat (np. `help abort`).
- υ **quit** Zakończenie pracy programu `lpd` (polecenie to działa tak samo jak polecenie `exit`).
- υ **restart nazwa_drukarki | all** Zrestartowanie rezydentnego programu obsługi drukarki. Używane zwykle, gdy program ten zakończy pracę bez wyraźnej przyczyny (co się czasem zdarza). Jeśli podany zostanie argument `all`, restartowane są wszystkie programy obsługi drukarek.
- υ **start nazwa_drukarki** Polecenie to powoduje uruchomienie drukarki, pozwalając na wydrukowanie oczekujących dokumentów, oraz uruchomienie programu obsługującego kolejkowanie dla danej drukarki.
- υ **status nazwa_drukarki** Wyświetlenie nazwy drukarki, informacji o tym, czy kolejkowanie jest włączone, czy włączone jest drukowanie, liczby oczekujących dokumentów oraz statusu programu rezydentnego obsługującego drukarkę. Jeśli w kolejce nie ma żadnych dokumentów, nie będą działać również programy rezydentne. Jeśli jednak program rezydentny nie działa mimo tego, że w kolejce oczekują na wydrukowanie jakieś dokumenty, należy go zrestartować (za pomocą polecenia `restart`).
- υ **stop nazwa_drukarki** Polecenie to powoduje zatrzymanie drukarki. Zadania drukowania są nadal kolejkowane, ale nie są drukowane. Jeśli rozpoczęto drukowanie dokumentu, zostanie ono dokończone. Polecenia `start` i `stop` powodują zmianę zawartości pliku `lock` w katalogu kolejki. Polecenie `stop` powoduje również zakończenie działania rezydentnego programu obsługi kolejkowania.

- **topq nazwa_drukarki id_zadania** Powoduje przesunięcie zadania drukowania o identyfikatorze `id_zadania` na początek kolejki.
- **topq nazwa_drukarki id_użytkownika** Przesunięcie wszystkich zadań, których właścicielem jest użytkownik o danym identyfikatorze, na początek kolejki (polecenie bardzo przydatne dla administratorów, którzy nie mają czasu czekać!).
- **up nazwa_drukarki** Używane do ponownego przyłączenia odłączonej polecienniem `down` drukarki.

Program `lpq` nie jest szczególnie przyjazny dla użytkownika, ale jego użycie jest w zasadzie jedyną metodą zarządzania drukarkami i ich kolejkami w Linuxie. Zadania te mogą nieco ułatwić różne nakładki na ten program.

Zarządzanie kolejkami drukowania za pomocą programów `lpq` i `lprm`

Zamiast korzystać z programu `lpq` można czasem użyć innych, bardziej wyspecjalizowanych poleceń. Najczęściej potrzebne są tylko dwa polecenia: jedno do wyświetlania zawartości kolejki i drugie, pozwalające usunąć z niej konkretne zadanie.

Za pomocą polecenia `lpq` można uzyskać informacje o zleceniach oczekujących na wydrukowanie. Składnia tego polecenia jest następująca:

```
lpq [-1] [-Pnazwa_drukarki] [id_zadania ...] [id_użytkownika ...]
```

Jeśli program `lpq` zostanie uruchomiony bez żadnych argumentów, wyświetli informacje o kolejce aktywnej drukarki, takie jak identyfikator właściciela zlecenia, pozycję zadania w kolejce, nazwy plików, które oczekują na wydrukowanie i ich sumaryczny rozmiar. Podając opcję `-1`, można uzyskać dodatkowe informacje o każdym zleceniu wydruku.

Jeśli potrzebne są informacje dotyczące innej drukarki, należy podać jej nazwę po opcji `-P`. Jeśli podany zostanie identyfikator zlecenia, wyświetlane będą tylko informacje go dotyczące, natomiast jeśli podany zostanie identyfikator użytkownika – wyświetcone zostaną dane o wszystkich zleceniach, których jest on właścicielem.



Ponieważ użytkownicy nie mają dostępu do katalogów kolejki, jedyną metodą na anulowanie zlecenia drukowania jest użycie polecenia `lprm`. Jeśli jesteś administratorem systemu, warto od czasu do czasu przypomnieć o tym użytkownikom - dzięki temu mniej będzie „niechcianych” wydruków.

Polecenie `lprm` służy do wycofania z kolejki zlecenia wydruku. Często zdarza się, że przez pomyłkę zamiast polecenia `lprm` wydawane jest polecenie `lpq`, które nie powoduje usunięcia zlecenia z kolejki. Jeśli chcesz wycofać zlecenie drukowania, musisz znać jego identyfikator. Jeśli jesteś zalogowany jako `root`, możesz również usunąć wszystkie zlecenia. Składnia polecenia `lprm` jest następująca:

```
lprm [-Pnazwa_drukarki] [-] [id_zadania ...] [id_użytkownika]
```

Argumentem polecenia `lprm` może również być pojedynczy myślnik – wówczas usuwane są wszystkie zadania, których właścicielem jest użytkownik wydający polecenie. Jeśli jesteś zalogowany jako `root`, usunięte zostaną wszystkie zadania. Pliki oczekujące na wydrukowanie przez określona drukarkę można usunąć podając opcję `-P`, na przykład polecenie:

```
lprm -Phplj -
```

powoduje usunięcie wszystkich zadań, których jesteś właścicielem, a które miały być wydrukowane na drukarce o nazwie `hplj`.



Jeśli jesteś zalogowany jako `root`, łatwo usunąć przypadkowo wszystkie zadania drukowania. Uważaj więc, by nie popełnić jakiegoś błędu składniowego!

Podanie identyfikatora zlecenia lub użytkownika powoduje usunięcie z kolejki określonego zlecenia lub wszystkich zleceń należących do określonego użytkownika. Jeśli nie zostaną podane żadne argumenty, usunięte zostanie zlecenie aktywne, którego właścicielem jest użytkownik wydający polecenie.

Po wycofaniu zlecenia `lprm` wyświetla na ekranie krótki komunikat, ale jeśli danego pliku nie ma w kolejce, nic nie zostanie wyświetlone (i będziesz się pewno zastanawiał, co się właściwie stało).

Jeśli użyjesz polecenia `lprm` z identyfikatorem zlecenia, które jest właśnie obsługiwane, może zdarzyć się, że albo zostanie ono wydrukowane do końca (jeśli zostało już przesłane do bufora drukarki), albo przerwane. W drugim przypadku zdarzyć się może, że drukarka przestanie odpowidać. Wtedy trzeba znaleźć identyfikator procesu obsługującego filtr wyjściowy (za pomocą polecenia `ps`) i zakończyć jego działanie.



Jeśli drukarka przestanie odpowidać i nie da się rozwiązać tego problemu za pomocą polecenia `lpc`, spróbuj zrestartować program `lpd`. Jeśli nawet to nie pomaga, prawdopodobnie jedynym rozwiązaniem będzie ponowne uruchomienie systemu.

Terminale

Większość systemów linuxowych korzysta tylko z jednej konsoli (klawiatury i ekranu komputera, w którym zainstalowany jest system). W takim przypadku nie są potrzebne żadne dodatkowe zabiegi konfiguracyjne.

Niektórzy administratorzy chcą jednak dodać kilka terminali tak, by możliwa była równoczesna praca wielu użytkowników (w końcu Linux jest systemem wielodostępnym).

Można to zrobić na dwa sposoby: podłączając terminale do portu szeregowego z tyłu komputera lub do portu znajdującego się na karcie multiport.

Używanie kart multiport

Karty multiport to łatwy i efektywny sposób na dodanie do systemu wielu portów szeregowych. Karty te dostarczane są przez wielu producentów w przeróżnych konfiguracjach. Udostępniają od 2 do 32 dodatkowych portów szeregowych, mogą również wykorzystywać różne typy gniazd (DB25, DB9 czy RJ11).

Jeśli zamierzasz używać karty multiport, upewnij się najpierw, czy dostępne są dla niej sterowniki linuxowe. Nie można użyć bez modyfikacji wersji sterowników dla UNIX-a czy Xenix-a. Ze względu na ich złożoność, modyfikacja kodu źródłowego jest jednak poza możliwościami zwykłego śmiertelnika.

Wraz z kartami multiport dostarczane są kompletne instrukcje dotyczące ich instalowania i konfigurowania terminali. Szczegóły tych operacji są zależne od sprzętu, dlatego nie będziemy się nimi zajmować.

Dodawanie terminali podłączonych przez port szeregowy

Aby dodać do systemu terminal, możesz również wykorzystać port szeregowy zamontowany w komputerze PC. Funkcję terminalu może pełnić prawdziwy terminal lub też inny komputer PC z działającym emulatorem terminalu.

Rodzaj kabla łączącego terminal z komputerem PC zależy od gniazd zamontowanych w obu urządzeniach. W większości przypadków jest to kabel DTE-to-DTE (ang. *Data Terminal Equipment*), ale czasem wymagany jest kabel DCE (ang. *Data Communication Equipment*). Ogólnie rzecz biorąc, terminal i inne komputery wymagają kabla DTE, natomiast modemy – DCE (kable DCE i DTE różnią się połączeniami).

Kabel łączący terminal z komputerem powinien być typu null-modem (niektóre przewody są w nim skrzyżowane, inne zwarte). Działać poprawnie będzie już kabel trójżyłowy (sygnały *send*, *receive* i masa), choć Linux wykorzystuje również sygnał Carrier Detect do sprawdzania, czy terminal jest podłączony i aktywny.

Kabel DCE (używany np. do podłączenia modemu – dlatego nazywany czasem kablem modemowym) łączy dwa gniazda bezpośrednio, tzn. styk nr 1 w jednej wtyczce odpowiada nr 1 w drugiej itd. Ważniejsze dla funkcjonowania połączenia DTE-to-DTE przewody (dla złączy typu DB25; numery styków dla złączy DB9 są inne, ale połączenia sygnałów pozostają bez zmian) zebrano w tabeli 33.1.

Tabela 33.1 Opis kabla DTE ze złączami DB25

Styk gniazda terminalu	Styk gniazda w komputerze	Sygnał
1	1	Ground

2	3	Transmit data / receive data
3	2	Receive data / transmit data
4	4	Ready to send
5	5	Clear to send
6	20	Data set ready/data terminal ready
7	7	Ground
8	20	Carrier detect/data terminal ready
20	6, 8	Data terminal ready/data set ready, carrier detect

Ponieważ większość użytkowników woli używać gotowych kabli, nie będziemy wdać się w szczegóły dotyczące ich montażu. Zamiast tego lepiej udać się do najbliższego sklepu komputerowego i wyjaśnić, jaki sprzęt chcemy połączyć, jaka jest wielkość (9 lub 25 styków) i rodzaj (męski – gdy na zewnątrz gniazda wystają styki, czy żeński – gdy nie ma styków) złącza.



Jeśli nie jesteś pewien, czy powinieneś użyć kabla modemowego, czy null-modem, warto zaopatrzyć się w przejściówkę, w której następuje skrzyżowanie odpowiednich linii. Urządzenie takie zamienia więc kabel modemowy na null-modem i odwrotnie.

Proces logowania

Aby zrozumieć, jakie dane zawierają pliki używane do konfigurowania terminali, warto najpierw zapoznać się z procesem logowania.

Proces ten rozpoczyna się od uruchomienia programu `/etc/init` podczas uruchamiania systemu. Program `init` jest odpowiedzialny za uruchomienie procesu `/etc/getty` dla każdego terminalu podłączonego do systemu. Dane o tych terminalach muszą być wpisane do plików `/etc/ttys` i `/etc/inittab`. Plik `/etc/ttys` zawiera listę portów i typów terminali, które są do nich podłączone. W pliku `/etc/inittab` zapisana jest pełna lista terminali, wraz z wszystkimi ich parametrami. Oba tym plikom przyjrzymy się dokładniej nieco później, w podrozdziale „Pliki konfiguracyjne terminalu: `/etc/ttys` i `/etc/inittab`”.

Jeśli z danych zapisanych w tych plikach wynika, że do danego portu podłączony jest terminal, proces `init` uruchamia dla niego proces `/etc/getty`, ustawiający parametry komunikacji i wyświetlający komunikat `login:`.

Gdy użytkownik loguje się używając określonego terminalu, proces `getty` wywołuje program `login`, który pobiera hasło. Program ten następnie sprawdza, czy identyfikator użytkownika i hasło są poprawne (na podstawie danych zawartych w pliku `/etc/passwd`); jeśli tak – wyświetlana jest zawartość pliku `/etc/motd`, a następnie uruchamiany jest domyślny dla danego użytkownika interpreter poleceń (również określony w pliku

/etc/passwd). Na koniec ustawiana jest wartość zmiennej TERM i program login kończy działanie.

Po zakończeniu logowania interpreter poleceń działa dalej, odczytując zawartość plików konfiguracyjnych, a następnie wyświetla znak zachęty i czeka na działania użytkownika.

Jak widać, w procesie logowania zamieszczonych jest całkiem sporo plików przechowywanych w katalogu /etc. Przyjrzymy się najważniejszym z nich (z punktu widzenia konfiguracji terminalu).

/sbin/getty i /etc/gettydefs

Wzmianki o programie /sbin/getty (w niektórych systemach /etc/getty) pojawiają się przy omawianiu problemów dotyczących terminali dość często, ale niewielu użytkowników zdaje sobie dokładne sprawę z funkcji spełnianych przez ten program. Służy on generalnie do ustawiania parametrów transmisji danych pomiędzy terminaliem a systemem, takich jak prędkość, protokół itp.

Program /sbin/getty wywoływany jest przez proces /etc/init. Po uruchomieniu otwiera on połaczenie przez port szeregowy czy inny port, do którego podłączony jest terminal, a następnie ustawia parametry transmisji danych do i z terminalu zgodnie z informacjami zawartymi w pliku /etc/gettydefs. Następnie do terminalu wysyłany jest komunikat login::

Proces getty umożliwia podanie w wierszu poleceń wielu modyfikujących jego zachowanie opcji, ale większość z nich nie jest szczególnie interesująca. Jeśli chcesz uzyskać pełniejsze informacje na temat składni tego polecenia, zajrzyj do dokumentacji na stronach man.

Plik /etc/gettydefs, zawierający parametry terminali używane przez program getty, składa się z wierszy o następującym formacie:

```
etykieta#parametry_startowe#parametry_końcowe#komunikat_login#następna_etykieta
```

etykieta powinna być unikalna, ponieważ na jej podstawie poszczególne wpisy są rozróżniane. parametry_startowe i parametry_końcowe pozwalają na kontrolowanie zachowania połączenia przed i po wykonaniu programu login.

komunikat_login to tekst, który zostanie wyświetlony na ekranie terminalu. Zwykle ma on postać login:, ale można zażądać wyświetlania dowolnego tekstu. następna_etykieta to etykieta wpisu, do którego należy przejść, gdyby użycie ustawień z wiersza bieżącego się nie powiodło. Takie rozwiązanie przydaje się przy połączeniach modemowych – próby nawiązania połączenia rozpoczyna się od największych prędkości przesyłu danych (np. 9600 bodów), stopniowo zmniejszając prędkość (poprzez 4800 i 2400 do 1200 bodów), aż do nawiązania połączenia. Dla terminali pole następna_etykieta wskazuje zwykle z powrotem na początek pliku.

Przykładowy fragment tego pliku może mieć następującą postać:

```
console# B19200 OPOST ONCLR TAB3 BRKINT IGNPAR ISTRIP IXON IXANY PARENB  
ECHO ECHOE ECHOK ICANON ISIG CS8 CREAD # B19200 OPOST ONCLR TAB3 BRKINT  
IGNPAR ISTRIP IXON IXANY PARENB ECHO ECHOE ECHOK ICANON ISIG CS8 CREAD  
#Console Login: #console  
9600H# B9600 # B9600 SANE IXANY PARENB TAB3 HUPCL #Login: #4800H  
4800H# B4800 # B4800 SANE IXANY PARENB TAB3 HUPCL #Login: #2400H  
2400H# B2400 # B2400 SANE IXANY PARENB TAB3 HUPCL #Login: #1200H  
1200H# B1200 # B1200 SANE IXANY PARENB TAB3 HUPCL #Login: #300H  
300H# B300 # B300 SANE IXANY PARENB TAB3 HUPCL #Login: #9600H
```

Jeśli zajrzesz do pliku gettydefs znajdującego się w Twoim systemie, znajdziesz jeszcze wiele innych wpisów, ale wszystkie one mają taki sam format, jak przedstawione w przykładzie. Pięć ostatnich wierszy dotyczy połączeń modemowych; pierwszy z nich definiuje parametry połączenia przy prędkości 9600 bodów. Jedyny parametr startowy – B9600 – ustawia prędkość transmisji na 9600 bodów. Parametry końcowe określają zachowanie terminalu (na przykład powodują zastępowanie znaku tabulacji trzema spacjajami). Wskaźnik do następnego wpisu prowadzi do połączenia z niższą prędkością przesyłania danych, co umożliwia połączenie z wolniejszymi modemami lub poprzez łączą słabej jakości.

Pierwszy wiersz przykładu jest typowy dla wszystkich systemów – opisuje on zachowanie się konsoli „systemowej”, czyli klawiatury i ekranu podłączonych bezpośrednio do komputera. Ponieważ są one podłączone na stałe, wskaźnik na następny wpis jest ustalony tak, że następuje zapłelenie.



Nie powinieneś raczej zmieniać ustawień w pliku gettydefs - o wiele łatwiej jest poszukać takiego wpisu, który odpowiada typowi terminalu, którego chcesz użyć. Jeśli jednak zmodyfikujesz jego zawartość, powinieneś wydać polecenie getty -c, aby uaktywnić wprowadzone modyfikacje.

Pliki konfiguracyjne terminala: /etc/ttys i /etc/inittab

Dane o konfiguracji terminalu zapisane są w plikach /etc/ttys i /etc/inittab. Pliki te mogą być modyfikowane za pomocą dowolnego edytora tekstów. Dostępne są również programy pozwalające modyfikować te pliki w wygodniejszy sposób (np. za pomocą menu).



Przed dokonaniem jakichkolwiek zmian w plikach konfiguracyjnych terminalu zrób ich kopię na wypadek, gdyby zmiany nie przyniosły oczekiwanych efektów. W tym celu wystarczy skopiować te pliki do plików o nazwach na przykład /etc/ttys.org i /etc/inittab.org.

Plik `/etc/ttys` zawiera dwie kolumny danych. Pierwsza z nich określa typ terminalu, natomiast druga – nazwę urządzenia. Oto przykładowa zawartość takiego pliku:

```
console tty1
console tty2
console tty3
console tty4
console tty5
console tty6
vt100 ttyp0
vt100 ttyp1
vt100 ttyp2
vt100 ttyp3
```

Typ terminalu w pierwszej kolumnie jest używany przy ustawianiu wartości zmiennej środowiskowej `TERM`.

Plik `/etc/inittab` zawiera informacje dotyczące zachowania każdego terminala. Jego format jest następujący:

```
ID:runlevel:akcja:proces
```

`ID` to jedno lub dwuznakowy identyfikator, inny dla każdego wpisu. W większości przypadków odpowiada on w jakiś sposób nazwie urządzenia, np. identyfikator `1` może określać terminal `tty1` itp.

Parametr `runlevel` określa możliwości terminalu w różnych stanach systemu operacyjnego (może on przybierać wartości od 0 do 6 oraz A, B i C, jak również kombinacje tych wartości). Jeśli żadna wartość nie zostanie podana, obsługiwane będą wszystkie stany systemu.

Zawartość pola `akcja` decyduje o tym, w jaki sposób obsłużyć następne pole (`proces`). Dostępne wartości to:

- boot** uruchomienie przy pierwszym odczytaniu pliku `inittab`;
- bootwait** uruchomienie przy pierwszym odczytaniu pliku `inittab`;
- initdefault** ustawienie domyślnego stanu systemu (runlevel);
- off** wyłączenie procesu, jeśli jest on aktywny;
- once** jednokrotne uruchomienie procesu;
- ondemand** uruchomienie procesu działającego przez cały czas (to samo co `respawn`);
- powerfail** wykonanie gdy `init` otrzyma sygnał o awarii zasilania;
- powerwait** wykonanie gdy `init` otrzyma sygnał o oczekiwaniu na zasilania;
- sysinit** uruchomienie przed uzyskaniem dostępu do konsoli;
- respawn** uruchomienie procesu działającego przez cały czas;

wait jednokrotne uruchomienie procesu.

Pole `akcja` pozwala określić zachowanie terminalu podczas uruchamiania systemu i wtedy, gdy zostanie zakończony obsługujący go proces `getty`.

Przykładowy plik `/etc/inittab` (zaczerpnięty co prawda z wcześniejszej wersji Linuxa, ponieważ w nowszych wersjach plik ten jest nieco bardziej skomplikowany) wygląda następująco:

```
#inittab for Linux
id:1:initdefault:
rc::bootwait:/etc/rc
1:1:respawn:/etc/getty 9600 tty1
2:1:respawn:/etc/getty 9600 tty2
3:1:respawn:/etc/getty 9600 tty3
4:1:respawn:/etc/getty 9600 tty4
```

Pierwsze dwa wiersze (po komentarzu) są używane podczas uruchamiania systemu. Drugi z nich nakazuje uruchomienie skryptu `/etc/rc`. Pozostałe powodują uruchomienie procesów `getty` dla konsoli od `tty1` do `tty4` z prędkością 9600 bodów.

Definicje terminali - plik `/etc/termcap`

Plik `/etc/termcap` (ang. *terminal capabilities*) zawiera instrukcje dotyczące komunikacji z większością typów terminali obsługiwanych przez system, w związku z tym jest on dość duży. Jeśli zamierzasz modyfikować ten plik, powinieneś najpierw wykonać jego kopię zapasową.

Zawartość pliku `termcap` jest podobna do zawartości pliku `/etc/printcap`, w którym znajdują się dane o poszczególnych typach drukarek. Każdy wpis określa kilka wariantów nazwy urządzenia oraz zestaw parametrów i odpowiadających im wartości odpowiednich dla różnych typów terminali. Ze względu na fakt, że niektóre bardziej rozbudowane terminali potrafią interpretować cały szereg symboli sterujących, poszczególne wpisy mogą być dość obszerne.

Przykładowe definicje prostych typów terminali – Vi550 i Vi603 – mogą mieć następującą postać:

```
vi550|visual 550 ansi x3.64:\n
:li#33:\n
:c1=\x030\x1E\x2J:tc=vi300:\n\n
vi603|visual603|visual 603:\n
:hs:m1:\n
:al=\x1E[L:bl=\x1E[J:ce=\x1E[K:c1=\x1E[H\x1E[J:\n
:cm=\x1E[%i%d;%dH:cs=\x1E[%i%d;%dr:dc=\x1E[P:dl=\x1E[M:\n
:ds=\x1EP2;1~\x1E\\:ei=\x1E[4l:fs=\x1E\\:\\\n
:i1=\x1E>\x1E[?31\x1E[?41\x1E[?7h\x1E[?8h\x1E[1;24r:im=\x1E[4h:\n
:mb=\x1E[5m:md=\x1E[1m:me=\x1E[m:mr=\x1E[7m:nd=\x1E[C:\n
:se=\x1E[27m:sf=\x1ED:so=\x1E[7m:sr=\x1EM:ts=\x1EP2~:ue=\x1E[24m:\n
:up=\x1E[A:us=\x1E[4m:tc=vt100:
```

Znaczenie poszczególnych kodów nie jest zwykle przedmiotem zainteresowania przeciętnego użytkownika czy administratora systemu linuxowego. Informacje takie mogą

być potrzebne dopiero wtedy, gdy trzeba skonfigurować nowy (nie obsługiwany przez system) typ terminalu.

Charakterystyki terminali zawarte w pliku `/etc/termcap` są używane przez plik `/etc/ttys`. Pierwsza kolumna pliku `/etc/ttys` zawiera nazwę terminalu, na podstawie której ustalana jest wartość zmiennej środowiskowej `TERM`. Dokładniejsze dane o parametrach terminalu odczytywane są z pliku `/etc/termcap` poprzez odszukanie wpisu odpowiadającego danej nazwie terminalu.



Większość terminali może emulować kilka różnych standardów. Jeśli nie potrafisz znaleźć w pliku `/etc/termcap` wpisu odpowiedniego dla terminalu, który chcesz podłączyć, poszukaj wpisu dla jednego z emulowanych typów terminali. Jest to na pewno rozwiązanie prostsze niż stworzenie nowego wpisu.

Dodawanie terminalu

Terminal można dodać do systemu linuxowego w taki sam sposób jak drukarkę – za pomocą polecenia `mknod`. Najpierw należy zdecydować, do którego portu będzie on podłączony. Porty szeregowe mają nazwy takie jak `/dev/ttys0` (port określany w systemie DOS jako COM1), `/dev/ttys1` (w systemie DOS – COM2) itd.

Większość komputerów klasy PC wyposażona jest w dwa porty szeregowe, choć możliwe jest podłączenie dodatkowych dwóch (czyli od `/dev/ttys0` do `/dev/ttys3`). Linux używa portów szeregowych w oparciu o ich adresy podawane przez BIOS. Zwykle są to następujące wartości:

```
ttyS0 (COM1)    0x03f8  
ttyS1 (COM2)    0x02f8  
ttyS2 (COM3)    0x03e8  
ttyS3 (COM4)    0x02e8
```

Jeśli w systemie dostępny jest tylko jeden port szeregowy, prawie zawsze jest on skonfigurowany jako COM1. Jeżeli nie jesteś pewny ustawień w Twoim systemie, możesz sprawdzić odpowiednie wartości za pomocą jakiegoś programu użytkowego (np. MSD.EXE dla systemu DOS) lub po prostu ustalić je metodą prób i błędów, zaczynając od najniższych wartości.

Aby utworzyć nowe urządzenie terminalu, trzeba użyć polecenia `mknod` (ang. *make node*); polecenie to opisane jest bardziej szczegółowo w podrozdziale „Polecenie `mknod`”), a następnie zmienić prawa dostępu do nowego pliku urządzenia tak, by mógł on być czytany i zapisywany przez użytkownika `root` lub `daemqn`. W większości systemów odpowiednie pliki urządzeń są konfigurowane już podczas instalacji.

Polecenie, które należy wydać, aby utworzyć plik terminalu, ma postać:

```
mknod -m 660 /dev/ttys0 c 4 64
```

Parametr `-m 660` ustala odpowiednie prawa dostępu do pliku `/dev/ttys0` to określenie portu szeregowego, odpowiadającego DOS-owemu portowi COM1. Parametr `c` określa, że urządzenie jest typu znakowego (większość terminali, za wyjątkiem bardzo szybkich modeli, pracuje jako urządzenia znakowe), a następujące po nim liczby definiują główny i poboczny numer urządzenia (w tym przypadku odpowiednio 4 i 64). Polecenia tworzące pliki urządzeń dla portów COM2 – COM4 miałyby postać:

```
mknod -m 660 /dev/ttys1 c 4 65  
mknod -m 660 /dev/ttys2 c 4 66  
mknod -m 660 /dev/ttys3 c 4 67
```

Po wykonaniu polecenia `mknod` należy jeszcze zmienić identyfikator właściciela pliku, na przykład wydając polecenie

```
chown root.tty /dev/ttys0
```

podstawiając za `/dev/ttys0` nazwę pliku urządzenia, do którego polecenie ma się odnosić. Właścicielem pliku będzie użytkownik `root.tty`. Należy również zmodyfikować odpowiednie wpisy w pliku `/etc/ttys`, ustawiając typ terminalu tak, aby proces jego inicjalizacji mógł przebiegać poprawnie. Plik `/etc/inittab` zawiera wpisy dla standar-dowych portów szeregowych, możesz więc użyć ich jako szablonu i dopasować przed-kość transmisji i inne parametry.

Polecenia stty oraz tset

Polecenie `stty` pozwala sprawdzić i ustawić opcje dotyczące obsługi terminalu. Jest ono bardzo złożone, posiada kilkadziesiąt różnych opcji modyfikujących działanie sterownika terminalu. Na szczęście tylko przy wyjątkowo „intensywnej” administracji będziesz musiał ich używać, więc w tym rozdziale po prostu zignorujemy większość szczegółów.

Aby sprawdzić aktualne ustawienia terminalu, wystarczy wydać polecenie `stty` bez żadnych parametrów. Dzięki temu można zweryfikować, czy informacje zapisane w plikach konfiguracyjnych `/etc/inittab` i `/etc/gettydefs` zostały poprawnie odczytane.

Polecenie `tset` również jest dość skomplikowane, ale większość jego opcji używana jest bardzo rzadko (szczególnie jeśli nie masz do czynienia z jakimś przedpotopowymi terminalami i niestandardowymi łączówkami). Polecenie to służy do inicjalizowania sterownika terminalu. Jeśli nazwa urządzenia nie zostanie określona jawnie jako parametr tego polecenia, użytą zostanie nazwa wskazywana przez zmienną środowiskową `TERM`.

Polecenie `tset` może zostać zapisane w jednym z plików inicjalizacyjnych przetwarzanych podczas logowania. Przykładowo użytkownik, który zawsze loguje się przez modem, może w pliku inicjalizacyjnym (`.profile`, `.cshrc` itp.) wpisać polecenie

```
tset -m dialup:vt100
```

Spowoduje ono ustawienie typu terminalu na `vt100` za każdym razem, gdy loguje się on do systemu przez modem. Oczywiście typ terminalu zostanie ustawiony na `vt100` na-

wet wtedy, jeśli w rzeczywistości użytkownik loguje się z terminalu innego rodzaju. Zapobiec takiej sytuacji można stosując polecenie:

```
tset -m dialup:?vt100
```

które spowoduje, że system zapyta o potwierdzenie typu terminalu:

```
TERM=(vt100) ?
```

Po wciśnięciu klawisza Enter typ terminala zostanie ustawiony na `vt100`, ale możliwe jest również podanie innej wartości.

Jak na razie polecenie to nie wygląda na szczególnie skomplikowane. Rogi pokazuje dopiero wtedy, gdy trzeba skonfigurować terminal podłączony na stałe. Przykładowo, jeśli trzeba uruchomić sterownik dla terminalu podłączonego przez port szeregowy, postać polecenia może być następująca:

```
eval `tset -s -Q -m dialup:?vt100 -m switch:z29`
```

Szczegóły składni tego polecenia nie są istotne dla większości administratorów. Jeśli jednak chcesz uzyskać dokładniejsze informacje na ten temat, zajrzyj na strony `man` dotyczące poleceń `tset` i `stty`.

Resetowanie terminalu

Od czasu do czasu zdarza się, że terminal podłączony przez port szeregowy przestaje zachowywać się poprawnie, bądź to nie wyświetlając znaku zachęty, bądź też wyświetlając różne „krzaczki”. Istnieją dwa proste sposoby wyjścia z takiej sytuacji. Jeśli jednak nie zadziałają, terminal powinien zostać wyłączony, a następnie uruchomiony ponownie (możesz również być zmuszony do wyłączenia procesów uruchomionych z tego terminalu).

Pierwszy sposób to wciśnięcie klawiszy `Control+j`, wpisanie polecenia `stty sane`, a następnie ponowne wciśnięcie `Control+j`. Terminal powinien zostać przywrócony do stanu używalności. Prawdopodobnie polecenie, które wpisujesz, nie zostanie wyświetlone na ekranie, więc wpisz je uważnie.

Jeśli terminal nadal nie zachowuje się jak należy, spróbuj wpisać `reset`, a następnie wcisnąć klawisz Enter lub `Control+j`. Jeśli to również nie pomaga, terminal zawiesi się i musi zostać zresetowany ręcznie.

Modemy

Proces dodawania modemu nie różni się zasadniczo od dodawania terminalu. W większości przypadków można zastosować procedurę omówioną w podrozdziale „Terminale”.

Modemy w systemach linuxowych używane są do łączenia się z siecią lub z innym systemem oraz do przyjmowania połączeń przychodzących. Jeśli modem służy do podłą-

czenia zdalnego terminalu, procedura jego instalacji jest taka, jak opisano w rozdziale „Terminale”; różni się tylko tym, że trzeba wybrać inne wpisy z pliku `/etc/inittab`. W przypadku modemu należy odszukać blok wpisów określający parametry połączeń z coraz mniejszymi prędkościami przesyłania danych.

Modemy, które mają być używane do połączeń z siecią i obsługi UUCP, zostaną omówione w rozdziale 37. „Praca w sieci” i rozdziale 39. „UUCP”. Rozdziały te zawierają również informacje o konfigurowaniu odpowiednich plików.

Jeśli modem ma służyć do łączenia się z innymi systemami, można go skonfigurować używając obsługiwanego za pomocą menu programu, wywoływanego z poziomu programu `setup`. Potrafi on automatycznie ustawić większość niezbędnych opcji.

Podsumowanie

W tym rozdziale podaliśmy najbardziej podstawowe informacje dotyczące urządzeń, dodawania ich do systemu i zarządzania nimi. Stosują się one do prawie wszystkich dystrybucji Linuxa, choć mogą istnieć pewne różnice w nazwach parametrów poszczególnych programów. Jeśli potrzebujesz dokładnych informacji na temat jakiegoś polecenia, zawsze możesz zajrzeć na strony `man` lub do dokumentacji.

Konfigurowanie i dodawanie do systemu urządzeń SCSI (ang. *Small Computer System Interface*) omówione jest w rozdziale 36. „Obsługa urządzeń SCSI”.

Jak skonfigurować system, by można było korzystać z poczty elektronicznej, doiesz się z rozdziału 40. „Konfigurowanie poczty”.

Polecenie `tar` i proces tworzenia kopii zapasowych przedstawione są w rozdziale 45. „Kopie zapasowe”.

Rozdział 34.

Procesy

Tim Parker

W tym rozdziale:

- υ Co trzeba wiedzieć o procesach
- υ Polecenie ps
- υ Polecenie kill

Każdy program działający w systemie linuxowym – uruchomiony przez użytkownika, przez systemu, czy też program rezydentny – jest procesem. Umiejętność zarządzania procesami jest dla administratora bardzo istotna (czasem nawet nieodzowna). W tym rozdziale przyjrzymy się bliżej temu problemowi. Nie będziemy oczywiście omawiać szczegółów technicznych przydzielania poszczególnym procesom czasu procesora czy pamięci operacyjnej. Omówimy jednak najważniejsze – z punktu widzenia administratora – zagadnienia, których znajomość jest niezbędna do zapewnienia bezawaryjnej pracy systemu.

Przy opisie systemów wielozadaniowych często używa się pojęć *proces* i *zadanie*. Często mogą być one używane zamiennie, ale przez zadanie przeważnie rozumie się proces uruchomiony przez interpreter polecen (który może również zatrudniać inne procesy). Proces jest najmniejszą niepodzieloną częścią programu, działającą w systemie Linux.

Co trzeba wiedzieć o procesach

Formalna definicja procesu brzmi: proces to program, który posiada własną wirtualną przestrzeń adresową. Oznacza to, że każdy program działający w systemie linuxowym jest procesem. Pojedyncze polecenie może uruchamiać wiele różnych procesów, szczególnie jeśli użyty zostanie mechanizm przekierowania czy potoków (ang. *pipe*). Poniższe polecenie powoduje uruchomienie trzech procesów, po jednym dla każdego wywoływanego programu:

```
nroff -man ps.1 | grep kill | more
```

Typy procesów

W systemie Linux rozróżniane są trzy rodzaje procesów, posiadające nieco inne cechy i atrybuty. Są to:

- **procesy interaktywne** – uruchamiane poprzez powłokę i przez nią kontrolowane; procesy takie mogą działać w tle lub na pierwszym planie;
- **procesy wsadowe** (ang. *batch process*), nie powiązane z żadnym terminaliem, ale wstawiane do kolejki i wykonywane sekwencyjnie;
- **procesy rezydentne** – czyli demony (ang. *daemons*); są one zwykle uruchamiane podczas startu systemu i pracują w tle przez cały czas działania Linuxa.

Użycie polecenia ps

Najprostszym sposobem na sprawdzenie, jakie procesy działają aktualnie w systemie, jest użycie polecenia `ps` (ang. *process status*). Udostępnia ono sporo różnego typu opcji, ale tylko kilka z nich wykorzystuje się w praktyce. Rozpoczniemy od przedstawienia najbardziej podstawowych informacji o tym poleceniu, a następnie omówimy niektóre z jego opcji.

Polecenie `ps` jest dostępne dla wszystkich użytkowników systemu, ale dla użytkownika `root` wyniki jego działania mogą być nieco inne niż dla zwykłego użytkownika.

Jeśli jesteś zalogowany jako zwyczajny użytkownik (tj. nie jako `root`) i wydasz polecenie `ps`, wyświetlane zostaną informacje o wszystkich procesach, których jesteś właścicielem. Możesz na przykład zobaczyć następujące dane:

```
$ ps
 PID  TTY      STAT      TIME      COMMAND
 229  2          S          0:00     /bin/login -- reksio
 257  2          S          0:00     -bash
 269  2          R          0:00     ps
```

Dane wyświetlane przez program ps

Dane wyświetlane przez program `ps` zawsze podawane są w kolumnach. Pierwsza kolumna ma nagłówek `PID` (ang. *Process ID*) i zawiera identyfikator procesu. Każdemu procesowi działającemu w systemie przypisany jest niepowtarzalny numer, dzięki któremu Linux może odróżnić go od innych procesów. Numerowanie procesów rozpoczyna się od zera po uruchomieniu Linuxa, a maksymalna dopuszczalna wartość zależy od systemu (często jest to 65535). Po przekroczeniu wartości maksymalnej numerowanie znów zaczyna się od zera, z pominięciem tych numerów, które zostały przydzielone wciąż aktywnym procesom. Zwykle najniższe numery przydzielone są procesom wchodząącym w skład jądra systemu i programom rezydentnym (demonom), uruchamianym

podczas startu systemu. Jeśli chcesz zrobić cokolwiek z procesem (na przykład zakończyć jego działanie), musisz znać jego identyfikator.

Kolumna opisana jako `TTY` zawiera informację o tym, z której konsoli dany proces został uruchomiony. Jeśli jesteś zalogowany jako zwykły użytkownik, będzie to nazwa Twojego terminalu lub konsoli. Jeżeli jesteś zalogowany na kilku konsolach, zobaczysz dane o procesach uruchomionych na wszystkich konsolach.

W kolumnie `STAT` znajduje się informacja o bieżącym stanie procesu. Najczęściej pojawiają się tu litery `S` (ang. *sleeping* – proces uśpiony) i `R` (ang. *running* – aktywny). Stan procesu może zmieniać się z uśpionego na aktywny i odwrotnie wiele razy w ciągu sekundy.

Kolumna `TIME` zawiera całkowitą ilość czasu procesora użytego przez proces do tej pory. Zwykle jest to wartość bardzo mała (większość procesów wykonuje się dość szybko). Jest to czas, przez jaki proces miał dostęp do procesora, a nie czas, który upłynął od momentu jego uruchomienia.

Ostatnia kolumna – `COMMAND` – zawiera polecenie, za pomocą którego proces został uruchomiony. Przeważnie jest to polecenie wpisane przez użytkownika z klawiatury terminalu, jednak `ps` na równych prawach traktuje też procesy wywołane przez inne procesy (nazywane procesami potomnymi – ang. *child processes*).

Interpretery poleceń uruchamiane przy logowaniu

Aby pomóc Ci odróżnić interpreter, który został uruchomiony podczas logowania, od pozostałych (uruchomionych później), jego nazwa poprzedzona jest myślakiem, na przykład tak:

```
$ ps
 PID   TTY      STAT      TIME      COMMAND
  46   v01      S         0:01     -bash
  75   v01      S         0:00     pdksh
  96   v01      R         0:00     bash
 123   v01      R         0:00     ps
```

W powyższym przykładzie widać, że interpreter poleceń o identyfikatorze 46 uruchomiony został przy logowaniu, natomiast inne interpretery (o identyfikatorach 75 i 96) zostały uruchomione później.

Zauważ, że na liście procesów zawsze znajduje się polecenie `ps`; jest to oczywiste, jeśli wziąć pod uwagę fakt, że polecenie to było uruchomione w trakcie swojego działania.

Uwagi dla użytkownika root

Kiedy zwykły użytkownik wydaje polecenie `ps`, uzyskuje listę własnych procesów. Jeśli jesteś zalogowany jako `root`, zobaczysz w takiej sytuacji listę wszystkich procesów w systemie, ponieważ `root` jest właścicielem ich wszystkich. Lista taka może być bardzo długa, szczególnie w systemach, w których pracuje jednocześnie kilku użytkowników.

Powinieneś więc skierować ją do pliku albo na wejście polecenia `more` lub `less`, na przykład za pomocą jednego z polecen:

```
ps | more  
ps > /tmp/ps_out
```

Przydatne opcje programu ps

Po dołączeniu do polecenia `ps` opcji `u` uzyskujemy kilka nowych informacji. Oto wynik wykonania takiego polecenia przez zwykłego użytkownika:

```
$ ps u  
USER  PID %CPU %MEM   SIZE  RSS TTY STAT START TIME COMMAND  
bill  281  0.2  3.0   1492  948  2   S    18:25  0:00  /bin/login -- Σ  
bill  
bill  284  0.2  2.4   1180  768  2   S    18:26  0:00  -bash  
bill  298  0.0  1.5   856   488  2   R    18:26  0:00  ps u
```

Najważniejsza nowość to kolumna `USER`, która pokazuje, kto uruchomił i posiada dany proces. Zamiast numerycznego identyfikatora użytkownika wyświetlany jest odpowiadający mu identyfikator tekstowy, odszukany przez polecenie `ps` w pliku `/etc/passwd`.

Opcja `u` powoduje również wyświetlenie kolumn zawierających dane o procentowym zużyciu czasu procesora (kolumna `%CPU`) i pamięci (kolumna `%MEM`). Dane te mogą być użyteczne w przypadku, gdy z nieznanych bliżej powodów system zwolni działanie. Można wówczas odszukać „winowiące” (w języku angielskim procesy konsumujące nadmierne ilości zasobów zwane są „memory hogs” i „CPU hogs”) i sprawdzić, czy przypadkiem nie jest to proces, który wymknął się spod kontroli i pochłania zasoby systemowe.

Kiedy wydasz polecenie `ps u`, będąc zalogowanym jako `root`, zobaczysz listę wszystkich procesów działających w systemie. Podobnie jak poprzednio, może zajść konieczność przesłania wyników polecenia do pliku lub na wejście programu `more`. W niektórych wersjach Linuxa po opcji `u` można również podać identyfikator użytkownika, co spowoduje wyświetlenie tylko procesów do niego należących, na przykład:

```
ps u bill
```

Ta składnia polecenia `ps` jest dozwolona w wersjach dostarczanych wraz z System V, ale nie działa w większości rozprowadzanych z Linuxem wersji programu `ps` opartych o BSD (w tym w wersji dostępnej na załączonym do książki dysku CD). Inne wersje tego polecenia dostępne są w węzłach FTP i BBS. Większość użytkowników może również użyć opcji `u` aby zobaczyć listę procesów uruchomionych przez innych użytkowników. Dzięki temu można sprawdzić, kto uruchamia procesy pochłaniające najwięcej zasobów, a administrator może łatwo zorientować się, jakie procesy uruchamiał użytkownik zgłoszający problemy z systemem.

Zwykły użytkownik może również zobaczyć listę wszystkich procesów działających w systemie (a nie tylko procesów uruchomionych przez siebie), używając opcji `a` (jeśli zastosujesz tę opcję, gdy jesteś zalogowany jako `root`, wyświetlana lista oczywiście nie zmieni się). Oto przykładowy wynik działania polecenia `ps a` wydanego przez zwykłego użytkownika:

```
$ ps a
PID  TTY  STAT   TIME  COMMAND
228  1    S      0:00  /bin/login -- root
230  3    S      0:00  /sbin/mingetty tty3
231  4    S      0:00  /sbin/mingetty tty4
232  5    S      0:00  /sbin/mingetty tty5
233  6    S      0:00  /sbin/mingetty tty6
236  1    S      0:00  -bash
248  1    S      0:12  /usr/bin/mc -P
250  p0   S      0:00  bash -rcfile .bashrc
281  2    S      0:00  /bin/login -- bill
284  2    S      0:00  -bash
300  2    R      0:00  ps a
```

Tak krótki wydruk pochodzi z systemu, w którym w zasadzie nic się nie dzieje. Większość pozycji opisuje procesy systemowe. W powyższym przypadku nie można określić, kto uruchomił dany proces – ale można połączyć opcje a oraz u:

```
$ ps au
USER  PID %CPU %MEM  SIZE  RSS  TTY  STAT  START  TIME  COMMAND
bill  281  0.1  3.0   1492  948  2    S      18:25  0:00  /bin/login -- Σ
bill
bill  284  0.1  2.4   1180  768  2    S      18:26  0:00  -bash
bill  302  0.0  1.5   856   492  2    R      18:27  0:00  ps au
root  228  0.0  3.0   1496  952  1    S      18:18  0:00  /bin/login -- Σ
root
root  230  0.0  1.0   740   316  3    S      18:18  0:00  /sbin/mingetty Σ
tty3
root  231  0.0  1.0   740   316  4    S      18:18  0:00  /sbin/mingetty Σ
tty4
root  232  0.0  1.0   740   316  5    S      18:18  0:00  /sbin/mingetty Σ
tty5
root  233  0.0  1.0   740   316  6    S      18:18  0:00  /sbin/mingetty Σ
tty6
root  236  0.0  2.5   1184  772  1    S      18:19  0:00  -bash
root  248  2.5  4.4   2456  1372  1   S      18:19  0:12  /usr/bin/mc -P
root  250  0.0  2.5   1188  772  p0   S      18:19  0:00  bash -rcfile Σ.
bashrc
```

Wyświetlane są kolumny zawierające te same informacje, jak w przypadku użycia opcji u, ale dotyczące wszystkich działających procesów. Porządek, w jakim podane zostaną opcje, nie jest istotny.

Opcja l pozwala na uzyskanie informacji o tym, przez jaki proces dany proces został uruchomiony:

```
$ ps l
FLAGS  UID  PID  PPID  PRI  NI  SIZE  RSS  WCHAN  STA  TTY  TIME  COMMAND
100100  501  281  1     0    0   1492  948  wait4  S    2    0:00
Σ /bin/login -- bill
      501  284  281  9     0   1180  768  wait4  S    2    0:00  -bash
100000  501  304  284  10    0   968   504          R    2    0:00  ps -l
```

Kolumna PPID (ang. *Parent Process ID*) zawiera identyfikator procesu, który uruchomił dany proces. Jak widać, program ps uruchomiony został przez powłokę bash, która z kolei została uruchomiona przez proces o numerze 1 – czyli proces init, który jest

„matką” wszystkich innych procesów (jeśli zastanawiasz się, jakie ma to konsekwencje – odpowiedź jest prosta: jeśli ten proces zakończy działanie, przestaną działać wszystkie inne procesy).



Wydając polecenie `ps` można, ale nie trzeba używać myślnika przed nazwami opcji.

Uwagi dla administratorów

W większości przypadków trzy przedstawione wcześniej wersje polecenia `ps` dostarczają wszystkich potrzebnych informacji. Jeśli potrzebne są dane o systemie jako całości, wystarczy wydać polecenia:

```
ps -ax  
ps -aux  
ps -le
```

Na ekranie otrzymasz wszystkie chyba dostępne w systemie informacje o procesach. Dokładniejsze dane o działaniu polecenia `ps` możesz znaleźć na stronach `man` (które w tym przypadku nie są – niestety – kompletne).

Polecenie kill

Od czasu do czasu zdarza się proces, który zawiesi terminal albo po prostu nic nie robi. Zwykle sytuacja taka jest wynikiem błędów tkwiących w różnych programach. W obu przypadkach jedyną drogą usunięcia takiego procesu jest wydanie polecenia `kill`.



Kiedy kończysz działanie procesu, a jesteś zalogowany jako `root`, zwróć szczególną uwagę na numer procesu, który zamierzasz usunąć, żeby przypadkiem nie popełnić mogącej mieć przykro skutki pomyłki. Nie powinieneś usuwać procesów systemowych, chyba że dokładnie wiesz, co chcesz przez to osiągnąć.

Aby użyć polecenia `kill`, musisz mieć dostęp do okna lub konsoli, w którym będziesz mógł wydawać polecenia. Jeśli terminal zawiesił się完全 – musisz znaleźć inny. Jako zwykły użytkownik możesz usuwać tylko własne procesy. Jako użytkownik `root` możesz manipulować wszystkimi procesami.

Musisz również znać identyfikator procesu, który chcesz usunąć (polecenie `kill` służy również do wysyłania innych sygnałów do procesów, nie tylko do ich usuwania – *przyp. tłum.*). Jest on wyświetlany przez polecenie `ps`. Poniżej podano przykład, w którym proces o nazwie `bad_prog` i identyfikatorze 292 nie zakończył się prawidłowo, w związku z czym trzeba go usunąć.

```
$ ps u
USER   PID %CPU %MEM   SIZE   RSS   TTY STAT START TIME   COMMAND
walter 281  0.2  3.0   1245  467    2   S   13:25 0:00   -bash
walter 292  9.2 12.0   2134  467    2   R   15:51 2:01   bad_prog
Σ$ kill 292
```

Polecenie `kill` nie wyświetla informacji potwierdzającej usunięcie procesu. Aby sprawdzić, czy wszystko przebiegło poprawnie, trzeba ponownie wydać polecenie `ps`.

Usuwanie procesów potomnych

Jeśli proces, który chcesz usunąć, podczas swojego działania uruchomił inne procesy, powinieneś usunąć również wszystkie procesy potomne. Czasem zdarza się, że usunięty proces „odradza” się po chwili – w takim przypadku usunąć należy proces, który go wywołuje (jego identyfikator można znaleźć w kolumnie `PPID` po wydaniu polecenia `ps`).

Jeśli pomimo wydania polecenia `kill` proces nadal działa, trzeba użyć mocniejszych argumentów. Polecenie `kill` udostępnia kilka poziomów działania. Wydane tylko z jednym argumentem – numerem procesu – próbuje grzecznie zakończyć proces, tzn. zamknąć otwarte pliki, zwolnić pamięć itd. Jeśli to nie pomaga, należy użyć opcji `-9`, która wymusza zakończenie procesu, na przykład tak:

```
kill -9 726
```

Jeśli nie działa nawet polecenie `kill -9`, możliwe, że masz do czynienia z jednym z „nieśmiertelnymi” procesów. Jedynym sposobem na zakończenie działania takiego stworzenia jest ponowne uruchomienie systemu.

Co można, a czego nie można usunąć

Generalnie nie możesz usunąć procesu nie należącego do Ciebie. Przy próbie zrobienia czegoś takiego otrzymasz komunikat:

```
kill: - Not owner
```

Oczywiście użytkownik `root` może usunąć każdy proces.

Podsumowanie

W tym rozdziale pokazaliśmy, w jaki sposób można uzyskać informacje o procesach działających w systemie oraz jak (w razie potrzeby) można je usunąć. Prawdopodobnie informacji tych nie będziesz musiał wykorzystywać szczególnie często (chyba, że jesteś programistą-eksperymentatorem), ale w każdym systemie od czasu do czasu zdarzają się sytuacje awaryjne. Problemy mają czasem mnożyć się przy wzroście liczby użytkowników.

kowników, a wtedy restart systemu jest rzeczą dość uciążliwą. Polecenie `ps` pozwala przeważnie na uniknięcie konieczności resetowania komputera.

Konfigurowanie i dodawanie do systemu urządzeń SCSI omówione jest w rozdziale 36. „Obsługa urządzeń SCSI”.

Jak skonfigurować system, by można było korzystać z poczty elektronicznej, dowiesz się z rozdziału 40. „Konfigurowanie poczty”.

Polecenie `tar` i tworzenie kopii zapasowych przedstawione są w rozdziale 45. „Kopie zapasowe”.

Rozdział 35.

Użytkownicy i konta

Tim Parker

W tym rozdziale:

- υ Konto administratora
- υ Konta użytkowników – plik `/etc/passwd`
- υ Użytkownicy systemowi
- υ Dodawanie nowego konta
- υ Usuwanie kont
- υ Grupy
- υ Polecenie `su`

Każdy użytkownik ma dostęp do systemu linuxowego tylko poprzez swoje konto. Każde konto użytkownika musi zostać skonfigurowane przez administratora (za wyjątkiem konta `root`, które jest zakładane podczas instalacji systemu). Choć z niektórych systemów linuxowych korzysta tylko jeden użytkownik, nie powinien on używać na co dzień konta `root`. W wielu systemach możliwa jest równoczesna praca kilku użytkowników, czy to dzięki konsolom wirtualnym, czy też przez modem, czy za pomocą terminalu. Umiejętność konfigurowania i zarządzania kontami użytkowników jest ważnym aspektem administrowania systemem.

W tym rozdziale przyjrzymy się następującym zagadnieniom:

- υ konto administratora (`root`),
- υ dodawanie nowych kont,
- υ usuwanie kont,
- υ grupy użytkowników,
- υ zarządzanie grupami.

Konto administratora

W trakcie instalowania systemu linuxowego tworzony jest jedno konto – konto administratora, o identyfikatorze `root`. Podczas gdy większość kont użytkowników jest zabezpieczona przed wyrządzeniem jakichkolwiek szkód w systemie, użytkownik `root` może za pomocą jednego polecenia zniszczyć cały system. Na użytkownika `root` nie są nałożone żadne ograniczenia.



Możliwości, jakie oferuje konto `root`, mogą uzależniać. Nie trzeba martwić się o prawa dostępu czy konfigurację oprogramowania; można zrobić wszystko i wszędzie. To powoduje, że większość nowych użytkowników używa konta `root` do codziennej pracy. Zwykle porzucają ten zwyczaj w momencie, gdy przez przypadek uda im się zniszczyć cały system. Brak ograniczeń nałożonych na konto `root` pociąga za sobą brak jakichkolwiek zabezpieczeń. Z tego powodu konto `root` powinno być używane tylko do zadań o charakterze administracyjnym. Nie używaj konta `root` do codziennej pracy!

Konto `root` powinno być używane tylko wtedy, gdy sytuacja naprawdę tego wymaga. Dobrym pomysłem jest zmiana znaku zachęty dla użytkownika `root`, dzięki czemu będziesz widział, że jesteś zalogowany jako `root` i być może zastanowisz się przed wydaniem jakiegoś niefortunnego polecenia. Możesz to zrobić ustalając odpowiednio wartość zmiennej środowiskowej `PS` – dokładniejsze informacje na ten temat znajdziesz w rozdziale 14. „Programowanie w języku powłoki”. Jeśli systemu używasz tylko Ty i uda Ci się go przez przypadek zniszczyć – to jeszcze niewielki problem. O wiele gorzej sytuacja wygląda, gdy zniszczysz efekty pracy innych użytkowników...

Mając na uwadze wszystkie powyższe ostrzeżenia, nie masz chyba wątpliwości, że pierwsze, co powinieneś zrobić po zainstalowaniu systemu, to założenie dla siebie konta do codziennego użytku. Hasło użytkownika `root` nie może być łatwe do odgadnięcia dla innych użytkowników systemu. Powinieneś również w miarę regularnie je zmieniać.

Warto także założyć konta służące do prac administracyjnych nie wymagających aż tak szerokich uprawnień, jakie oferuje `root`, na przykład do tworzenia kopii zapasowych na taśmiech. Użytkownik ten może mieć np. dostęp do całego systemu plików (tak jak `root`), ale bez prawa zapisu. Podobne konta mogą zostać stworzone do obsługi poczty, Internetu itp. Przemyśl dobrze przywileje, jakie trzeba nadać każdemu użytkownikowi – dzięki temu system będzie bezpieczniejszy i unikniesz przypadkowych uszkodzeń.

Jeśli chodzi o ścisłość, konto administratora nie musi nazywać się `root`. Może mieć dowolną nazwę, ale jego identyfikator numeryczny musi być ustawiony na 0 (w pliku `/etc/passwd`).

Konta użytkowników - plik /etc/passwd

Nawet jeśli jesteś jedynym użytkownikiem systemu, warto wiedzieć kilka rzeczy o kontach użytkowników i zarządzaniu nimi, choćby dlatego, że powinieneś posiadać konto do codziennej pracy (inne niż `root`). Musisz więc umieć dodać do systemu nowego użytkownika. Jeśli inne osoby również będą korzystać z systemu, czy to przez modem, czy też bezpośrednio, powinieneś założyć dla każdej z nich osobne konto. Można też utworzyć konta ogólnodostępne, np. dla przyjaciół tylko okazjonalnie korzystających z Twojego systemu.

Każda osoba używająca systemu powinna posiadać własny identyfikator oraz hasło. Jedynym wyjątkiem mogą być użytkownicy typu `guest`, którzy powinni mieć jednak bardzo ograniczone prawa dostępu. Dzięki założeniu osobnego konta dla każdego użytkownika znacznie poprawia się bezpieczeństwo systemu. Jako administrator masz również lepsze pojęcie o tym, kto i w jakim celu z niego korzysta.

Wszystkie informacje o kontach użytkowników przechowywane są w pliku `/etc/passwd`. Jego właścicielem powinien być `root`, a identyfikator grupy powinien być ustawiony na `0` (grupa `root` albo `system`, zależnie od wpisu w pliku `/etc/groups`). Prawa dostępu powinny być tak ustawione, by zapisywać do tego pliku mógł tylko `root`, natomiast prawo odczytu powinno mieć wszyscy użytkownicy (o grupach i prawach dostępu będziemy jeszcze mówić w dalszej części tego podrozdziału). Wpisy w pliku `/etc/passwd` mają następujący format:

```
nazwa_użytkownika:hasło:ID_użytkownika:ID_grupy:komentarz:katalog_domowy:  
powłoka_domyślna
```

Listing 35.1 przedstawia przykładową zawartość takiego pliku.

Listing 35.1. Plik `/etc/passwd` utworzony podczas instalacji Linuxa

```
root:jyGSAETCyfRDE:0:0:root:/root:/bin/bash  
bin:*:1:1:bin:/bin:  
daemon:*:2:2:daemon:/sbin:  
adm:*:3:4:adm:/var/adm:  
lp:*:4:7:lp:/var/spool/lpd:  
sync:*:5:0:sync:/sbin:/bin/sync:  
shutdown:*:6:0:shutdown:/sbin:/sbin/shutdown:  
halt:*:7:0:halt:/sbin:/sbin/halt:  
mail:*:8:12:mail:/var/spool/mail:  
news:*:9:13:news:/var/spool/news:  
uucp:*:10:14:uucp:/var/spool/uucp:  
operator:*:11:0:operator:/root:  
games:*:12:100:games:/usr/games:  
gopher:*:13:30:gopher:/usr/lib/gopher-data:  
ftp:*:140:50:FTP User:/home/ftp:  
nobody:*:-1:100:Nobody:/:
```

Każdy wiersz składa się z siedmiu pól rozdzielonych dwukropkami. Jeśli dane pole nie ma zawierać żadnych informacji, dwukropki muszą pozostać (tzn. każdy wiersz musi zawierać sześć dwukropków). Pola te mają następujące funkcje:

nazwa_uzytkownika:	niepowtarzalny identyfikator użytkownika;
hasło:	hasło dostępu do systemu, zakodowane tak, by jego odczytanie nie było możliwe;
ID_uzytkownika:	identyfikator numeryczny, inny dla każdego użytkownika, używany przez system;
ID_grupy:	liczba określająca grupę, do której należy użytkownik;
komentarz:	zwykle prawdziwe imię i nazwisko użytkownika, ale można tu wpisać dowolny tekst;
katalog_domowy:	katalog, w którym użytkownik znajdzie się po zalogowaniu;
powłoka_domyślna	program uruchamiany po zakończeniu procesu logowania.

Każdemu polu przyjrzymy się nieco dokładniej. Powinieneś orientować się, w jaki sposób są one wykorzystywane przez programy. Taki plik, jak `/etc/passwd`, używany jest nie tylko w systemie Linux, ale we wszystkich systemach UNIX-owych.

Identyfikatory użytkowników

Identyfikatorem użytkownika może być dowolny tekst, zwykle o długości do ośmiu znaków, inny dla każdego użytkownika. Ponieważ jest on podstawą identyfikacji podczas pracy w sieci, powinien być prosty i oczywisty; nie od rzeczy jest na przykład kombinacja taka jak pierwsza litera imienia + nazwisko lub jego część (np. `tszydl` czy `wkuc`) – taka konwencja jest dość często stosowana w większych systemach.

Zauważ, że wszystkie litery w powyższych przykładach są małe. Wielkość liter jest istotna w systemach UNIX-owych, więc identyfikator `tszydl` nie jest tożsamy z `TSzydl`. Zwykle używa się identyfikatorów składających się tylko z małych liter. Dozwolone, choć rzadko używane, są podkreślenia, przecinki, cyfry i niektóre inne symbole.

W niewielkich systemach często używa się identyfikatorów brzmiących nieco bardziej przyjaźnie, na przykład `michal` czy `iwona`. Jeśli dwóch użytkowników nosi te same imiona, trzeba znaleźć sposób na ich rozróżnienie (np. `tomek` i `tom`).

Niektórzy użytkownicy wolą, żeby ich identyfikatorem było przewisko czy cokolwiek innego. Wszystko jest w porządku w niewielkich sieciach i do zastosowań „rozrywkowych”, ale staje się dziwactwem w pracy, gdzie tylko utrudnia życie innym osobom próbującym skontaktować się z danym użytkownikiem.

Hasła

System przechowuje hasła użytkowników w postaci zakodowanej. Hasło może być zmienione przez użytkownika po zalogowaniu się (co wymaga znajomości tegoż hasła) lub też przez użytkownika `root`. Pole pliku `/etc/passwd` zawierające hasło jest bardzo wrażliwe na wszelkiego rodzaju modyfikacje, które w zasadzie zawsze powodują zablokowanie dostępu do danego konta (aż do zmiany hasła przez administratora systemu).



W niektórych wersjach UNIX-a dla zapewnienia większego bezpieczeństwa hasła nie są przechowywane w pliku `/etc/passwd`. Jeśli w Twoim systemie zamiast zakodowanych haseł w pliku `passwd` znajduje się znak `*`, oznacza to, że istnieje inny plik, w którym zapisane są ich rzeczywiste wartości (ta technika nazywa się *password shadowing* - ukrywanie haseł).

Systemy korzystające z Yellow Pages lub Network Information Service nie wykorzystują tego pola, a zamiast tego używają wspólnego pliku zawierającego dane o użytkownikach. Pozwala to każdemu użytkownikowi na zalogowanie się do systemu za pomocą dowolnego komputera podłączonego do sieci.

Podczas logowania program `login` sprawdza, czy wprowadzone hasło odpowiada zaktualizowanemu. Jeśli nie, użytkownik nie uzyska dostępu do systemu.

Pole to może również być wykorzystane do ograniczenia dostępu do systemu. Jeśli dane konto nie powinno być używane, wystarczy do pola zawierającego hasło wprowadzić pojedynczą gwiazdkę (*). Ma to zastosowanie również w przypadku użytkowników systemowych, takich jak `lp` czy `sync`. Gwiazdka w polu hasła spowoduje zablokowanie dostępu do systemu dla określonego użytkownika.

Można pozostawić to pole puste, co daje wszystkim możliwość dostania się do systemu bez podawania hasła. Użytkownik, który ma taką możliwość, powinien mieć bardzo ograniczone prawa dostępu do plików w systemie, ponieważ każdy może zalogować się używając jego identyfikatora. Ogólnie nie należy jednak pozostawiać pustych haseł, chyba że używasz systemu wyłącznie do zabawy i w systemie plików nie są zapisane żadne istotne dane.

Nie powinieneś próbować wprowadzić zakodowanego hasła do pliku `/etc/passwd` ręcznie. Nie jest możliwe odtworzenie metody kodowania haseł, więc prawie na pewno skończy się to zablokowaniem dostępu do danego konta (aż do zmiany hasła przez administratora systemu).

Numeryczny identyfikator użytkownika

Z każdym identyfikatorem tekstowym użytkownika związany jest również identyfikator numeryczny. To on, a nie identyfikator tekstowy, jest używany przez system do rozróżniania użytkowników. Programy wyświetlające informacje o użytkownikach dokonują

translacji numeru na odpowiadający mu identyfikator tekstowy (na podstawie zawartości pliku `/etc/passwd`) dla wygody czytającego.

W większości systemów UNIX-owych numery użytkowników mają wartości od 100 w górę, ponieważ numery od 0 do 99 zarezerwowane są dla użytkowników systemowych. W podanym wcześniej przykładowym pliku `/etc/passwd` można zauważyć, że użytkownikowi `root` przypisany jest numer 0 – jest to reguła obowiązująca w każdym systemie. Użytkownikowi `nobody` (używanemu do obsługi sieciowych systemów plików NFS) przypisany jest numer -1, który jest nieprawidłowy. Przy zakładaniu nowych kont dobrze jest numerować użytkowników kolejno, rozpoczynając od wartości 100, następnie 101 itd.

Identyfikator grupy

Identyfikator grupy zapisany w pliku `/etc/passwd` jest również wartością liczbową. Decyduje on o tym, do której grupy należał będzie dany użytkownik po zalogowaniu. Grupy używane są dla ułatwienia zarządzania prawami dostępu do plików ale w wielu małych systemach ich możliwości nie są wykorzystywane. Numery grup rozpoczynają się od 0. Grupa o nazwie `users` zwykle posiada numer 100.

Identyfikator grupy używany jest do weryfikowania praw dostępu do plików, oraz przy tworzeniu i modyfikowaniu plików. Jeśli w Twoim systemie funkcjonuje tylko jedna grupa użytkowników, nie musisz martwić się o przydzielanie prawidłowych wartości identyfikatorów grupy. Jeśli chcesz założyć kilka grup, musisz również zmodyfikować plik `/etc/group`.

Komentarz

Pole komentarza używane jest przez administratora systemu po to, aby wpis można było łatwiej zidentyfikować. Zwykle zawiera ono prawdziwe imię i nazwisko użytkownika, czasem również wydział czy numer ewidencyjny. Nazywane jest czasem polem GE-COS – nazwa ta pochodzi od pierwszego systemu, w którym go używano.

Niektóre programy bazują na informacjach zapisanych w tym polu przy wyświetlanie danych o użytkownikach, nie należy więc wpisywać tam żadnych poufnych danych. Systemy poczty na przykład informują na jego podstawie o tym, kto wysłał daną wiadomość. Choć nie trzeba używać tego pola, w większych systemach ułatwia ono życie zarówno administratorowi, jak i innym użytkownikom.

Katalog domowy

Pole to określa, w jakim katalogu będzie znajdował się użytkownik po zalogowaniu się do systemu. Każdy użytkownik powinien posiadać własny katalog domowy, w którym nie są na niego nałożone żadne ograniczenia. W nim zapisywane są pliki konfiguracyjne różnych programów. Na ten katalog powinna wskazywać wartość zmiennej środowiskowej `HOME`.

Zwykle wszystkie katalogi domowe użytkowników zebrane są w jednym miejscu. W systemie Linux jest to przeważnie katalog `/home`. Czasem – jeśli administrator jest przyzwyczajony do innego ich umieszczenia – mogą znajdować się gdzieś indziej (na przykład w katalogu `/usr` czy `/user`; zdarza się również, szczególnie w większych systemach, że katalogi domowe użytkowników znajdują się w kilku różnych podkatalogach – *przyp. tłum.*).

Powłoka domyślna

Pole określające powłokę domyślną zawiera nazwę programu, który zostanie uruchomiony po zakończeniu logowania. W większości przypadków jest to interpreter poleceń `bash` czy `tcsh`, ale może to być dowolna aplikacja, taka jak program do obsługi poczty itp. – na przykład dla użytkownika `uucp` jest to program `uucp`. Jeśli pole to pozostawione jest puste, system uruchomi interpreter domyślny (zwykle `bash`, ale zależy to od konfiguracji systemu).

Wiele wersji Linuxa pozwala użytkownikowi na zmianę wartości tego pola za pomocą polecenia `passwd -s` lub `chsh`. Jeśli używasz któregoś z tych poleceń, możesz zmienić program uruchamiany tylko na jeden z tych, których nazwy zapisane są w pliku `/etc/shells` (plik ten można modyfikować dowolnym edytorem tekstów). Pozwala to na zachowanie wyższego poziomu bezpieczeństwa. Administrator może w tym polu wpisać dowolny program. Jeśli w Twoim systemie używany jest plik `/etc/shells`, powinien on mieć takie same prawa dostępu, jak plik `/etc/passwd`, ponieważ konsekwencje płynące z możliwości jego modyfikacji przez niepowołane osoby są równie niebezpieczne jak nieautoryzowany dostęp do pliku `/etc/passwd`.

Użytkownicy systemowi

Podany wcześniej przykładowy plik `/etc/passwd` zawiera kilkanaście wpisów, wprowadzonych przez system podczas instalacji. Wszyscy zdefiniowani w nim użytkownicy spełniają w systemie linuxowym specjalną rolę. Kilku z nich jest szczególnie wartych uwagi:

root	administrator – użytkownik, na którego nie są nałożone żadne ograniczenia;
daemon	konto używane dla procesów systemowych;
bin	właściciel niektórych plików wykonywalnych;
sys	właściciel niektórych plików wykonywalnych;
adm	właściciel plików administracyjnych;
uucp	konto używane do komunikacji za pomocą protokołu UUCP.

W innych systemach mogą również być założone konta do specyficznych zadań; ich nazwy zwykle wyjaśniają przeznaczenie – na przykład użytkownik `postmaster` przeznacza-

czony jest do obsługi poczty. Większości z tych wpisów nie należy modyfikować. W polu hasła mają one wpisaną gwiazdkę, co zabezpiecza przed ich wykorzystaniem do zalogowania się do systemu.

Dodawanie nowego konta

Nowego użytkownika można dodać do systemu na dwa sposoby: ręcznie, edytując plik `/etc/passwd`, bądź też za pomocą skryptu automatyzującego wszystkie czynności, dzięki czemu minimalizowane jest ryzyko popełnienia błędów, co jest szczególnie ważne dla mało doświadczonych administratorów lub w przypadku dodawania większej liczby użytkowników. Tylko użytkownik `root` może modyfikować plik `/etc/passwd`.



Przed dokonaniem jakichkolwiek zmian w pliku `/etc/passwd` zrób jego kopię zapasową. Jeśli przez przypadek uszkodzisz go, może się zdarzyć, że nie będziesz mógł się zalogować nawet jako `root` (do systemu można wówczas dostać się tylko w trybie administracyjnym). Na wypadek problemów dobrze mieć kopię tego pliku na dyskietce startowej.

Aby dodać wpis do pliku `/etc/passwd`, można użyć dowolnego edytora tekstów. Dane nowego użytkownika można dopisać na końcu pliku; każdemu użytkownikowi musi odpowiadać oddzielnny wiersz. Upewnij się, że numery identyfikacyjne użytkowników się nie powtarzają. Przykładowo, aby dodać użytkownika o identyfikatorze `bill`, numerze identyfikacyjnym 103 i numerze grupy 100, którego katalogiem domowym jest `/home/bill`, a domyślnym interpreterm poleceń – `bash`, powinieneś dodać wiersz:

```
bill::103:100:Bill Smalwood:/home/bill:/bin/bash
```

Pole określające hasło należy pozostawić puste, ponieważ nie da się wprowadzić zaktualizowanego hasła samodzielnie. Następnie należy zmienić hasło używając polecenia `passwd`:

```
passwd bill
```

Po wydaniu tego polecenia należy dwukrotnie wprowadzić hasło. Użytkownik `bill` powinien je zmienić jak najszybciej (na takie, które będzie znał tylko on). Niektórzy administratorzy jako pierwsze hasło wpisują jakiś łatwy do odgadnięcia tekst, na przykład `password` albo identyfikator użytkownika, zmuszając jednocześnie użytkownika do zmiany hasła zaraz po zalogowaniu się. Jest to dopuszczalne, jeśli pierwsze logowanie następuje zaraz po założeniu hasła. W przeciwnym przypadku konto przez dłuższy czas może pozostać niezabezpieczone, stwarzając potencjalne zagrożenie dla bezpieczeństwa systemu.

Po wpisaniu odpowiednich danych do pliku `/etc/passwd` należy utworzyć katalog domowy użytkownika, o ile nie istnieje on jeszcze w systemie. Jego właścicielem musi być osoba, która ma go używać:

```
mkdir /home/bill  
chown bill /home/bill
```

Każdy użytkownik musi należeć do jakiejś grupy. Jeśli w Twoim systemie jest zdefiniowana tylko jedna grupa, dodaj identyfikator nowego użytkownika do odpowiedniego wiersza w pliku `/etc/group`. Plik ten jest omówiony bardziej szczegółowo w podrozdziale „Grupy”.

Na koniec należy jeszcze skopiować do katalogu domowego nowego użytkownika pliki konfiguracyjne interpreterów poleceń itp., na przykład tak:

```
cp /home/iwona/.profile /home/bill/.profile  
chown bill /home/bill/.profile
```

Pliki takie warto przejrzeć na wypadek występowania w nich nieprawidłowych wpisów, na przykład ustawiających wartość zmiennej `HOME` czy katalog poczty. W przypadku używania powłoki C plik konfiguracyjny nazywa się `.login` lub `.cshrc`, w przypadku powłoki `bash` oraz `pdksh` – `.profile`.

Ogólnie proces dodawania nowego użytkownika składa się z trzech etapów:

1. dodania wpisu do pliku `/etc/passwd`,
2. utworzenia katalogu domowego i przypisania mu odpowiedniego właściciela,
3. skopiowania plików konfiguracyjnych do katalogu nowego użytkownika.

Niektóre dystrybucje Linuxa zawierają polecenie przeniesione z wersji Berkley BSD UNIX o nazwie `vipw`, które wywołuje domyślny edytor (zwykle `vi`) na tymczasowej kopii pliku `/etc/passwd`; dzięki takiemu rozwiązaniu nie jest możliwe, aby dwóch użytkowników edytowało równocześnie ten sam plik, co zapobiega potencjalnym problemom. Przed zapisaniem pliku sprawdzana jest jego poprawność syntaktyczna, co dodatkowo podnosi poziom bezpieczeństwa.

Często można również użyć skryptów automatyzujących dodawanie nowych użytkowników, które nazywają się `adduser` albo `useradd`. Zwykle zadają one pytanie o początkowe hasło, które można pozostawić puste. Główną ich zaletą jest fakt, że automatycznie kopiąją pliki konfiguracyjne dla wszystkich obsługiwanych powłok, często również same ustawiają poprawne wartości zmiennych środowiskowych. Może to znacząco ułatwić proces dodawania nowych użytkowników.



Hasła w systemie linuxowym to jeden z najbardziej newralgicznych, z punktu widzenia bezpieczeństwa, problemów. Do każdego konta powinno być przypisane bezpieczne hasło, chyba że masz pewność, że nikt obcy nie korzysta z systemu (pewność taką możesz mieć tylko wtedy, gdy nie jesteś połączony do sieci, a komputer jest pod klawiszem). Hasła przypisuje się i zmienia poleceniem `passwd`. Administrator może zmienić każde hasło w systemie, natomiast zwykły użytkownik może zmienić tylko swoje hasło.

Usuwanie kont

Podobnie jak dodawanie nowego konta, usuwanie można przeprowadzić na dwa sposoby: ręcznie lub używając odpowiedniego skryptu, który zwykle nazywa się `deluser` lub `userdel`. Skrypt pyta o identyfikator użytkownika, którego należy usunąć z systemu, a następnie wykonuje wszystkie niezbędne czynności: usuwa wpis z pliku `/etc/passwd`, może również posprzątać w katalogach kolejek i usunąć katalog domowy.

Jeśli chcesz usunąć konto ręcznie, najpierw usuń odpowiedni wiersz z pliku `/etc/passwd`. Następnie możesz usunąć pliki, których właścicielem jest nieistniejący już użytkownik. Możesz również usunąć jego katalog domowy wraz z całą zawartością za pomocą polecenia:

```
rm -r /home/nazwa_użytkownika
```

`/home/nazwa_użytkownika` to pełna ścieżka dostępu do katalogu domowego użytkownika. Upewnij się najpierw, że w katalogu tym nie ma żadnych plików, które chciałbyś zatrzymać!

Następnie należy usunąć plik poczty, który zwykle znajduje się w katalogu `/usr/spool/mail/nazwa_użytkownika`. Powinieneś sprawdzić, czy w pliku `/etc/aliases` nie ma wpisów dotyczących usuwanego użytkownika. Plik ten modyfikuje się za pomocą polecenia `newaliases`.

Na koniec wyłącz wszystkie zadania programów `cron` i `at`, których właścicielem był usuwany użytkownik. Dane o nich możesz uzyskać używając polecenia `crontab`.

Jeśli musisz pozostawić danego użytkownika w systemie z jakiegoś powodu, najłatwiej uniemożliwić mu logowanie, wpisując w miejsce hasła w pliku `/etc/passwd` gwiazdkę. Tak zablokowane konto może być ponownie uaktywnione poleceniem `passwd`.

Proces ręcznego usuwania użytkownika składa się z następujących etapów:

1. usunięcie wpisów z plików `/etc/passwd` i `/etc/group`,
2. usunięcie poczty użytkownika,
3. wyłączenie zadań `at` i `cron`,
4. usunięcie katalogu domowego (o ile pliki w nim zawarte nie będą już potrzebne).

Czasem trzeba tymczasowo zablokować konto użytkownika (na przykład kiedy wyjeździ on na dłuższe wakacje i nie będzie używały systemu). W takim przypadku należy w miejsce hasła wpisać do pliku `/etc/passwd` gwiazdkę. Można również dodać gwiazdkę przed pierwszym znakiem hasła – po powrocie użytkownika będzie można ją usunąć, a hasło użytkownika nie ulegnie zmianie.

Grupy

Każdy użytkownik w systemie UNIX-owym musi należeć do jakieś grupy. Grupa to logiczny zbiór użytkowników. Przykładowo, grupę mogą stanowić wszyscy pracownicy jednego wydziału, którzy powinni mieć dostęp do jakichś wspólnych danych, czy możliwość używania danego urządzenia, na przykład skanera czy kolorowej drukarki laserowej. W systemie może być dowolnie dużo grup. Użytkownik może być w danym momencie przypisany tylko do jednej grupy. Jeśli jednak należy jeszcze do innych (każdy użytkownik może należeć do dowolnej liczby grup), może w każdej chwili sam przyłączyć się do odpowiedniej grupy.

Prawa dostępu dla grup można ustalić tak, by ich członkowie mieli np. dostęp do danych urządzeń, plików czy systemów plików, natomiast nie mieli ich inni użytkownicy. Przykładowo, wszyscy pracownicy księgowości powinni mieć dostęp do rachunków firmy, do których jednak nie powinni mieć dostępu pozostali pracownicy – takie ograniczenie może być z łatwością zrealizowane za pomocą grup.

W wielu systemach linuxowych istnieje tylko jedna grupa użytkowników. Takie rozwiązanie jest akceptowalne w małych, najwyższej kilkusobowanych systemach. Dostęp do plików i urządzeń określany jest wówczas nie w oparciu o przynależność do grupy, ale o indywidualne ustawienia praw dostępu dla każdego pliku. Zalety używania grup ujawniają się, gdy z systemu zaczyna korzystać większa liczba użytkowników – na przykład dzieci i przyjaciele – wówczas, używając grup, można z łatwością manipulować prawami dostępu do plików.

Informacje o grupach przechowywane są w pliku `/etc/groups`, którego struktura jest podobna do struktury pliku `/etc/passwd`. Plik `/etc/groups` pochodzący ze świeżo zainstalowanego systemu przedstawiony jest na listingu 35.2.

Listing 35.2. Plik `/etc/groups` utworzony podczas instalacji Linuxa

```
root::0:root
bin::1:root,bin,daemon
daemon::2:root,bin,daemon
sys::3:root,bin,adm
adm::4:root,adm,daemon
tty::5:
disk::6:root
lp::7:daemon,lp
mem::8:
kmem::9:
wheel::10:root
mail::12:mail
news::13:news
uucp::14:uucp
man::15:
floppy:x:19:
games::20:
gopher::30:
dip::40:
ftp::50:
nobody::99:
users::100:
pppusers:x:230:
popusers:x:231:
slipusers:x:232:
```

Każdy wiersz składa się z czterech pól, rozdzielonych dwukropkami (dwa dwukropki obok siebie oznaczają, że dane pole jest puste):

`nazwa_grupy:hasło_grupy:ID_grupy:użytkownicy`

Pola te mają następujące znaczenie:

nazwa_grupy: niepowtarzalna nazwa grupy, składająca się maksymalnie z ośmiu znaków;

- hasło_grupy:** pole to zwykle pozostawiane jest puste (w niektórych wersjach systemu tu przechowywane jest hasło, które użytkownik musi podać, chcąc przyłączyć się do grupy); większość wersji Linuxa nie obsługuje tego pola;
- ID_grupy:** numer identyfikacyjny, różny dla każdej grupy, używany przez system operacyjny;
- użytkownicy:** lista użytkowników należących do danej grupy.

W każdym systemie istnieje pewna liczba grup tworzonych podczas instalacji, służących do obsługi samego systemu operacyjnego (jak na przykład `bin`, `mail`, `uucp`, `sys` itd.). Na powyższym wydruku są to wszystkie grupy oprócz ostatnich czterech. Nigdy nie powinieneś pozwolić użytkownikom, by należeli do jednej z tych grup, ponieważ daje im to prawa dostępu zagrażające bezpieczeństwu systemu. Mogą do nich należeć tylko użytkownicy systemowi.

Grupy systemowe

Jak widać na listingu 35.2, istnieje kilka grup systemowych, do których nie należą zwykli użytkownicy systemu. Funkcje niektórych z nich są następujące:

- root/wheel/system** zwykle używana, aby uzyskać prawa użytkownika `root` za pomocą polecenia `su`; grupa ta jest właścicielem większości plików systemowych;
- daemon** używana do obsługi katalogów kolejki dla poczty, drukarki itp.;
- kmem** używana przez programy wymagające bezpośredniego dostępu do pamięci (np. `ps`);
- sys** właściciel niektórych plików systemowych; w niektórych systemach grupa ta zachowuje się tak samo jak `kmem`;
- tty** właściciel plików obsługujących terminal.

Domyślana grupa dla zwykłych użytkowników nazywa się `users` i posiada numer identyfikacyjny 100 (w niektórych systemach funkcję tę pełni grupa `group` o numerze 50).

Dodawanie nowej grupy

Aby utworzyć nową grupę, można ręcznie edytować plik `/etc/groups`, ale można również w tym celu użyć skryptu o nazwie `addgroup` czy `groupadd`. Z punktu widzenia administratora ręczna edycja pliku jest nawet wygodniejsza, ponieważ umożliwia od razu wgląd w całą strukturę grup. Poza tym nie we wszystkich systemach dostępne są odpowiednie skrypty.

Jeśli chcesz modyfikować zawartość pliku `/etc/groups`, powinieneś najpierw wykonać jego kopię zapasową. Za pomocą dowolnego edytora tekstów dodaj nowy wiersz dla każdej grupy, którą zamierzasz utworzyć. Upewnij się, że nie popełniłeś żadnych błędów składniowych. Poniżej znajdują się dwa wiersze, które spowodują utworzenie dwóch nowych grup; do każdej z nich należeć będzie jeden użytkownik:

```
rachunki::101:bill  
skaner::102:iwona
```

Numery identyfikacyjne tych grup to 101 i 102. Nowe wiersze można dodać w dowolnym miejscu pliku. Identyfikatory użytkowników mających należeć do danej grupy powinny znajdować się na końcu wiersza. W powyższym przykładzie do każdej z grup należy tylko jeden użytkownik – w następnym podrozdziale dowiesz się, w jaki sposób można dodać do grupy więcej niż jednego użytkownika. Definicje grup nie muszą pojawiać się w kolejności zgodnej z ich numerami identyfikacyjnymi.

Po dokonaniu zmian w pliku `/etc/groups` powinieneś sprawdzić, czy nadal ma on prawidłowo ustawione prawa dostępu. Jego właścicielem powinien być użytkownik `root` (lub `system`). Nikt prócz właściciela nie powinien mieć prawa zapisu do tego pliku.

Dodawanie użytkowników do grupy

Każdy użytkownik może być członkiem wielu grup; w takim przypadku jego identyfikator powinien pojawić się w każdym wierszu odpowiadającym grupie, do której on przynależy (identyfikatory użytkowników w pliku `/etc/groups` powinny być rozdzielone przecinkami). Nie ma formalnego ograniczenia liczby użytkowników mogących należeć do danej grupy, ale w praktyce nie jest aż tak dobrze: liczba użytkowników należących do grupy jest efektywnie ograniczana przez maksymalną długość wiersza w pliku `/etc/groups` (255 znaków). Istnieją metody obejścia tego problemu, ale w większości systemów nie będzie to konieczne.

Oto fragment pliku `/etc/groups` zawierającego definicje kilkuosobowych grup:

```
rachunki::101:bill,iwona,michal,root,szymon  
scanner::102:iwona,janek,root  
cad::103:michal,piotr,wojtek
```

Identyfikatory użytkowników mogą pojawiać się w dowolnej kolejności.

W danym momencie użytkownik może należeć tylko do jednej grupy. O tym, do której grupy będzie należał po zalogowaniu, decyduje wpis w pliku `/etc/passwd`.

Usuwanie grupy

Jeśli zdecydujesz, że dana grupa nie jest już potrzebna, możesz usunąć odpowiedni wpis z pliku `/etc/groups`. Powinieneś również sprawdzić w pliku `/etc/passwd`, czy któryś użytkownik nie loguje się jako członek tej grupy (w takim przypadku bowiem nie będzie mógł się zalogować). Możesz również wyszukać w systemie plików pliki i katalogi należące do danej grupy i zmienić ich przynależność (bądź też usunąć). Jeśli tego nie zrobisz, inni użytkownicy mogą nie mieć dostępu do tych plików.

W niektórych wersjach Linuxa do usuwania wpisów z pliku `/etc/group` przeznaczony jest skrypt o nazwie `delgroup` lub `groupdel`.

Polecenie su

Czasem zachodzi potrzeba wydania jakiegoś polecenia jako inny użytkownik. Jeśli jesteś zalogowany jako `root`, a chcesz stworzyć pliki, których właścicielem będzie `bill`, łatwiej zalogować się jako `bill` niż zmieniać potem prawa dostępu do plików. Podobnie wygląda sytuacja, jeśli jesteś zalogowany jako zwykły użytkownik, a potrzebujesz na chwilę uprawnień administratora. W takich sytuacjach z pomocą przychodzi polecenie `su`.

Polecenie `su` zmienia identyfikator użytkownika i przyznaje wszystkie uprawnienia, jakie posiada inny użytkownik. Jego parametrem jest identyfikator użytkownika, którym chcesz się na chwilę stać, na przykład:

```
su root
```

Po wydaniu takiego polecenia system zapyta Cię o hasło. Jeśli wprowadzisz je poprawnie, będziesz miał wszystkie uprawnienia użytkownika `root` (do czasu, aż wydasz polecenie `exit` lub wcisniesz Control+d). Jeśli jesteś zalogowany jako `root`, możesz stać się innym użytkownikiem bez podawania hasła, na przykład wydając polecenie:

```
su bill
```

Po wcisnięciu kombinacji klawiszy Control+d znowu staniesz się użytkownikiem `root`. Jeśli jesteś zalogowany jako zwykły użytkownik, a chcesz przełączyć się na chwilę na inne konto zwykłego użytkownika, również musisz podać hasło.

Podsumowanie

W tym rozdziale przyjrzaliśmy się plikom `/etc/passwd` oraz `/etc/groups`, za pomocą których kontrolowany jest dostęp do systemu. Jak widać, ich budowa jest prosta i mogą być łatwo modyfikowane dowolnym edytorem tekstów. Pamiętaj o tym, że są to pliki bardzo ważne dla prawidłowego działania systemu, więc należy wykonywać ich kopie zapasowe i sprawdzać, czy mają przypisane odpowiednie prawa dostępu.

Konfigurowanie i dodawanie do systemu urządzeń SCSI omówione jest w rozdziale 36. „Obsługa urządzeń SCSI”.

Jak skonfigurować system, by można było korzystać z poczty elektronicznej, dowiesz się z rozdziału 40. „Konfigurowanie poczty”.

Polecenie `tar` i tworzenie kopii zapasowych przedstawione są w rozdziale 45. „Kopie zapasowe”.

Rozdział 36.

Obsługa urządzeń SCSI

Tim Parker

W tym rozdziale:

- υ Nowsze standardy SCSI
- υ Obsługiwane urządzenia SCSI
- υ Sterowniki SCSI
- υ SCSI – rozwiązywanie problemów

SCSI (ang. *Small Computer System Interface*; skrót ten wymawia się *skazi*) jest standardową metodą komunikacji pomiędzy komputerem a urządzeniami peryferyjnymi. Posiada wiele zalet, które powodują, że jest lepszy od innych interfejsów (takich jak IDE), niestety, zwykle jest również nieco droższy.

SCSI używa dedykowanej karty kontrolera (zintegrowanej z niektórymi lepszymi płytami głównymi), do której podłączany jest łańcuch urządzeń. Mogą to być urządzenia wewnętrzne – wtedy kabel ma formę płaskiej taśmy, lub zewnętrzne – kabel do ich podłączenia jest okrągły, ekranowany. Każde urządzenie musi posiadać inny identyfikator (od 0 do 7, ale nowsze sterowniki dopuszczają do 16 urządzeń). Zwykle karcie kontrolera przypisany jest numer 7, natomiast dyskowi twardemu, z którego uruchamiany jest system – 0.

Główną zaletą interfejsu SCSI jest jego duża prędkość. Ponadto w urządzeniach SCSI zamontowana jest elektronika służąca do obsługi interfejsu, co powoduje, że jest im łatwiej się ze sobą porozumieć. Następną zaletą jest fakt, że system sam się konfiguruje. Po podłączeniu do magistrali SCSI np. skanera (mającego oczywiście ustawiony unikalny identyfikator) urządzenie samo przesyła do sterownika dane o sobie, dzięki czemu cały proces konfiguracji może odbyć się automatycznie.

Na końcu łańcucha SCSI musi być zamontowany terminator. Jest to w najprostszym przypadku zestaw oporników, zapewniający właściwe warunki rozchodzenia się fal elektromagnetycznej. Niektóre urządzenia SCSI posiadają terminator wewnętrzny, który powinien być aktywny tylko wtedy, gdy urządzenie podłączone jest na końcu łańcucha.

Niektóre urządzenia potrafią również samodzielnie wykryć taką sytuację i załączyć terminator. Aby uniknąć niespodzianek, powinieneś przejrzeć dołączoną do urządzeń dokumentację.

Urządzenia SCSI mogą komunikować się poprzez magistralę SCSI bez angażowania płyty głównej czy procesora, np. skaner może wysyłać polecenia bezpośrednio do dysku twardego czy drukarki, co wpływa na zwiększenie sprawności systemu.

Nowsze standardy SCSI

Kilka lat temu dostępny był tylko jeden rodzaj interfejsu SCSI. W tej chwili funkcjonuje przynajmniej pięć jego wersji, z których niektóre są kompatybilne ze sobą, inne nie. Wszystkie wersje dzielą się na trzy kategorie, nazywane SCSI-I, SCSI-II i SCSI-III. SCSI-I to tradycyjny, ośmiobitowy interfejs mogący obsługiwać do siedmiu urządzeń. Można go rozpoznać po prostokątnych, 50-nóżkowych gniazdach. SCSI-I był dość ograniczony pod względem szybkości, dlatego wprowadzone zostały nowsze systemy. Ważne jest, by nie mylić nazw złączy używanych przez urządzenia z nazwami standardów, ponieważ nie zawsze są one zgodne. SCSI-II pozwala na dołączenie do 15 urządzeń poprzez o wiele mniejsze, choć również 50-nóżkowe gniazda w kształcie litery D. SCSI-III również pozwala na dołączenie 15 urządzeń, ale używa szerszych, 68-nóżkowych złączy w kształcie litery D.

Na pewno spotkałeś się z takimi określeniami standardów jak Fast, Wide czy UltraWide SCSI. Są to określenia związane z tym, czy wtyk kabla wewnętrznego ma 50 czy 68 nóżek (i odpowiednio zewnętrznego 40 czy 68 nóżek). Połączenia wewnętrzne i zewnętrzne mają różne szerokości, co może być nieco mylące. W tabeli 36.1 zebrane zostały podstawowe typy interfejsów SCSI wraz z ich własnościami.

Tabela 36.1. Standardy SCSI

Nazwa	Magistrala, do której podłączony jest sterownik	Liczba nóżek złącza wewnętrznego	Liczba nóżek złącza zewnętrznego	Maksymalna szybkość transmisji danych (MB/s)
SCSI (SCSI-I)	EISA/ISA/PCI	50	50	10
FastWide (SCSI-II)	EISA	50 i 68	68	20
Fast-20 Wide (SCSI-II)	EISA	50 i 68	68	20
Fast Wide (SCSI-III)	PCI	68	68	20
Fast-20 Wide (SCSI-III)	PCI	68	68	20
Differential Fast-20 Wide (Differential SCSI)	PCI	68	68	20

Dostępne są również karty sterowników dopuszczające transmisję danych nawet z prędkością 40 MB/s, ale nie są one kompatybilne z większością urządzeń. Differential SCSI to standard, który różni się nieco pod względem elektrycznym w sposobie przesyłania danych

(używana jest metoda różnicowa) – dzięki temu możliwe jest stosowanie dłuższych przewodów i większych prędkości przesyłu. Obecnie dostępnych jest niewiele urządzeń obsługujących ten standard. Również karty sterowników są droższe niż dla pozostałych odmian. W obrębie jednego łańcucha SCSI nie można mieszać urządzeń standardowych i różnicowych.

Nie będziemy wgłębiać się w szczegóły interfejsu SCSI, ponieważ nie są one istotne dla Linuxa. Jeśli chcesz uzyskać więcej informacji, przejrzyj dokumentację dołączaną do sterownika, która zwykle zawiera teoretyczny opis interfejsu.

Obsługiwane urządzenia SCSI

Nie można, niestety, założyć, że jeżeli Linux obsługuje interfejs SCSI, to każde urządzenie SCSI będzie działać poprawnie. Do każdej wersji systemu dołączany jest plik, w którym znajdują się informacje o obsługiwanych urządzeniach. Zajrzyj do niego przed zakupem nowej karty kontrolera czy innego urządzenia.

Niektóre urządzenia rozprowadzane są wraz z poprawkami w kodzie jądra systemu (ang. *patch*), dzięki którym urządzenie może być obsługiwane. Musisz upewnić się wtedy, czy poprawki przeznaczone są do Twojej wersji jądra, a następnie przekompilować jądro z nowymi sterownikami. Czasem poprawki nie są rozprowadzane z urządzeniami, ale można je znaleźć na stronach WWW producenta.

Sterowniki SCSI

Każdemu urządzeniu w systemie Linux musi odpowiadać plik urządzenia – pod tym względem urządzenia SCSI nie są żadnym wyjątkiem. W większości przypadków Linux jest rozprowadzany wraz z pełnym zestawem takich plików, które wymagają tylko skonfigurowania. Jeśli nie jesteś jeszcze za pan brat ze sterownikami, plikami urządzeń oraz głównymi i pobocznymi numerami urządzeń, zajrzyj do rozdziału 33. „Urządzenia”.

Dyski twarde

Dyski SCSI zawsze są urządzeniami blokowymi, a ich numer główny powinien mieć wartość 8. W przeciwieństwie do systemów BSD, Linux nie obsługuje bezpośredniego dostępu do dysków SCSI.

Dla każdego dysku rezerwowanych jest 16 numerów pobocznych. Numer 0 reprezentuje cały dysk, numery od 1 do 4 – partycje podstawowe, od 5 do 15 – partycje rozszerzone.

W Linuxie numery poboczne dysków są przydzielane dynamicznie, rozpoczynając od najniższej wartości identyfikatora SCSI. Zgodnie z przyjętą powszechnie konwencją dyski twarde SCSI nazywają się `/dev/sdX` (na przykład `/dev/sda`, `/dev/sdb`), a ich poszczególne partycje – `/dev/sdXY` (na przykład `/dev/sda1`, `/dev/sda2`).

Linux nie sprawia problemów podczas dzielenia dysków SCSI na partycje, ponieważ potrafi komunikować się bezpośrednio z kontrolerem. Każdy dysk jest widziany tak, jak widzi go kontroler, z numerami bloków zaczynającymi się od 0 do najwyższego numeru (i przy założeniu, że wszystkie bloki są pozbawione błędów). W ten sposób łatwo jest uzyskać informacje o geometrii dysku (dla porównania, DOS wymaga odwzorowania numerów bloków na adresy w formacie głowica/cylinder/sektor, które nie są zbyt wygodne, choć pozwalają na bezpośrednią manipulację zawartością dysku).

Możesz przeznaczyć cały dysk twardy dla systemu Linux – wtedy program instalacyjny zajmie się zakładaniem partycji. Można również podzielić dysk na partycje linuxowym lub DOS-owym programem `fdisk`, dzieląc dostępną przestrzeń pomiędzy różne systemy operacyjne. W systemach obsługujących zarówno urządzenia SCSI jak i IDE, może być konieczna zmiana ustawień BIOS-u tak, by można było uruchomić system z dysku SCSI.

Napędy CD-ROM

Napędy CD-ROM, w których wielkość bloku danych wynosi 512 lub 2048 bajtów, będą w systemie linuxowym działać prawidłowo, nie będą natomiast działać pozostałe typy napędów. Nie powinno to raczej sprawiać problemów, ponieważ inne napędy są rzadkością.

Dyski CD-ROM zapisywane są w kilku różnych formatach i – niestety – nie wszystkie z nich dają się odczytać w systemie linuxowym. Najważniejszym, międzynarodowym standardem jest ISO9660, ale nie wszystkie płyty są z nim zgodne, ponieważ został on wprowadzony dość długo po rozpowszechnieniu się napędów CD-ROM.

Numer główny CD-ROM-u z interfejsem SCSI to 11, natomiast numery poboczne przydzielane są dynamicznie (pierwsze urządzenie otrzyma numer 0, drugie – 1 itd.). Po szczególne napędy nazywają się `/dev/srX`, np. `/dev/sr0`, `/dev/sr1`, lub też `/dev/scdX`, np. `/dev/scd0`, zależnie od wersji systemu.

Aby możliwy był odczyt danych z dysku CD-ROM, należy zamontować zawarty na nim system plików. Można to zrobić ręcznie lub zamieścić odpowiednie polecenie w jednym z plików inicjalizacyjnych. Ma ono postać:

```
mount nazwa_urządzenia punkt_zamontowania, np.:  
mount /dev/sr0 /mnt/cdrom
```

Jeśli Twój CD-ROM nie montuje się poprawnie po wydaniu takiego polecenia, przyczyną może być niewłaściwy format dysku, brak katalogu, w którym system plików ma być zamontowany, lub brak wpisu w pliku `/etc/fstab`. Wpis taki powoduje, że domyślnym typem systemu plików zapisanego na dysku CD-ROM jest ISO9660. W takim przypadku możesz użyć pełniejszej wersji polecenia:

```
mount -t iso9660 /dev/sr0 /mnt/cdrom
```

Aby możliwe było zamontowanie systemu plików zapisanego na płycie CD-ROM, jądro systemu musi obsługiwać system plików ISO9660. Jeśli tak nie jest, trzeba je przekompilować załączając odpowiednie opcje.

Linux próbuje zablokować otwieranie napędu w czasie, gdy dysk jest zamontowany. Jest to zabezpieczenie przed próbą zmiany dysku bez poinformowania o tym systemu operacyjnego. Nie wszystkie napędy pozwalają na to, ale jeśli nie możesz wyjąć dysku, to najprawdopodobniej oznacza to, że najpierw trzeba go odmontować polecienniem `umount punkt_zamontowania` (nawet jeśli nie jest aktualnie używany).

Napędy taśmowe

Linux obsługuje kilka typów napędów taśmowych SCSI. Przed zakupem powinieneś sprawdzić, czy dany model jest obsługiwany, przeglądając dokumentację dołączoną do dystrybucji. Najpopularniejsze modele, takie jak Archive Viper firmy QIC, Exabyte, Wangtek 5150S i napędy DAT działają w systemie Linux bez zarzutu.

Napędy taśmowe mają zwykle numer główny 9, a numery poboczne przydzielane są dynamicznie. Nazwy napędów nieprzewijanych mają przeważnie postać `/dev/nrstX` (np. `/dev/nrst0`), a napędów przewijanych – `/dev/rstX` (np. `/dev/rst1`). W przypadku napędów nieprzewijanych najstarszy bit numeru pobocznego jest ustawiony, więc pierwsze urządzenie tego typu miało numer główny 9, a poboczny 128.

Ogólnie Linux obsługuje zarówno urządzenia o stałej, jak i o zmiennej długości bloku, pod warunkiem, że jego długość jest mniejsza od długości bufora sterownika, która jest ustawiona w większości dystrybucji na 32 kB (ale może zostać zmieniona). Parametry napędu – takie jak długość bloku, buforowanie czy gęstość taśmy – ustawiane są polecienniem `ioctl$` lub za pomocą programu `mt`.

Inne urządzenia

Dostępnych jest również wiele innych urządzeń opartych na interfejsie SCSI, takich jak skanery, drukarki, dyski wymienialne itp. Takie urządzenia obsługiwane są przez ogólny sterownik SCSI. Sterownik ten udostępnia interfejs służący do wysyłania poleceń do dowolnego urządzenia SCSI.

Sterownik ogólny SCSI używa trybu znakowego i ma numer główny 21. Numery poboczne są przydzielane dynamicznie – 0 dla pierwszego urządzenia, 1 dla drugiego itd. Odpowiednie urządzenia nazywają się `/dev/sg0`, `/dev/sg1` itd.

SCSI - rozwiązywanie problemów

Wiele często spotykanych przy pracy z interfejsem SCSI problemów da się łatwo rozwiązać. Najtrudniej jest znaleźć ich źródło. Czasem pomaga czytanie komunikatów diagnostycznych wyświetlanych podczas uruchamiania systemu i inicjalizacji magistrali SCSI.

Poniżej zebraliśmy najczęściej występujące problemy z urządzeniami SCSI wraz z rozwiązaniami, które pomogą w większości przypadków.

Wszystkie identyfikatory przydzielone są jednemu urządzeniu SCSI: Urządzenie ma przypisany ten sam numer, który jest przypisany kontrolerowi (zwykle 7). Należy zmienić ustawienie odpowiedniej zwołki.

Wszystkie dostępne numery LUN są zajęte przez dane urządzenie: Prawdopodobnie posiadasz starą wersję oprogramowania firmowego. W pliku `/usr/src/linux/drivers/scsi/scsi.c` w definicji zmiennej `blacklist` zawarta jest lista „nieposłusznego” urządzeń – możesz spróbować dodać swoje urządzenie do tej listy i przekompilować jądro. Jeśli to nie pomaga, skontaktuj się z producentem.

Komunikat „time out”: Upewnij się, że przerwania, z których korzysta kontroler, są ustawione prawidłowo oraz że nie ma żadnych konfliktów IRQ, DMA czy portów wejścia/wyjścia z innymi urządzeniami.

Komunikaty „sense error”: Komunikaty te spowodowane są zwykle wadliwymi przewodami lub brakiem terminatorów. Upewnij się, że łańcuch SCSI jest prawidłowo zakończony z obu stron. Nie używaj terminatora w środku łańcucha – to również sprawia problemy. Jeśli używasz długich przewodów, możesz spróbować zamontować aktywne terminatory zamiast pasywowych.

Napęd taśmowy nie jest rozpoznawany podczas uruchamiania systemu: Spróbuj uruchomić system z włożoną taśmą.

Jądro obsługujące sieć nie pracuje poprawnie z urządzeniami SCSI: Procedura `autoprobe` używana dla wielu sterowników sieciowych może zakłócać działanie sterownika SCSI. Spróbuj wyłączać po kolej sterowniki, aż znajdziesz ten sprawiający problemy, a następnie skonfiguruj go ponownie.

Urządzenie SCSI zostało wykryte, ale system nie potrafi z niego korzystać: Prawdopodobnie nie masz odpowiedniego pliku urządzenia. Plik taki powinien znaleźć się w katalogu `/dev`, powinien również mieć przypisany odpowiedni główny i poboczny numer urządzenia. Odpowiedni plik można stworzyć za pomocą programu `mkdev`.

Kontroler SCSI nie działa prawidłowo, jeśli używa mapowanych adresów wejścia / wyjścia: Problem ten jest dość powszechny w przypadku sterowników Trantor 128 i Seagate; jest on spowodowany nieprawidłowym buforowaniem portów wejścia / wyjścia. Aby rozwiązać ten problem, należy wyłączyć buforowanie portu używanego przez kontroler (w ustawieniach XCMOS) lub, jeśli nie jest to możliwe, całkowicie wyłączyć buforowanie portów wejścia / wyjścia.

System nie potrafi znaleźć żadnych urządzeń SCSI (komunikat `scsi: 0 hosts`): Procedura `autoprobe` dla sterownika opiera się na danych podawanych przez BIOS i nie potrafi prawidłowo uruchomić kontrolera. Problem ten występuje w przypadku urządzeń: Adaptec 152x, Adaptec151x, Adaptec AIC-6260, Adaptec AIC-6360, Future Domain 1680, Future Domain TMC-950, Future DomainTMC-8xx, Trantor T128, Trantor T128F, Trantor T228F, Seagate ST01, Seagate ST02 i Western Digital 7000. Sprawdź, czy BIOS kontrolera jest załączony i czy nie powoduje konfliktu z BIOS-em jakiegoś innego urządzenia. Jeśli BIOS jest

załączony, znajdź jego „podpis” uruchamiając program systemu DOS `debug`. Przykładowo, użyj polecenia `d=c800:0` programu `debug`, aby sprawdzić, czy kontroler potwierdzi swoją obecność (jeśli sterownik używa adresu `0xc8000`). Jeśli karta nie odpowiada, zmień ustawienia adresu.

System SCSI czasem zawiesza się: Możliwych powodów jest wiele, nie wyłączając uszkodzenia karty sterownika. Sprawdź ją jakimś programem diagnostycznym. Spróbuj użyć innego kabla. Jeśli system zawiesza się tylko wtedy, gdy pracują równocześnie jakieś urządzenia, jest to prawdopodobnie wina oprogramowania firmowego – skontaktuj się wówczas z producentem. Możesz również sprawdzić, czy na dysku twardym nie występują uszkodzone sektory, które mogą być przyczyną najróżniejszych objawów.

Podsumowanie

Interfejs SCSI ma opinię trudnego w obsłudze, ale w rzeczywistości jest to jeden z najwydajniejszych i najbardziej elastycznych systemów. Większość doświadczonych użytkowników systemów UNIX-owych preferuje interfejs SCSI, ponieważ łatwo do niego dostosować jądro systemów UNIX-owych (nie wyłączając Linuxa).

Jak można ułatwić sobie codzienną pracę za pomocą języka powłoki, dowiesz się z rozdziału 14. „Programowanie w języku powłoki”.

Proces konfigurowania systemu tak, by mógł on pracować jako serwer internetowy, opisany jest w rozdziale 47. „Konfigurowanie węzła internetowego”.

O tym, jak używać systemów kontroli wersji, aby łatwiej zarządzać tworzonymi aplikacjami, możesz przeczytać w rozdziale 56. „Systemy kontroli wersji”.

Rozdział 37.

Praca w sieci

Tim Parker

W tym rozdziale:

- υ TCP/IP
- υ Wymagania sprzętowe
- υ Pliki konfiguracyjne
- υ Testowanie konfiguracji i rozwiązywanie problemów

TCP/IP

Linux oferuje użytkownikowi pełną implementację TCP/IP, protokołu używanego po-wszechnie w Internecie i sieciach lokalnych opartych o systemy UNIX-owe. Wszystko, czego potrzebujesz, by stworzyć własną sieć lub podłączyć się do istniejącej, to karta sieciowa, trochę kabla i kilka zmian w plikach konfiguracyjnych.

Proces konfiguracji sieci przebiega zawsze tak samo, bez względu na to, czy łączysz ze sobą dwa komputery stojące w jednym pokoju, czy też podłączasz się do sieci, w której jest już 5000 innych komputerów.

TCP/IP jest protokołem otwartym, tzn. jego specyfikacja techniczna jest ogólnodostępna i każdy może go zaimplementować w swoim programie czy sprzęcie. Dzięki temu stał się on bardzo popularny – dostępne są wersje tego protokołu dla wszystkich chyba platform. Jego największą zaletą jest fakt, że możliwe jest nawiązanie połączenia bez względu na to, jaki system operacyjny używany jest w komputerze.

Ujmując rzecz precyzyjniej, TCP/IP nie jest pojedynczym protokołem, ale zestawem kilkunastu protokołów, z których każdy jest przeznaczony do innych zadań. Wszystkie te protokoły używają jednak wspólnych komponentów do wysyłania i odbierania pakietów danych.

Dwa główne protokoły wchodzące w skład TCP/IP to Transmission Control Protocol i Internet Protocol (od nich właśnie wzięła się nazwa TCP/IP). Protokoły można podzielić ze względu na funkcje, do jakich są przeznaczone.

Transport. Poniższe protokoły używane są do przenoszenia danych pomiędzy systemami.

- υ **TCP** (Transmission Control Protocol). Usługa oparta na połączeniu, tzn. oba porozumiewające się systemy komunikują się przez cały czas.
- υ **UDP** (User Datagram Protocol). Usługa bezpołączeniowa, tzn. nie wymagająca bezpośredniego połączenia pomiędzy maszynami przez cały czas.

Kierowanie przepływem danych (ang. *routing*). Poniższe protokoły stworzono do adresowania danych tak, by jak najszybciej trafiły na miejsce przeznaczenia. Obsługują one również dzielenie większych porcji informacji na małe bloki.

- υ **IP** (Internet Protocol). Obsługuje transmisję danych.
- υ **ICMP** (Internet Control Message Protocol). Obsługuje przesyłanie informacji o statusie IP, takich jak komunikaty o błędach czy o zmianach struktury sieci, mogących mieć wpływ na kierowanie przepływem danych.
- υ **RIP** (Routing Information Protocol). Jeden z kilku protokołów używanych do znajdowania najlepszej drogi przepływu danych.
- υ **OSPF** (Open Shortest Path First). Inny protokół służący do znajdowania najlepszej drogi przepływu danych.

Adresowanie. Poniższe protokoły obsługują adresowanie maszyn w sieci, zarówno poprzez numery IP, jak i nazwy symboliczne.

- υ **ARP** (Address Resolution Protocol). Określa niepowtarzalny numer sieciowy maszyny.
- υ **DNS** (Domain Name System). Znajduje numer maszyny na podstawie jej nazwy.
- υ **RARP** (Reverse Address Resolution Protocol). Określa numer maszyny w sieci, ale odwrotnie niż ARP.
- υ **BOOTP** (Boot Protocol). Pozwala uruchomić urządzenie podłączone do sieci na podstawie informacji dostarczonych przez serwer.

Usługi dla użytkowników. Do tych aplikacji użytkownicy mają bezpośredni dostęp.

- υ **FTP** (File Transfer Protocol). Pozwala na łatwy transport plików z jednego komputera na drugi za pomocą protokołu TCP.
- υ **TFTP** (Trivial FTP). Prosta aplikacja do transmisji plików, oparta na protokole UDP.

TELNET. Pozwala na logowanie się do systemu z konsoli zdalnej.

Obsługa bramek sieciowych. Poniższe usługi pozwalają przekazywać informacje o kierowaniu danymi, informacje o statusie i obsługiwać przepływ danych w sieciach lokalnych.

- υ **EGP** (Exterior Gateway Protocol). Przesyła informacje o kierowaniu danymi do sieci zewnętrznych.
- υ **GGP** (Gateway-to-Gateway Protocol). Przesyła informacje o kierowaniu danymi pomiędzy bramkami internetowymi.
- υ **IGP** (Interior Gateway Protocol). Przesyła informacje o kierowaniu danymi w obrębie sieci lokalnej.

Inne. Ważne usługi nie pasujące do żadnej z powyższych kategorii.

- υ **NFS** (Network File System). Pozwala na montowanie katalogów jednego komputera w innym komputerze i dostęp do nich tak, jakby były katalogami lokalnymi.
- υ **NIS** (Network Information Service). Ułatwia zarządzanie kontami i hasłami użytkowników.
- υ **RPC** (Remote Procedure Call). Pozwala aplikacjom komunikować się ze sobą za pomocą wywołań funkcji.
- υ **SMTP** (Simple Mail Transfer Protocol). Protokół transmisji poczty pomiędzy maszynami.
- υ **SNMP** (Simple Network Management Protocol). Używany do otrzymywania informacji o konfiguracji TCP/IP i oprogramowania. Do prawidłowej pracy wymaga zainstalowania pętli zwrotnej.

Choć od czasu do czasu zdarzają się uaktualnienia standardu TCP/IP poprawiające działanie dostępnych funkcji lub implementujące nowe usługi, kolejne wersje pozostają kompatybilne z poprzednimi.

Wymagania sprzętowe

W zasadzie można skonfigurować TCP/IP nawet bez karty sieciowej czy podłączenia do sieci, używając tak zwanej pętli zwrotnej (ang. *loopback*). Metoda ta pozwala na porozumiewanie się protokołu TCP/IP z innym oprogramowaniem bez opuszczania komputera. Tworzona jest po prostu pętla pomiędzy wyjściem i wejściem programu. Jest to metoda stosowana powszechnie do testowania konfiguracji TCP/IP, również niektóre programy wymagają pętli zwrotnej do poprawnej pracy. Sterownik pętli zwrotnej zawsze ma numer sieciowy 127.0.0.1.

Jeśli jednak chcesz podłączyć się do sieci, karta sieciowa jest niezbędna. W systemie Linux używane są karty Ethernet, zaprojektowane właśnie do obsługi protokołów TCP/IP. Często spotkasz się z określeniem pakiet (ang. *pocket*). Pakiet to pewna ilość danych i instrukcji sterujących, złożona razem przez protokół TCP/IP i przesyłana po przez sieć. Wszystkie dane przed przesłaniem dzielone są na pakiety, a następnie ponownie łączone w miejscu przeznaczenia.

Większość kart Ethernet dostępnych dziś na rynku jest kompatybilna z systemem Linux, ale przed zakupem nowego sprzętu zawsze warto sprawdzić to w dokumentacji. Wszystkie popularniejsze karty sieciowe (włączając w to modele Plug-and-Play przeznaczone dla systemu Windows 95) współpracują z Linuxem, czasem jednak konieczne jest ręczne ustawienie numeru przerwania IRQ i adresu wejścia / wyjścia.

Jeżeli zamierzasz łączyć się z siecią przez telefon, wówczas nie potrzebujesz karty sieciowej, ale modemu kompatybilnego z usługą, której zamierzasz używać. Przykładowo, aby użyć protokołu SLIP (Serial Line Interface Protocol), potrzebujesz modemu pracującego z prędkością co najmniej 14400 kbps (protokół V.32bis).

Pliki konfiguracyjne

Załóżmy, że posiadasz typowy komputer PC i kartę Ethernet i chcesz skonfigurować ją tak, by móc używać sieci opartej na TCP/IP. W większości takich przypadków podana poniżej procedura zadziała. Ze względu jednak na mnogość wersji Linuxa i możliwość potencjalnych konfliktów ze wszystkimi chyba urządzeniami w systemie, a także specyficzne wymagania niektórych kart, poniższy tekst powinien być traktowany tylko jako przewodnik.

Jeśli TCP/IP nie działa prawidłowo po skonfigurowaniu zgodnie z poniższymi wskazówkami, powinieneś uważnie przejrzeć jeszcze raz pliki konfiguracyjne i na podstawie komunikatów o błędach spróbować zlokalizować problem. W razie niepowodzenia pomocy możesz szukać w grupach dyskusyjnych poświęconych Linuxowi i w innych źródłach informacji.

Aby móc skonfigurować TCP/IP, powinieneś najpierw zainstalować oprogramowanie do obsługi sieci. Również jądro systemu musi obsługiwać sieć – jeśli tak nie jest, musisz je przekompilować (obsługa sieci instalowana jest domyślnie, ale jeśli instalacja systemu przebiegała w jakiś niestandardowy sposób, sieć może nie być zainstalowana).

Najpierw zajmiemy się konfigurowaniem karty sieciowej. Następnie przedstawimy modyfikacje tego procesu niezbędne przy konfigurowaniu połączenia modemowego.

Zanim zaczniesz

Zanim przystąpisz do modyfikowania plików systemowych, powinieneś poświęcić parę minut na znalezienie kilku informacji, które będą później potrzebne. Najlepiej zapisz je

gdzieś na kartce, aby do różnych plików konfiguracyjnych nie wprowadzić przypadkiem różnych wartości.

Adres IP

Najpierw musisz poznać swój adres IP (nazywany również adresem internetowym). Na jego podstawie identyfikowany jest każdy komputer podłączony do sieci. 32-bitowy adres używany w sieciach opartych na protokole TCP/IP jest podzielony na cztery osmiobitowe części, rozdzielone kropkami, na przykład 155.25.25.16 czy 147.23.145.2.

Dla wygody, adresy IP dzielą się na dwie części; pierwsza identyfikuje samą sieć, natomiast druga jest numerem komputera w danej sieci.

Jeśli chcesz samodzielnie podłączyć się do Internetu, musisz skontaktować się z Internet Network Information Center, instytucją, która przydzielają numery IP tak, by się nie powtarzały (opierając się na wielkości sieci). Jeśli nie planujesz podłączenia do Internetu, możesz wybrać sobie dowolny adres IP.

Dla zapewnienia elastyczności adresy IP przydzielane są w zależności od wielkości sieci. Sieci dzielą się na trzy kategorie: A (jeden bajt jest adresem sieci, pozostałe trzy – adresem maszyny w podsieci, co oznacza, że podsieć może zawierać ponad 16 milionów komputerów), B (dwa bajty adresu sieciowego i dwa bajty adresu maszyny w podsieci – czyli ponad 65000 komputerów) i C (trzy bajty adresu sieciowego i jeden dla identyfikacji w podsieci, co oznacza, że w sieci może pracować do 254 komputerów, ponieważ numery 0 i 255 są zarezerwowane). Większość sieci to sieci klasy B i C.

Istnieją ograniczenia co do wartości pierwszego bajtu adresu: w sieci klasy A musi to być liczba z zakresu od 0 do 127, w sieci klasy B – od 128 do 191, a w sieci klasy C – od 192 do 223. Ograniczenie to spowodowane jest faktem, że starsze bity pierwszego bajtu adresu zawierają informacje o klasie sieci. Nie można również używać numerów 0 i 255, zarezerbowanych dla specjalnych celów.

Wiadomości przesyłane za pomocą protokołu TCP/IP zawierają w nagłówku adres IP zarówno komputera, który wysłał dane, jak i komputera, który ma je odebrać. Jeśli zamierzasz podłączyć się do istniejącej sieci, powinieneś dowiedzieć się, jakiego adresu możesz użyć. Jeśli budujesz własną sieć, którą zamierzasz podłączyć do Internetu, powinieneś skontaktować się z Network Information Center. Jeżeli jednak z Internetem i innymi sieciami zamierzasz łączyć się co najwyżej przez telefon, możesz wybrać sobie dowolny numer IP.

Jeśli konfigurujesz tylko sterownik pętli zwrotnej, w ogóle nie potrzebujesz numeru IP. Domyślną wartością jest wtedy 127.0.0.1.

Maska podsieci

Następnym numerem, który będzie potrzebny, jest maska podsieci. Znając numer IP łatwo go znaleźć samemu. Zamiast adresu sieci należy wpisać 255, a pozostałe bajty ustawić na 0. Przykładowo, jeśli chcesz podłączyć się do sieci klasy C, maska podsieci bę-

dzie miała postać 255.255.255.0. Dla sieci klasy B jest to 255.255.0.0, a dla klasy A – 255.0.0.0.

Jeśli konfigurujesz sterownik pętli zwrotnej, prawidłową maską jest 255.0.0.0.

Adres sieci

Adres sieci to – w zależności od klasy sieci – pierwszy bajt, dwa lub trzy pierwsze bajty adresu IP odpowiednio dla klas A, B i C. Na przykład jeśli łączysz się z siecią klasy B, a Twój numer IP to 147.120.43.31, to adresem sieci jest 147.120.0.0. Mówiąc ściśle, jest to wynik wykonania bitowej operacji AND na adresie IP i masce podsieci.

Aby otrzymać adres sieci z adresu IP, należy zastąpić w nim numer komputera zerami. Jeśli podłączasz się do sieci klasy C, a Twój adres IP to 201.12.5.23, adresem sieci będzie 201.12.5.0.

Przy konfigurowaniu pętli zwrotnej adres ten jest niepotrzebny.

Adres rozgłoszenia

Adres rozgłoszenia (ang. *broadcast*) jest używany wtedy, gdy pakiet danych ma trafić do wszystkich urządzeń podłączonych do sieci. Jest to adres sieci, w którym zera zamienione są na 255 (na przykład jeśli Twoim adresem IP jest 147.120.42.31, adres sieci jest równy 147.120.0.0, a adres rozgłoszenia – 147.120.255.255).

Przy konfigurowaniu pętli zwrotnej adres ten jest niepotrzebny.

Adres bramki sieciowej

Adres bramki to numer IP komputera, który jest skonfigurowany jako bramka łącząca sieć lokalną z inną siecią (np. Internet). Jeśli konfigurujesz własną sieć nie podłączoną do innych sieci, adres ten nie jest potrzebny.

Zwykle bramka ma taki sam numer IP jak Twój komputer, z tym że ostatni bajt ma wartość 1, np. jeśli Twój numer IP to 147.120.43.31, to numerem bramki jest prawdopodobnie 147.120.43.1. Taka konwencja przyjęła się już od pierwszych wersji protokołu TCP/IP.

Przy konfigurowaniu pętli zwrotnej adres ten jest niepotrzebny.

Adres serwera nazw

W wielu większych sieciach funkcjonuje komputer przeznaczony do tłumaczenia nazw symbolicznych na numery IP i odwrotnie. Zamiana taką dokonywana jest za pomocą systemu DNS (Domain Name System). Jeśli do Twojej sieci podłączone jest takie urządzenie, jego adresem jest adres serwera nazw.

dzenie, właśnie jego adres jest adresem serwera nazw. Jeśli to Twój komputer ma działać jako serwer nazw (co wymaga dodatkowych kroków konfiguracyjnych, nie opisanych tutaj), powinieneś podać numer sterownika pętli zwrotnej, czyli 127.0.0.1.

Przy konfigurowaniu pętli zwrotnej adres ten jest niepotrzebny, ponieważ komputer łączy się tylko sam ze sobą.

Konfigurowanie interfejsu pozornego

Interfejs pozorny jest sposobem na przydzielenie komputerowi numeru IP w przypadku, gdy korzysta on tylko z interfejsów SLIP i PPP. Interfejs pozorny rozwiązuje problem pojawiający się w systemach nie podłączonych na stałe do sieci, w których jedynym prawidłowym adresem IP, pod który można wysłać dane, jest 127.0.0.1. Chociaż za pomocą protokołów SLIP i PPP możesz połączyć się ze światem, to kiedy interfejs nie jest aktywny, nie posiadasz wewnętrznego adresu IP, którego mogłyby używać aplikacje.

Problem ten jest szczególnie dokuczliwy, gdy trzeba używać aplikacji, które do swojego działania wymagają podania prawidłowego adresu IP. Przykładowo niektóre edytory tekstów czy narzędzia zarządzające pulpitem wymagają podania takiego adresu. Interfejs pozorny pozwala na ustawienie adresu IP dla Twojego komputera, który nie jest wykorzystywany do niczego oprócz oszukiwania aplikacji.

Konfigurowanie takiego interfejsu jest bardzo proste. Jeśli w pliku `/etc/hosts` jest wpisany adres IP komputera, wystarczy wydać polecenia:

```
ifconfig dummy nazwa_komputera  
route add nazwa_komputera
```

Spowodują one utworzenie połączenia z Twoim własnym adresem IP. Jeśli wcześniej nie wpisałeś adresu IP do pliku `/etc/hosts`, musisz to najpierw zrobić.

Pliki konfiguracyjne - szczegóły

Konfiguracja systemu TCP/IP dla Linuxa nie jest trudna i polega w zasadzie na wpisaniu za pomocą dowolnego edytora tekstów odpowiednich adresów IP do kilku plików konfiguracyjnych. Przed wprowadzeniem zmian dobrze zrobić kopię zapasową modyfikowanych plików na wypadek problemów.

Pliki konfiguracyjne systemu TCP/IP są bardzo podobne we wszystkich systemach UNIX-owych, więc jeśli kiedykolwiek konfigurowałeś TCP/IP w UNIX-ie, nie znajdziesz w tym rozdziale nic nowego. Jeśli jest to Twój pierwszy raz – po prostu postępuj zgodnie z podanymi niżej wskazówkami.

Pliki rc

Pliki `rc` (ang. *run command*) są przetwarzane podczas uruchamiania systemu. Proces ten kontrolowany jest przez program `init`. Zwykle za ich pośrednictwem uruchamiane są procesy obsługujące pocztę, drukarki itp. Tu również następuje inicjalizacja połączeń TCP/IP. W większości systemów linuxowych pliki `rc` znajdują się w katalogu `/etc/rc.d`.

Informacje dotyczące konfiguracji TCP/IP zapisane są w dwóch plikach: `rc.inet1` (w nim ustawiane są parametry sieci) i `rc.inet2` (tu uruchamiane są programy rezydentne obsługujące TCP/IP). W niektórych systemach pliki te połączone są w jeden większy plik o nazwie `rc.inet` lub `rc.net`.

Zanim zaczniesz modyfikować dane zapisane w tych plikach, najpierw upewnij się, że są one faktycznie wczytywane przez program `init`. Jest tak, jeśli w pliku `/etc/inittab` lub `/etc/rc.d/rc.M` znajdują się polecenia nakazujące odczytanie tych plików.

Niektóre wersje Linuxa do uruchamiania programów obsługujących TCP/IP używają tylko pliku `/etc/inittab`, w innych zaś w pliku tym znajduje się polecenie wywołujące plik `/etc/rc.d/rc.M` (jeśli uruchamiany jest tryb wielodostępny), w którym zapisane są odpowiednie informacje. W takim przypadku protokół TCP/IP nie jest uruchamiany w trybie administracyjnym.

Jeśli odpowiednie wpisy w którymś z powyższych plików za pomocą symbolu `#` na początku wiersza zaznaczone są jako komentarz – powinieneś usunąć ten symbol. Przykładowe wpisy w pliku `rc.M` mogą wyglądać tak:

```
/binhostname `cat /etc/HOSTNAME | cut -f1 -d .`  
/bin/sh /etc/rc.d/rc.inet1  
/bin/sh /etc/rc.d/rc.inet2
```

Jeśli w pliku `/etc/inittab` nie znajdziesz odniesienia do pliku `rc.inet` lub odniesienia do pliku, w którym znajduje się polecenie przetwarzania tego pliku (na przykład `/etc/rc.d/rc.M` w dystrybucji Slackware), musisz sam dopisać odpowiednie polecenia. Pamiętaj, że modyfikujesz jedne z najważniejszych plików w systemie, więc wykonaj ich kopię zapasową, najlepiej na dyskietce startowej.

Zwykle w plikach inicjalizacyjnych zawarta jest spora liczba komentarzy, znacznie ułatwiających konfigurację, na przykład takich:

```
#IPADDR="127.0.0.1"  
#NETMASK=""  
#NETWORK="127.0.0"  
#BROADCAST=""  
#GATEWAY=""
```

W takim przypadku wystarczy usunąć znaki `#` i wpisać odpowiednie numery. Jeśli któryś z wpisów nie jest Ci potrzebny (na przykład numer bramki internetowej), po prostu pozostaw w odpowiednim wierszu symbol komentarza.

W pliku `rc.inet1` występują też odwołania do programów `ifconfig` i `route`. `ifconfig` jest programem służącym do konfigurowania interfejsów sieciowych, natomiast `ro-`

`ute` jest używany do określania drogi przesyłania danych. Odwołania do tych programów konfigurujące sterownik pętli zwrotnej powinny mieć następującą postać:

```
/sbin/ifconfig lo 127.0.0.1  
/sbin/route add -net 127.0.0.0
```

Żaden z tych wierszy nie może być zaznaczony jako komentarz. Sterownik pętli zwrotnej musi być skonfigurowany, aby system TCP/IP mógł działać prawidłowo.

Poniżej tych wpisów prawdopodobnie znajduje się dość spora liczba wierszy zaznaczonych jako komentarz, służących do konfigurowania interfejsów sieciowych. Na początek spróbuj usunąć symbol komentarza z wiersza wyglądającego mniej więcej tak:

```
/etc/ifconfig eth0 ${IPADDR} netmask ${NETMASK} broadcast ${BROADCAST}
```

Jeśli spowoduje to problemy podczas uruchamiania systemu, spróbuj zamiast niego użyć wpisu

```
/etc/ifconfig eth0 ${IPADDR} netmask ${NETMASK}
```

`eth0` to etykieta pierwszej karty Ethernet w systemie.

Na koniec, jeśli w sieci działa bramka internetowa, znajdź odpowiednie wiersze i wpisz tam jej adres. Zanim to jednak zrobisz, warto upewnić się, że wszystko inne jest skonfigurowane poprawnie – o wiele łatwiej jest znaleźć błąd w konfiguracji, jeśli bramka sieciowa nie jest używana.

W pliku `rc.inet2` znajdują się polecenia uruchamiające programy obsługi TCP/IP. W większości przypadków nie powinieneś go modyfikować. Możesz sprawdzić, czy wiersz wywołujący program `inetd` nie jest zaznaczony jako komentarz. Odpowiedni fragment będzie prawdopodobnie wyglądał tak:

```
if [-f ${NET}/inetd  
then  
        echo -n " inetd"  
        ${NET}/inetd  
else  
        echo" no INETD found. INET cancelled."  
        exit 1  
fi
```

Jeśli przeczytałeś rozdział 14. „Programowanie w języku powłoki” albo wiesz coś o programowaniu w innych językach, łatwo zrozumiesz, o co chodzi w powyższym fragmencie. Powoduje on sprawdzenie, czy w systemie znajduje się program `inetd`, jeśli tak – program ten zostaje uruchomiony. Jeśli odpowiedniego pliku nie ma, generowany jest komunikat o błędzie i dalsza część pliku nie jest przetwarzana.

Inne programy, takie jak `named` czy `routed`, nie są wymagane do poprawnej pracy TCP/IP. Jeśli nie jesteś pewny, czy chcesz korzystać z oferowanych przez nie możliwości, możesz ich nie załączać.

Możesz jeszcze chcieć uruchomić program `syslogd`, który potrafi zapisać do pliku wszystkie komunikaty generowane przez programy rezydentne (co ułatwia znajdowanie

ewentualnych błędów). Ścieżka dostępu do pliku, w którym rejestrowane będą komunikaty, określana jest w pliku `/etc/syslog.conf`.

Omówiliśmy już wszystkie modyfikacje, które należy wprowadzić do plików `rc`. Po zainstalowaniu i przetestowaniu systemu TCP/IP możesz kolejno załączać pozostałe programy, takie jak `routed`, `named` itd.

/etc/hosts

W pliku `/etc/hosts` znajduje się lista adresów IP i nazw symbolicznych, które się do nich odnoszą. Jest to dobre miejsce na wpisanie danych o najczęściej odwiedzanych serwerach – dzięki temu zamiast numerami IP można będzie posługiwać się ich nazwami. W małych sieciach można tu wpisać dane o wszystkich maszynach – nie trzeba będzie wówczas używać programu `named`.

W pliku `/etc/hosts` musi znajdować się wpis odpowiadający komputerowi lokalnemu – `localhost` (czasem również ma on nazwę `loopback`; odpowiada mu adres 127.0.0.1), prawdopodobnie jest tam również wpis z nazwą Twojego komputera (jeśli nadaleś mu nazwę podczas instalacji oprogramowania). Nie trudź się wpisywaniem zbyt wielu nazw dopóki nie jesteś pewny, że TCP/IP pracuje prawidłowo. Oto przykładowa zawartość pliku `/etc/hosts`:

```
127.0.0.1      localhost
147.12.2.42    merlin.tpci merlin
```

Jak widać, format wpisów jest prosty: najpierw numer IP, następnie nazwa, oddzielona od numeru tabulatorem. Można podać więcej niż jedną nazwę komputera. W powyższym przykładzie komputer o nazwie `merlin` jest częścią sieci o nazwie `tpci`, przewidziano więc również możliwość zaadresowania go jako `merlin.tpci`.

Bardziej rozbudowana wersja tego pliku może mieć postać:

```
127.0.0.1      localhost
147.12.2.42    merlin.tpci merlin
147.12.2.43    wizard.tpci wizard
147.12.2.44    arthur.tpci arthur komp_michala
```

W powyższym przykładzie zdefiniowanych jest kilka nazw komputerów działających w obrębie jednej sieci (posiadających takie same numery sieci). Jeden z komputerów dostępny jest pod trzema różnymi nazwami.

Jeśli używasz tylko sterownika pętli zwrotnej, jedynym wpisem w pliku `/etc/hosts` powinien być numer 127.0.0.1 z nazwą `localhost` i ewentualne nazwy Twojego komputera.

/etc/networks

W pliku `/etc/networks` znajduje się adres sieci lokalnej i adresy sieci, z którymi często się łączysz. Plik ten jest używany przez program `route`, uruchamiany w pliku

`rc.inet1`. Dzięki niemu zamiast używać adresu sieci można podać jej nazwę, np. zamiast wpisywać 149.23.24 wpisać po prostu `siec_wojtko`.

W pliku `/etc/networks` powinny znaleźć się wpisy dla każdej sieci, dla której będziesz używał polecenia `route`. W przeciwnym przypadku generowane będą komunikaty o błędach, a sieć nie będzie działała prawidłowo.

Poniżej przedstawiamy przykładową zawartość pliku `/etc/networks`. Pamiętaj, że adresy sieci zawierają tylko jeden, dwa lub trzy pierwsze bajty (zależnie od wielkości sieci), pozostałe bajty są równe zero. W pliku `/etc/networks` znajdują się musi przynajmniej wpis `loopback` i `localnet`.

```
loopback      127.0.0.0
localnet      147.13.2.0
siec_wojtko   197.32.1.0
big_net       12.0.0.0
```

/etc/host.conf

Dane zapisane w pliku `/etc/host.conf` decydują o sposobie, w jaki nazwy symboliczne tłumaczone są na numery IP. Zwykle zawiera on dwa wiersze:

```
order hosts, bind
multi on
```

Mówią one, że przy tłumaczeniu nazw najpierw należy sprawdzić plik `/etc/hosts`, a następnie szukać pomocy w serwerze nazw (jeśli taki istnieje w sieci). Wpis `multi` pozwala na użycie więcej niż jednego numeru IP dla danego komputera (ma to zastosowanie w przypadku używania bramki internetowej lub w przypadku komputerów podłączonych jednocześnie do kilku sieci).

Jeśli zawartość pliku `/etc/host.conf` jest taka, jak pliku przykładowego, nie trzeba go modyfikować.

resolv.conf

Plik `resolv.conf` jest używany przez program tłumaczący nazwy. Zawiera on adres serwera nazw i nazwę Twojej domeny (jeśli taką posiadasz – a musisz posiadać, jeśli jesteś podłączony do Internetu).

Plik `resolv.conf` dla systemu `merlin.tpcicom` (dla którego nazwą domeny jest `tpcicom`) może na przykład zawierać następujący wiersz:

```
domain tpcicom
```

Jeżeli w sieci działa serwer nazw, w pliku tym powinien również znaleźć się wpis zawierający jego numer IP:

```
domain tpci.com
nameserver 182.23.12.4
```

Jeśli w sieci działa kilka serwerów nazw (co często zdarza się w większych sieciach), każdy z nich powinien być wpisany w osobnym wierszu.

Jeżeli Twój system nie posiada nazwy domenowej, nie musisz przejmować się tym plikiem.

/etc/protocols

W systemach UNIX-owych plik `/etc/protocols` używany jest do identyfikacji i znajdowania numerów wszystkich dostępnych w systemie protokołów transmisji danych (każdemu z protokołów przypisany jest inny numer, ale nie jest to w tej chwili istotne). Ten plik zwykle nie wymaga modyfikacji; jest on aktualizowany automatycznie podczas instalacji oprogramowania. Poniżej zamieszczamy przykładową zawartość takiego pliku:

```
# Internet protocols (IP)
ip      0      IP
icmp   1      ICMP
ggp    3      GGP
tcp    6      TCP
egp    8      EGP
pup   12      PUP
udp   17      UDP
hello  63      HELLO
```

Jeśli w Twoim systemie plik ten wygląda nieco inaczej, nie jest to powód do zmartwień. Nie trzeba go modyfikować, ale warto wiedzieć, jakie informacje zawiera.

/etc/services

W pliku `/etc/services` przechowywane są dane o wszystkich dostępnych usługach sieciowych. Jego również nie należy edytować.

Plik ten składa się z wpisów o formacie `nazwa_usługi numer_portu/protokół [inne_nazwy ...]`, na przykład:

```
# network services
echo      7/tcp
echo      7/udp
discard   9/tcp     sink     null
discard   9/udp     sink     null
ftp       21/tcp
telnet    23/tcp
smtp      25/tcp     mail     mailx
tftp      69/udp
# specific services
login     513/tcp
who       513/udp     whod
```

Choć nie trzeba modyfikować zawartości tego pliku, powinieneś orientować się, jakiego typu informacje on zawiera – pozwoli Ci to nieco lepiej zrozumieć zasadę działania systemu TCP/IP.

/etc/hostname lub /etc/HOSTNAME

W pliku `/etc/hostname` lub `/etc/HOSTNAME` (zależnie od systemu) przechowywana jest nazwa systemu, na przykład:

```
merlin.tpci
```

I to wszystko. Nazwa systemu używana jest przez większość protokołów i wiele aplikacji, więc ważne jest jej prawidłowe ustawienie. Można ją zmienić modyfikując zawartość tego pliku i ponownie uruchamiając komputer, ale w większości systemów dostępny jest specjalny program służący do tego celu.

W systemie Linux dostępny jest program `hostname`, który wyświetla aktualną nazwę systemu, oraz `uname`, który uruchomiony z opcją `-n` wyświetla nazwę węzła:

```
$ hostname  
merlin.tpci.com  
$ uname -n  
merlin
```

W niektórych wersjach Linuksa polecenie `hostname` wyświetla tylko nazwę komputera, nie dodającą nazwy domeny. Omówiliśmy już wszystkie pliki, które są istotne z punktu widzenia konfiguracji TCP/IP. Możesz teraz zresetować komputer i sprawdzić, czy wszystko działa jak należy.

Testowanie konfiguracji i rozwiązywanie problemów

Przyjrzyj się dokładnie informacjom wyświetlonym przy uruchamianiu systemu. Jeśliauważysz komunikaty o błędach, pomogą Ci one w zlokalizowaniu problemu. Jeśli nie wystąpią żadne problemy, wyświetcone zostaną komunikaty o ładowaniu poszczególnych programów obsługi TCP/IP.

Polecenie netstat

Za pomocą programu `netstat` łatwo sprawdzić, czy konfiguracja TCP/IP jest poprawna. Program ten wyświetla informacje o statusie połączeń sieciowych. Jest to najczęściej używanym programem do diagnozowania problemów z TCP/IP.

Oprócz wymienionych niżej opcji tego programu, dostępnych jest jeszcze wiele innych – jeśli chcesz dowiedzieć się o nich czegoś więcej, zajrzyj na strony `man` lub do jakiejś dobrej książki na temat pracy w systemach UNIX-owych.

Wyświetlanie danych o połączeniach

Polecenie `netstat` bez żadnych parametrów wyświetla informacje o wszystkich aktywnych połączeniach (bez względu na to, czy aktualnie odbywa się przez nie transmisja danych). Aby wyświetlić dane również o połączeniach biernych, użyj opcji `-a`.

Dane wyjściowe programu `netstat` wyświetlane są w kolumnach o nazwach: `Proto` (typ protokołu), `Recv-Q` i `Send-Q` (ilość danych odebranych i wysłanych), `Local Address` (adres lokalny), `Foreign Address` (adres zdalny) oraz `state` – aktualny stan połączenia. Oto fragment takich danych:

```
merlin> netstat -a
Active Internet connections (including servers)
Proto Recv-Q Send-Q Local Address      Foreign Address      (state)
ip    0      0      *.*                  *.*                  ESTABLISHED
tcp   0      2124   tpcl.login        oscar.1034          ESTABLISHED
tcp   0      0      tpcl.1034         prudie.login       ESTABLISHED
tcp   11212  0      tpcl.1035         trejis.1036        ESTABLISHED
tcp   0      0      tpcl.1021         reboc.1024        TIME_WAIT
tcp   0      0      *.1028            *.*                LISTEN
tcp   0      0      *.*                *.*                CLOSED
udp   0      0      localhost.1036    localhost.syslog
udp   0      0      *.1034            *.*                *
udp   0      0      *.*                *.*                *
udp   0      0      *.*                *.*                *
```

W powyższym przykładzie widać trzy aktywne połączenia (ich status jest oznaczony jako `ESTABL.`), przez jedno z nich wysyłane są dane (wartość większa od zera w kolumnie `Send-Q`). Gwiazdka oznacza, że dla danego adresu nie odnaleziono jeszcze punktu końcowego.

Statystyki interfejsu sieciowego

Za pomocą polecenia `netstat -i` można sprawdzić, jak ogólnie zachowuje się interfejs sieciowy (i na przykład sprawdzić działanie karty sieciowej).

Polecenie to wyświetla nazwę interfejsu, maksymalną liczbę znaków, jaką może zawierać pakiet (MTU), liczbę pakietów odebranych bez błędów (RX-OK), liczbę pakietów odebranych z błędami (RX-ERR), liczbę pakietów odrzuconych (RX-DRP) i liczbę pakietów, które nie mogły być odebrane (RX-OVR). Takie same dane są dostępne dla pakietów wysyłanych. Poniżej przedstawiamy przykładowy wynik działania tego polecenia:

```
merlin> netstat -i
Kernel Interface table
Iface MTU     Met      RX-OK      RX-ERR      RX-DRP      RX-OVR      TX-OK      TX-ERR      TX-DRP
Σ      TX-OVR Flags
lo     2000      0       0          0          0          0         12          0          0      0
```

Σ	BLRU										
eth0	1500	0	218	0	0	0	144	0	0	0	0
Σ	BRU										

Tablica kierowania przepływem danych (routing table)

Tablice kierowania przepływem danych są aktualniane w sposób ciągły w oparciu o dane dotyczące połączeń z innymi komputerami. Jeśli chcesz uzyskać informacje o nich (o ile są dostępne w Twoim systemie), powinieneś wydać polecenie `netstat -r`.

W poszczególnych kolumnach wyświetlane zostaną dane o maszynie docelowej, adres bramki, która zostanie użyta do przesyłania danych, znacznik, czy dana droga jest aktywna (U) i czy prowadzi do bramki (G) czy komputera (H), licznik odniesień (Refs) pokazujący, ile aktywnych połączeń może korzystać z danej drogi równocześnie, liczbę pakietów przesyłanych do tej pory daną drogą (Use) i nazwę używanego interfejsu.

```
merlin> netstat -r
Kernel routing table
Destination     Gateway      Genmask      Flags   Metric  Ref    Use   Iface
localnet        *           255.255.0.0  U        0       0    262   eth0
loopback        *           255.0.0.0   U        0       0     12    lo
default         *           0.0.0.0     U        0       0     0    eth0
```

ping

Program `ping` (ang. *Packet Internet Groper*) używany jest do wysyłania i odbierania pakietów danych po to, by upewnić się, że połączenie jest aktywne. Jeśli komputer docelowy odbierze pakiet, w którym znajduje się żądanie odpowiedzi, natychmiast wysyła odpowiedź do Twojego komputera.

System, w którym uruchomiony jest program `ping` nie przestaje wysyłać pakietów aż do przerwania działania programu (na przykład kombinacją klawiszy *Control+c*). Po przerwaniu jego działania wyświetlane jest krótkie podsumowanie. Oto przykładowy wynik działania polecenia `ping`:

```
prudie> ping merlin
PING merlin: 64 data bytes
64 bytes from 142.12.120.12: icmp_seq=0, time=20. ms
64 bytes from 142.12.120.12: icmp_seq=1, time=10. ms
64 bytes from 142.12.120.12: icmp_seq=2, time=10. ms
64 bytes from 142.12.120.12: icmp_seq=3, time=20. ms
64 bytes from 142.12.120.12: icmp_seq=4, time=10. ms
64 bytes from 142.12.120.12: icmp_seq=5, time=10. ms
64 bytes from 142.12.120.12: icmp_seq=6, time=10. ms
--- merlin PING Statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip - min/avg/max = 10/12/20
```

Jeśli ping nie był w stanie połączyć się z komputerem docelowym, wyświetlane są komunikaty o błędach. Można również wywołać polecenie ping localhost, które pokaże, czy sterownik pętli zwrotnej jest skonfigurowany prawidłowo.

Program ping jest przydatny, ponieważ podaje cztery ważne informacje: czy oprogramowanie TCP/IP działa prawidłowo, czy urządzenia sieciowe może być prawidłowo adresowane, czy można połączyć się z komputerem zdalnym (testując adresowanie i kierowanie danych) oraz weryfikuje działanie odpowiedniego oprogramowania w systemie zdalnym.

Podsumowanie

W tym rozdziale dowiedziałeś się, jak zainstalować, skonfigurować i przetestować połączenie Ethernet z Twoim komputerem. Następny rozdział omawia protokoły SLIP i PPP, używane w wielu systemach do łączenia się z Internetem. Ostatnim protokołem używanym w systemie Linux jest UUCP (UNIX-to-UNIX Copy), który jest omówiony w rozdziale „UUCP”.

Jeśli zamierzasz zbudować sieć łączącą kilka komputerów, na pewno nie będziesz miał z tym większych kłopotów. Jeśli posiadasz dwa komputery, wygodne i wydajne jest połączenie systemu Linux z komputerem pracującym pod kontrolą systemu DOS czy Windows (pod warunkiem, że zainstalujesz w nim oprogramowanie TCP/IP).

Możesz teraz przejść do następnego rozdziału, przedstawiającego interfejsy SLIP i PPP, lub też przejść do innych tematów.

Jeśli chcesz dowiedzieć się, jak powinna być skonfigurowana sieć, abyś mógł czuć się w niej bezpiecznie, przejdź do rozdziału 42. „Bezpieczeństwo w sieci”.

Jak skonfigurować NFS, NIS i YP, dowiesz się z rozdziału 43. „NFS” i 44., „NIS oraz YP”.

O konfigurowaniu węzła internetowego pracującego pod kontrolą systemu Linux możesz przeczytać w rozdziale 47. „Konfigurowanie węzła internetowego”.

Rozdział 38.

SLIP i PPP

Tim Parker

W tym rozdziale:

- υ Konfiguracja protokołu SLIP
- υ Konfiguracja protokołu PPP
- υ Używanie systemu DNS z protokołami SLIP i PPP

Większość systemów linuxowych i UNIX-owych połączonych jest z Internetem za pomocą protokołów SLIP (ang. *Serial Line Internet Protocol*) lub PPP (ang. *Point-to-Point Protocol*). Oba te protokoły służą do obsługi połączenia modemowego (przez modem synchroniczny, asynchroniczny lub ISDN). Linux obsługuje również ulepszoną wersję protokołu SLIP – CSLIP (ang. *Compressed SLIP*).

Konfigurację tych protokołów można przeprowadzić podczas instalowania TCP/IP, ale można również odłożyć ją do czasu, gdy będziesz chciał połączyć się z Internetem (ponieważ większość usługodawców internetowych wymaga łączenia się za pomocą tych protokołów).

Sam proces konfigurowania protokołów SLIP i PPP nie jest skomplikowany. Jeśli będziesz trzymał się podanych niżej wskazówek, całość zajmie Ci tylko kilka minut.

Konfigurowanie interfejsu pozornego

Interfejs pozorny jest sposobem na przydzielenie komputerowi numeru IP w przypadku, gdy korzysta on tylko z interfejsów SLIP i PPP. Interfejs pozorny rozwiązuje problem pojawiający się w systemach nie podłączonych na stałe do sieci, w których jedynym prawidłowym adresem IP, pod który można wysłać dane, jest 127.0.0.1. Problem ten jest szczególnie widoczny, gdy musisz używać aplikacji wymagających podania prawidłowego adresu IP, którego zwykle nie posiadasz jeśli nie jesteś podłączony do sieci.

Konfigurowanie takiego interfejsu jest bardzo proste. Jeśli w pliku `/etc/hosts` jest wpisany adres IP komputera, wystarczy wydać polecenie tworzące odpowiedni interfejs i powiadomić system, że może za jego pośrednictwem przesyłać dane:

```
ifconfig dummy nazwa_komputera  
route add nazwa_komputera
```

Powyższe polecenia spowodują utworzenie połączenia z Twoim własnym adresem IP. Jeśli wcześniej nie wpisałeś adresu IP do pliku `/etc/hosts`, musisz to najpierw zrobić, dopisując do niego wiersz:

```
127.0.0.1      loopback
```

W niektórych wersjach Linuxa zamiast programów `ifconfig` i `route` używa się odpowiednich programów obsługiwanych za pomocą menu. W większości przypadków sterownik pętli zwrotnej jest konfigurowany automatycznie przy instalacji Linuxa. Sprawdź, czy w pliku `/etc/hosts` znajduje się wiersz zawierający adres 127.0.0.1. Jeśli nie, dodaj go i uruchom ponownie komputer.

Konfiguracja protokołu SLIP

Protokół SLIP używany jest przy kontaktowaniu się z usługodawcą internetowym i do połączeń pomiędzy dwoma komputerami. Nawiązywanie połączenia przez modem odbywa się w zwykły sposób, a następnie SLIP przejmuje kontrolę nad połączeniem. Sterownik SLIP powinien być częścią jądra systemu – jeśli tak nie jest, należy je przekompilować. Większość sterowników SLIP obsługuje również protokół CSLIP, umożliwiający kompresję nagłówków, co prowadzi do uzyskaniawiększej szybkości przesyłania danych (nie każdy usługodawca internetowy pozwala jednak na połączenia CSLIP – powinieneś to sprawdzić, zanim zaczniesz konfigurację).

W większości systemów linuxowych do obsługi protokołu SLIP musi być przeznaczony jeden z portów szeregowych, co oznacza, że nie będzie go można używać do innych celów. Kernel używa programu SLIPDISC (ang. *SLIP Discipline*) do kontrolowania portu szeregowego i blokowania dostępu do niego aplikacjom próbującym wykorzystać go w inny sposób nawet wtedy, gdy protokół SLIP nie jest wykorzystywany.

Protokół SLIP

Połączenia SLIP obsługiwane są przez dwa programy: `dip` i `slattach`. Oba mogą zostać użyte do zainicjalizowania połączenia. Nie można natomiast do tego celu użyć standardowych programów komunikacyjnych.

Przeznaczenie obu programów jest nieco różne. `slattach`, który po prostu podłącza się do portu szeregowego, używany jest gdy istnieje stałe połączenie z serwerem SLIP (wtedy niepotrzebna jest obsługa modemu ani potwierdzanie transmisji). Program `dip` obsługuje inicjalizowanie połączenia, logowanie i potwierdzanie transmisji. Powinieneś go używać, jeśli łączysz się przez modem. Można go również użyć, by skonfigurować system jako serwer SLIP, pozwalając innym łączyć się z Twoim komputerem.

SLIP jest protokołem bardzo prostym, ponieważ w transmisji danych biorą udział tylko dwa urządzenia: Twój komputer i serwer. Po nawiązaniu połączenia, SLIP przesyła numer IP, który będzie później używany. Niektóre systemy wysyłają ten sam numer za każdym razem (adresowanie statyczne), inne natomiast mają różne numery przy różnych połączeniach (adresowanie dynamiczne). Konfigurowanie obu typów połączeń wymaga nieco innych zabiegów.

Najłatwiejszym sposobem na to, aby przeznaczyć port szeregowy do połączeń SLIP, jest użycie programu `slattach`. Jest on dostępny w przeważającej większości systemów linuxowych. Jego argumentem powinna być nazwa portu szeregowego. Przykładowo, aby do połączeń SLIP przeznaczyć drugi port szeregowy, wydaj polecenie

```
slattach /dev/cua1 &
```

Gdyby po poleceniu tym nie postawić znaku `&` powodującego uruchomienie go w tle, konsola przestałaby być dostępna. Powyższe polecenie można również wpisać do któregoś z plików inicjalizacyjnych.

Po uruchomieniu programu `slattach`, port szeregowy jest podłączany do pierwszego urządzenia SLIP (zwykle `/dev/s10`). Jeśli używasz więcej niż jednego portu do połączeń SLIP, musisz uruchomić program `slattach` odpowiednią liczbę razy. Domyślnie większość systemów linuxowych używa dla takich portów interfejsu CSLIP. Jeśli nie odpowiada Ci taka sytuacja, powinieneś explicite zażądać użycia protokołu SLIP:

```
slattach -p slip /dev/cua1 &
```

Ważne jest, aby po obu stronach połączenia ustawiony był taki sam protokół – w przeciwnym przypadku prawidłowe przesyłanie danych będzie niemożliwe (istnieją jeszcze dwie wersje protokołu SLIP: `slip6` – protokół sześciobitowy, oraz protokół, który potrafi dostosować się do wersji używanej przez serwer). Nie można na przykład nawiązać połączenia, jeśli jeden komputer używa protokołu `slip6`, a drugi CSLIP.

Po przydzieleniu protokołu SLIP portu szeregowego, należy skonfigurować interfejs sieciowy tak samo, jak dla wszystkich innych urządzeń, za pomocą polecień `ifconfig` i `route`. Przykładowo, jeśli Twój komputer nazywa się `merlin`, a chcesz połączyć się z komputerem `arthur`, powinieneś wydać polecenia:

```
ifconfig s10 merlin-slip pointtopoint arthur
route add arthur
```

Polecenie `ifconfig` służy do konfiguracji interfejsu sieciowego o nazwie `merlin-slip` (lokalny adres interfejsu SLIP), natomiast polecenie `route` powoduje dodanie odpowiedniego wpisu do tablic kierowania przesyłem danych.

Jeśli chcesz używać protokołu SLIP do łączenia się z Internetem, musisz posiadać adres IP i odpowiedni wpis w pliku `/etc/hosts`.

Po wykonaniu polecień `ifconfig` i `route` możesz przetestować połączenie – powinno działać bez problemów. Jeśli w przyszłości będziesz chciał usunąć interfejs SLIP, najpierw należy usunąć odpowiedni wpis z tablic kierowania przesyłem danych, później wyłączyć interfejs polecienniem `ifconfig`, a na koniec zakończyć działanie procesu `slattach`. Można to zrobić w następujący sposób:

```
route del arthur
ifconfig sl0 down
```

Aby zakończyć działanie procesu `slattach`, trzeba podać jego numer identyfikacyjny (wyświetlany przez polecenie `ps`) jako argument polecenia `kill` (patrz rozdział 34. „Procesy”).

Jeśli posiadasz dedykowane połączenie z serwerem SLIP i chcesz używać programu `slattach`, plik `rc.inet1` powinieneś zmodyfikować następująco:

```
IPADDR="123.12.3.1"          #adres IP Twojego komputera
REMADDR="142.12.3.12" #adres IP serwera SLIP

slattach -p cslip -s 19200 /dev/ttys0      #ustawienie predkosci i portu
/etc/ifconfig sl0 $IPADDR pointtopoint $REMADDR up
/etc/route add default gw $REMADDR
```

Wiersze podobne do powyższych zwykle znajdują się już w tym pliku, tyle że zaznaczone są jako komentarz. Informacje o prędkości połączenia, protokole i porcie powinieneś oczywiście dostosować do swoich potrzeb. Jeśli serwer nie obsługuje protokołu `cslip`, zamiast niego jako parametru polecenia `slattach` użyj wartości `slip`.

Jeśli serwer SLIP przydziela adresy IP dynamicznie, nie możesz wpisać adresu IP do pliku konfiguracyjnego. Większość serwerów SLIP wyświetla wtedy informację zawierającą przydzielony numer IP, który może zostać przechwycony za pomocą programu `dip`.

dip

Program `dip` znakomicie upraszcza łączenie się z serwerem SLIP. Aby go użyć, będziesz potrzebował skryptu, zawierającego wszystkie polecenia związane z nawiązaniem połączenia. W skrypcie takim zwykle następuje również przesłanie identyfikatora i hasła, co jeszcze bardziej upraszcza logowanie.

Przykładowy skrypt `dip` zamieszczony jest na stronach `man` dotyczących tego programu – jeśli chcesz go użyć, skieruj go do pliku i dostosuj do swoich potrzeb. Oto prosty skrypt (jeśli jest taka potrzeba, możesz oczywiście wpisać go ręcznie, nie zapominając o dostosowaniu danych do swojego systemu).

```
#
# sample.dip   Dialup IP connection support program.
#
# Version:      @(#)sample.dip 1.40 07/20/93
#
# Author: Fred N. van Kempen, <waltje@uWalt.NL.Mugnet.ORG>
#
main:
    # First of all, set up our name for this connection.
    # I am called "uwalt.hacktic.nl"  (== 193.78.33.238)
    get $local uwalt.hacktic.nl

    # Next, set up the other side's name and address.
    # My dialin machine is called 'xs4all.hacktic.nl'
    Σ(== 193.78.33.42)
```

```

get $remote xs4all.hacktic.nl
# Set netmask on s10 to 255.255.255.0
netmask 255.255.255.0
# Set the desired serial port and speed.
port cuao2
speed 38400
# Reset the modem and terminal line.
# This seems to cause trouble for some people!
reset

# 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_error
send MYLOGIN\n
wait ord: 20
if $errlvl != 0 goto password_error
send MYPASSWD\n

loggedin:
# We are now logged in.
wait SOMETEXT 15
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

Σdone
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 1

```

```
exit:  
    exit
```

Obecnie dostępnych jest kilka wersji tego skryptu, również na dołączonej do książki płycie CD-ROM. Jeśli masz dostęp do Internetu, możesz znaleźć je w węzłach FTP i w grupach dyskusyjnych.

Konfiguracja protokołu PPP

Protokół PPP jest używany częściej niż SLIP, głównie dlatego, że ma większe możliwości. Jego funkcje podzielone są na dwie grupy: protokół HLDC (ang. *High Level Data Link Control*), który definiuje zasady przesyłania bloków informacji pomiędzy maszynami, oraz funkcje rezydentnego programu `pppd`, obsługującego protokół po nawiązaniu połączenia przez HDLC.

Podobnie jak w przypadku protokołu SLIP, najpierw następuje normalne połączenie pośród dwoma komputerami poprzez modem, a dopiero później PPP przejmuje kontrolę nad połączeniem. Zanim uda Ci się ustanowić połączenie PPP, musisz skonfigurować sterownik pętli zwrotnej. Powinieneś również posiadać prawidłowo działający system tłumaczenia nazw, obojętnie, czy zapisany w pliku `/etc/hosts`, czy w postaci serwera DNS (patrz podrozdział „Używanie systemu DNS z protokołami SLIP i PPP” w dalszej części tego rozdziału).

Zakładanie konta PPP

Jeśli zamierzasz pozwolić innym użytkownikom łączyć się z Twoim systemem za pomocą protokołu PPP, ze względów bezpieczeństwa najlepiej jest założyć konto przeznaczone tylko do tego celu, o nazwie `ppp`. Nie jest to konieczne w przypadku, gdy nie przewidujesz, by ktoś miał łączyć się z systemem przez telefon. Możesz również użyć protokołu PPP logując się jako dowolny użytkownik. Jednak dla podniesienia poziomu bezpieczeństwa warto założyć konto PPP, szczególnie jeśli zamierzasz zezwolić na obsługę połączeń przychodzących. Proces zakładania konta `ppp` nie różni się niczym od zakładania konta dla nowego użytkownika – można użyć skryptu `adduser` czy `newuser`, bądź też ręcznie zmodyfikować zawartość pliku `/etc/passwd`.

Przykładowy wpis w pliku `/etc/passwd` definiujący konto użytkownika o identyfikatorze `ppp` (o numerze `UID` równym 201 oraz numerze `GID` równym 51) może mieć następującą postać:

```
ppp:*:201:51:PPP account:/tmp:/etc/ppp/ppscript
```

W tym przypadku dla użytkownika `ppp` hasło jest zablokowane (nikt więc nie będzie mógł się zalogować na to konto) i, ponieważ nie zostały utworzone żadne pliki, katalogiem domowym jest `/tmp`. Program uruchamiany domyślnie po zalogowaniu to `/etc/ppp/ppscript`, będący skryptem, w którym powinieneś zapisać wszystkie informacje konfiguracyjne.

Oto przykładowa zawartość skryptu `pppscript`:

```
#!/bin/sh
mesg n
stty -echo
exec pppd -detach silent modem crtscts
```

Pierwszy wiersz wymusza uruchomienie programu w interpreterze `sh`, bez względu na to, jaki jest aktualnie używany (patrz rozdział 14. „Programowanie w języku powłoki”). Drugi wyłącza wszystkie próby zapisu do konsoli używanej przez PPP. Polecenie `stty` w trzecim wierszu powoduje, że dane wysyłane przez zdalny komputer nie będą wyświetlane na ekranie. Ostatnie polecenie uruchamia program rezydentny `pppd`, obsługujący protokół PPP. Program `pppd` zostanie omówiony nieco później.

Nawiązywanie połączenia - program `chat`

Aby można było uruchomić PPP, wcześniej trzeba nawiązać połączenie z komputerem zdalnym. Można to zrobić na wiele sposobów; jednym z nich jest użycie programu `chat`. Program ten jest popularny głównie dlatego, że korzysta ze skryptów o formacie bardzo podobnym do skryptów UUCP.

Aby użyć programu `chat`, trzeba wydać polecenie przypominające nieco wiersze pliku konfiguracyjnego protokołu UUCP o nazwie `/etc/Systems`. Przykładowo, aby połączyć się z komputerem dostępnym pod numerem telefonu 555-1234, można wydać polecenie:

```
chat "" ATZ OK ATDT5551234 CONNECT "" login: ppp word: secret1
```

Parametry tego programu to pary spodziewany tekst-tekt do wysłania. Po uruchomieniu programu nie musimy czekać na przesłanie żadnego tekstu, więc pierwszy tekst jest pusty. Wysyłamy polecenie `ATZ`, resetujące modem i oczekujemy na potwierdzenie (`OK`). Następnie wybieramy numer (`ATDT5551234`) i oczekujemy na komunikat `CONNECT`. Po uzyskaniu połączenia nie wysyłamy nic i czekamy na zachętą `login:` ze strony zdalnego komputera. Po jej uzyskaniu wysyłamy identyfikator `ppp` i czekamy na pytanie o hasło (`word:`). Na koniec wysyłane jest hasło (`secret1` – tu należy oczywiście podstawić wartość oczekiwana przez system zdalny) i program `chat` kończy działanie, pozostawiając jednak otwarte połączenie.

Jeśli druga strona połączenia nie wysyła tekstu `login:` zaraz po uruchomieniu modemu, możesz być zmuszony do użycia polecenia `BREAK`, aby „obudzić” system zdalny. Można to zrobić wydając polecenie:

```
chat - v "" ATZ OK ATDT5551234 CONNECT "" ogin: -BREAK- ogin: ppp word:
secret1
```

Istnieje jednak dość poważny problem związany z bezpieczeństwem, ponieważ dowolny użytkownik, który wykona polecenie `ps ef` zobaczy na ekranie pełną treść polecenia, jakie wydałeś przy uruchamianiu programu `chat` – nie wyłączając hasła. Można łatać obejść ten problem zapisując odpowiednie parametry w pliku, którego nazwę należy następnie przekazać jako parametr polecenia `chat`:

```
chat -f nazwa_pliku
```

Plik `nazwa_pliku` powinien zawierać następującą treść:

```
"" ATZ OK. ATDT5551234 CONNECT "" login: ppp word: secret1
```

W skrypcie można również wykryć często spotykane problemy, np. gdy linia jest zajęta (`BUSY`) lub nie udało się nawiązać połączenia (`NO CARRIER`). Takie komunikaty generowane przez modem można poprzedzić słowem `ABORT`, które powoduje zakończenie działania skryptu, gdy odebrana zostanie kotaś z tych informacji. Przykładowo zmodyfikowany skrypt wygląda następująco:

```
ABORT BUSY ABORT `NO CARRIER` "" ATZ OK. ATDT5551234 CONNECT "" login:  
ppp word: secret1
```

Potrzebne były dwa polecenia `ABORT`, ponieważ pojedyncze polecenie `ABORT` może przyjąć tylko jeden argument. Reszta skryptu pozostała bez zmian. Pojedynczy cudzysłów wokół składającego się z dwóch wyrazów parametru `NO CARRIER` zapewnia jego prawidłową interpretację.

Uruchamianie pppd

Po nawiązaniu połączenia i zalogowaniu się na konto PPP, można uruchomić program rezydentny `pppd`, który przejmie kontrolę nad połączeniem PPP. Jeśli Twój komputer używa portu `/dev/cua1` (oznaczenie pierwszego portu szeregowego podłączonego do modemu) do połączeń PPP z prędkością 38400 bodów, odpowiednie polecenie ma postać:

```
pppd /dev/cua1 38400 crtscts defaultroute
```

Nakazuje ono kernelowi przełączyć interfejs obsługujący `/dev/cua1` na PPP i ustawić połączenie IP z komputerem zdalnym. Opcja `crtsccts`, używana prawie zawsze przy połączeniach o prędkości powyżej 9600 bodów, załączca potwierdzanie transmisji.

Twój system będzie używał własnego numeru IP, co nie powinno sprawiać problemów większości przypadków. Jeśli jednak chcesz wymusić użycie innego numeru (zarówno dla systemu lokalnego, jak i zdalnego), możesz to zrobić podając go w wierszu polecień w następującym formacie:

```
lokalny_adres_IP:zdalny_adres_IP
```

Jeśli chcesz zmienić tylko jeden adres, drugą część po prostu pozostaw pustą, na przykład tak:

```
147.23.43.1:
```

Powyższe polecenie spowoduje wymuszenie użycia przez komputer lokalny adresu 147.23.43.1.

Za pomocą polecenia `chat` uruchomionego jako argument polecenia `pppd` można od razu nawiązać połączenie i przekazać je do programu `pppd`. Najwygodniej w tym celu zapisać parametry programu `chat` w pliku i użyć opcji `-f`:

```
pppd connect "chat -f nazwa_pliku" /dev/cua1 38400 -detach crtscts modem  
Σ defaultroute
```

Plik `nazwa_pliku` zawiera pary tekst wysyłany-tekst oczekiwany, jak omówiliśmy to już wcześniej. Opcja `-detach` nakazuje programowi `pppd` nie odłączać się od konsoli i rozpoczęć działanie w tle. Opcja `modem` powoduje obserwację portu szeregowego i rozłączenie „rozmowy” w przypadku zerwania połączenia.

Program `pppd` rozpoczyna nawiązywanie połączenia PPP od wymiany adresów IP, następnie ustawia parametry transmisji. Później `pppd` daje znać oprogramowaniu sieciowemu, że w użyciu jest łącze PPP, ustawiając odpowiedni interfejs dla `/dev/ppp0` (jeśli jest to pierwsze połączenie PPP). Na koniec wykonane zostaje odpowiednie polecenie `route`, kierujące dane przez to połączenie.

Testowanie konfiguracji

`pppd` wysyła komunikaty bezpośrednio do programu `syslog` (taką samą sytuację mamy w przypadku programu `chat`, o ile został uruchomiony z opcją `-v`). Jeśli masz jakieś kłopoty z połączeniem PPP, powinieneś nakazać programowi `syslog` rejestrowanie wszystkich przychodzących informacji w pliku, a potem je przejrzeć. Musisz być jednak świadomym faktu, że zapisywane są również hasła i identyfikatory użytkowników – z tego powodu nie powinieneś domyślnie używać opcji `-v` polecenia `chat`.

Jeśli w pliku `/etc/syslog.conf` nie ma wpisu określającego nazwę pliku, w którym należy rejestrować komunikaty, wszystkie komunikaty przesyłane do programu `syslog` są tracone. Jeśli więc chcesz rejestrować informacje pochodzące z programów `pppd` i `chat`, powinieneś do tego pliku dodać wiersz:

```
daemon.*      /tmp/ppp-log
```

Powoduje on zapisywanie wszystkich informacji od programów rezydentnych do pliku `/tmp/ppp-log` (można oczywiście użyć dowolnej nazwy pliku). W wielu wersjach Linuxa separatorem w tym pliku musi być znak tabulacji, nie spacja. Kiedy skrypt zacznie działać poprawnie, pamiętaj o usunięciu tego wiersza, ponieważ rozmiary pliku z rejestrowanymi danymi szybko stają się nie do przyjęcia.

PPP a bezpieczeństwo

PPP to wspaniały protokół do komunikowania się przez modem, ale luki w bezpieczeństwie, które tworzy, są tak wielkie, że mógłby przez nie przejechać autobus. Drobne błędy w jego konfiguracji powodują, że każdy może uzyskać dostęp do komputera lub użyć połączenia PPP do łączenia się z innymi komputerami. Dla wyeliminowania tych problemów wprowadzono technikę uwierzytelniania (zwaną po angielsku *authentication*), polegającą na sprawdzaniu, czy komputer/osoba na każdym z końców połączenia faktycznie jest tym, za kogo się podaje, i czy wolno mu używać połączenia.

PPP używa dwóch metod uwierzytelniania: PAP (ang. *Password Authentication Protocol*, protokół uwierzytelniania haseł) i CHAP (ang. *Challenge Handshake Authentication Protocol*, protokół uwierzytelniania przez wezwanie do potwierdzenia). PAP przypomina nieco procedurę logowania. Kiedy jeden komputer wysyła identyfikator użytkownika i hasło do drugiego, komputer odbierający sprawdza te informacje po swojej stronie. Metoda ta jest bardzo prosta, ale ma jedną bardzo poważną wadę: każdy może przechwycić połączenie i pod słuchać przesypane hasło.

CHAP rozwiązuje ten problem, dlatego jest używany o wiele częściej. Pierwszy komputer wysyła dowolny ciąg znaków do drugiego komputera razem ze swoją nazwą. Drugi komputer używa takiego samego ciągu, tej nazwy oraz swojej nazwy do przeprowadzenia pewnych operacji na przesłanym ciągu znaków, po czym odsyła go wraz ze swoją nazwą. Pierwszy komputer przeprowadza podobną operację, po czym jeśli otrzymany ciąg znaków zgadza się z pierwszym wysłanym, połączenie uważane jest za bezpieczne. Dodatkowym zabezpieczeniem jest fakt, że taka procedura powtarzana jest wielokrotnie podczas trwania połączenia.

Kiedy dwa systemy łączą się, domyślnie nie używają mechanizmów uwierzytelniania. Kiedy uwierzytelnianie jest jednak załączone, jeden z końców połączenia próbuje użyć protokołu CHAP. Jeśli nie uda się nawiązać połączenia, wypróbowywany jest protokół PAP. Jeżeli również za pomocą tego protokołu nie uda się potwierdzić tożsamości komputera z drugiej strony połączenia, połączenie jest przerywane. Aby przy połączeniach używać mechanizmów uwierzytelniania, trzeba w pliku `/etc/ppp/options` podać opcję `auth`. Jeśli jednak komputery, z którymi się łączysz, nie obsługują tych mechanizmów, nawiązanie połączenia nie będzie możliwe.

Informacje potrzebne do używania CHAP i PAP przechowywane są w plikach `/etc/ppp/chap-secrets` i `/etc/ppp/pap-secrets`. Jeśli zamierzasz używać uwierzytelniania dla wszystkich połączeń (co jest bardzo dobrym pomysłem), powinieneś zadbać o odpowiednią zawartość tych plików. Jeśli skonfigurujesz oba te pliki i zastosujesz opcję `auth` w pliku `/etc/ppp/options`, nikt nie będzie mógł połączyć się z Twoim systemem bez potwierdzenia swojej tożsamości.

Plik `/etc/ppp/chap-secrets` składa się z czterech kolumn danych, zawierających odpowiednio nazwę klienta, nazwę serwera, tekst-hasło i opcjonalną listę adresów IP. Zachowanie systemu jest różne w zależności od tego, czy komputer lokalny żąda uwierzytelnienia od komputera próbującego się połączyć, czy też sam jest wzywany do uwierzytelnienia. Kiedy komputer lokalny ma potwierdzić swoją tożsamość, `pppd` sprawdza, czy w pliku tym znajduje się wpis z nazwą klienta zgodną z nazwą systemu i odpowiednią nazwą serwera, po czym używa podanego tekstu do skonstruowania wysyłanej wiadomości. Pojedynczy wpis w pliku `/etc/ppp/chap-secrets` może mieć następującą postać:

```
#client           server           string           addresses
merlin.tpc1.com   big_guy.big_net.com  "I hate DOS"
```

W takim przypadku do konstruowania wiadomości potwierdzającej używany jest tekst `I hate DOS` (otoczony cudzysłowem, ponieważ składa się z więcej niż jednego wyrazu). Wraz z nazwą systemu jest on wysyłany do `big_guy.big_net.com`, gdzie jest

kodowany wraz z nazwą tamtego systemu i odsyłany z powrotem. Tu jest odkodowywany i jeśli otrzymany zostanie ponownie tekst I hate DOS, oznacza to, że wszystko jest w porządku. W pliku /etc/ppp/chap-secrets może oczywiście znajdować się więcej wpisów, na przykład:

```
#client          server          string          adresses
merlin.tpc1.com   big_guy.big_net.com "I hate DOS"
merlin.tpc1.com   chatton.cats.com   "Meow, meow, meow"
merlin.tpc1.com   roy.sailing.ca    "Hoist the spinnaker"
```

Kiedy to Twój komputer żąda potwierdzenia, procedura jest odwrócona. Program pppd szuka wpisu z nazwą zdalnego komputera w polu z nazwą klienta i z nazwą lokalnego komputera w polu serwera. Odpowiednie wpisy mają więc postać:

```
#client          server          string          adresses
big_guy.big_net.com merlin.tpc1.com "Size isn't everything"
```

Jak widać, plik /etc/ppp/chap-secrets powinien składać się z par wpisów, w których pola klienta i serwera są zamienione miejscami (ponieważ uwierzytelnianie może przebiegać w obie strony), na przykład tak:

```
#client          server          string          adresses
merlin.tpc1.com   big_guy.big_net.com "I hate DOS"
big_guy.big_net.com merlin.tpc1.com "Size isn't everything"
merlin.tpc1.com   chatton.cats.com   "Meow, meow, meow"
chatton.cats.com  merlin.tpc1.com "Here, Kitty, Kitty"
merlin.tpc1.com   roy.sailing.ca    "Hoist the spinnaker"
roy.sailing.ca    merlin.tpc1.com "Man overboard"
```

Rozmiar takiego pliku może być dość znaczny, jeśli łączysz się z większą liczbą komputerów. Na szczęście dozwolone jest użycie symboli wieloznaczkowych, takich jak symbol gwiazdki:

```
#client          server          string          adresses
merlin.tpc1.com   big_guy.big_net.com "I hate DOS"
big_guy.big_net.com merlin.tpc1.com "Size isn't everything"
merlin.tpc1.com   chatton.cats.com   "Meow, meow, meow"
chatton.cats.com  merlin.tpc1.com "Here, Kitty, Kitty"
merlin.tpc1.com   roy.sailing.ca    "Hoist the spinnaker"
*                 merlin.tpc1.com "Man overboard"
```

W podanym wyżej przykładzie ostatni wpis pozwala każdej innej maszynie połączyć się z komputerem lokalnym za pomocą tego samego tekstu. Oczywiście, w pliku chap-secrets zdalnego komputera musi znajdować się taki sam wpis. Jest to nieco mniej bezpieczny sposób, niż użycie osobnego tekstu dla każdego komputera, ale może zaoszczędzić sporo czasu.

Pole zawierające adresy IP, nie wykorzystane w powyższych przykładach, pozwala używać zarówno adresu IP, jak i nazwy symbolicznej klienta. Jest to konieczne w przypadku, gdy zdalny komputer wysyła inny numer IP za każdym razem, co normalnie spowodowałby błąd przy uwierzytelnianiu. Jeśli pole to jest puste, akceptowany jest dowolny adres IP. Jeśli jest w nim myślnik, dostęp z danego komputera jest zabroniony.

Plik `/etc/ppp/pap-secrets` jest podobny do pliku `/etc/ppp/chap-secrets`. Poszczególne pola określają nazwę klienta, serwera, tekst tajnego hasła i dostępne dla systemu aliasy. Pliki te jednak różnią się nieco wyglądem, ponieważ nazwy klienta i serwera nie są nazwami pełnymi, a hasło jest pojedynczym blokiem tekstu, na przykład:

```
# /etc/ppp/pap-secrets
#   user      server      string      addresses
merlin    darkstar    yG55Sj28    darkstar.big_net.com
darkstar  merlin     5Drbt32    merlin.tpci.com
merlin    chatton    MeowMeow  chatton.cats.com
chatton   merlin     74GRr34    merlin.tpci.com
```

Dwa pierwsze wiersze odpowiadają za połączenia z komputerem `darkstar`. Pierwszy wiersz określa, w jaki sposób ma przebiegać uwierzytelnianie w przypadku, gdy wymaga go komputer `darkstar`. Nazwa użytkownika zapisana w pierwszej kolumnie jest przesyłana do komputera zdalnego, druga kolumna określa nazwę, pod jaką widziany będzie serwer. Stwarza to problem: program `pppd` nie ma możliwości poznania pełnej nazwy komputera zdalnego – może tylko poznać jego numer IP. Można również podać numer IP lub nazwę komputera zdalnego na końcu polecenia uruchamiającego proces `pppd`, na przykład tak:

```
pppd ..... remotename chatton user merlin
```

W powyższym przykładzie serwer nazywa się `chatton`, natomiast Twój komputer – czyli klient – `merlin`. Dzięki możliwości podania opcji `user` można wymusić użycie innej niż rzeczywista nazwy systemu lokalnego.

Używanie systemu DNS z protokołami SLIP i PPP

Jeśli protokoły SLIP i PPP używasz do czegoś więcej niż czytania poczty czy przeglądania grup dyskusyjnych, prawdopodobnie wygodnym rozwiązaniem będzie użycie systemu DNS. Najłatwiej zrobić to wpisując adres IP serwera nazw do pliku `/etc/resolv.conf`. Przykładowo, jeśli masz dostęp do serwera nazw o adresie IP 145.2.12.1, Twój plik `/etc/resolv.conf` powinien wyglądać tak:

```
#/etc/resolv.conf
domain      merlin.com          #domena lokalna
nameserver  145.2.12.1          #serwer nazw internetowych
```

Jeśli taki wpis istnieje, SLIP czy PPP wysyła prośbę o przetłumaczenie nazwy na adres IP do serwera nazw i czeka na odpowiedź. Im łatwiejszy jest dostęp do serwera nazw, tym wydajniejsza będzie taka procedura. Z tego powodu powinieneś wybierać możliwie najbliższy serwer nazw (pod względem czasu dostępu).

Użycie tego sposobu ma jednak swoje wady. Każde tłumaczenie adresu musi przejść przez linię SLIP czy PPP, co może spowodować zwolnienie działania aplikacji i zwiększenie ruchu w sieci.

Problem zmniejszenia wydajności można obejść konfigurując bufor serwera nazw. W tym celu trzeba zmodyfikować zawartość pliku `/etc/named.boot`. Jeśli chcesz skonfigurować swój komputer jako bufor serwera nazw, plik ten powinien wyglądać następująco:

```
; /etc/named.boot
directory    /var/named
cache        .db.cache          ;cache-only
primary      0.0.147.in-addr.arpa db.cache      ;loopback
```

W powyższym przykładzie do określenia sterownika pętli zwrotnej użyta została lokalna nazwa sieciowa w formacie IN-ADDR.ARPA. Lista serwerów nazw będzie przechowywana w pliku `.db.cache`.

Podsumowanie

Konfigurowanie protokołów PPP i SLIP w systemie linuxowym nie jest skomplikowane, ale wymaga poświęcenia uwagi szczegółów przy modyfikowaniu plików konfiguracyjnych. Po skonfigurowaniu, protokoły te mogą być używane do uzyskiwania dostępu do Internetu czy dostępu do innych maszyn używających tych protokołów. Jeśli chcesz uzyskać informacje na tematy pokrewne, przejdź do innych rozdziałów.

Ogólne zasady pracy w sieci omówione są w rozdziale 37. „Praca w sieci”.

NIS oraz YP, systemy, które dodadzą elastyczności systemowi linuxowemu podłączonemu do sieci, opisane są w rozdziale 44. „NIS oraz YP”.

O konfigurowaniu własnego węzła internetowego przeczytać możesz w rozdziale 47. „Konfigurowanie węzła internetowego”.

Rozdział 39.

UUCP

Tim Parker

W tym rozdziale:

- υ Konfiguracja protokołu UUCP
- υ Połączenie UUCP
- υ Komunikacja bezpośrednia
- υ Skrypty logowania
- υ Modyfikowanie harmonogramu dostępu
- υ UUCP a bezpieczeństwo
- υ Używanie protokołu UUCP

Protokół UUCP (ang. *UNIX to UNIX Copy*) opracowany został jako prosty protokół do połączeń modemowych pomiędzy systemami UNIX-owymi. Dziś najczęściej używany jest do przesyłania poczty, pozwalaając na korzystanie z niej za pośrednictwem modemu systemom nie podłączonym na stałe do sieci. Może również być używany do przeglądania grup dyskusyjnych i innych podobnych usług, nie wymagających stałego połączenia. UUCP tworzy połączenie pomiędzy dwoma systemami UNIX-owymi. Nie może być użyty do korzystania z systemu zdalnego (typu FTP czy Telnet). Choć posiada zabezpieczenia adekwatne do większości zastosowań, właśnie ten protokół jest najczęściej używany do włamań, ponieważ administratorzy przez niedopatrzenie czy niewiedzę nie konfigurują odpowiednio tych zabezpieczeń.

W systemach linuxowych używa się kilku wersji UUCP, które są kompatybilne ze sobą pod względem przesyłania danych, ale różnią się procesem konfiguracji. Często podczas instalacji systemu można wybrać pomiędzy wersją Taylor UUCP i HDB (HoneyDanBer) UUCP. Wielu użytkowników Linuxa woli wersję Taylor, choć użytkownicy innych systemów UNIX-owych wybierają nowszą wersję HDB. W tym rozdziale przyjrzymy się obu tym wersjom (inne są spotykane względnie rzadko, nie będziemy się więc nimi zajmować). W pierwszej części rozdziału zajmiemy się konfiguracją UUCP, druga omawia używanie tego protokołu.

Konfiguracja protokołu UUCP

Większość plików konfiguracyjnych dotyczących UUCP znajduje się w katalogu `/usr/lib/uucp`. Choć proces konfiguracji może wydawać się okropnie skomplikowany, w rzeczywistości tylko kilka plików wymaga modyfikacji (w każdym z nich trzeba zmienić jeden lub dwa wpisy).

Wersje Taylor i HDB różnią się całkowicie pod względem konfiguracji, omówimy je więc oddzielnie. Nie musisz się martwić o to, jaka wersja działa na drugim końcu połączenia – nie ma to żadnego znaczenia, o ile tylko system jest skonfigurowany prawidłowo.

W niektórych wersjach Linuxa dostępne są skrypty nieco ułatwiające konfigurację (częściej skrypty takie rozprowadzane są z wersją HDB, choć kilka skryptów jest również dostępnych dla wersji Taylor UUCP). Jeśli zdecydujesz się na ich użycie, mimo wszystko powinieneś sprawdzić wszystkie wpisy „ręcznie”.

W podanym niżej przykładzie zakładamy, że nazwą naszego komputera jest `merlin` i chcemy połączyć się za pomocą UUCP z komputerem `arthur`. Podczas konfigurowania swojego systemu powinieneś zachować format wszystkich podanych przykładów, wprowadzając oczywiście poprawki dotyczące nazw systemów.

Konfigurowanie Taylor UUCP

Poniżej przedstawiamy pliki biorące udział w procesie konfigurowania Taylor UUCP, wraz z ich krótkim opisem:

- υ **/usr/lib/uucp/config** – definiuje nazwę komputera lokalnego;
- υ **/usr/lib/uucp/sys** – określa, jak wywoływać systemy zdalne;
- υ **/usr/lib/uucp/port** – opisuje każdy port używany do nawiązywania połączenia i jego parametry;
- υ **/usr/lib/uucp/dial** – definiuje metody porozumiewania się z modemem przy nawiązywaniu połączeń;
- υ **/usr/lib/uucp/dialcodes** – w nim zapisywane są pełne odpowiedniki symbolicznych kodów numerów telefonicznych; rzadko używany, gdy istnieje połączenie bezpośrednie;
- υ **/usr/lib/uucp/call** – może zawierać identyfikator i hasło systemu zdalnego, ale obecnie jest rzadko używany;
- υ **/usr/lib/uucp/passwd** – zawiera identyfikatory użytkowników i hasła używane, gdy ktoś chce połączyć się z Twoim systemem; używany tylko w przypadku, gdy zamiast zwykłego logowania aktywny jest program `uucico`.

Aby nie komplikować problemu, pominiemy tu całą teorię i przejdziemy od razu do omówienia przykładowej konfiguracji. Jedyne, co musisz zrobić, to podstawić własne dane, takie jak numery telefonów, pliki urządzeń, nazwy systemów itp. Proces powinien być powtarzony dla każdego systemu, z którym chcesz się łączyć.

Pierwszy plik, który należy zmodyfikować, zawiera nazwę systemu i kilka innych ogólnych parametrów. Plik `/usr/lib/uucp/config` powinien zawierać jeden wiersz definiujący nazwę Twojego systemu:

```
nodename      merlin
```

Słowo `nodename` musi rozpoczynać wiersz, po nim muszą następować spacje lub znaki tabulacji, a następnie nazwa systemu lokalnego. Informacja w tym pliku może być wprowadzona podczas instalowania systemu – wtedy nie trzeba jej modyfikować. Powinieneś jednak upewnić się, że jest ona prawidłowa – w przeciwnym przypadku połączenie nie będzie działało.



Aby można było używać UUCP, Twój system musi posiadać nazwę. Dla kompatybilności ze starszymi wersjami UUCP powinna ona składać się z siedmiu lub mniej znaków. Najlepiej, jeśli jest to ta sama nazwa, którą podałeś podczas instalacji systemu. Ułatwi to życie użytkownikom, którzy będą chcieli połączyć się z Twoim systemem. Przykładowo, jeśli pełna nazwa domenowa systemu ma postać `merlin.tpci.com`, rozsądnie będzie nadać mu nazwę UUCP `merlin`.

Potrzebne będą również informacje o systemie, z którym zamierzasz się połączyć. Należy je wprowadzić do pliku `/usr/lib/uucp/sys`. Zwykle plik ten zawiera kilka przykładowych wpisów, można więc użyć jednego z nich jako szablonu, usuwając również znaki komentarza (#). Wpis określający parametry połączenia z komputerem `arthur` może wyglądać na przykład tak:

```
# system: arthur (Bill Smallwood's Linux system)
system      arthur
time        Any
phone       555-1212
port        com1
speed       9600
chat        login: merlin password: secret1
```

Pierwszy wiersz powyższego przykładu jest komentarzem. Następne wiersze określają różne parametry połączenia, takie jak nazwa systemu zdalnego (`arthur`), czas, kiedy można łączyć się z systemem (wartość `Any` oznacza brak ograniczeń), numer telefonu (włącznie z numerem kierunkowym itp.), port szeregowy, którego należy użyć (`com1`), prędkość połączenia (9600 bodów) oraz skrypt `chat`, automatyzujący logowanie (w tym przypadku odpowiadający na monit `login:` tekstem `merlin`, a następnie wysyłający hasło `secret1` po odebraniu tekstu `password:`).

Większość połączeń wymaga podania identyfikatora i hasła – muszą się one znaleźć w pliku konfiguracyjnym, ponieważ UUCP nie dopuszcza sesji interaktywnych. Może to stanowić pewien problem, ponieważ pozwala innym użytkownikom na odczytanie hasła dostępu, ale ponieważ używane jest ono tylko do obsługi UUCP, nie jest to poważne zagrożenie. Poza tym odpowiednio ustalone prawa dostępu do tego pliku powinny zabezpieczać przed odczytaniem go przez użytkowników systemu.



Nie wszystkie serwery wymagają podania hasła przy połączeniach UUCP. W niektórych używa się hasła pustego, w jeszcze innych - uucp.

Nazwa portu podana w pliku `/usr/lib/uucp/sys` nie musi zgadzać się z faktyczną nazwą urządzenia w systemie Linux – odpowiednie połączenie zapewnia wpis w pliku `/usr/lib/uucp/port`, w którym dla modemu o prędkości 9600 bodów powinien znaleźć się wpis podobny do takiego:

```
# com1 device port
port      com1
type      modem
device    /dev/cua0
speed    9600
dialer   Hayes
```

Pierwszy wiersz w powyższym przykładzie określa nazwę portu, używaną również w pliku `/usr/lib/uucp/sys`. Typ połączenia (zwykle `modem`) określony jest w kolejnym wierszu. Faktyczne urządzenie, do którego odnosi się dany port, określone jest w wierszu trzecim. W wielu systemach można tu wpisać identyfikator `/dev/modem`, który jest dołączaniem do odpowiedniego urządzenia.

Dalej następuje maksymalna prędkość połączenia modemowego. Jako ostatnia podana jest informacja o typie modemu, a właściwie sposobie, w jaki ma on wybrać numer. Jest to pozostałość po czasach, gdy modemy nie potrafiły same wybierać numeru i używały do tego celu urządzenia o nazwie dialer.

Typ modemu podany w pliku `/usr/lib/uucp/port` musi być zdefiniowany w pliku `/usr/lib/uucp/dial`; oto przykładowy wpis:

```
# Hayes modem
dialer   Hayes
chat     "" ATZ OK ATDT\T CONNECT
```

Symbol `\T` automatycznie zastępowany jest przez numer, z jakim należy się połączyć. W niektórych systemach dane z plików `/usr/lib/uucp/dial` i `/usr/lib/uucp/port` połączone są w jeden wpis w pliku `/usr/lib/uucp/sys`, bezpośrednio opisujący wszystkie parametry modemu i połączenia.

Komputer na drugim końcu połączenia (w tym przypadku `arthur`) musi posiadać w plikach konfiguracyjnych odpowiednie wpisy dla komputera `merlin`. Będą one podobne do przedstawionych powyżej, różnice wystąpią tylko w nazwach systemów, numerze

telefonu, ewentualnie również w skrypcie `chat` i typie modemu. Dopóki oba komputery nie są skonfigurowane prawidłowo, uzyskanie połączenia nie będzie możliwe.

W niektórych systemach dostępny jest program `uuchck`, który weryfikuje wpisy w plikach konfiguracyjnych i wyświetla odpowiednie podsumowanie. Program ten możesz również zdobyć w węzłach FTP i BBS (namiary na nie możesz znaleźć w dodatku A „Węzły FTP i grupy dyskusyjne poświęcone Linuxowi”).

Domyślnie Taylor UUCP pozwala systemowi zdalnemu po połączeniu się wykonać tylko wybrane polecenia. Zwykle są to polecenia `rmail` i `rnews`. Jeśli chcesz pozwolić na używanie jakichś dodatkowych poleceń, musisz dodać wiersz zawierający ich nazwy do pliku `/usr/lib/uucp/sys`; na przykład, aby zezwolić na używanie poleceń `rmail`, `rnews`, `rsmtp` i `rdataupdate`, należy dodać następujący wiersz:

```
system      chatton  
...  
commands    rmail   rnews   rsmtp   rdataupdate
```

Wszystkie polecenia muszą znajdować się w ścieżce przeszukiwania używanej przez programy obsługi UUCP (czyli w zasadzie program `uuxqt`).

Jeśli chcesz przesyłać pliki pomiędzy dwoma komputerami, musisz również zmodyfikować inne pliki konfiguracyjne. Kiedy system zdalny wysyła plik do Twojego komputera, powinieneś być on ze względów bezpieczeństwa zapisany w katalogu `/usr/spool/uucppublic` lub `/var/spool/uucppublic`. Nie powinieneś pozwalać na zapisanie plików gdziekolwiek indziej, ponieważ może się to skończyć zniszczeniem plików systemowych. Przyjęło się, że przesyłane pliki trafiają do katalogu `/usr/spool/uucppublic` lub `/usr/spool/uucp/system`, gdzie `system` to nazwa systemu zdalnego.

Katalogi, do których trafiają mają pliki przesyłane przez UUCP i z których można wysyłać pliki, można określić w pliku `/usr/lib/uucp/sys`:

```
system      chatton  
...  
local-send  ~/send  
local-receive ~/receive
```

Przy takiej konfiguracji użytkownicy Twojego systemu mogą wysyłać pliki znajdujące się w podkatalogu `send` katalogu `uucp`, a każdy plik przychodzący zostanie zapisany w podkatalogu `receive`. Można zezwolić na wysyłanie plików z kilku różnych miejsc, na przykład z katalogów domowych użytkowników:

```
local-send  ~/send  /home
```

Jeśli chcesz umożliwić obsługę żądań transferu pochodzących z komputera zdalnego, powinieneś dodać dwa wpisy:

```
remote-send      /usr/lib/uucppublic  
remote-request  /usr/lib/uucppublic
```

Powodują one, że komputer zdalny może pobierać pliki z katalogu `/usr/lib/uucppublic`, może również w tym samym katalogu zapisywać pliki. Również tu można zezwolić na korzystanie z kilku katalogów, oddzielając ich nazwy spacjami.

UUCP pozwala także na przesyłanie danych do innego komputera za pomocą komputera pośredniczącego (ang. *hopping*). Przykładowo, jeśli chcesz wysyłać pocztę do systemu `warlock`, ale połączyć się z nim możesz tylko przez systemem `wizard`, powinieneś poinformować o tym fakcie system UUCP. W tym celu do pliku `/usr/lib/uucp/sys` powinieneś dodać polecenie `forward`:

```
system      wizard
...
forward     warlock
```

Następnie należy dodać wpis dla systemu `warlock`, który poinformuje UUCP, że poczta przychodząca od Ciebie przechodzić będzie przez system `wizard`:

```
system warlock
...
forward-to   merlin
```

Wpis `forward-to` gwarantuje, że pliki zwrócone przez system `warlock` będą przesyłane do systemu `merlin`, czyli do Twojego komputera. W przeciwnym przypadku byłyby one usuwane jako niemożliwe do dostarczenia.

Domyślnie Taylor UUCP nie pozwala na pośredniczenie w przesyłaniu poczty. Administrator powinien dokładnie rozważyć zezwolenie na taką działalność ze względu na potencjalne duże obciążenie łącza modemowego plikami pochodzącyimi z innych systemów.

Konfigurowanie HDB UUCP

HDB UUCP jest systemem nowszym niż Taylor UUCP. Pod wieloma względami jego konfiguracja jest łatwiejsza, ale tak naprawdę oba systemy nie jest trudno skonfigurować.

Nazwa systemu lokalnego nie jest ustawiana w żadnym pliku konfiguracyjnym – używana jest nazwa podana podczas instalacji sieci. Można ją zmienić za pomocą polecenia `hostname`.

Nazwy systemów zdalnych przechowywane są w pliku `/usr/lib/uucp/Systems` (w niektórych starszych wersjach plik ten nazywa się `/usr/lib/uucp/L.sys`). Każdemu z systemów odpowiada osobny wiersz o formacie:

```
nazwa_systemu  ogr_czasowe    typ_urządzenia prędkość      telefon
Σ              skrypt_logowania
```

Pole `nazwa_systemu` określa nazwę systemu zdalnego. Pole `ogr_czasowe` pozwala ustalić, w jakich godzinach może nastąpić połączenie. `typ_urządzenia` to etykieta odpowiadająca odpowiedniemu wpisowi w pliku `/usr/lib/uucp/Devices`. `prędkość` określa maksymalną prędkość przesyłania danych (ograniczoną zwykle prędkością modemu z jednej lub z drugiej strony połączenia), `telefon` to numer telefonu, do którego podłączony jest system zdalny, a `skrypt_logowania` określa identyfikator użytkownika i hasło, które

należy podać po połączeniu się z systemem (podobnie jak skrypt `chat` w systemie Taylor UUCP). Przykładowy wpis dla systemu zdalnego `arthur` może mieć postać:

```
arthur Any      ACU      9600    555-1212      login: uucp password: Σ  
secret1
```

Wartość `Any` w polu `ogr_czasowe` nie nakłada żadnych ograniczeń co do czasu połączenia. Typ urządzenia o etykiecie `ACU` musi być zdefiniowany w pliku `/usr/lib/uucp/Devices` (w starszych wersjach plik ten nazywa się `/usr/lib/uucp/L-devices`).

Wpisy w pliku `/usr/lib/uucp/Devices` (lub w starszych wersjach `/usr/lib/uucp/L-devices`) zawierają dane o urządzeniach, których można użyć do połączeń z systemami zdalnymi. Mają one składnię:

```
typ_urządzenia sterownik      dialer_line      prędkość      dialer  
Σ      [token Dialer ...]
```

`typ_urządzenia` to etykieta, która będzie używana dla danych ustawień. `sterownik` to nazwa urządzenia, które użyte będzie do komunikacji (zwykle port szeregowy, jak np. `/dev/tty2a` czy `/dev/modem`). Pole `dialer_line` nie jest używane, należy w nie wpisać myślnik. Parametr `prędkość` określa maksymalną prędkość przesyłania danych, a `dialer` to nazwa pliku, w którym znajdują się instrukcje dotyczące obsługi danego urządzenia. Przykładowy wpis dla modemu Hayes 9600 podłączonego do drugiego portu szeregowego może wyglądać tak:

```
ACU      tty2A      -      9600      dialHA96
```

Opisuje on urządzenie o etykiecie `ACU`, podłączone do portu `/dev/tty2A` (część `/dev` nazwy urządzenia została pominięta – HDB UUCP dopuszcza takie uproszczenie), używające programu zapisanego w pliku `dialHA96` do sterowania usługą wybierania numerów. Dla większości popularnych modemów dostępne są gotowe pliki, których nazwy należy wpisać w polu `dialer`.

Jeśli nie jest dostępny plik opisujący komunikację z modelem, należy stworzyć odpowiedni wpis w pliku `/usr/lib/uucp/Dialers` o formacie:

```
dialer translacje      oczekuj      wyslij ...
```

`dialer` to etykieta odpowiadająca użytej w pliku `/usr/lib/uucp/Devices`. `translacje` to tablica translacji używana podczas wybierania numeru, pozwalająca zamienić odpowiednie znaki na pauzy itp. Pola `oczekuj` i `wyslij` to skrypt `chat`, dzięki któremu ustawiane są parametry pracy modemu. Znaki białe w tym skrypcie są ignorowane, chyba że znajdują się w cudzysłowach. Oto przykładowy wpis dla modemu Hayes 1200 Smartmodem:

```
hayes1200      =,-,      "" AT\r\c OK\r \EATDT\T\r\c CONNECT
```

Przy wybieraniu numeru znaki `=` i `-` zamieniane są na `,` (przecinek) oznaczający pauzę. Dalej następują polecenia inicjujące modem i wybierające numer. Ponieważ w pliku `Dialers` znajdują się gotowe wpisy dla większości popularnych modemów, nie będziemy szczegółowo omawiać tego zagadnienia.

Problem zezwalania na transfer plików jest nieco bardziej złożony niż w przypadku Taylor UUCP, ponieważ HDB UUCP pozwala na bardziej szczegółową konfigurację. W książkach poświęconych UUCP na temat ten poświęconych jest prawie sto stron – my skoncentrujemy się na niezbędnym minimum.

Dane dotyczące praw dostępu do systemu i plików zawarte są w pliku `/usr/lib/uucp/Permissions`. Format pojedynczego wpisu w takim pliku jest następujący:

```
MACHINE=nazwa_zdalna LOGNAME=uucp \
COMMANDS=rmail:rnews:uucp \
READ=/usr/spool/uucppublic:/usr/tmp \
WRITE=/usr/spool/uucppublic:/usr/tmp \
SENDFILES=yes REQUEST=no
```

Pole `MACHINE` określa nazwę komputera zdalnego. `LOGNAME` to identyfikator, za pomocą którego komputer ten (lub Twój komputer) loguje się do systemu, `COMMANDS` – poleceń, jakie może wykonać, `READ` i `WRITE` – lista katalogów z których może czytać i do których może zapisywać pliki, pole `SENDFILES` określa, czy dopuszcza się przesyłanie plików, a `REQUEST` – czy dozwolone jest żądanie przesłania plików z Twojego systemu. Ukośniki na końcu każdego wiersza służą do ukrycia znaku nowego wiersza – dzięki temu cały wpis jest traktowany jak jeden wiersz, a jednocześnie zachowana jest dobra czytelność tekstu. Jest to rozwiązanie typowe w systemach UNIX-owych.

Kompletny wpis dla komputera zdalnego o nazwie `wizard` może mieć następującą postać:

```
MACHINE=wizard LOGNAME=uucp1 \
COMMANDS=rmail:uucp \
READ=/usr/spool/uucppublic \
WRITE=/usr/spool/uucppublic \
SENDFILES=yes REQUEST=yes
```

Komputer ten może pobierać i wysyłać pliki do katalogu `/usr/spool/uucppublic`. Może uruchomić programy `mail` oraz `uucp`. Jeśli nie chcesz, by komputer zdalny przesyłał pliki do Twojego systemu, ustaw wartość `SENDFILES` na równą `no`. Podobnie, jeśli nie życzysz sobie, by jakiekolwiek pliki były pobierane z Twojego systemu, ustaw na `no` wartość `REQUEST`.

Połączenie UUCP

Nawiązanie połączenia UUCP składa się z kilku etapów. Łatwiej będzie zrozumieć rolę plików konfiguracyjnych, jeśli przyjrzymy się przebiegowi typowej sesji UUCP. Proces nawiązujący połączenie UUCP i obsługujący przesyłanie informacji nazywa się `uucico` (ang. *UUCP Call In/Call Out*). Połączenie może być zrealizowane przez uruchomienie `uucico` z nazwą zdalnego komputera podaną jako parametr:

```
uucico -s arthur
```

Przy uruchomieniu, program `uucico` szuka danych o systemie zdalnym w pliku `/usr/lib/uucp/sys` (Taylor UUCP) lub `/usr/lib/uucp/Systems` (HDB UUCP). Po odnalezieniu ich, odczytuje odpowiednie dane z innych plików konfiguracyjnych

(`/usr/lib/uucp/port` i `/usr/lib/uucp/dial` dla Taylor UUCP albo `/usr/lib/uucp/Devices` i `/usr/lib/uucp/Dialers` dla HDB UUCP), uruchamia połączenie modemowe i tworzy plik blokujący dostęp do portu szeregowego innym aplikacjom (sprawdzając najpierw, czy urządzenie nie jest używane przez inną aplikację – czyli czy nie istnieje plik LCK, np. `LCK..cua0` dla pierwszego modemu).

Po wykonaniu skryptu chat inicjalizującego modem i wybierającego numer, do logowania do komputera zdalnego używany jest skrypt chat zapisany w pliku `/usr/lib/uucp/sys` lub `/usr/lib/uucp/System`. Po zakończeniu logowania, w systemie zdalnym uruchamiany jest również program uucico, po czym oba programy wymieniają potwierdzenia i rozpoznają się transfer danych.

Po zakończeniu sesji komputer lokalny upewnia się, że komputer zdalny nie ma nic więcej do przesyłania i przerzuca połączenie. Program uucico kończy działanie.

Komunikacja bezpośrednia

Jeśli dwa komputery połączone są bezpośrednio przez port szeregowy, do przesyłania danych również można użyć protokołu UUCP. Jedyna zmiana w opisany wyżej procesie konfiguracji dotyczy konfiguracji portu. Zamiast używać urządzenia modemu, użyć należy połączenia bezpośredniego. Przykładowo, dla systemu Taylor UUCP wpis w pliku `/usr/lib/uucp/sys` będzie zawierał wiersz:

```
port      direct1
```

a w pliku `/usr/lib/uucp/port` znajdzie się wpis:

```
port      direct1
type     direct
speed    38400
device   /dev/cua1
```

Prędkość transmisji i nazwę portu szeregowego należy dostosować do rzeczywistych wartości używanych w systemie. Podobnie rzecz wygląda w przypadku HDB UUCP, z tym że odpowiednie wpisy powinny znaleźć się w plikach `/usr/lib/uucp/Systems` i `/usr/lib/uucp/Devices`.

Skrypty logowania

Skrypty logowania są częścią pliku `/usr/lib/uucp/sys` lub `/usr/lib/uucp/System`; często okazują się one najtrudniejszym krokiem konfiguracji połączenia. Jeśli łączysz się z typowym systemem UNIX-owym, wysyłane powinny być tylko standardowe zachęty do wprowadzenia identyfikatora i hasła, ale w niektórych systemach nie jest to takie proste. Z tego powodu przyjrzymy się temu problemowi nieco dokładniej.

Ogólnie rzecz biorąc, skrypt logujący składa się z par wzorzec-akcja. Wzorzec wyszukiwany jest wśród znaków wysyłanych przez drugi koniec połączenia, natomiast akcja określa tekst, który ma zostać wysłany przez komputer lokalny. Prosty skrypt może wyglądać tak:

```
login: merlin password: secret1
```

W takim przypadku system czeka na odebranie tekstu `login:`, następnie wysyła tekst `merlin`, czeka na tekst `password:` i wysyła `secret1`. Skrypt można troszkę skrócić, zdając sobie sprawę, że w zasadzie interesuje nas tylko moment zakończenia komunikatu, na który czekamy:

```
gin: merlin word: secret1
```

Takie rozwiązanie ma też inne zalety, na przykład jeśli system zdalny zamiast `login:` wysyła tekst `Login:`, skrócona forma – w przeciwieństwie do formy pełnej – zareaguje prawidłowo.

Użyteczna jest możliwość wymuszenia na maszynie zdalnej zrestartowania procesu `getty`. W tym celu należy wstawić myślnik i słowo `BREAK`, co powoduje wysłanie sekwencji resetującej połączenie po upływie określonego czasu, na przykład tak:

```
ogin: -BREAK-ogin: merlin sword: secret1
```

W takim przypadku, jeśli komputer zdalny nie wyśle zachęty `login:`, po pewnym czasie Twój komputer wysyła sekwencję resetującą połączenie i ponownie czeka na zachęte.

Można używać również kilku znaków specjalnych. Oto znaczenie najważniejszych z nich.

- `\c` Zapobiega wysyłaniu znaku powrotu karetki (tylko przy wysyłaniu).
- `\d` Opóźnienie o 1 sekundę (tylko przy wysyłaniu).
- `\p` Pauza trwająca ułamek sekundy (tylko przy wysyłaniu).
- `\t` Wysłanie znaku tabulacji.
- `\r` Wysłanie znaku powrotu karetki.
- `\s` Wysłanie spacji.
- `\n` Wysłania znaku nowego wiersza.
- `\` Wysłanie znaku `\`.

Czasem trzeba wysłać jeden lub więcej z podanych wyżej znaków, zanim komputer zdalny zechce się odezwać. Przykładowo, poniższy skrypt wysyła przed próbą nawiązania kontaktu kombinację powrót karetki + znak nowego wiersza:

```
\n\r\p ogin: merlin word:secret1
```

To zwykle wystarcza, by w systemie zdalnym uruchomiony został proces getty obsługujący odpowiedni port.

Modyfikowanie harmonogramu dostępu

Zarówno Taylor, jak i HDB UUCP pozwalają na podanie, kiedy można łączyć się z danym systemem. W przykładach podanych wcześniej nie było żadnych ograniczeń czasowych (w odpowiednim polu występowało słowo `Any`), ale takie ograniczenia mogą być bardzo użyteczne - czy to ze względu na różne taryfy telefoniczne, czy też po prostu na ograniczony czas działania komputera po drugiej stronie połączenia.

Jeśli chcesz zezwolić na połączenia w poszczególne dni tygodnia, użyj dwuliterowych skrótów ich angielskich nazw (`Mo` – poniedziałek i dalej `Tu`, `We`, `Th`, `Fr`, `Sa`, `Su`), `wk` dla dni roboczych, `Any` dla dostępu bez ograniczeń i `Never` dla zablokowania dostępu. Dozwolone są również kombinacje tych wartości. Godziny dostępu podaje się w formie zakresu, w formacie 24-godzinnym. Jeśli nie zostaną określone godziny, w których można się połączyć, połączenie jest możliwe przez cały dzień.

Daty i godziny podaje się nie oddzielając ich spacjami, natomiast poszczególne wpisy rozdziela się przecinkami. Oto przykładowe definicje ograniczeń, prawidłowe zarówno w systemie Taylor, jak i HDB:

Wk1800-0730	dostęp w dni robocze od 18.00 do 7.30;
MoWeFr	dostęp w poniedziałki, środy i piątki;
Wk2300-2400, SaSu	dostęp w dni robocze od 23.00 do 24.00 i w weekendy przez całą dobę.

UUCP a bezpieczeństwo

Prawa dostępu do plików konfiguracyjnych UUCP powinny być uważnie ustawione, tak aby protokół mógł funkcjonować prawidłowo i nie zagrażał bezpieczeństwu systemu. Najprościej można zapewnić to w ten sposób, że wszystkie pliki konfiguracyjne powinny być własnością użytkownika `uucp`, grupy `uucp`. W tym celu w katalogu `/usr/lib/uucp` należy wydać polecenia:

```
chown uucp *
chgrp uucp *
```

Ze względów bezpieczeństwa dla użytkownika `uucp` powinieneś założyć jakieś dobre hasło. Niektóre wersje Linuxa pozostawiają domyślnie to hasło puste, co powoduje, że system stoi otworem dla każdego, komu tylko przyjdzie do głowy zalogować się jako `uucp`.

Prawa dostępu do plików powinny umożliwiać czytanie i zapisywanie (i wykonywanie dla katalogów) tylko dla ich właściciela. Prawo odczytu dla kogokolwiek innego może dać mu informacje o hasłach pochodzące z plików konfiguracyjnych.

Kiedy program obsługuje UUCP loguje się do systemu zdalnego, musi podać identyfikator użytkownika i hasło. Dane te są przechowywane w plikach `/usr/lib/uucp/sys` lub `/usr/lib/uucp/Systems`, powinny więc być zabezpieczone przed nieautoryzowanym dostępem poprzez ustalenie odpowiednich praw dostępu.

Jeśli kilka innych systemów łączy się z Twoim, możesz dla wszystkich utworzyć jedno wspólne konto i hasło, można również zrobić to dla każdego systemu zdalnego z osobna – w tym celu wystarczy dodać dla każdego systemu wpis do pliku `/etc/passwd` i zdefiniować dla niego (za pomocą programu `passwd`) odpowiednie hasło. System zdalny będzie mógł zalogować się używając takiego konta. Na użytkownikach korzystających z protokołu UUCP należy wymusić użycie tylko programu `uucico`. Odpowiedni wpis w pliku `/etc/passwd` może wyglądać tak:

```
uucp1:123:52:UUCP Login for  
Arthur:/usr/spool/uucppublic:usr/lib/uucp/uucico
```

Katalogiem domowym użytkownika `uucp1` w powyższym przykładzie jest `/usr/spool/uucppublic`, a po zalogowaniu uruchamiany jest program `uucico`. Jeśli utworzysz osobne konto dla każdego łączącego się z Twoim systemu, możesz dokładniej kontrolować prawa dostępu.

Należy uważnie sprawdzić, jakie polecenia może wykonać dany użytkownik. Odpowiednie ustawienia znajdują się w plikach konfiguracyjnych UUCP. Jeśli pliki są przesyłane dalej przez Twój system, powinieneś wiedzieć kto je przesyła i dokąd trafiają.

Najważniejszą jednak sprawą jest zapewnienie, aby z systemu mogli korzystać tylko zaufani użytkownicy. Nie otwieraj dostępu dla wszystkich, bo jest to proszenie się o kłopoty. Regularnie sprawdzaj dane konfiguracyjne i prawa dostępu do plików zawierających newralgiczne informacje.

Używanie UUCP

Po skonfigurowaniu systemu UUCP można używać go do przesyłania plików i poczty. Najpierw trzeba jednak poznać składnię adresów, która nieco różni się od używanej po-wszechnie w Internecie:

```
komputer!cel
```

`komputer` to nazwa komputera zdalnego, natomiast `cel` to nazwa pliku, do którego chcesz się dobrać, bądź identyfikator użytkownika. Na przykład, aby wysłać pocztę do użytkownika `yvonne` w systemie `arthur`, można użyć polecenia `mail` w następujący sposób:

```
mail arthur!yvonne
```

UUCP pozwala również na przejście przez kilka komputerów, zanim plik trafi do adresata. Można dzięki temu zaoszczędzić na rachunkach telefonicznych lub też dostać się do większej sieci za pomocą jednego tylko połączenia. Założymy, że chcesz wysłać pocztę do użytkownika `bill` w systemie `warlock`, ale nie masz z nim połączenia. Masz za to połączenie z systemem `arthur`, który jest połączony z systemem `warlock` i skonfigurowany do przesyłania poczty od Ciebie. W takim przypadku możesz wysłać pocztę poleciением:

```
mail arthur!warlock!bill
```

Podczas dekodowania adresu odczytywana jest tylko pierwsza część (`arthur`), pozostała jest przesyłana razem z pocztą do systemu `arthur`. W tym systemie ponownie następuje odczytanie adresu – teraz adresatem jest system `warlock`; jeśli dozwolone jest pośrednictwo w przekazywaniu poczty, wiadomość zostanie przesłana do tego systemu. Droga przesyłania wiadomości może być bardziej skomplikowana, na przykład poniższy adres powoduje przesłanie wiadomości do użytkownika `alex` w systemie `vader` za pośrednictwem systemów `arthur`, `warlock` i `chatton`:

```
mail arthur!warlock!chatton!vader!alex
```

W skomplikowanych przypadkach trudno jednak nie pomylić kolejności, w jakiej trzeba podać nazwy komputerów. Możliwość taka może być dość użyteczna w przypadku, gdy pewna liczba użytkowników posiada lokalne połączenia pomiędzy komputerami – dzięki temu możliwe jest skonfigurowanie nawet dość złożonej sieci (symbol wykryznika w adresie nazywany jest po angielsku *bang*, cały adres może więc zostać odczytany jako `arthur-bang-warlock-bang-chatton-bang-vader-bang-alex`).



Niektóre interpreterы poleceń nie akceptują znaku `!`, ponieważ jest on zarezerwowany do innych celów - na przykład powłoka C używa go do przywoływania poprzednio wydanego polecenia. W takim przypadku należy zabronić jego interpretacji, na przykład poprzedzając go symbolem `\`:

```
mail arthur\!warlock\!bill
```

W zależności od tego, jak skonfigurowany jest system, może on albo od razu po wysłaniu wiadomości łączyć się z innym systemem, albo wstawać wiadomości do kolejki oczekującej na wysłanie przy najbliższym połączeniu, szczególnie jeśli połączenia są dozwolone tylko w określonych godzinach. Dane dotyczące ograniczania czasu połączeń zapisane są w pliku `/usr/lib/uucp/sys` lub `/usr/lib/uucp/Systems`.

Musisz pamiętać o tym, że jeśli systemy, przez które przesyłasz dane, nie łączą się od razu z następnym systemem, proces przesyłania wiadomości może się znacznie przedłużyć. Jeśli na przykład komputer łączy się z następnym systemem raz na dobę, w najgorszym przypadku może wprowadzić nawet 24-godzinne opóźnienie. Każdy następny komputer pośredniczący powoduje dodatkowe opóźnienie.



Nie możesz założyć, że dane przesyłane przez UUCP będą poufne. W systemie zdalnym może je przeglądać każdy, kto ma dostęp do katalogu UUCP. Nie w każdym systemie prawa dostępu są ustalone prawidłowo. Jeśli musisz przesłać poufne dane, najlepiej je zakoduj, podając adresatowi odpowiedni klucz (oczywiście nie za pomocą poczty elektronicznej).

W systemie UUCP wszystkie zlecenia transferu traktowane są jako *zadania* (ang. *jobs*) – z tym terminem możesz spotkać się dość często, przeglądając dokumentację dotyczącą UUCP. *Zadanie* to polecenie, które ma zostać wykonane w systemie zdalnym, plik, który ma zostać przesłany, czy też dowolne inne dane, które mają zostać przesłane pomiędzy dwoma systemami.

Przesyłanie poczty

Większość programów do obsługi poczty akceptuje adresy w formacie UUCP bez żadnych dodatkowych zabiegów konfiguracyjnych, co upraszcza znaczco używanie tego protokołu. W poprzednim podrozdziale przedstawiliśmy przykład użycia polecenia `mail` z adresem w formacie UUCP.

Można wykorzystać wszystkie opcje modyfikujące zachowanie polecenia `mail`. Dla przykładu spróbujmy za pomocą tego polecenia przesłać zawartość pliku `dane_1` do użytkownika `yvonne` w systemie `chatton` przez system `arthur`, dołączając nagłówek z tematem. W tym celu należy wydać polecenie:

```
mail -s "Plik z danymi" arthur\!chatton\!yvonne < dane_1
```

Praktycznie wszystkie programy pocztowe, również te pracujące w systemie X, akceptują oprócz zwykłych adresów internetowych adresy w formacie UUCP, ale warto sprawdzić to w dokumentacji przed zainstalowaniem nowego pakietu pocztowego.

Przesyłanie plików

Do przesyłania plików za pomocą protokołu UUCP służy polecenie `uucp`, którego składnia jest następująca:

```
uucp [opcje] źródło cel
```

Opcje obsługiwane przez to polecenie różnią się nieco w zależności od wersji i typu systemu UUCP, ale większość wersji obsługuje opcje zebrane poniżej.

- c Zapobiega kopiowaniu pliku do kolejki przed wysłaniem. Domyslnie plik, który ma zostać przesłany, jest kopowany do katalogu kolejki. Za pomocą opcji `-c` można określić położenie tego katalogu.
- f Zapobiega odtwarzaniu struktury katalogów w systemie docelowym. Domyslnie struktura katalogów jest odtwarzana, można również zażądać tego explicit, podając opcję `-d`.

-m Wysyła wiadomość do osoby, która wydała polecenie `uucp`, potwierdzającą zakończenie kopiowania danych.

-nuzytkownik Wysyła po zakończeniu kopiowania wiadomość do użytkownika w systemie zdalnym.

Ustawienia domyślne odpowiadają większości użytkowników, choć opcja `-m` może być przydatna, jeśli chcesz otrzymać potwierdzenie przesłania danych.

Parametry `źródło` i `cel` określają nazwy odpowiednich plików lub katalogów, analogicznie jak w przypadku polecenia `cp`. Trzeba jednak pamiętać, że jeśli nazwa odnosi się do systemu zdalnego, musi posiadać prawidłowy format UUCP. Przykładowo, jeśli chcesz przesyłać plik `dane_1` z bieżącego katalogu do katalogu `/usr/spool/uucppublic` w systemie `arthur`, powinieneś wydać polecenie:

```
uucp dane_1 arthur\!/usr/spool/uucppublic
```

Zauważ, że przed ścieżką dostępu do katalogu w systemie zdalnym pojawiła się nazwa tego systemu. Katalog `/usr/spool/uucppublic` jest zwykle jedynym katalogiem, do którego można przesyłać dane. Adresat danych musi sam zadbać o skopiowanie ich do odpowiedniego katalogu.

Jeśli chcesz wysłać plik do użytkownika `bill`, umieszczając go w osobnym podkatalogu, i powiadomić go o tym fakcie oraz otrzymać potwierdzenie wysłania pliku, powinieneś wydać polecenie:

```
uucp -m -nbill dane_1 arthur\!/usr/spool/uucppublic/bill/
```

Jeśli chcesz skopiować plik z komputera zdalnego, musisz znać jego położenie i mieć dostęp do katalogu i prawo odczytu tego pliku (lub też ktoś musi skopiować dla Ciebie ten plik do katalogu `uucppublic`). Poniższe polecenie skopiuje plik `bigfile` z katalogu `/usr/tmp` w systemie `chatton` do lokalnego katalogu `/home/reksio`:

```
uucp chatton\!/usr/tmp/bigfile /home/reksio
```

UUCP pozwala na użycie symboli wieloznacznych, ale należy otoczyć je cudzysłowami (żeby zapobiec ich interpretacji przez interpreter polecen powłoki). Aby skopiować wszystkie pliki, których nazwa zaczyna się od liter `chap`, zapisane w katalogu `/usr/bill/book` (zakładając, że masz odpowiednie prawa dostępu) do lokalnego katalogu `/usr/bigbook`, wydaj polecenie:

```
uucp "warlock\!/usr/bill/book/chap*" /usr/bigbook
```

Możliwe jest także przesyłanie plików za pośrednictwem innych systemów, co wymaga posiadania odpowiednich praw dostępu we wszystkich systemach pośredniczących. Można również przesyłać pliki z jednego systemu do innego (co prawda trudno znaleźć zastosowanie dla tej sztuczki), na przykład tak:

```
uucp arthur\!/usr/lib/uucppublic/bigfile warlock\!/usr/lib/uucppublic/
```

Powyższe polecenie powoduje przesyłanie pliku `bigfile` z systemu `arthur` do systemu `warlock`. Prostszym rozwiązaniem byłoby wydanie odpowiedniego polecenia w systemie `arthur` lub `warlock` – nie występują wówczas problemy z konfiguracją praw dostępu.

Sprawdzanie transferu

Z pomocą polecenia `uustat` można sprawdzić, jaki jest aktualny status transferu, który został „zamówiony”, ale nie dotarł jeszcze na miejsce. Po jego wydaniu pojawi się lista wszystkich wydanych przez Ciebie zleceń kopiowania o następującym formacie:

```
ID zadania system użytkownik data polecenie rozmiar
```

system jest nazwą pierwszego komputera, do którego dane są wysyłane (czyli nie zawsze jest to nazwa systemu docelowego), użytkownik to identyfikator osoby, która wydała zlecenie kopiowania, pole data określa, kiedy dane zadanie zostało zlecone, polecenie to dokładna treść polecenia, które zostanie wykonane po nawiązaniu połączenia, a rozmiar to sumaryczna wielkość danych, które zostaną przesłane.

Jeżeli wydasz polecenie `uustat` jako zwykły użytkownik, otrzymasz informacje tylko o tych zadaniach, które sam zleciłeś. Jeśli polecenie to wydasz jako `root`, wyświetlane zostaną dane o wszystkich czekających na skopiowanie plikach. Jako zwykły użytkownik również możesz uzyskać informacje o wszystkich zleceniach, używając opcji `-a`:

```
uustat -a
```

Aby anulować wysłanie jakiegoś pliku, użyj opcji `-k` polecenia `uustat` wraz z identyfikatorem zadania; na przykład, aby anulować wysłanie pliku, któremu został przyznany numer identyfikacyjny 17, należy wydać polecenie:

```
uustat -k 17
```

Możesz anulować wysłanie tylko tych plików, które sam zleciłeś (chyba że jesteś zalogowany jako administrator).

Podsumowanie

Protokół UUCP jest dość łatwy do skonfigurowania. Oferuje on prosty sposób przesyłania poczty czy plików do innych systemów. Jest to jedna z najłatwiejszych metod komunikacji w niewielkich sieciach, ponieważ do nawiązania połączenia wystarcza połączenie modemowe lub bezpośrednie.

Choć popularność protokołu UUCP spada dzięki rozwojowi sieci LAN, nadal jest on rozwiązaniem najtańszym i dobrym dla kogoś, kto potrzebuje połączyć tylko kilka komputerów. Jest również świetnym sposobem na połączenie się z systemami przyjaciół, pozwalającym na przykład na wygodne przesyłanie poczty.

O tym, jak skonfigurować Linuxa aby możliwe było korzystanie z poczty elektronicznej, przeczytać możesz w rozdziale 40. „Konfigurowanie poczty”.

Używanie grup dyskusyjnych omówione jest w rozdziale 41. „Konfigurowanie grup dyskusyjnych”.

Metody zabezpieczania systemu przed włamaniem są tematem rozdziału 42. „Bezpieczeństwo w sieci”.

Rozdział 45. „Kopie zapasowe”, omawia bardzo istotną problematykę tworzenia kopii zapasowych.

Rozdział 40.

Konfigurowanie poczty

Tim Parker

W tym rozdziale:

- υ Jak działa poczta elektroniczna
- υ Konfigurowanie programu `sendmail`
- υ Używanie programu `sendmail` w wersji 8
- υ `smail`
- υ Modyfikacja zachowania `smail`

System obsługi poczty elektronicznej w Linuxie składa się z dwóch warstw: agenta poczty, czyli programu, którego używasz do wysyłania i odbierania poczty (ang. MUA, Mail User Agent), i warstwy systemowej (ang. MTA, Mail Transport Agent), która obsługuje procesy wysyłania i nadawania wiadomości.

Najpopularniejsze w systemach linuxowych programy typu MTA to `sendmail` i `smail`. Programów typu MUA są dziesiątki i tylko do Ciebie należy wybór tego najodpowiedniejszego. Program `sendmail` oparty jest na systemie poczty opracowanym w Uniwersytecie Kalifornijskim w Berkley. Dostępnych jest kilka jego wersji, różniących się nieco możliwościami. Drugi dość powszechnie używany program o nazwie `smail` opracowany został przez Curta Nolla i Ronaldą Karra. Również ten program jest dostępny w kilku wersjach.

Programy `sendmail` i `smail` rozprowadzane są wraz z większością dystrybucji Linuxa, co może nieco zakłopotać użytkownika nie wiedzącego, na który system powinien się zdecydować. W przypadku niewielkich systemów (jakimi zwykle są komputery linuxowe, również te podłączone do sieci) oba te programy działają równie dobrze. Pod pewnymi względami `smail` jest nieco łatwiejszy w konfigurowaniu, głównie dlatego, że jest nowszy. Mimo to `sendmail` jest bardziej elastyczny i lepiej sprawdza się w większych systemach.

Jak działa poczta elektroniczna

Kiedy piszesz list czy wiadomość za pomocą jednego z programów do obsługi poczty, takiego jak np. Elm, Pine czy mail, aplikacja przekazuje jego treść do programu MTA, takiego jak sendmail lub smail. W systemie może działać kilka takich programów (na przykład jeden do obsługi sieci lokalnej i drugi do UUCP), ale zwykle dla wygody używa się tylko jednego. Ścisłe rzecz ujmując, wiadomość nie jest przekazywana bezpośrednio do programu MTA, ale do programu rmail (nazwa ta jest zwykle tylko aliasem nazwy jednego z tych programów).

Jeśli wiadomość adresowana jest do kogoś w sieci lokalnej (lub w tym samym systemie), MTA powinien to zauważać na podstawie adresu. MTA musi również rozpoznawać i tłumaczyć aliasy, dzięki którym możliwe jest adresowanie sieci, systemów czy użytkowników za pomocą różnych identyfikatorów. Jeśli wiadomość jest przeznaczona dla użytkownika innej sieci, MTA musi umieć nawiązać połączenie z komputerem, który przekaże pocztę dalej. Połączenie to może być oparte zarówno na TCP, jak i UUCP. Jeśli połączenie realizowane jest za pomocą TCP, często używa się protokołu SMTP (Simple Mail Transfer Protocol). MTA musi również radzić sobie z takimi problemami, jak niemożność dostarczenia poczty z powodu błędного adresu komputera czy użytkownika – w takim przypadku poczta musi być albo zignorowana, albo lepiej odesłana do nadawcy.

Routing (czyli kierowanie drogą dostarczenia) poczty jest również ważnym aspektem pracy programów MTA i różni się w zależności od użytego schematu adresowania. Jeśli adres jest oparty na protokole TCP (czyli ma postać nazwy domenowej), MTA próbuje dostarczyć pocztę bezpośrednio do komputera docelowego w oparciu o adres IP, pozostawiając routing oprogramowaniu sieciowemu wchodzącemu w skład systemu TCP/IP.

Konfigurowanie programu sendmail

Najczęściej używanym programem typu MTA jest sendmail, rozpowszechniany z większością dystrybucji Linuxa. Jest to system niebyvale potężny i elastyczny, co niestety powoduje, że nie jest najłatwiejszy w konfiguracji i utrzymaniu. Mimo to skonfigurowanie go do prostych zadań jest całkiem łatwe, o czym przekonasz się, czytając ten rozdział. Jeśli wybrałeś właśnie ten program, zawarte tu informacje powinny wystarczyć dla skonfigurowania systemu poczty w dowolnej sieci (może za wyjątkiem najbardziej skomplikowanych).

Ze względu na swoją złożoność sendmail jest często dostarczany razem z programem użytkowym o nazwie IDA pod wspólną nazwą sendmail+IDA. Program IDA znacznie upraszcza konfigurację programu sendmail, dlatego jest bardzo często używany w systemach linuxowych. Dzięki niemu sendmail staje się najprostszym w użyciu programem do obsługi transportu poczty.



Jeśli w Twoim systemie zainstalowany jest tylko program `sendmail`, warto rozważyć załadowanie pakietu `sendmail+IDA` z któregoś z węzłów FTP lub BBS. Oferowane przez ten pakiet udoskonienia z nawiązką wynagrodzą trud włożony w załadowanie odpowiednich plików. Wiele dystrybucji oferuje program `sendmail` w wersji 8, do którego zwykle nie jest dodawany program `IDA`. Informacji o najświeższych wersjach programu `sendmail` i `IDA` szukaj w węzłach BBS i FTP.

System `sendmail` (bez programu `IDA`) przechowuje większość informacji konfiguracyjnych w pliku `/etc/sendmail.cf` (lub, w niektórych systemach, `/usr/lib/sendmail.cf`). Język używany w tym pliku różni się zupełnie od innych plików konfiguracyjnych i jest stosunkowo złożony. Zresztą obejrzyj ten plik za pomocą programu `more` i sam spróbuj domyślić się, o co w nim chodzi.

Plik `sendmail.cf` zawiera dane o czynnościach podejmowanych domyślnie przez system `sendmail`. Inne pliki służące do konfiguracji tego programu to:

- υ **decnexttable:** zamienia ogólne adresy na adresy DECnet;
- υ **genericfrom:** zamienia adresy wewnętrzne na ogólne;
- υ **mailertable:** określa reguły specjalnego traktowania serwerów i domen;
- υ **pathtable:** definiuje ścieżki UUCP do zdalnych komputerów i domen;
- υ **uucpxtable:** wymusza dostarczanie poczty z adresem DNS za pomocą UUCP;
- υ **uucprelays:** umożliwia „skróty” do komputerów zdalnych;
- υ **xaliases:** zamienia adresy ogólne na wewnętrzne.

Każdemu z tych plików przyjrzymy się nieco bardziej szczegółowo dalszej części tego rozdziału. Jak już wspomniano, pliki te trudno modyfikować ręcznie. Użycie programu `IDA` znacznie poprawia sytuację, ponieważ umożliwia on skonfigurowanie systemu `sendmail` przez podanie odpowiednich opcji.

System `sendmail+IDA` za pomocą preprocesora takiego jak `m4` lub `dbm` generuje na podstawie wprowadzonych danych odpowiednie pliki konfiguracyjne.

Plik `sendmail.cf`

W przypadku użycia programu `sendmail+IDA`, plik `sendmail.cf` nie jest edytowany ręcznie. Jest on generowany po zakończeniu konfiguracji. Cała procedura konfiguracyjna opiera się na danych zapisanych w pliku `sendmail.m4`, zawierającym podstawowe takie informacje, jak nazwa systemu, specyficzne ścieżki dostępu, nazwa domyślnego programu obsługi poczty itp. Choć plik ten może stać się dość długi, w większości in-

stalacji używających do przesyłania poczty protokołów UUCP lub SMTP wystarczy wprowadzić do niego tylko podstawowe informacje.

Jedną z ważniejszych sekcji w pliku `sendmail.m4` jest sekcja określająca położenie plików i katalogów. Zwykle zaczyna się ona od wiersza definiującego zmienną `LIBDIR`, który może mieć następującą postać:

```
dnl #define(LIBDIR, /usr/local/lib/mail)
```

Katalog `LIBDIR` określa miejsce, gdzie `sendmail+IDA` szuka plików konfiguracyjnych i tablic kierowania przepływem danych (ang. *routing tables*). Zwykle modyfikacja wartości domyślnej nie jest konieczna, ponieważ jest ona ogólnie przyjęta w systemach linuxowych. Jeżeli ścieżka zapisana w pliku `sendmail.m4` jest prawidłowa, nie należy jej zmieniać. Jest ona zwykle również na stałe zapisana w pliku wykonywalnym programu `sendmail` i nie musi być nadpisywana przez plik `sendmail.m4` (czy generowany na jego podstawie plik `sendmail.cf`). Jeśli musisz zmienić tę wartość, powinieneś usunąć z początku wiersza tekst `dnl`, który jest znacznikiem komentarza, wprowadzić odpowiednią dla Twojego systemu ścieżkę i ponownie wygenerować plik `sendmail.cf`.

Program dostarczający pocztę jest określony wartością zmiennej `LOCAL_MAILER_DEF`, której definicja może mieć postać:

```
define (LOCAL_MAILER_DEF, mailers.linux)dnl
```

Wiersz ten jest potrzebny, ponieważ `sendmail` nie zajmuje się dostarczaniem poczty. Dba o to inny program. Domyślnie używany jest program zdefiniowany w pliku `mailers.linux`, którym prawie zawsze jest `deliver`. Powinieneś jednak na wszelki wypadek sprawdzić, co zawiera plik `mailers.linux` (zapisany w tym samym katalogu, co plik `sendmail.m4`, czyli zwykle `/usr/local/lib/mail`). Typowo jego zawartość jest następująca:

```
#mailers.linux
Mlocal, P=/usr/bin/deliver, F=S1smFDMP, S=10, R=25/10, A=deliver $u
Mprog, P=/bin/sh, F=lsDFMeuP, S=10, A=sh -c $u
```

Nazwa programu dostarczającego pocztę musi być również wprowadzona do pliku `Sendmail.mc`, który jest używany do generowania pliku `sendmail.cf`. Jeśli używasz programu innego niż `deliver`, powinieneś również tam skorygować odpowiedni wpis (jeśli używasz programu `deliver`, nie musisz przejmować się zawartością tego pliku). Plik `Sendmail.mc` jest wczytywany przy przetwarzaniu pliku `sendmail.m4`, za co odpowiada znajdujący się zwykle na początku pliku `sendmail.m4` wiersz

```
include (Sendmail.mc)dnl
```

Możliwe, że będziesz musiał dodać kilka wartości do definicji zmiennej `PSEUDODOMAINS`, używanej do obsługi systemów nie posiadających nazwy domenowej, na przykład dostępnych przez UUCP. Odpowiednie ustawienie wartości tej zmiennej zapobiega próbom tłumaczenia tych nazw przez DNS (co zawsze kończy się niepowodzeniem). Odpowiedni wiersz może mieć następującą postać:

```
define (PSEUDODOMAINS, BITNET UUCP)dnl
```

Można również użyć zmiennej `PSEUDONYMS`, pozwalającej na ukrycie nazwy Twojego komputera przed światem zewnętrznym. Oznacza to np. że bez względu na to, czy poczta została wysłana z systemu `merlin.tpci.com` czy `chatton.tpci.com`, odbiorca otrzyma tylko adres `tpci.com` – nazwy komputerów lokalnych zostały ukryte. Jeśli wartość tej zmiennej jest zdefiniowana, `sendmail` przyjmuje pocztę od wszystkich maszyn w sieci lokalnej, np. definicja:

```
define(PSEUDONYMS, tpci.com)dnl
```

pozwala wszystkim komputerom w sieci `tpci.com` wysyłać pocztę za pośrednictwem systemu `sendmail`.

Aby określić nazwę komputera lokalnego, należy odpowiednio ustawić wartość zmiennej `DEFAULT_HOST`. Zwykle nazwa ta pokrywa się z nazwą serwera poczty (lub Twojego komputera, jeśli nie jesteś podłączony do sieci). Poniższy wiersz określa nazwę domyślnego serwera poczty:

```
define(DEFAULT_HOST, merlin.tpci.com)dnl
```

Jeśli nie ustawisz wartości tej zmiennej, poczta nie będzie prawidłowo zwracana do Twojego systemu.

Jeśli system nie pracuje jako bramka internetowa (albo do innej sieci dostępnej z sieci lokalnej), można skonfigurować go tak, by cała poczta była przesyłana do innego komputera, który powinien rozesłać ją dalej. Zrobić to można ustawiając wartości zmiennych `RELAY_HOST` (zmienna ta określa nazwę komputera, do którego ma być przekazywana poczta) i `RELAY_MAILER` (określa protokół używany do transportu poczty do tego komputera). Aby na przykład przesyłać pocztę do systemu o nazwie `wizard`, można zdefiniować wartości tych zmiennych w następujący sposób:

```
define(RELAY_HOST, wizard)dnl  
define(RELAY_MAILER, UUCP-A)dnl
```

Położenie tablic konfiguracyjnych

W pliku `sendmail.m4` znajdują się wiersze określające położenie tablic konfiguracyjnych. Zwykle znajdują się one w katalogu określonym przez wartość zmiennej `LIBDIR`. Ta sekcja pliku może wyglądać tak:

```
define(ALIASES, LIBDIR/aliases)dnl  
define(DOMAINTABLE, LIBDIR/domaintable)dnl
```

i tak dalej dla pozostałych plików (zwykle jest ich około siedmiu). Możesz zmienić te wartości, jeśli chcesz przenieść odpowiednie pliki w inne miejsce. Najlepiej jednak pozostawić je tam, gdzie są.

Plik `decnexttable` używany jest do tłumaczenia nazw domenowych na nazwy w standardzie DECnet. Jest to pozostałość po wcześniejszych wersjach programu `sendmail` i prawdopodobnie nigdy nie przyda się użytkownikowi systemu linuxowego (chyba że pracuje on w systemie DECnet).

Plik `domaintable` używany jest do wymuszania wykonania pewnych poleceń po użyciu DNS. Plik ten, prawie nigdy nie używany w Linuxie, pozwala na rozwijanie nazw skróconych. Założymy na przykład, że często wysyłasz pocztę do komputera o nazwie `okropnie_dluga_nazwa.w_okropnie_duzej_sieci.com` i nie masz ochoty wpisywać jej za każdym razem. Możesz umieścić w pliku `domaintable` wpis

```
okropnie_dluga_nazwa.w_okropnie_duzej_sieci.com      dlugi.com
```

dzięki któremu każdy adres postaci `bill@dlugi.com` zostanie przetłumaczony do formy `bill@okropnie_dluga_nazwa.w_okropnie_duzej_sieci.com`. Wpis w tym pliku może również być użyteczny do poprawiania często popełnianych błędów typograficznych, na przykład jeśli użytkownicy często przez pomyłkę próbują wysłać pocztę do systemu `abcdef.com`, którego poprawną nazwą jest `abcdfe.com`, możesz do tego pliku dodać wiersz

```
abcdef.com      abcdef.com
```

Pierwszą wartością jest prawidłowa nazwa systemu, natomiast druga wartość określa nazwę skróconą lub często używaną nazwę nieprawidłową.

Tabela `genericfrom` jest używana do ukrywania nazw komputerów lokalnych i identyfikatorów użytkowników, które są zamieniane na jakiś ogólniejszy identyfikator. Jest ona rzadko używana w systemach linuxowych, ponieważ ogólnie przyjętą konwencją jest podawanie w wysyłanych wiadomościach prawdziwego identyfikatora i adresu. Plik komplementarny, `xaliases`, tłumaczy ogólne adresy na adresy lokalne.

Tablica `mailertable` używana jest do określania reguł specjalnego traktowania serwerów i domen. Najczęściej stosuje się ją do określenia, za pomocą jakiego protokołu można połączyć się z poszczególnymi domenami. Plik ten nie musi być modyfikowany, jeśli system korzysta tylko z protokołu UUCP, ale jeśli chcesz używać również SMTP lub DNS, powinieneś sprawdzić jego zawartość.

Plik `mailertable` jest przetwarzany od ostatniego wiersza w góre. Z tego powodu najczęściej używane reguły należy umieścić na końcu pliku, a bardziej specyficzne na początku. Reguły określone są w formacie:

```
protokół      modyfikator      adres_pośredni adres_docelowy
```

Protokół definiowany na początku wiersza może przyjąć jedną z trzech wartości:

TCP-A TCP z adresami w formacie internetowym

TCP-U TCP z adresami w formacie UUCP

UUCP-A UUCP z adresami w formacie internetowym

`modyfikator` może być jednym ze znaków:

! usuwa nazwę komputera z adresu przed przesaniem wiadomości (używane przy adresowaniu w formacie UUCP)

, nie modyfikuje adresu

- : usuwa nazwę komputera tylko wtedy, gdy podane są nazwy komputerów pośredniczących.

Jeśli na przykład chcesz spowodować, aby poczta do systemu `roy.sailing.org` była przesyłana przez system `wizard` za pomocą protokołu UUCP, powinieneś dodać do pliku `mailertable` wiersz:

```
UUCP-A,wizard roy.sailing.org
```

Można również zdefiniować bardziej ogólne reguły, na przykład wpis

```
TCP-A, wizard chatton.com
```

powoduje, że poczta przesyłana do sieci `chatton.com` będzie obsługiwana przez lokalny serwer poczty za pomocą TCP.

Tablica `pathtable` pozwala na bezpośrednie określenie drogi przesyłania danych do zdalnych komputerów i sieci. Poszczególne wpisy mają format podobny do definicji aliasów dla systemu UUCP i są posortowane alfabetycznie. Tablica `pathtable` jest używana rzadko, ponieważ systemy linuxowe zwykle radzą sobie z routingu bez „prowadzenia za rączkę”.

Plik `uucprelays` pozwala na „zwarcie” ścieżki UUCP w przypadku, gdy istnieje lepsza droga pozwalająca na dostarczenie poczty. Przykładowo, jeśli użytkownicy często wysyłają pocztę do systemu określonego ścieżką `wizard!bignet!merlin!tpci`, a w systemie utworzone zostało bezpośrednie połączenie z systemem `tpci`, możesz za pomocą wpisu w pliku `uucprelays` skierować pocztę krótszą drogą. Plik ten jest dość rzadko używany w systemach linuxowych.

Tablica `uucpxtable` używana jest, gdy do dostarczenia poczty należy użyć adresu UUCP. Pozwala ona na przetłumaczenie adresu DNS na adres UUCP. Jeśli używasz serwera poczty innego niż aktualny system albo chcesz łączyć się z poszczególnymi komputerami za pomocą protokołu UUCP ze względu na niezawodność, powinieneś skorzystać z możliwości udostępnianych przez ten plik.

W pliku tym znajdują się pary nazwa UUCP – nazwa DNS, na przykład:

```
chatton chatton.com
```

Taki wpis informuje program `sendmail`, że poczta mająca trafić do systemu `chatton.com` powinna być przesyłana za pomocą protokołu UUCP do systemu o nazwie `chatton`. Dzięki temu adres `yvonne@chatton.com` zostanie przetłumaczony do postaci `chatton!yvonne`, która może zostać obsłużona przez protokół UUCP.

Tworzenie pliku `sendmail.cf`

Po wpisaniu danych do pliku `sendmail.m4` i innych związanych z nim plików, można wygenerować plik `sendmail.cf`. Do tego celu używa się preprocesora `m4`. Kiedy plik `sendmail.m4` jest gotowy, należy wydać polecenie:

```
make sendmail.cf
```

podstawiając oczywiście odpowiednią nazwę pliku (o ile została ona zmieniona – na przykład jeśli plik konfiguracyjny nazywał się `tpci.m4`, powinieneś podstawić nazwę `tpci.cf`).

Po utworzeniu pliku `sendmail.cf`, należy skopiować go do katalogu `/etc` i uruchomić program `sendmail` poleceniem:

```
/usr/lib/sendmail -bd -q1h
```

lub zresetować komputer (ponieważ `sendmail` zwykle uruchamiany jest z jednego z plików inicjalizacyjnych `rc`). Ścieżki dostępu do katalogów, w których zapisane są pliki konfiguracyjne, mogą być inne – w takim przypadku należy oczywiście skopiować plik `sendmail.cf` do odpowiedniej lokacji.

Używanie programu `sendmail` w wersji 8

Najnowsza wersja programu `sendmail` ma numer 8. Nie przejmuj się tym, że nie wiadłeś wersji 6 ani 7 – wersje o takich numerach nigdy nie istniały. Po wersjach 5.X pojawiła się od razu wersja z numerem 8 (jednym z ważniejszych ulepszeń tej wersji jest ochrona przed spamem – czyli niechcianą pocztą: sama ta cecha może być wystarczającym powodem dla aktualnienia wersji oprogramowania!).

Szczegóły konfiguracji systemu `sendmail` 8 nie różnią się od konfiguracji innych wersji, za wyjątkiem obsługi aż czterech wersji protokołu UUCP:

- ⦿ **uucp-old** (to samo co **uucp**) Klasyczny UUCP, używający adresów w formacie `komputer!uzytkownik`, potrafiący wysłać wiadomość tylko pod jeden adres (kiedy wiadomość adresowana jest do kilku osób, do każdej z nich wysyłana jest osobna kopia). Ta wersja powinna być używana, tylko jeśli kompatybilność ze starymi systemami UUCP jest bezwzględnie konieczna.
- ⦿ **uucp-new** (wcześniej znane jako **suucp**) Protokół podobny do UUCP, ale pozwalający za pomocą polecenia `rmail` wysłać wiadomość do więcej niż jednego adresata. Poza tym nie wprowadza żadnych znaczących zmian w porównaniu z poprzednią wersją.
- ⦿ **uucp-dom** Pozwala na użycie adresów w formacie domenowym. Ta wersja protokołu może być niekompatybilna z niektórymi systemami, z którymi przyjdzie Ci się łączyć.
- ⦿ **uucp-udom** Kombinacja wersji `uucp-new` i `uucp-dom`, obsługująca poprawnie zarówno adresy domenowe jak i UUCP.

Którankolwiek z wersji programu obsługi UUCP wybierzesz, odpowiedni plik powinien zostać skopiowany (lub dołączony) do standardowego programu obsługi UUCP.

smail

Pod względem funkcjonalności system `smail` jest bardzo podobny do systemu `sendmail`, ale proces jego konfiguracji jest inny. Pod pewnymi względami program `smail` jest łatwiejszy w obsłudze niż `sendmail`, dlatego może stanowić dobrą alternatywę w mniejszych systemach. Jeśli zdecydujesz się na użycie systemu `smail`, będziesz musiał ręcznie modyfikować pliki konfiguracyjne, ponieważ skrypty czy programy automatyzujące ten proces są w zasadzie niedostępne.

System `smail` używa wielu opcji i parametrów, których nie trzeba modyfikować, dlatego przyjrzymy się tylko tym, które są naprawdę niezbędne. Jeśli chcesz dowiedzieć się więcej o opcjach i parametrach, których nie omawiamy w tym rozdziale, zajrzyj na strony `man` poświęcone programowi `smail`. Tu skoncentrujemy się na łatwej i szybkiej konfiguracji programu `smail` do najczęściej wykonywanych zadań.

Konfigurowanie programu smail

Aby program `smail` mógł działać poprawnie, niezbędne jest utworzenie kilku dowiązań do niego. Dwa najważniejsze to `/usr/bin/rmail` i `/usr/lib/sendmail` (w niektórych systemach `/usr/sbin/sendmail`). Dowiązania te są konieczne, ponieważ większość programów kieruje pocztę wychodzącą do programu `rmail` lub `sendmail`, a powinna trafić ona do `smail`. Utworzenie odpowiednich dowiązań spowoduje, że proces przekierowania danych do programu `smail` stanie się „przezroczysty” i nie trzeba będzie zmieniać konfiguracji programów pocztowych.

Powinieneś więc sprawdzić, czy programy `rmail` i `sendmail` są dowiązaniami do programu `smail`; jeśli nie, powinieneś utworzyć odpowiednie dowiązania. Dowiązania (symboliczne) zaznaczane są podczas wyświetlania danych o zawartości katalogów w następujący sposób:

```
lrwxrwxrwx    1      root      root   6 Sep 16:35    plik1 -> plik2
```

Strzałka (`->`) symbolizuje istnienie dowiązania symbolicznego. Jeśli dowiązania nie istnieją, można je utworzyć poleceniami:

```
ln -s /usr/local/bin/smail /usr/bin/rmail  
ln -s /usr/local/bin/smail /usr/bin/sendmail
```

Jeśli może się zdarzyć, że poczta wychodząca lub przychodząca będzie przesyłana również przez SMTP, potrzebne będzie jeszcze jedno dowiązanie

```
ln -s /usr/local/bin/smail /usr/bin/smtpd
```

kierujące do programu `smail` również dane adresowane do programu `smtpd` (oczywiście należy podstawić odpowiednie dla systemu ścieżki dostępu do tych programów). W takim przypadku należy również sprawdzić, czy konfiguracja TCP dopuszcza używanie tego protokołu. Wiersz pliku `/etc/services` o postaci

```
smtp    25/tcp #Simple Mail Transfer Protocol
```

nie powinien być zaznaczony jako komentarz (symbolem komentarza w tym pliku jest znak `#` w pierwszej kolumnie). Taki wpis pozwala na nawiązanie połączenia SMTP poprzez port TCP o numerze 25 (jest to wartość domyślna).

Jeśli program `smail` ma działać w tle przez cały czas, powinieneś uruchomić go w jednym z plików inicjalizacyjnych (zwykle `rc.inet2`), wpisując tam wiersz:

```
/usr/local/bin/smail -bd -q15m
```

Opcja `-bd` powoduje, że `smail` ładuje się jako program rezydentny, a `-q15m` nakazuje przetwarzanie poczty co 15 minut. Jeżeli chcesz, aby poczta obsługiwana była częściej, podstaw odpowiednią wartość. Zbyt częste obsługiwanie poczty obciąża jednak niepotrzebnie system.

Jeśli wolisz, by program `smail` nie działał cały czas, ale był uruchamiany przez `inetd` gdy jest potrzebny, powinieneś usunąć wpis z pliku `rc` (lub zaznaczyć go jako komentarz) i do pliku `/etc/inetd.conf` wpisać następujący wiersz:

```
smtp    stream  tcp      nowait  root    /usr/sbin/smtpd      smtpd
```

Aby powyższy wpis działał prawidłowo, musi być utworzone dowiązanie do programu `smail` o nazwie `smtpd`.

Zmiany w plikach konfiguracyjnych systemu `smail` zależą od tego, jakiego protokołu używasz do przesyłania poczty: UUCP (łatwiejszy w konfiguracji) czy TCP. Oba przypadki przyjrzymy się osobno. Można również skonfigurować obie metody przesyłania poczty.

Konfigurowanie `smail` z protokołem UUCP

Konfigurowanie systemu `smail` tak, aby używało do przesyłania danych protokołu UUCP jest bardzo proste. Odpowiednie dane należy w prowadzić do pliku konfiguracyjnego `/usr/lib/smail/config`. Często w tym samym katalogu znajduje się plik `config.sample`, którego można użyć jako szablonu.

Do modyfikacji zawartości pliku `/usr/lib/smail/config` można użyć dowolnego edytora tekstu potrafiącego zapisać plik w formacie ASCII. Wprowadzić trzeba poprawki dotyczące czterech zmiennych:

- visible_domain** nazwa domeny, do której należy Twój system,
- visible_name** pełna nazwa domenowa komputera,

uucp_name	nazwa UUCP komputera (zwykle taka sama jak <code>visible_name</code>),
smart_host	nazwa serwera UUCP.

Każdy parametr podawany jest w formacie `nazwa=wartość`. Po obu stronach znaku równości nie powinno być spacji. Tekst od symbolu `#` do końca wiersza traktowany jest jako komentarz.

Ustawianie lokalnych nazw domenowych

Rozpocznij od wprowadzenia nazw domenowych dla komputerów lokalnych. Znajdź w pliku `/usr/lib/smail/config` fragment, który określa wartość zmiennej `visible_domain`; zwykle ma on postać:

```
#Our domain name
visible_domain=tpci
```

Zmienna `visible_domain` zawiera nazwę domenową Twojego komputera. Jeśli możliwe ma być użycie kilku nazw, wszystkie one powinny zostać wyszczególnione, na przykład:

```
#Our domain name
visible_domain=tpci:tpci.com:tpci.UUCP:uucp
```

Wartość tej zmiennej (oraz nazwa komputera lokalnego zwracana przez polecenie `hostname`) wykorzystywana jest do rozstrzygnięcia, czy wiadomość jest adresowana do systemu lokalnego, czy należy przesłać ją dalej.

Jeśli Twój system jest prawidłowo zarejestrowany w plikach mapujących UUCP, powinieneś do zmiennej `visible_domain` dodać również nazwę `uucp`, tak jak to przedstawiono w powyższym przykładzie. Warto również dodać wersje nazwy lokalnej zawierające najczęstsze pomyłki typograficzne (dla podanego wyżej przykładu mogłyby to być nazwy `tpci` czy `tpci.com`).

Ustawianie domeny lokalnej dla poczty wychodzącej

Kiedy wiadomość ma wyjść poza sieć lokalną, jako część informacji o trasie jej dostarczana jest pełna nazwa domenowa komputera-nadawcy. Określana jest ona na podstawie wartości zmiennej `visible_name`, której definicja może na przykład mieć postać:

```
#Our domain name for outgoing mail
visible_name=tpci.com
```

Ogólnie rzecz biorąc, nazwa ta powinna składać się z nazwy komputera i jednej z nazw domen podanych w zmiennej `visible_domain`, w przeciwnym przypadku listy odsyłane jako odpowiedzi na wysyłane z Twojego systemu wiadomości nie będą mogły do niego trafić.

Zmienna `visible_name` zawiera zwykle pełną nazwę domenową systemu (i ile system takową posiada) lub nazwę domenową, która występuje w tablicach kierowania przesyłem danych.

Inne nazwy UUCP

W pliku `/usr/lib/smail/config` znajduje się również definicja zmiennej `uucp_name`. Jest ona opcjonalna i nie musisz jej używać, jeśli wartości zmiennych `visible_domain` i `visible_name` są ustawione prawidłowo. Może jednak być przydatna wtedy, gdy na przykład zmienisz nazwę swojego komputera. Normalnie wymagałoby to wprowadzenia zmian we wszystkich tablicach mapowania UUCP, ale można tego uniknąć, podając starą nazwę systemu w zmiennej `uucp_name`. W pozostałych przypadkach jej wartość powinna być taka sama jak wartość zmiennej `visible_name`:

```
#UUCP mapping name  
uucp_name=tpci.com
```

Jeśli nazwa Twojego komputera się zmieniła, możesz pozostawić starą wartość w zmiennej `uucp_name`, co pozwoli uniknąć konieczności aktualniania tablic mapowania UUCP.

Ustawianie serwera UUCP

Niektóre komputery wykorzystują serwer UUCP, tzn. komputer, który obsługuje przesyłanie poczty do i z innych sieci. W takim przypadku należy go podać jako wartość zmiennej `smart_host` definiowanej w pliku `/usr/lib/smail/config`:

```
# Smart host  
smart_host=merlin
```

W takim przypadku program `smail` przesyła pocztę adresowaną do innych sieci do komputera `merlin` (jego pełna nazwa domenowa to `merlin.tpci.com` – jest ona określana na podstawie wartości zmiennych `smart_host` i `visible_name`), który zajmuje się przesyaniem jej dalej. Komputer ten musi być dostępny poprzez UUCP (czyli powinien posiadać odpowiednie wpisy w plikach konfiguracyjnych UUCP – patrz rozdział 39. „UUCP”).

Konfigurowanie smail z protokołem TCP

Jeśli do przesyłania poczty chcesz wykorzystać połączenie sieciowe, musisz wprowadzić odpowiednie modyfikacje do pliku `/usr/lib/smail/config`, zawierającego dane o typach połączeń wraz z nazwami systemów zdalnych. Istnieje kilka sposobów konfigurowania poczty w sieci lokalnej. Można na przykład użyć usługi NFS tak, aby jeden plik konfiguracyjny był dostępny dla wszystkich systemów, protokołu POP (ang. *Post Office Protocol*) czy IMAP (ang. *Interactive Mail Access Protocol*), które pozwalają na przetwarzanie wszystkich wiadomości w centralnym komputerze, lub też skonfigurować każdy z komputerów niezależnie od innych. Sam proces konfiguracji niewiele się różni w każdym z tych przypadków; różnica polega głównie na tym, że dane trzeba wprowadzać ręcznie.

dzić albo do wspólnego pliku konfiguracyjnego, dostępnego poprzez NFS lub SMTP, albo do plików konfiguracyjnych każdego z komputerów z osobna.

Proces konfiguracji rozpoczniemy od ustawienia lokalnych nazw domenowych w zmiennych `visible_domain` i `visible_name`. Były one omówione bardziej szczegółowo w podrozdziale dotyczącym konfigurowania `smail` dla UUCP, tu ograniczymy się tylko do podania przykładów:

```
#Our domain name
visible_domain=tpci.com
#Our domain name for outgoing mail
visible_name=tpci.com
```

Te definicje ustalają nazwę domenową systemu lokalnego – na jej podstawie `smail` rozstrzyga, czy poczta adresowana jest do systemu lokalnego, czy należy przesyłać ją dalej. Wartość zmiennej `visible_domain` będzie dodawana do wszystkich wiadomości wychodzących jako adres zwrotny (zamiast nazwy zwracanej przez polecenie `hostname`). Najczęściej wartości zmiennej `visible_domain` i `visible_name` są takie same.

Następnym krokiem konfiguracji jest ustawienie nazwy komputera zajmującego się wymianą poczty z innymi sieciami. Jeśli to Twój system pełni taką funkcję, nie musisz wpisywać żadnych wartości. Zmienne, których wartości należy ustawić, to `smart_path` (określa nazwę komputera obsługującego pocztę; nazwa ta jest ustalana przez dołączenie do wartości zmiennej `smart_path` wartości zmiennej `visible_domain`), oraz `smart_transport` (która określa protokół używany do przesyłania poczty; przeważnie jest to protokół SMTP).

Oto przykładowy fragment pliku `/usr/lib/smail/config`, określający wartości tych zmiennych:

```
#smart host routing
#smart host name
smart_host=merlin
# communications protocol to smart host
smart_transport=smtp
```

Modyfikacja zachowania programu smail

Konfiguracja opisana w poprzednich podrozdziałach wystarcza dla większości systemów linuxowych. Ponieważ jednak system `smail` składa się z trzech modułów (router, modułu transportującego i dostarczającego), możliwe jest dostosowanie jego zachowania do praktycznie każdej, nawet najbardziej nietypowej sieci. W typowych sieciach jedynym elementem, który może wymagać dalszej konfiguracji, jest router, dlatego zajmiemy się nim nieco bardziej szczegółowo.

W większości przypadków zachowanie każdego elementu programu `smail` kontrolowane jest przez plik konfiguracyjny (lub kilka plików) zapisany w katalogu `/usr/lib/smail`. Zarówno w standardowych dystrybucjach, jak i w węzłach FTP, dostępnych jest wiele przykładowych plików konfiguracyjnych. Modyfikacja takiego pliku jest znacznie łat-

twiejsza, niż tworzenie go „od zera”. Liczba dostępnych opcji i niektóre szczegóły konfiguracji różnią się w zależności od wersji programu `smail` – należy więc sprawdzić, czy pliki przykładowe są przeznaczone dla właściwej wersji tego programu.

Router odpowiada za tłumaczenie adresów docelowych, kierowanie wiadomości do następnego komputera na drodze przesyłu, oraz określanie protokołu, jakiego należy użyć do przesyłania danych. Jeśli wiadomość adresowana jest do użytkownika w systemie lokalnym (co rozpoznawane jest przez porównanie adresu z wartością odpowiedniej zmiennej definiowanej w pliku `/usr/lib/smail/config`), jest przekazywana do modułu dostarczającego.

Jeśli wiadomość ma dotrzeć do maszyny zdalnej, odpowiednie sterowniki (określone w pliku `/usr/lib/smail/routers`) decydują na podstawie adresu, do którego komputera dane powinny zostać przesłane. Plik `/usr/lib/smail/routers` zawiera nazwy sterowników routera. Każdy z nich (w kolejności określonej przez kolejność wpisów w tym pliku) otrzymuje adres docelowy wiadomości i sprawdza, czy na podstawie takiego adresu potrafi określić drogę, jaką wiadomość powinna zostać skierowana.

W większości przypadków nie będą konieczne żadne zmiany w pliku `/usr/lib/smail/routers`. Domyślnie używana konfiguracja działa w następujący sposób:

- tłumaczenie nazwy bezpośrednio na numer IP za pomocą funkcji bibliotecznej `gethostbyaddr`;
- tłumaczenie za pomocą nazwy symbolicznej używając funkcji `gethostbyname`;
- tłumaczenie za pomocą bazy danych o nazwach skróconych (zapisanej w pliku `/usr/lib/smail/paths`);
- dla adresów UUCP – sprawdzenie, czy komputer docelowy jest dostępny bezpośrednio;
- przesyłanie poczty do komputera obsługującego wymianę poczty z innymi sieciami, o ile wszystkie inne metody nie daly rezultatu.

Ten sposób routingu działa dobrze w większości systemów, ale powinieneś zaznaczyć wiersz dotyczący UUCP jako komentarz, jeśli protokół ten nie został prawidłowo skonfigurowany lub też w ogóle nie zamierzasz go wykorzystywać (w przeciwnym przypadku zostaniesz zasypany komunikatami o błędach generowanymi przez system `smail`).

Jeśli system jest połączony bezpośrednio do Internetu, konfiguracja programu `smail` musi być nieco zmodyfikowana, ponieważ nie potrafi on bezpośrednio obsługiwać formatu MX. Należy wówczas zaznaczyć jako komentarz wszystkie wiersze w pliku `routers` i zamiast tego użyć routera BIND (jeśli nie jest on dostarczony z Twoją wersją Linuxa, możesz go załadować z węzłów BBS i FTP).

Jeżeli używasz zarówno połączeń SLIP/PPP, jak i UUCP, możesz spotkać się z sytuacją, że `smail` zbyt długo czeka na połączenie. W takim przypadku powinieneś przestać kolejność akcji w pliku `/usr/lib/smail/routers` tak, aby najpierw przeglądana była baza danych o nazwach skróconych. Ponieważ UUCP jest bardziej wydajny przy

połączeniach SLIP/PPP niż SMTP, możesz w zasadzie całkowicie wyłączyć tłumaczenie za pomocą funkcji `gethostbyaddr` i `gethostbyname`.

Kiedy router znajdzie już najlepszą drogę dostarczenia poczty, określa również protokół, za pomocą którego ma ona zostać przesłana. W tym punkcie adres docelowy może zostać zmodyfikowany. Przykładowo, jeśli `chatton@bigcat.com` jest dostępny przez UUCP, adres zostanie zmieniony na `bigcat!chatton`. W innych przypadkach adres docelowy może zostać wzbogacony o pewne szczegóły, na przykład adres `chatton@bigcat.com` może zostać uzupełniony o nazwę komputera `chatton@whiskas.bigcat.com`, jeśli zapewni to efektywniejsze dostarczenie wiadomości.

Niektóre routery UUCP używają również pliku `/usr/lib/smail/paths`, w którym definiowane są aliasy pozwalające na dostęp do systemów z wykorzystaniem serwerów pośredniczących. Plik ten zawiera posortowany zestaw wpisów, składających się z dwóch pól rozdzielonych znakiem tabulacji; pierwsze z nich określa nazwę systemu docelowego, natomiast drugie – pełną ścieżkę w formacie UUCP. W pliku tym nie jest dozwolone używanie komentarzy.

Podsumowanie

W tym rozdziale przyjrzaliśmy się konfiguracji programów `sendmail` i `smail` dla systemów poczty opartych zarówno na protokole UUCP, jak i TCP. To, który z tych programów zainstalujesz w swoim systemie, zależy głównie od indywidualnych upodobań. Oba programy działają dobrze i powinny zapewnić Ci bezproblemowy dostęp do poczty elektronicznej.

Konfigurowanie systemu linuxowego tak, aby można było korzystać z grup dyskusyjnych, omówione jest w rozdziale 41. „Konfigurowanie grup dyskusyjnych”.

W rozdziale 42. „Bezpieczeństwo w sieci” omówione są metody zabezpieczania się przed atakami intruzów z zewnątrz.

Tworzenie kopii zapasowych (które mogą uchronić Cię przed koniecznością ponownego konfigurowania systemu poczty) jest tematem rozdziału 45. „Kopie zapasowe”.

Metody automatyzowania codziennych zadań przedstawione są w rozdziale 46. „cron i at”.

Rozdział 41.

Konfigurowanie grup dyskusyjnych

Tim Parker

W tym rozdziale:

- υ Usenet i grupy dyskusyjne
- υ NNTP
- υ Konfiguracja przeglądarek grup dyskusyjnych

Jeśli posiadasz dostęp do Internetu, wcześniej czy później będziesz chciał skorzystać z dobrodziejstw oferowanych przez grupy dyskusyjne sieci Usenet. Jest to jeden z najbardziej dynamicznych (i najbardziej kontrowersyjnych) aspektów Internetu. Choć można korzystać tylko z list korespondencyjnych, większość użytkowników jest zainteresowana szczególnie dostępem do sieci Usenet.

Usenet został stworzony po to, aby ułatwić użytkownikom dostęp do grup dyskusyjnych (ang. *newsgroups*). Grupa dyskusyjna pozwala każdemu, kto ma do niej dostęp, na publiczną rozmowę z pozostałymi członkami grupy. Usenet jest obsługiwany przez miliony sieci w setkach krajów, a korzystają z niego setki milionów użytkowników.

Każdy komputer, który połączony jest z Internetem, czy to bezpośrednio, czy przez bramkę internetową lub połączenie modemowe, może uzyskać dostęp do sieci Usenet. W tym celu trzeba posiadać odpowiednie oprogramowanie, które potrafi załadować i odesłać wiadomości, oraz przeglądarkę, umożliwiającą czytanie i pisanie artykułów.

Oprogramowanie implementujące przekazywanie wiadomości z Usenetu poprzez sieci lokalne nazywane jest NNTP (ang. *Network News Transfer Protocol*). Za pomocą NNTP system może współpracować z każdym innym obsługującym Usenet. NNTP wchodzi w skład większości dystrybucji Linuxa, dzięki czemu nie trzeba szukać dodatkowego oprogramowania.

Usenet i grupy dyskusyjne

Usługa Usenet składa się z dwóch elementów. Najpierw trzeba załadować do systemu wszystkie wiadomości, za co odpowiedzialne jest oprogramowanie transportujące (NNTP dla połączeń TCP i CNews dla UUCP), a następnie połączyć je i przekonwertować do czytelnej dla użytkownika postaci. Pierwotnie grupy dyskusyjne korzystały wyłącznie z protokołu UUCP, dlatego większość oprogramowania obsługiwała ten standard, a funkcje umożliwiające przesyłanie danych w inny sposób zostały dodane później. Ponieważ większość z Was używa będzie protokołów TCP/IP, w tym rozdziale skoncentrujemy się na oprogramowaniu NNTP.

Transfer wiadomości odbywa się za pomocą techniki nazywanej trasowaniem rozpływowym (ang. *flooding*). Jeden komputer łączy się z drugim i wysyła do niego wszystkie artykuły (proces przesyłania artykułów nazywa się po angielsku *newsfeed*). Ten z kolei łączy się i wysyła artykuły do następnego komputera lub kilku komputerów. Takie rozwiązanie pozwala na uniknięcie centralnego rozprowadzania danych. W każdym systemie znajduje się lista komputerów, do których należy przesłać dane.

W systemie biorącym udział w przesyłaniu artykułów może zostać dodany nowy artykuł. Do każdego z artykułów dołączona jest lista serwerów, które go już „widziały” (nazywana ścieżką, ang. *Path*), co pozwala zapobiegać wielokrotnemu przesyłaniu tych samych danych. System, do którego trafia artykuł, dopisuje swój identyfikator do ścieżki, używając notacji UUCP.

Można ograniczyć obszar, na którym dany artykuł ma być rozprowadzany, przez odpowiedni wpis w jego nagłówku. Dzięki temu możesz napisać artykuł, który będzie czytany tylko w sieci lokalnej (komputery sieci lokalnej po prostu nie prześlą go dalej).

Aby zapobiec powstawaniu duplikatów artykułów krążących w sieci Usenet, każdy z nich posiada swój identyfikator (zapisany w polu `Message-ID` nagłówka), składający się z niepowtarzalnego numeru i nazwy systemu, w którym artykuł został napisany.

Identyfikator artykułu używany jest przez każdy komputer w momencie rozpoczęcia transferu danych. W każdym systemie znajduje się plik o nazwie `history`, zawierający identyfikatory dostarczonych do systemu artykułów, na którego podstawie system stwierdza, czy chce, aby dana wiadomość została przesłana, czy też ma już jej własną kopię. Za przesyłanie odpowiednich informacji odpowiedzialny jest protokół `ihave/sendme`.

Za pomocą protokołu `ihave/sendme` jeden komputer przesyła do drugiego listę wszystkich posiadanych artykułów i czeka, aż tamten prześle informacje o tym, które z nich chce otrzymać. Artykuły są przesyłane pojedynczo, w odpowiedzi na sygnały `sendme`. Następnie proces może zostać odwrócony. Takie rozwiązanie działa dobrze, ale jest dość niewygodne przy przesyłaniu większych ilości danych. Z tego powodu (oraz z powodu niewielkiej prędkości przesyłu danych poprzez modemowe połączenia UUCP) protokół `ihave/sendme` raczej nie jest używany w przypadku, gdy trzeba regularnie przesyłać duże ilości artykułów. Przykładowo, przesłanie 100 MB artykułów za pomocą protokołu `ihave/sendme` każdego dnia jest praktycznie niewykonalne.

Inną metodą jest przesyłanie za pośrednictwem szybkiego połączenia wszystkich posiadanych artykułów do następnego komputera (ang. *batching*), który może usunąć te, które już ma (porównując ich numery identyfikacyjne z zapisanymi w pliku `history`). Ta metoda jest jednak również dość uciążliwa, ponieważ komputer odbierający otrzymuje dużo danych i musi je przetworzyć, co obciąża system.

Istnieją zasadniczo trzy metody dostępu do artykułów przechowywanych w innym systemie. Za pomocą NNTP można załadować artykuły, używając techniki podobnej do protokołu `ihave/sendme`. Można również zamówić artykuły, które powstały po określonej dacie. Trzecim sposobem jest czytanie artykułów po jednym, bez konieczności ładowania ich do swojego komputera. W tym celu trzeba jednak zalogować się do systemu zdalnego, co w dzisiejszych czasach często nie stanowi problemu.

NNTP

NNTP może działać w dwóch trybach: aktywnie i biernie. Tryb aktywny (podobny do protokołu CNews `ihave/sendme`) polega na tym, że system wysyłający (klient) oferuje dany artykuł odbierającemu (serwer), który go przyjmuje lub odrzuca. Ten tryb powoduje duże obciążenie serwera, który musi przetworzyć wszystkie posiadane artykuły.

Tryb bierny polega na tym, że komputer odbierający zamawia wszystkie artykuły, które przybyły od określonej daty (za pomocą polecenia `newnews`). Następnie usuwa artykuły powtarzające się lub niepotrzebne, wykorzystując polecenie `article`. Jest to rozwiązanie mniej obciążające serwer, który musi tylko wysłać większą liczbę artykułów, ale ze względów bezpieczeństwa serwer przed przesaniem danych powinien upewnić się, że zamawiający ma prawo czytać otrzymywane informacje.

NNTP został zaimplementowany dla systemu Linux przez Stana Barbera i Phila Lapleya jako program rezydentny `nntpd`. Zwykle jest on rozprowadzany w postaci kodu źródłowego, ponieważ do poprawnego działania wymaga wprowadzenia kilku informacji dotyczących systemu.

System `nntpd` składa się z programu-serwera i dwóch programów-klientów (jednego dla trybu aktywnego, drugiego dla biernego). Dodatkowo większość wersji `nntpd` zawiera zamiennik programu `inews`.

Alternatywą dla `nntpd` może być program INN (InterNetNews), opracowany przez Richa Salza. Jest on również rozprowadzany z wieloma dystrybucjami Linuxa. Pozwala na korzystanie zarówno z UUCP, jak i TCP, ale przeznaczony jest raczej dla większych komputerów. Jeśli przewidujesz korzystanie z wielu grup dyskusyjnych, powinieneś zamiast systemu `nntpd` zainstalować program INN. Ponieważ `nntpd` wystarcza dla większości użytkowników, skoncentrujemy się na jego konfiguracji. Dokładniejsze informacje na temat programu INN znajdziesz w dokumentacji dołączonej do oprogramowania lub dostępnej w węzłach BBS i FTP oraz w postaci dokumentu FAQ posyłanego często na listy dyskusyjne.

Kiedy NNTP otrzymuje artykuł z innego komputera, przekazuje go do jednego z programów obsługujących grupy dyskusyjne. Zwykle jest to program `rnews` lub `inews`. Możliwe jest również skonfigurowanie NNTP tak, by współpracował z programem `relaynews`, pozwalającym na przesyłanie wszystkich artykułów do następnych komputerów (ang. *batching* – proces ten został omówiony wcześniej). Aby można było używać NNTP, należy skonfigurować plik `/usr/lib/news/history`, zawierający informacje niezbędne do przesyłania danych za pomocą niektórych protokołów.

Instalowanie serwera NNTP

Serwer NNTP o nazwie `nntpd` zwykle rozprowadzany jest w postaci kodu źródłowego. Trzeba go skompilować w systemie lokalnym, po wprowadzeniu do kodu źródłowego pewnych informacji o systemie. Informacje te można dostarczyć za pomocą programu zapisanego zwykle w pliku o nazwie `/usr/lib/news/common/conf.h`. Uruchom ten program (jest on w zasadzie zestawem makropoleceń) i odpowiedz na wszystkie zadane pytania. Jeśli nie możesz go znaleźć, użyj polecenia `find`:

```
find / -name conf.h -print
```

Proces instalacji NNTP zaczyna od utworzenia katalogu, w którym będą przechowywane przychodzące artykuły. Może to być na przykład `/usr/spool/news/.tmp` (lub `/var/spool/news/.tmp`). Właścicielem tego katalogu powinien być użytkownik `news`, należy więc wydać polecenia:

```
mkdir /usr/spool/news/.tmp  
chown news.news /usr/spool/news/.tmp
```

Serwer NNTP można skonfigurować na dwa sposoby. Może on działać przez cały czas, wówczas należy uruchomić go w jednym z plików `rc` (zwykle `rc.inet2`) w trakcie uruchamiania systemu. Można również uruchamiać go tylko wtedy, gdy jest potrzebny, za pośrednictwem programu `inetd`.

Jeśli `nntpd` ma działać przez cały czas, upewnij się (sprawdzając w pliku `/etc/inetd.conf`), że nie jest on wywoływany również przez program `inetd`, ponieważ spowoduje to konflikt.

Jeżeli chcesz, aby `nntpd` uruchamiany był przez proces `inetd`, co może nieco odciążyć system w czasie, gdy grupy dyskusyjne nie muszą być obsługiwane, musisz wprowadzić odpowiednie informacje do pliku `/etc/inetd.conf`. Za pomocą dowolnego edytora ASCII dodaj do niego wiersz:

```
nntp stream tcp nowait news /usr/etc/in.nntpd nntpd
```

Taki wpis może się tam już znajdować, zaznaczony jako komentarz (znak `#` w pierwszej kolumnie). W takim przypadku wystarczy usunąć symbol komentarza.

Bez względu na to, czy `nntpd` działa cały czas, czy jest uruchamiany przez proces `inetd`, w pliku `/etc/services`, określającym usługi dostępne za pośrednictwem protokołu TCP/IP, musi znajdować się wpis:

```
nntp 119/tcp readnews untp
```

W większości wersji Linuxa taki wpis istnieje, ale jest zaznaczony jako komentarz – usuń wtedy znak #.

Konfigurowanie nntpd

Po skompilowaniu programu `nntpd` (które następuje automatycznie po uruchomieniu programu `conf.h`) powinieneś skonfigurować plik `/usr/lib/news/nntp_access` i zdecydować, które serwery i w jaki sposób będą mogły łączyć się z Twoim komputerem za pomocą `nntpd`. Wpisy w tym pliku mają format:

```
nazwa_serwera      read|xfer|both|no      post|no      zabronione
```

`nazwa_serwera` musi być albo pełną nazwą domenową, albo adresem IP. Można również podać część nazwy lub numeru IP, zezwalając na dostęp wszystkich systemów wchodzących w skład podsieci. Jeżeli nazwa komputera, który próbuje się połączyć, dokładnie zgadza się z zawartością pola `nazwa_serwera`, plik nie jest przeszukiwany dalej. Jeśli natomiast pasuje tylko częściowo (tzn. zgadza się numer podsieci) przeszukiwanie jest kontynuowane. Aby określić prawa dostępu dla wszystkich komputerów, można zamiast nazwy serwera użyć słowa `default`.

Prawa dostępu definiowane są w drugim polu. Rozpoznawane są cztery wartości:

- read** komputer może otrzymywać artykuły;
- xfer** komputer może wysyłać artykuły;
- both** komputer może otrzymywać i wysyłać artykuły;
- no** brak dostępu do artykułów.

Trzecie pole mówi o tym, czy komputer zdalny może wysyłać nowe artykuły. Jeśli znajduje się tu wartość `post`, komputer może wysyłać nowe artykuły, a lokalny system NNTP zadba o skompletowanie informacji w ich nagłówkach. Jeśli słowo `no` pojawi się w drugim lub trzecim polu, system nie będzie mógł wysyłać nowych artykułów.

Ostatnie pole zawiera listę grup, do których komputer nie będzie miał dostępu. Pole to rozpoczyna się od wykrzyknika i ma format listy, której poszczególne elementy rozdzielane są przecinkami (taki format jest również używany w programie CNews), na przykład wpis:

```
chatton.bignet.com both post !alt.local
```

pozwala komputerowi `chatton.bignet.com` na wysyłanie i odbieranie artykułów ze wszystkich grup dyskusyjnych za wyjątkiem grup `alt` i `local`. Serwer może również wysyłać nowe artykuły.

Prawdopodobnie będziesz chciał skonfigurować plik `/usr/lib/news/nntp_access` tak, aby wszystkim systemom zdalnym przyznać jakieś domyślne uprawnienia, a następnie stworzyć konkretne wpisy dla systemów, z którymi faktycznie chcesz współpracować. Oto przykładowa zawartość pliku `/usr/lib/news/nntp_access`:

```
#wpis domyslny
default          xfer      no
#pelny dostep dla chatton
chatton.bignet.com both      post
#tylko odczyt dla brutus
brutus.bignet.com   read      no
```

Dzięki takiemu rozwiązaniu każdy komputer będzie mógł przesyłać artykuły do Twojego komputera (ale nie będzie mógł wysyłać nowych artykułów), natomiast komputer `chatton.bignet.com` posiada pełne uprawnienia, a `brutus.bignet.com` tylko prawo odczytu artykułów.

W niektórych systemach NNTP zaimplementowane są systemy autoryzacji, które zapewniają przed dostępem do danych przez osoby niepowołane. Ponieważ jak na razie nie działają one dość sprawnie, najlepiej ich nie używać. Jeśli posiadasz nowszą wersję `nntpd`, poszukaj w dokumentacji informacji o zastosowaniu tych mechanizmów.

Konfiguracja przeglądarek grup dyskusyjnych

Przeglądarka grup dyskusyjnych przedstawia w sposób wygodny dla użytkownika dane zapisane przez takie programy, jak `nntpd` czy `CNews`. Pozwala czytać, zapisywać, drukować i wykonywać inne operacje na artykułach, włączając oczywiście możliwość odpowiadania na nie. Można zapoznać się z tematami poruszonymi w danej grupie, zapisać się do grupy i wykonywać inne czynności.

Przeglądarki różnią się między sobą złożonością, użytecznością i wygodą obsługi. Wraz z dystrybucjami Linuxa dostarczane są zarówno aplikacje graficzne, jak i działające w trybie tekstowym. Programiści przystosowują również swoje ulubione przeglądarki działające pierwotnie w systemach UNIX czy DOS.

Nie możemy omówić wszystkich dostępnych przeglądarek, ale pokażemy, jak skonfigurować najpopularniejsze z nich. Te informacje w połączeniu z dokumentacją powinny pozwolić Ci uporać się z konfigurowaniem dowolnej przeglądarki. Ponieważ prawie z każdą dystrybucją rozprowadzane są programy `trn` i `tin`, właśnie im przyjrzymy się bliżej w tym rozdziale.

Konfigurowanie programu trn

Przeglądarka `trn`, wykorzystywana powszechnie przez użytkowników UNIX-a, oparta jest na klasycznej już przeglądarce `rn` (ang. *read news*). Główną nowością, jaką wprowadził `trn`, była możliwość śledzenia wątków (ang. *thread*, są to artykuły powiązane tematycznie). W większości systemów program `trn` jest gotowy do użycia bez żadnych dodatkowych zabiegów konfiguracyjnych, o ile nie chcesz korzystać z obsługi wątków.

Aby `trn` mógł odnaleźć artykuły powiązane ze sobą tematycznie, musi posiadać bazę danych o poszczególnych wątkach. Jest ona zwykle przechowywana w pliku `/usr/local/bin/rn/mthreads`, a tworzy ją program `mthreads`. Najlepiej uruchamiać go w regularnych odstępach czasu za pomocą programu `cron` (zwykle tak często, jak ładowane są nowe artykuły). Program `trn` będzie działał bez programu `mthreads`, ale nie będzie umożliwiał obsługi wątków.

Polecenie `mthreads` bez żadnych argumentów generuje bazę danych tylko dla ostatnio załadowanych artykułów we wszystkich grupach dyskusyjnych. Aby poindeksować wszystkie grupy od zera, wydaj polecenie:

```
mthreads all
```

Spowoduje ono sprawdzenie zawartości pliku `/usr/lib/news/active` i rozpoczęcie indeksowania wszystkich grup wyszczególnionych w tym pliku.

Jeśli chcesz indeksować tylko niektóre grupy, możesz przekazać ich nazwy jako argumenty wywołania polecenia `mthreads` (w wierszu poleceń lub pliku `crontab`), na przykład polecenie

```
mthreads rec.auto.antique
```

spowoduje poindeksowanie bazy wątków grupy `rec.auto.antique`. Możliwe jest też poindeksowanie kilku grup – należy wówczas podać ich nazwy rozdzielone przecinkami jako argumenty polecenia `mthreads`. Możesz również indeksować całe gałęzie grup, na przykład tak:

```
mthreads alt
```

Aby nie indeksować którejś z grup, poprzedź jej nazwę wykrzyknikiem; polecenie:

```
mthreads rec.auto,rec.audio,!rec.audio.tech
```

spowoduje poindeksowanie grup gałęzi `rec.auto` i `rec.audio`, za wyjątkiem grupy `rec.audio.tech`.

Jeśli nowe artykuły ładujesz bardzo często, możesz rozważyć uruchomienie programu `mthreads` jako programu rezydentnego, dzięki czemu nie trzeba uruchamiać go co jakiś czas, a nadchodzące artykuły są przetwarzane natychmiast – obciąża to jednak nieco bardziej zasoby systemu. Aby uruchomić program `mthreads` jako program rezydentny, należy użyć opcji `-d`. Domyślnie nowe artykuły indeksowane są co 10 minut. Polecenie uruchamiające ten program można umieścić w jednym z plików `rc`.

Konfigurowanie programu `tin`

W przeciwieństwie do programu `trn`, `tin` nie wymaga interwencji użytkownika do zbudowania bazy danych o wątkach – tworzy ją za każdym razem, gdy użytkownik wchodzi na listy dyskusyjne. Indeksowanie przebiega dość szybko, chyba że grupa zawiera więcej niż 500 nowych artykułów.

Plik zawierający indeks zapisywany jest w katalogu domowym użytkownika pod nazwą `.tin/index/nazwa_grupy_dyskusyjnej`. Jego rozmiary mogą być całkiem spore, dlatego warto rozważyć utworzenie jednego indeksu, z którego będą mogli korzystać wszyscy użytkownicy. Taki efekt można uzyskać zmieniając właściciela programu `tin`:

```
chown news.news tin
```

Baza danych o wątkach będzie teraz zapisywana w pliku `/usr/spool/news/.index`.

Możesz również uruchomić program rezydentny `tind`, który zadba o aktualizację bazy danych na bieżąco. Kod źródłowy tego programu dostarczany jest wraz z niektórymi dystrybucjami Linuxa, ale rzadko jest on rozpowszechniany w wersji skompilowanej – dla tego do jego zainstalowania konieczny będzie kompilator języka C i program narzędziowy `make`.

Podsumowanie

Po skompilowaniu, zainstalowaniu i skonfigurowaniu systemu NNTP i przeglądarki, możesz zacząć korzystać z możliwości oferowanych przez grupy dyskusyjne. Oprócz przeglądarek `tin` i `trn` dostępne są też inne programy tego typu, na przykład `nn`, również zawierający obsługę wątków. Wybór jednego z nich jest kwestią indywidualnych preferencji. Najlepiej, jeśli sam sprawdzisz, który najbardziej Ci odpowiada.

Jeśli chcesz zabezpieczyć swój system przed hakerami, zajrzyj do rozdziału 42. „Bezpieczeństwo w sieci”.

O tworzeniu kopii zapasowych systemu plików przeczytasz w rozdziale 45. „Kopie zapasowe”.

Automatyzacja zadań, takich jak ładowanie artykułów z grup dyskusyjnych w ciągu nocy, opisana jest w rozdziale 46. „cron i at”.

Rozdział 42.

Bezpieczeństwo w sieci

Tim Parker

W tym rozdziale:

- υ Słabe hasła
- υ Bezpieczeństwo plików
- υ Dostęp przez modem
- υ UUCP
- υ Dostęp poprzez sieć lokalną
- υ Śledzenie intruza
- υ Przygotowywanie się na najgorsze

Omówienie wszystkich zagadnień związanych z bezpieczeństwem systemów zajęłoby kilka tomów, z konieczności więc omówimy tylko te najprostsze. Przyjrzymy się najbardziej podstawowym mechanizmom zabezpieczania się przed włamaniem poprzez łącza telefoniczne i niektórym aspektom zabezpieczeń w sieci lokalnej. Nie będziemy omawiać skomplikowanych rozwiązań, które są trudne do wprowadzenia i wymagają specjalistycznej wiedzy, a ponadto mogą być zastosowane tylko w niektórych systemach.

Zamiast tego przyjrzymy się podstawowym metodom zabezpieczania systemu, które są zarówno proste, jak i skuteczne. Wielu administratorów albo nie wie, co można zrobić, aby zabezpieczyć się przed włamaniem, albo też lekceważy taką możliwość. Włamania są jednak wyjątkowo częste, więc nie warto igrać z ogniem.

Słabe hasła

Może się to wydawać nieprawdopodobne, ale najczęściej intruzi dostają się do systemów (obojętnie czy przez sieć lokalną, czy połączenie modemowe), wykorzystując słabe – czyli łatwe do odgadnięcia – hasła użytkowników. Jeśli użytkownicy stosują słabe hasła, nawet najlepszy system zabezpieczeń nie może zabezpieczyć przed włamaniem.

Jeśli zarządzasz systemem, w którym jest kilku użytkowników, powinieneś wymagać od nich zmiany hasła co pewien czas (na przykład co sześć czy osiem tygodni). Najlepsze hasła składają się z kombinacji cyfr i liter, których nie można znaleźć w żadnym słowniku.

Czasem zmiana haseł nie wystarcza – powinieneś wtedy rozważyć użycie jednego z programów komercyjnych lub dostępnych na licencji public domain, wymuszających użycie mocnych haseł. Pakiety takie zwykle rozprowadzane są w postaci kodu źródłowego, przed użyciem konieczne jest więc ich skompilowanie.

Bezpieczeństwo plików

Dbałość o bezpieczeństwo systemu powinna zaczynać się już na poziomie praw dostępu do plików, które powinny być ustawiane bardzo uważnie. Jeśli chcesz zabezpieczyć nowe pliki przed niepowołanymi osobami, ustaw odpowiednią wartość zmiennej `umask`.

Oczywiście kwestie bezpieczeństwa plików mają znaczenie tylko wtedy, gdy z systemu korzysta jeszcze ktoś oprócz Ciebie. Jeśli tak jest, warto pomyśleć nad globalnym ustawieniem wartości tej zmiennej dla wszystkich użytkowników, zapewniając, że nowe pliki będą miały przypisane prawa dostępu nadające ich właścicielom i nikomu innemu prawo do odczytu i zapisu. To w zasadzie wszystko, co można zrobić dla zabezpieczenia plików.

Szczególnie ważne pliki (na przykład zawierające dane o pracownikach) warto dodatkowo zabezpieczyć za pomocą jakiegoś programu szyfrującego – dostępnych jest wiele tego rodzaju aplikacji. W większości z nich do kodowania i dekodowania używa się tego samego hasła

Dostęp przez modem

Dla większości użytkowników Linuxa problem zabezpieczania się przed włamaniem poprzez bramkę internetową nie jest szczególnie istotny z prostego powodu – nie posiadają oni bezpośredniego połączenia z Internetem. Trzeba jednak pomyśleć o zabezpieczeniu się przed włamaniemi przez modem.

Modemy są najczęściej używanym „oknem na świat” w systemach linuxowych. Za ich pośrednictwem można używać systemu z komputera zdalnego, uzyskać dostęp do Internetu itp. Zabezpieczenie linii używanych przez modem jest prostą i efektywną metodą powstrzymania przypadkowych szperaczy.

Modemy oddzwaniające

Najbezpieczniejszą techniką kontrolowania dostępu do modemu jest użycie modemów oddzwaniających (ang. *callback modems*). Modemy tego typu pozwalają użytkownikowi połączyć się w zwykły sposób, następnie przerywają połączenie, wyszukując w bazie danych

numer telefonu użytkownika, który chciał się zalogować, i same nawiązują połączenie. Największą wadą takiego rozwiązania są wysokie koszty, dlatego nie jest ono stosowane zbyt często.

Dodatkowe problemy pojawiają się, gdy użytkownicy często zmieniają miejsce, z którego dzwonią, co wiąże się ze zmianą numeru telefonu. Modemy oddzwaniające dają się również oszukać za pomocą przekazywania rozmowy – rozwiązania oferowanego przez wiele nowocześniejszych aparatów telefonicznych.

Problemy z modemami

Typowy modem telefoniczny może być źródłem problemów, jeśli nie kończy prawidłowo połączenia (nie odwiesza słuchawki po zakończeniu sesji). Wynika to często ze złej konfiguracji modemu lub niewłaściwego podłączenia.

Problemy powodowane przez nieprawidłowe podłączenie modemu mogą wydawać się banalne, ale w wielu systemach połączenia montowane na własną rękę nie zapewniają prawidłowej kontroli nad linią i możliwe jest pozostawienie nie zakończonego połączenia. Wtedy następny dzwoniący kontynuuje sesję poprzedniego użytkownika.

Aby zabezpieczyć się przed tego typu problemami, wymień wszystkie robione ręcznie połączenia na okablowanie pochodzące od pewnego producenta. Warto również kilka czy kilkanaście razy sprawdzić, czy po zakończeniu sesji połączenie jest prawidłowo przerywane.

Problem ten może być również spowodowany nieprawidłową konfiguracją oprogramowania. Sprawdź w dokumentacji modemu, czy Twój skrypt potrafi odwiesić słuchawkę po zakończeniu sesji i przy zerwaniu połączenia. Problem ten rzadko występuje w przypadku popularnych modemów, ale mając jakiś mniej znany model warto się upewnić, czy nie będzie on powodował problemów, sprawdzając, czy po zakończeniu połączenia linia jest zwalniana.

Bardzo skuteczną metodą zapobiegania włamaniom przez modem jest po prostu odłączanie go w czasie, kiedy nie jest potrzebny. Ponieważ włamania zwykle mają miejsce po normalnych godzinach pracy systemów, można po prostu wyłączać modem na noc. Można również użyć pliku `crontab` do czasowego wyłączania portu szeregowego, do którego podłączony jest modem.

W wielu systemach wyłączanie modemu nie jest praktycznym rozwiązaniem, ale często warto je rozważyć. Jeśli dostęp w godzinach nocnych jest konieczny, można na przykład pozostawić jeden modem załączony, wyłączając pozostałe. W większych systemach zwykle liczba modemów działających po godzinach pracy jest znacznie mniejsza, niż pracujących normalnie.

Jak modem obsługuje połączenie

Aby użytkownik mógł otrzymać dostęp do systemu linuxowego przez modem, w systemie musi działać proces `getty`. Jest on uruchamiany przez proces `init` dla każdego portu szeregowego. Proces `getty` jest odpowiedzialny za pobranie identyfikatora użytkownika, ustalenie parametrów transmisji (na przykład typu terminalu i prędkości przesyłu danych) oraz obserwowanie, czy czas oczekiwania na odpowiedź nie został przekroczony. W systemie Linux porty szeregowe i porty kart multiport kontrolowane są przez plik `/etc/ttys`.

W niektórych systemach możliwe jest założenie dodatkowego hasła, wymaganego do połączeń modemowych. Użytkownik łączący się za pośrednictwem modemu będzie musiał wprowadzić dwa hasła. Jeśli hasło do połączeń modemowych jest obsługiwane, ustawia się je w pliku `/etc/dialups`.

W systemach linuxowych w pliku `/etc/dialups` przechowywana jest lista portów wymagających podania hasła, natomiast same hasła przechowywane są w innym pliku (na przykład `/etc/d_passwd`). Podobne rozwiązanie można również zastosować do połączeń UUCP.

UUCP

Przy projektowaniu systemu UUCP brano pod uwagę problemy bezpieczeństwa, ale było to wiele lat temu – od tego czasu wymagania stawiane tego typu systemom zmieniły się bardzo znacznie. Znaleziono również sporo luk w bezpieczeństwie stwarzanych przez ten system, choć większość z nich została naprawiona poprzez modyfikacje w kodzie źródłowym. System UUCP nadal wymaga, by poświęcić nieco uwagi prawidłowej konfiguracji – wtedy będzie działał poprawnie i bezpiecznie.

Jeśli nie zamierzasz używać UUCP, usuń z pliku `/etc/passwd` użytkownika `uucp` (lub przynajmniej zablokuj logowanie dopisując gwiazdkę na początku hasła). Usunięcie użytkownika `uucp` nie zakłóci w żaden sposób działania systemu, o ile nie jest używany protokół UUCP.

Wszystkim katalogom i plikom systemu UUCP powinny być przypisane tak ograniczone prawa dostępu, jak to tylko możliwe (zwykle są to katalogi `/usr/lib/uucp`, `/usr/spool/uucp` i `/usr/spool/uucppublic`). Odpowiednie prawa możesz ustalić za pomocą poleceń `chown`, `chmod` i `chgrp`. Właścicielem tych katalogów powinien być użytkownik `uucp`, grupa `uucp`. Warto również regularnie sprawdzać, czy wszystkie pliki mają przypisane prawidłowe prawa dostępu.

To, czy dany system ma prawo korzystać z protokołu UUCP, ustalane jest na podstawie zawartości kilku plików. Pliki te (na przykład `/usr/lib/uucp/Systems` i `/usr/lib/uucp/Permissions`) powinny być również własnością użytkownika `uucp` i tylko on powinien mieć do nich dostęp. To zabezpieczy przed ich modyfikacją przez niepowołane osoby.

Katalog `/usr/spool/uucppublic` często jest celem włamań, ponieważ każdy łączący się system ma w nim prawo zapisu i odczytu. Pewnym zabezpieczeniem może być utworzenie dwóch podkatalogów: jednego do odbierania, a drugiego do nadawania plików. Można również utworzyć osobny podkatalog dla każdego systemu łączącego się za pośrednictwem UUCP.

Dostęp poprzez sieć lokalną

Choć sieci lokalne rzadko stanowią zagrożenie same w sobie (ponieważ zwykle korzystają z nich zaufane osoby), są jednak jednym z najłatwiejszych sposobów na włamanie się do systemu. Najpoważniejszy problem polega na tym, że jeśli choć jedna maszyna ma jakieś słabe punkty (jeśli chodzi o zabezpieczenia), to żaden inny komputer w sieci nie jest bezpieczny. Z tego powodu każde zabezpieczenie powinno działać na wszystkich bez wyjątku komputerach w sieci.

Idealny system bezpieczeństwa w sieci lokalnej wymusza zastosowanie odpowiednich procedur uwierzytelniania dla każdego połączenia. Niestety, takie rozwiązanie może powodować konflikty z wieloma istniejącymi programami i rozwiązaniami systemowymi. Dlatego w Linuxie istnieje pojęcie komputera zaufanego (ang. *trusted host*). Taki komputer może połączyć się z systemem bez żadnych problemów, pod warunkiem, że jego nazwa znajduje się w odpowiednim pliku – w większości przypadków nie jest wymagane nawet hasło! Wszystko, co musi zrobić włamywacz, to poznać nazwę komputera zaufanego i połączyć się, podając taką nazwę. Sprawdź więc dokładnie zawartość plików `/etc/hosts.equiv`, `/etc/hosts` i `.rhosts`.

Jednym z rozwiązań jest używany ostatnio dość powszechnie system Kreberos, opracowany przez MIT. Kreberos wprowadza pojęcie komputera bardzo bezpiecznego (ang. *very secure host*), który działa jako serwer uwierzytelniający połączenia. Szyfrowanie przesyłanych wiadomości zabezpiecza również przed odczytywaniem ich nagłówków przez włamywaczy. System Kreberos uwierzytelnia wszystkie wiadomości przesyłane w obrębie sieci.

Ze względu na naturę większości sieci, systemy linuxowe są narażone na ataki intruzów posiadających dużą wiedzę o systemie. Znane są dosłownie setki problemów stwarzanych przez oprogramowanie TCP/IP. Jako pierwszy krok w zabezpieczaniu systemu powinieneś więc wyłączyć wszystkie te usługi i protokoły, których nie wykorzystujesz, ponieważ ktoś może użyć ich do włamania się do systemu.

Śledzenie intruza

Większość włamań dokonywanych jest przez ludzi ciekawych, jakiego typu dane przechowywane są w systemie, ale nie mających na celu dokonania zniszczeń. Często włamują się do systemu co jakiś czas, rozglądają się, grają w jakieś gry i wylogowują się niczego nie zmieniając. Z tego powodu trudno zauważyc, że ktoś się włamał – wówczas zdany jesteś tylko na łaskawość intruza. Może nie przyjdzie mu do głowy niczego niszczyc.

Czasem również system może być wykorzystywany jako odskocznia przy atakowaniu innego – wtedy administrator tego systemu stwierdzi, że włamanie nastąpiło z Twojego komputera i możesz mieć kłopoty.

„Śledzić” użytkowników systemu możesz włączając proces rejestrujący dane użytkownika za każdym razem, gdy łączy się on i rozciera z systemem, nazywany procesem audytu (ang. *auditing*) lub procesem monitorowania połączeń. Nie we wszystkich wersjach Linuxa proces taki jest dostępny. Dokładniejszych informacji szukaj na stronach `man`.

Jeśli proces audytu działa w Twoim systemie, powinieneś często przeglądać generowany plik. Dobrze jest również napisać prosty skrypt wyświetlający podsumowanie, zawierające dane o tym, kto ile czasu był podłączony do systemu, w jakich godzinach itp. To może pozwolić Ci wyłapać wszelkie anomalie czy zauważać coś, co nie zgadza się z Twoimi informacjami (na przykład logowanie się po godzinach pracy itp.). Taki skrypt możesz z łatwością napisać na przykład w języku `gawk`. Dostępne są również gotowe programy do tego celu.

Przygotowywanie się na najgorsze

Załóżmy, że ktoś się włamuje. Co możesz zrobić w takim przypadku? Oczywiście kopii zapasowych nie da się przecenić – pomogą Ci one odzyskać wszystkie usunięte czy uszkodzone pliki. Ale poza tym, co jeszcze można zrobić?

Po pierwsze, powinieneś dowiedzieć się, w jaki sposób dokonano włamania i zabezpieczyć tę drogę tak, by włamanie nie mogło się już powtórzyć. Jeśli nie jesteś pewny, włącz wszystkie modemy i terminale i uważnie sprawdź wszystkie pliki konfiguracyjne, szukając nieprawidłowych danych. Gdzieś musi być błąd, ponieważ inaczej nie byłoby włamania. Sprawdź również hasła i listę użytkowników – być może zapomniałeś usunąć jakieś konto, które dawno już nie powinno istnieć.

Jeśli jesteś ofiarą powtarzających się ataków, spróbuj załączyć system monitorowania połączeń. Być może dzięki temu będziesz mógł wyśledzić, jaką drogą nieproszony gość dostaje się do systemu i co w nim robi.

Powinieneś również mieć świadomość, że włamania są nielegalne – w ostatczności możesz więc skontaktować się z organami ścigania.

Podsumowanie

Przestrzegając kilku podanych wyżej prostych zasad, powinieneś uchronić się przed „zwykłymi” hakerami. Niestety, nie wystarczą one, by zabezpieczyć się przed najlepszymi – im nie przeszkladzą nawet wojskowe systemy ochrony. Wprowadzając opisane wcześniej zabezpieczenia, na pewno nie wyrządisz żadnych szkód w systemie, więc warto wprowadzić je w każdym systemie połączonym z innymi systemami za pomocą modemu lub sieci lokalnej.

Konfiguracja sieciowego systemu plików (NFS) opisana jest w rozdziale 43. „NFS”.

Metody tworzenia kopii zapasowych omawia rozdział 45. „Kopie zapasowe”.

Problematykę związaną z konfigurowaniem systemu tak, by mógł on działać jak serwer WWW, podejmuje rozdział 47. „Konfiguracja węzła internetowego”.

Rozdział 57. „Dostosowywanie jądra systemu” przedstawia metody wprowadzania zmian do jądra systemu.

Rozdział 43.

NFS

Tim Parker

W tym rozdziale:

- υ Konfiguracja NFS w systemie Linux

NFS (Network File System, sieciowy system plików), stworzony przez firmę Sun Microsystems, umożliwia współużytkowanie plików i katalogów pomiędzy systemami UNIX-owymi. Katalog sieciowy udostępniany poprzez NFS widziany jest w każdym systemie tak, jakby był katalogiem lokalnym. Przykładowo, jeśli masz dostęp do systemu linuxowego, w którym zainstalowane jest mnóstwo gier, a nie masz miejsca, by je wszystkie zainstalować u siebie, możesz za pomocą NFS skonfigurować swój system tak, aby gry te były widoczne w jakimś katalogu lokalnym i używać ich bez żadnych problemów. Dzięki NFS proces łączenia się z innym komputerem i transportu danych za każdym razem, gdy uruchamiasz grę, jest niewidoczny dla użytkownika (może za wyjątkiem opóźnień wprowadzanych przez sieć).

NFS może być używany w sieciach różnego typu, choć został zaprojektowany dla sieci opartych na TCP/IP. Ze względu na jego popularność, pojawiły się implementacje dla różnych systemów operacyjnych, co pozwala na współużytkowanie plików i katalogów w sieciach niejednorodnych.

W UNIX-ie i Linuxie NFS działa w trybie peer-to-peer (równy z równym). Oznacza to, że Twój komputer może zachowywać się zarówno jako klient, jak i serwer usługi NFS, a nawet spełniać obie te funkcje równocześnie.

Wiele osób używa NFS w pracy, ale boi się skonfigurować go we własnym systemie linuxowym, ponieważ uważają, że jego konfiguracja jest złożona, trudna, i wymaga dużej wiedzy o systemie operacyjnym. Z tego powodu wiele osób rezygnuje z NFS, choć jest to jedna z najbardziej pozytycznych usług oferowanych przez TCP/IP. Jak jednak przekonasz się czytając ten rozdział, konfigurowanie NFS w systemie Linux nie jest ani takie skomplikowane, ani długotrwałe, jak się powszechnie uważa. Oczywiście abyś mógł docenić jego zalety, w skład sieci musi wchodzić więcej niż jeden komputer...



Niektóre nowsze produkty, takie jak VisionFS, znacznie upraszczają używanie dysków sieciowych. Wersje tych programów są dostępne również dla Linuxa, ale niestety są to produkty komercyjne.

Konfiguracja NFS w systemie Linux

NFS intensywnie korzysta z usługi RPC (Remote Procedure Call, zdalne wywoływanie procedur). Z tego powodu przed uruchomieniem NFS trzeba sprawdzić, czy program obsługujący RPC działa poprawnie. W niektórych systemach możesz to zrobić wydając polecenie:

```
rpcinfo -p
```

Wynikiem jego działania powinna być lista wszystkich serwerów RPC działających w Twoim systemie, na przykład:

```
[root@linux reksio]# rpcinfo -p
program vers  proto port
100000  2      tcp   111    portmapper
100000  2      udp   111    portmapper
300019  1      udp   737
100001  13     udp   791    rstatd
100001  3      udp   791    rstatd
100001  2      udp   791    rstatd
100001  1      udp   791    rstatd
100001  13     udp   796    rstatd
100001  3      udp   796    rstatd
100001  2      udp   796    rstatd
100001  1      udp   796    rstatd
```

Jeśli RPC działa prawidłowo, powinieneś zobaczyć przynajmniej pięć pozycji: dwie dla UDP, dwie dla TCP oraz jedną dla programu `pcnfsd`, obsługującego NFS. W powyższym przykładzie nie ma pozycji `pcnfsd`, więc wiemy, że program obsługi NFS nie jest aktywny.

Programy obsługujące NFS muszą najpierw zostać zainstalowane: Linux najczęściej pozwala na to przy instalacji systemu. Jeśli nie zainstalowałeś ich wtedy, zrób to teraz, używając programu instalacyjnego dołączonego do Twojej wersji Linuxa. W niektórych systemach (np. Caldera Linux i wiele dystrybucji Slackware) program obsługi NFS jest domyślnie instalowany i uruchamiany przy starcie systemu, bez względu na to, czy jest faktycznie używany.

Konfiguracja serwera linuxowego

W większości wersji Linuxa NFS jest uruchamiany i wyłączany za pomocą skryptu `/etc/nfs`. Jeśli chcesz uruchamiać go automatycznie, powinieneś utworzyć dowiązanie do tego skryptu o nazwie `/etc/rc2.d/Sname`. Aby prawidłowo wyłączać NFS, powinieneś również stworzyć dowiązanie `/etc/rc0.d/Kname`. Ręcznie możesz go uruchomić i zatrzymać poleceniami:

```
/etc/nfs start  
/etc/nfs stop
```



W niektórych wersjach Linuxa używane są inne nazwy plików i katalogów, na przykład wspomniany wyżej skrypt `/etc/nfs` może być zapisany pod nazwą `/etc/rc.d/init.d/nfs` albo `/etc/init.d/nfs`.

Po wydaniu polecenia uruchamiającego NFS na ekranie powinny zostać wyświetcone komunikaty potwierdzające uruchomienie odpowiednich programów:

```
$ /etc/nfs start  
Starting NFS services: exportfs mountd nfsd pcnfsd biod(x4)  
Starting NLM services: statd lockd
```

natomiasz przy wyłączaniu usługi NFS, komunikaty o zakończeniu ich działania:

```
$ /etc/nfs stop  
NFS shutdown: [NFS Shutdown Complete]
```

Jeśli linuxowy system plików ma być dostępny za pośrednictwem usługi NFS dla innych komputerów, musi być wymieniony w pliku `/etc(exports`. W niektórych systemach NFS uruchamia się automatycznie, jeśli plik `/etc(exports` istnieje podczas uruchamiania systemu – wywoływany jest wówczas program `exportfs`, pozwalający innym komputerom korzystać z systemu plików za pośrednictwem NFS. Jeśli zmodyfikujesz plik `/etc(exports`, powinieneś wydać polecenie `exportfs` jeszcze raz albo ponownie uruchomić komputer, aby uaktywnić zmiany. Składnia wpisów w pliku `etc(exports` jest następująca:

```
katalog [-opcja, opcja ...]
```

`katalog` to ścieżka dostępu do katalogu, który ma być udostępniany (eksportowany, według terminologii NFS) za pomocą NFS, a opcje mogą przybierać następujące wartości:

- | | |
|------------------------------|---|
| ro | eksportowanie w trybie tylko do odczytu (domyślnie: do odczytu i zapisu); |
| rw=nazwy_komputerów | zezwolenie na zapis tylko dla wymienionych komputerów, dla pozostałych – tylko odczyt; |
| anon=uid | jeśli żądanie NFS pochodzi z nieznanego serwera, używa identyfikatora użytkownika <code>uid</code> dla określenia praw dostępu; |
| root=nazwy_komputerów | nadaje uprawnienia użytkownika <code>root</code> użytkownikom <code>root</code> pochodzący z wyszczególnionych komputerów; |

access=klient

umożliwia używanie NFS tylko wyszczególnionym klientom. Pole `klient` może zawierać nazwę komputera lub całej grupy sieciowej.

Podany niżej przykładowy plik `/etc(exports` ułatwi zrozumienie zastosowania tych opcji. Symbol

```
# oznacza początek komentarza.
/usr/stuff      -ro          #eksport tylko do odczytu dla wszystkich
/usr   -access=clients    #eksport dla grupy o nazwie clients
/usr/public     #eksport do odczytu i zapisu dla każdego
```

Jeśli zmodyfikujesz zawartość pliku `/etc(exports`, powinieneś zrestartować programy obsługi NFS. Polecenie `exportfs` wyświetla nazwy wszystkich eksportowanych systemów plików.



Niektóre systemy automatycznie generują plik o nazwie `/etc/xtab`, zawierający informacje o systemach plików. Nie należy modyfikować tego pliku, ponieważ NFS może przestać działać prawidłowo. Plik ten jest tworzony przez program `exportfs`.

W niektórych systemach linuxowych do eksportowania katalogów używa się polecenia `share` (wiele wersji Linuxa nie obsługuje tego polecenia, ponieważ jego funkcje pokrywają się funkcjami pliku `/etc(exports`). Jego składnia jest następująca:

```
share -F nfs -o opcje -d opis ścieżka
```

Parametr `nfs` określa, że katalog ma być eksportowany za pośrednictwem usługi NFS. `opcje` mogą przyjmować takie same wartości, jak opcje podawane w pliku `/etc(exports`. Można również podać opisową nazwę eksportowanego katalogu, używając opcji `-d`. Jeśli na przykład chcesz udostępnić do odczytu i zapisu katalog `/usr/public`, możesz wydać polecenie:

```
share -F nfs -d "Server public directory" /usr/public
```

Można również użyć jednocześnie kilku opcji, na przykład:

```
share -F nfs -o ro=artemis,anon=200 -d "Book material" /usr/reksio/book
```

Powyższe polecenie spowoduje udostępnienie katalogu `/usr/reksio/book`, opisanego jako `Book material`, dla każdego w trybie do odczytu i zapisu, za wyjątkiem komputera `artemis`, który będzie miał tylko możliwość odczytu. Prawa dostępu anonimowych użytkowników będą określone na podstawie praw dostępu użytkownika o identyfikatorze numerycznym `200`.

Polecenie `share` bez parametrów zwykle wyświetla informacje o wszystkich eksportowanych katalogach.

Konfigurowanie klienta

Eksportowany przez serwer system plików można zamontować bez żadnych problemów polecienniem `mount` o następującej składni:

```
mount -F nfs [-o opcje] serwer:katalog punkt_zamontowania
```

Opcja `-F nfs` określa, że chodzi o system plików dostępny przez NFS. `serwer:katalog` to nazwa serwera i ścieżka dostępu do katalogu, który ma zostać zamontowany. `punkt_zamontowania` określa katalog, w którym zdalny system plików będzie dostępny w komputerze lokalnym (katalog ten musi istnieć). W niektórych wersjach Linuxa składnia polecenia `mount` może być nieco inna, na przykład czasem zamiast opcji `-F nfs` należy podać `-f NFS`. Dokładne informacje o składni znajdują się w dokumentacji.

Podane niżej przykładowe polecenie montuje system plików znajdujący się w katalogu `/usr/public` systemu `artemis` w katalogu lokalnym `/usr/artemis`:

```
mount -F nfs artemis:/usr/public /usr/artemis
```

Katalog `/usr/artemis` musi zostać wcześniej utworzony.

opcje mogą być kombinacją następujących wartości:

- rw** montuje system plików do odczytu i zapisu (ustawienie domyślne);
- ro** montuje system plików tylko do odczytu;
- timeo=x** określa maksymalny czas, jaki należy poczekać przed daniem za wygraną, jeśli montowanie się nie powiedzie (w dziesiątych sekundach);
- retry=x** w razie niepowodzenia powoduje powtórzenie próby montowania x razy;
- soft** wymusza poddanie się, jeśli od serwera nie przychodzi potwierdzenie zamontowania;
- hard** powoduje, że klient kontynuuje próby montowania aż do skutku;
- intr** pozwala na przerwanie montowania przez naciśnięcie dowolnego klawisz; w przeciwnym przypadku próby montowania są powtarzane do skutku.

Poniższe polecenie powoduje próbę zamontowania katalogu `/usr/public` udostępnianego w systemie `artemis` w trybie tylko do odczytu, i poddanie się, jeśli komputer zdalny nie potwierdzi zamontowania systemu plików:

```
mount -F nfs -o soft,ro artemis:/usr/public /usr/artemis
```

Polecenie `mount` bez parametrów wyświetla listę wszystkich zamontowanych systemów plików.

Istnieje jeszcze prostsza metoda montowania często używanych systemów plików. Jeśli nazwy systemów plików i punkty ich zamontowania wraz z opcjami zostaną umieszczone w pliku `/etc/fstab` lub `/etc/vfstab` (zależnie od wersji Linuxa), będzie można je zamontować wydając polecenie `mount` tylko z jednym parametrem – punktem zamontowania, na przykład

```
mount /usr/artemis
```

Podsumowanie

Jak widać, usługa NFS nie jest aż tak trudna w konfiguracji, jak się powszechnie uważa. W ciągu kilku minut możesz skonfigurować system tak, aby wygodnie dzielić pliki i katalogi z innymi użytkownikami. NFS to szybki i wygodny sposób dostępu do aplikacji i plików zapisanych w innych systemach. Choć szczegóły konfiguracji zależą od wersji systemu, rzut okiem na strony `man` pozwoli Ci ustalić odpowiednią składnię poszczególnych poleceń przedstawionych w tym rozdziale.

NIS oraz YP, pozwalające na łatwiejszy dostęp do systemu z dowolnego komputera podłączonego do sieci, opisane są w rozdziale 44. „NIS i YP”.

Rozdział 46. „cron i at”, przedstawia sposoby automatyzowania zadań, które muszą być wykonywane regularnie.

Konfigurowanie serwera internetowego omówione jest w rozdziale 47. „Konfigurowanie węzła internetowego”

Rozdział 44.

NIS i YP

Tim Parker

W tym rozdziale:

- υ Konfiguracja domeny NIS
- υ Programy rezydentne obsługujące NIS
- υ Konfigurowanie komputera-nadzorcy NIS
- υ Konfigurowanie nadzorców zastępczych
- υ Klienci NIS

System NIS (ang. *Network Information Service*, sieciowy system informacyjny) umożliwia dostęp do plików, które normalnie byłyby dostępne tylko lokalnie, z dowolnego punktu sieci. Takie rozwiązanie ułatwia życie zarówno użytkownikom, jak i administratorom systemu. Podstawą systemu NIS jest wspólny dla całej sieci plik z hasłami, dzięki czemu użytkownik nie musi posiadać konta w każdym z podłączonych do sieci komputerów. Zamiast tego komputer-nadzorca (ang. *NIS master*) kontroluje dostęp do wszystkich innych komputerów.



YP (Yellow Pages) to system będący poprzednikiem NIS. Nazwa systemu została zmieniona ze względu na prawa autorskie, ale większość terminologii używanej w opisie systemu NIS pochodzi z systemu YP.

W tym rozdziale dowiesz się, w jaki sposób skonfigurować system NIS w prostej sieci. Niestety, nie wszystkie sieci są proste – w niektórych przypadkach szczegóły architektury i konfigurowania sieci mogą stać się bardzo skomplikowane. Choć ogólne zasady działania systemu NIS pozostają takie same, konfigurowanie bardziej złożonych sieci może wymagać dodatkowych zabiegów – co gorsza innych praktycznie w każdym przypadku. Dlatego pozostawimy ten temat na boku, skupiając się na prostych sieciach, do których zwykle podłączone są komputery linuxowe.

Nazwy plików używanych do konfigurowania systemu NIS zebrane są w tabeli 44.1.

Tabela 44.1. Pliki używane do konfigurowania systemu NIS

Nazwa pliku	Funkcja
/etc/ethers	Odwzorowywanie adresów Ethernet MAC na IP
/etc/group	Informacje o prawach dostępu dla grup
/etc/hosts	Odwzorowywanie adresów IP na nazwy symboliczne
/etc/netmasks	Maski IP
/etc/passwd	Informacje o kontach użytkowników
/etc/protocols	Odwzorowywanie protokołów i numerów sieciowych
/etc/rpc	Numery RPC
/etc/services	Odwzorowywanie numerów portów na protokoły TCP/IP

Podczas konfigurowania komputera-nadzorcza i nadzorców zastępczych przyjrzymy się najczęściej używanym plikom konfiguracyjnym, zobaczymy również, jakie modyfikacje należy wprowadzić do plików konfiguracyjnych komputerów mających pracować w systemie NIS.

Konfiguracja domeny NIS

W systemie NIS do logicznego grupowania komputerów używa się pojęcia domeny. Domena NIS zawiera jeden komputer skonfigurowany jako nadzorca i jeden lub więcej komputerów skonfigurowanych jako nadzorce zastępcze. Nadzorca zastępczy przejmuje wszystkie funkcje nadzorcza w takich sytuacjach, jak załamanie systemu czy problemy z siecią. Dzięki takiemu rozwiązaniu system działa przez cały czas, bez względu na dostępność komputera-nadzorcza. Konfigurowanie nadzorcza zastępczego jest bardzo proste, więc w każdej sieci powinien znaleźć się co najmniej jeden komputer pełniący tę rolę. Można również podzielić obsługę NIS pomiędzy komputer-nadzorcę i jego zastępcę. Zmiany w plikach konfiguracyjnych nadzorcza są automatycznie rozprowadzane do zastępców.

Obszar domeny NIS nie musi pokrywać się z domeną internetową, choć z taką sytuacją mamy do czynienia w większości systemów (czyli, innymi słowami, cała sieć jest domeną NIS). Domena NIS musi posiadać nazwę, która również może, ale nie musi, pokrywać się z nazwą domeny internetowej. Zamiast jednej dużej domeny można też stworzyć mniejsze domeny NIS, na przykład osobną dla każdego wydziału w przedsiębiorstwie.

Aby skonfigurować domenę NIS, powinieneś najpierw wybrać dla niej jakąś nazwę i ustalić numer IP komputera-zarządcy i zarządcy zastępczego. Jeśli konfigurujesz więcej niż jedną domenę, musisz zdecydować, które komputery będą należały do której domeny.

ny. Każdy komputer w obrębie domeny musi zostać wyszczególniony w odpowiednim pliku konfiguracyjnym.

Konfiguracja domeny NIS wymaga zalogowania się w każdym systemie podłączonym do sieci i ustawienia we wszystkich systemach takiej samej nazwy domeny za pomocą polecenia:

```
domainname nazwa_domeny
```

Aby ustawić nazwę domeny, musisz mieć uprawnienia administratora. Ponieważ jednak takie polecenie zachowuje ważność tylko do momentu zrestartowania komputera, najlepiej zapisać je w jednym z plików `rc` (zwykle znajdujących się w katalogu `/etc/rc.d`). Szczegóły dotyczące tej operacji mogą różnić się w zależności od wersji Linuxa.

W niektórych wersjach Linuxa dostępne są skrypty automatyzujące te zadania, na przykład w systemie Caldera OpenLinux dostępny jest program narzędziowy, po uruchomieniu którego trzeba tylko podać nazwę domeny i adresy IP komputera-nadzorcy i zastępców (program ten ogranicza ich liczbę do najwyżej dwóch).

Programy rezydentne obsługujące NIS

System NIS korzysta z kilku programów rezydentnych działających w systemie komputera-nadzorcy i jego zastępców. Jednym z nich jest program `ypserv`, obsługujący wszystkie żądania pochodzące od komputerów-klientów.

Po stronie klientów działa program `ypbind`, który jest odpowiedzialny za łączenie się z nadzorcą podczas uruchamiania systemu i podejmowanie wszelkich działań związanych z przesyaniem danych o użytkownikach i konfiguracji sieci. Proces łączenia się programu `ypbind` z komputerem-nadzorcą nazywany jest wiązaniem (ang. *binding*).

Proces wiązania się z nadzorcą rozpoczyna się w momencie, kiedy komputer-klient wysyła do wszystkich komputerów informację o tym, że poszukuje adresu IP nadzorcy systemu NIS. Nadzorca w odpowiedzi na taką informację powinien przesłać swój numer IP i numer portu, na którym może nastąpić połaczenie. Jeśli odezwie się więcej niż jeden nadzorca, używany jest numer pierwszego z nich. Jeśli nikt się nie odezwie, informacja jest ponownie wysyłana do sieci – aż do skutku.

Aby sprawdzić, z którym komputerem-nadzorcą system jest aktualnie związany, należy wydać polecenie `ypwhich`:

```
$ ypwhich  
merlin
```

Konfigurowanie komputera-nadzorcy NIS

Konfigurowanie komputera-nadzorcy NIS jest czynnością dość prostą. Należy zacząć od sprawdzenia, czy wszystkie informacje w plikach `/etc/passwd` i `/etc/group` są aktualne; usuń wpisy o wszystkich nieistniejących już użytkownikach i grupach, sprawdź, czy wszystkie katalogi domowe i interpretery poleceń uruchamiane przy logowaniu są ustalone prawidłowo. Zwróć uwagę na to, czy dla każdego konta założone jest hasło. Błąd w plikach konfiguracyjnych za pomocą NIS rozprzestrzenia się na całą sieć, znacznie obniżając poziom bezpieczeństwa.

Po przygotowaniu plików upewnij się, że jesteś zalogowany jako `root` – tylko wtedy będziesz posiadał odpowiednie prawa dostępu i pełny dostęp do systemu. Pliki mapowania systemu NIS generowane są z plików istniejących w systemie za pomocą programu `ypinit` z opcją `-m` (która oznacza, że komputer ma być nadzorcą, ang. *master*):

```
/usr/sbin/ypinit -m
```

Ścieżka dostępu do programu `ypinit` może być inna w Twoim systemie – sprawdź ją, jeśli po wydaniu powyższego polecenia generowany jest komunikat o błędzie.

Po uruchomieniu, program `ypinit` przegląda wszystkie pliki NIS wymienione w pliku `/var/yp` i generuje pliki mapowania, które będą używane przez procesy klientów. W niektórych systemach plik spełniający funkcję taką, jak `/var/yp`, nazywa się nieco inaczej – na przykład w systemie SCO UNIX jest to jeden z plików zapisanych w katalogu `/etc/yp`; powinieneś to sprawdzić w dokumentacji. Plik `/var/yp` zawiera listę plików mapowania, które mają zostać wygenerowane, i zwykle nie wymaga modyfikacji.

Program `ypinit` tworzy nowy katalog, zazwyczaj o nazwie `/var/yp/nazwa_domeny`, w którym zapisuje wszystkie wygenerowane pliki. Jeśli konfigurujesz kilka domen obsługiwanych przez tego samego nadzorce, pliki utworzone dla każdej z domen zostaną umieszczone w osobnym podkatalogu.

Na koniec program `ypinit` zapyta o komputery spełniające funkcję nadzorców następujących – powinieneś podać ich nazwy. Nazwy te są zapisywane w pliku w katalogu `/var/yp/nazwa_domeny`.



W niektórych wersjach Linuksa pliki związane z NIS znajdują się w innych niż przykładowe katalogach – jeśli nie potrafisz ich znaleźć, użyj polecenia `find`.

Po wygenerowaniu odpowiednich plików możesz uruchomić program rezydentny `ypserv`. Najlepiej zautomatyzować ten proces, wpisując do pliku `rc` polecenie (które być może już w nim się znajduje)

```
if [ -f /etc/yp/ypserv -a -d /var/yp/`domainname` ]
then
    /etc/yp/ypserv
fi
```

Powyższy fragment skryptu powoduje uruchomienie programu `ypserv`, o ile jest on zainstalowany w systemie i zostały wygenerowane odpowiednie pliki. W pierwszym wierszu nazwa polecenia `domainname` otoczona jest odwróconym pojedynczym cudzysłowem, dzięki czemu polecenie to zostanie wykonane, a w miejsce, w którym się ono znajduje, zostanie podstawiony wynik jego wykonania (czyli faktyczna nazwa domeny). Jeśli odpowiedni katalog istnieje, uruchamiany jest program `ypserv`. Powinieneś podstawić właściwe dla Twojego systemu ścieżki dostępu.

Można również uruchomić program `ypserv` ręcznie (jeśli jesteś zalogowany jako `root`), wydając polecenie

```
/etc/yp/ypserv
```

Następnie w systemie komputera-nadzorcy należy uruchomić program `ypbind` (w przeciwnym przypadku program `ypserv` nie będzie potrafił odnaleźć plików mapowania). W tym celu możesz do jednego z plików `rc` dopisać następujące wiersze:

```
if [ -d /var/yp ]
then
    /etc/yp/ypbind
fi
```

Tu również powinieneś w razie potrzeby podstawić odpowiednie dla Twojego systemu ścieżki dostępu. Jeśli jesteś zalogowany jako `root`, możesz uruchomić program `ypbind` z wiersza poleceń. Upewnij się, że ścieżka dostępu jest prawidłowa.

Jeśli chcesz szybko sprawdzić, czy programy obsługi NIS działają poprawnie, wydaj polecenie:

```
ypmatch reksio passwd
```

Polecenie `ypmatch` wysyła do nadzorcy NIS żądanie, by dopasował do pierwszego argumentu wiersz z pliku podanego jako drugi argument. W przykładzie sprawdzony zostanie plik `passwd` (`passwd` to w zasadzie alias nazwy `passwdbyname`) w poszukiwaniu wiersza zaczynającego się od tekstu `reksio`. Jako wynik działania tego polecenia powinien zostać wyświetlony odpowiedni wiersz.

Konfigurowanie nadzorców zastępczych

Aby skonfigurować nadzorcę zastępczego, komputer-nadzorca musi już być skonfigurowany i działać. Gdy jesteś pewny, że pracuje on prawidłowo, zaloguj się jako administrator systemu, który ma działać jako nadzorca zastępczy. Wcześniej należy również ustawić nazwę domeny – sprawdź więc, czy istnieje odpowiednie odwołanie do programu `domainname` w jednym z plików `rc` lub uruchom ten program samodzielnie.

Aby skonfigurować nadzorcę zastępczą i skopiować wszystkie potrzebne pliki konfiguracyjne z komputera-nadzorcy, wydaj polecenie (podstawiając ścieżkę dostępu właściwą dla Twojego systemu):

```
/etc/yp/ypbind
```

Za pomocą polecenia `ypwhich` sprawdź, czy połączenie przebiegło prawidłowo. Powinno ono zwrócić nazwę komputera-nadzorcy. Na koniec wydaj polecenie:

```
/etc/yp/ypinit -s nazwa_komputera-nadzorcy
```

Opcja `-s` powoduje, że dany komputer będzie zachowywał się jak nadzorca zastępczy. Zostaną założone odpowiednie katalogi, a z komputera-nadzorcy przesłane zostaną wszystkie potrzebne pliki mapowania.

Po zakończeniu tych czynności możesz sprawdzić, czy wszystko jest w porządku, używając podobnie jak w poprzednim podrozdziale polecenia `ypmatch`:

Aby regularnie aktualniać pliki mapowania w komputerach-zastępcach, używa się polecenia `ypxfer`. Argumentem tego polecenia powinna być nazwa pliku, który ma być przesyłany, np.:

```
ypxfer passwdbyname
```

Większość administratorów automatyzuje uaktualnianie plików mapowania albo dodając odpowiedni wpis do pliku `crontab`, dzięki czemu transfer odbywa się regularnie (zwykle w nocy), albo też umieszczając odpowiednie polecenia w skrypcie uruchamianym przez administratora.

Klienci NIS

Konfigurowanie komputera korzystającego z systemu NIS wymaga prawidłowego ustalenia nazwy domeny (za pomocą programu `domainname`, który może być uruchamiany w jednym z plików `rc`) oraz wydania polecenia `ypbind`, wiążącego komputer z odpowiednim komputerem-nadzorcą.

Kiedy zachodzi potrzeba znalezienia jakiegoś wpisu w pliku `/etc/passwd` lub `/etc/group`, najpierw przeszukiwane są pliki lokalne, a w razie niepowodzenia do nadzorcy wysyłane jest zapytanie. Aby było to możliwe, w pliku `/etc/passwd` musi znajdować się wpis:

```
+*:0:0:::
```

Zgodnie z formatem pliku `/etc/passwd`, jest to prawidłowy wpis, choć nie zawiera żadnych informacji. Znak `+` poleca przesyłać zapytanie do nadzorcy NIS. Może on znajdować się w dowolnym miejscu w pliku – po dojściu do niego wysyłane jest zapytanie, a jeśli ono nie przyniesie rezultatu, przeszukiwanie pliku jest kontynuowane.

Podsumowanie

Konfigurowanie systemu NIS nie jest bardzo złożone. Najtrudniejszą częścią jest doprowadzenie do odpowiedniej postaci plików używanych do generowania plików mapowania (głównie przez usuwanie starych wpisów) i zapewnienie odpowiedniego poziomu bezpieczeństwa. Konfigurowanie komputera-nadzorcy i nadzorcy zastępczego w sumie nie trwa dłużej niż pół godziny. Często odnalezienie w systemie odpowiednich programów (w różnych wersjach znajdują się one w różnych katalogach) zajmuje więcej czasu niż sama konfiguracja.

Współużytkowanie plików i katalogów za pośrednictwem usługi NFS opisane jest w rozdziale 43. „NFS”.

Konfigurowanie programów `cron` i `at`, pozwalających na automatyczne wykonywanie pewnych zadań w tle, omówione jest w rozdziale 46. „`cron` i `at`”.

Jak założyć i skonfigurować węzeł internetowy w oparciu o system Linux, dowiesz się z rozdziału 47. „Konfigurowanie węzła internetowego”.

Rozdział 45.

Kopie zapasowe

Tim Parker

W tym rozdziale:

- υ Po co tworzyć kopie zapasowe
- υ Inwentaryzowanie zapisanych danych
- υ Używanie programu `tar` do tworzenia kopii zapasowych

Trzy podstawowe zasady administratora systemu to: 1) należy wykonywać kopie zapasowe, 2) należy wykonywać kopie zapasowe i 3) należy wykonywać kopie zapasowe. Liczba osób, które utraciły wartościowe dane, nie wspominając o czasie, które spędziły na mozołnym odtwarzaniu plików konfiguracyjnych, jest ogromna. Nawet jeśli nie posiadasz napędu taśmowego czy innego odpowiednio pojemnego nośnika, powinieneś przyzwyczajać się do wykonywania kopii zapasowych swoich danych. Jak to robić – dowiesz się w tym rozdziale.

Jeżeli z Twojego systemu korzysta wielu użytkowników, jesteś podłączony do sieci, działa poczta itd., to chyba nie masz wątpliwości, że kopie zapasowe powinny być wykonywane nawet codziennie. Jeżeli jednak system służy głównie do zabawy, sprawa ta nie jest aż tak poważna – w razie problemów tracisz „tylko” wszystkie pliki konfiguracyjne. Jednak kopie zapasowe należy wykonywać i w tym przypadku, co najwyżej z mniejszą częstotliwością.

Po co tworzyć kopie zapasowe

Kopia zapasowa może zawierać dane z całego systemu plików lub też jego fragmentu i używana jest po to, aby w razie potrzeby można było odzyskać potrzebne pliki. W większości systemów do przechowywania danych używa się napędów taśmowych, ale nadają się do tego również wyjmowalne dyski twarde czy nawet dyskietki.

Źródeł zagrożeń w nowoczesnym systemie komputerowym jest bardzo wiele, od awarii dysku twardego, przez przerwy w zasilaniu do zwykłych pomyłek. Kopia zapasowa okazuje się wówczas wręcz wybawieniem. Choć często trudno jest zmusić się do jej utworzenia, zwykle czas stracony na to zwraca się z nawiązką w przypadku problemów. Można również uprościć sobie pracę używając programu `cron`.

Jednym z zagrożeń w systemie Linux jest sama natura systemu: ponieważ jest to system wielozadaniowy i wielodostępny, równocześnie może być otwartych wiele plików. Przez cały czas odbywa się wymiana danych pomiędzy pamięcią i dyskiem twardym (nawet wtedy, gdy nie jest zalogowany żaden użytkownik i nie działa żaden program uruchomiony przez użytkownika).

Wiele informacji o systemie plików jest również przechowywanych w pamięci. Choć są one często zapisywane na dysku, to jednak nie następuje to natychmiast, w efekcie czego przerwanie tego procesu może prowadzić do utraty zawartości plików systemowych. Pliki na dysku mogą pozostać w stanie przejściowym, nie odpowiadającym rzeczywistemu stanowi systemu plików.

Choć uszkodzenie systemu plików może być spowodowane przez najróżniejsze czynniki – z których nie wszystkie są zależne od administratora systemu – zadaniem administratora jest nie dopuścić do utraty ważnych danych niezależnie od okoliczności i w razie problemów odtworzyć cały system plików w możliwie krótkim czasie.

Wałą kwestią jest również zdecydowanie, gdzie będą przechowywane nośniki zawierające kopie zapasowe systemu. Większość administratorów (głównie używających systemu linuxowego w domu) wykonuje kopie systemu na takich nośnikach, jak taśmy czy dyskietki, i przechowuje je w tym samym miejscu, w którym znajduje się system. Należy upewnić się, że nie występują tam silne pola magnetyczne (występujące w pobliżu różnych urządzeń, na przykład głośników, monitorów, modemów czy telewizorów), nie jest za ciepło itd. Warto jednak rozważyć przechowywanie nośników gdzieś zupełnie indziej – takie rozwiązanie pozwala im przetrwać również poważniejsze (ale i o wiele rzadsze) katastrofy, takie jak na przykład pożar, który może zniszczyć zarówno system, jak i kopie danych.

Nośniki

Nośnikiem używanym najczęściej do tworzenia kopii zapasowych jest taśma. Jest ona popularna ze względu na stosunkowo niską cenę, niewygórowane wymagania co do warunków przechowywania i rozsądną prędkość transmisji danych. Proces zapisu i odczytu danych jest pewny, a taśmy są przenośne pomiędzy systemami. Niestety, aby używać taśm, trzeba posiadać napęd taśmowy. Jeśli nie dysponujesz napędem taśmowym, musisz rozważyć inne rozwiązania.

Jedną z alternatyw są dyski wyjmowalne różnych typów, takie jak Iomega Bernoulli czy ZIP. Dyski z danymi, zwykle zapakowane w specjalne pojemniki, mogą być wyjmowane i przechowywane z dala od systemu. Niektóre z tych dysków, podobnie jak taśmy, pozwalają na zapis cykliczny.

Inna możliwość to po prostu użycie dodatkowego dysku twardego – ceny tych urządzeń stale spadają. Można więc do Twojego komputera (lub dowolnego innego komputera w sieci) podłączyć dysk przeznaczony wyłącznie do przechowywania kopii zapasowych i zapisywać na nim pełną kopię systemu. Nagrywarki dysków CD-ROM i WORM również dobrze nadają się do tworzenia kopii zapasowych.

Dyskietki to ostatnia deska ratunku w większych systemach. Nie są jednak złym rozwiązaniem do przechowywania niewielkich plików. Choć pojawiło się kilka napędów o większej gęstości zapisu, brak sterowników dla systemu Linux nie zachęca do ich użycia.

Harmonogram tworzenia kopii zapasowych

Jednym z najważniejszych aspektów tworzenia kopii zapasowych jest regularność. Jest to szczególnie ważne w systemach, z których korzysta wielu użytkowników, i w których zawartość systemu plików zmienia się bardzo intensywnie. Jeśli tylko Ty używasz systemu, możesz tworzyć kopie wtedy, kiedy uznasz to za stosowne.

Dla większości systemów wykorzystywanych przez kilku użytkowników, stale komunikujących się z Internetem i dokonujących rutynowych operacji w systemie plików, konieczne jest codzienne tworzenie kopii zapasowych. Niekoniecznie pełnych – wystarczy, że zapiszesz tylko te pliki, które uległy zmianie od utworzenia ostatniej kopii (jest to tzw. system przyrostowy, ang. *incremental backup*).

Większość administratorów wykonuje kopie zapasowe w nocy lub wcześnie rano, ponieważ wtedy system jest stosunkowo najmniej obciążony (zarówno w sensie obciążenia procesora, jak i liczby otwartych plików). Proces ten można zautomatyzować poleceнием `cron` lub `at` (patrz rozdział 46. „*cron i at*”), co zwalnia operatora od konieczności jego ręcznej obsługi i umożliwia wybranie najdogodniejszej z punktu widzenia systemu pory. Po wykonaniu operacji wystarczy tylko sprawdzić, czy wszystko poszło dobrze, zanotować dane o nowej kopii i wymienić nośnik.

W systemach mało obciążonych i takich, z których korzysta jedna osoba, kopie można wykonywać w dowolnym momencie, ale nadal warto zautomatyzować ten proces, o ile system pracuje nieprzerwanie. Jeśli Twój komputer jest włączony tylko wtedy, gdy go używasz, powinieneś przyzwyczaić się do tworzenia kopii zapasowych w czasie, gdy robisz coś innego.

Byli użytkownicy systemów DOS i Windows często podchodzą do problemu tworzenia kopii zapasowych w ten sposób, że każdorazowo zapisują nowe dane na tym samym nośniku. Praktyka ta nie ma większego sensu – posiadanie jednej tylko kopii danych bardzo często okazuje się niewystarczające, nie pozwala bowiem wrócić do poprzednich wersji plików. Przypuśćmy, że usunąłeś przed tygodniem jakiś plik, zapisując go wcześniej dla bezpieczeństwa na taśmie. Teraz orientujesz się, że był to błąd, ale nie masz już żadnych szans odzyskania pliku, ponieważ na taśmie zapisane są wcześniejsze dane (takie postępowanie zabezpiecza tylko przed awarią, a biorąc pod uwagę złożliwość rzeczy martwych, jeśli już zdarzy się awaria, to pewno okaże się, że taśma również uległa uszkodzeniu – *przyp. tłum.*).

W idealnym przypadku kopie powinny być przechowywane przez kilka dni - lub nawet tygodni - przed ponownym użyciem nośnika. Jest to szczególnie ważne w systemach wykorzystywanych przez wielu użytkowników, zdarza się bowiem, że ktoś przypomina sobie, że dwa miesiące temu usunął plik, bez którego nie może żyć - a Ty właśnie dwie minuty temu skończyłeś zapisywać na tej taśmie dzisiejszy stan systemu. Istnieją metody na zabezpieczenie się przed takimi przypadkami - omówimy je za chwilę. To, jaki system tworzenia kopii powinieneś zastosować, zależy od przeznaczenia systemu, ale generalnie powinna to być co najmniej jedna pełna kopia co tydzień, uzupełniona codziennymi kopiami przyrostowymi, przy czym kopie powinny być przechowywane na co najmniej dwa tygodnie wstecz.

Wykonanie pełnej kopii systemu sprawdza się do utworzenia obrazu całej zawartości systemu plików i wymaga nośnika o pojemności odpowiadającej wielkości tego ostatniego. W celu zmniejszenia wymaganej objętości nośnika można użyć kompresji danych, jednak możliwość ta nie zawsze jest dostępna. Inną alternatywą jest użycie kilku jednostek nośnika (taśm). Ponieważ jednak program `cron` nie jest w stanie zmieniać ich automatycznie, przy tworzeniu kopii w takim systemie niezbędna jest obecność administratora. Nietrudno się domyślić, że wykonanie pełnej kopii zapasowej na wielu nośnikach o małej pojemności jednostkowej (na przykład na dyskietkach) jest procesem żmudnym i długotrwałym.

Kopie przyrostowe (czasem również zwane różnicowymi, ang. *differential backup*) zawierają jedynie te pliki, które zmieniły się od momentu utworzenia poprzedniej kopii. Nie we wszystkich systemach pliki posiadają atrybut pozwalający to łatwo stwierdzić (atrybut taki nie występuje również w najpowszechniejszym używanym w Linuxie systemie plików `ext2`), ale wtedy z pomocą może przyjść data ostatniej modyfikacji pliku.

Utworzenie kopii przyrostowej w Linuxie nie jest rzeczą łatwą, chyba że ograniczysz się do części systemu plików, w której dokonywanych jest najwięcej modyfikacji, na przykład `/home`. Takie rozwiązanie to tzw. kopia częściowa (ang. *partial backup*). Zauważ, że w dowolnym systemie można tworzyć kopie przyrostowe za pomocą procesu działającego cały czas w tle i rejestrującego wszystkie zmiany w systemie w pliku, a następnie zapisującego pliki, które się zmieniły. Oplacalność takiego rozwiązania jest jednak dyskusyjna.

Jak często należy robić kopie zapasowe? Najprostsza reguła jest taka, że nie powinieneś dopuścić do utraty informacji. Dla wielu oznacza to kopiowanie codzienne. Założmy, że pracując nad dokumentem lub programem tracisz wszystkie dane wprowadzone od czasu zapisania ostatniej kopii zapasowej. Ile czasu zajmie Ci ich odtworzenie? Jeśli więcej, niż zajęłoby zrobienie kopii zapasowej, zrób kopię. W pozostałej części tego rozdziału przyjmiemy, że nośnikiem jest taśma magnetyczna, ale nasze rozważania odnoszą się również do innych mediów.

Jaki jest zatem najwydajniejszy system tworzenia kopii zapasowych, zakładając, że konieczne jest ich regularne tworzenie? Dla średniej wielkości systemów (obsługujących kilku użytkowników nie generujących nadmiernego „ruchu” w systemie plików) warto zalecić codzienne tworzenie kopii, co wymaga od 10 do 14 taśm, zależnie od tego, czy kopie będą tworzone również w weekendy.

Każdą taśmę należy opisać zgodnie z jej przeznaczeniem, np. „Dzień 1”, „Dzień 2” itd. Taśm tych używaj cyklicznie, dzięki czemu będziesz mógł zawsze wrócić do stanu sprzed dwóch tygodni (przy założeniu, że kopie nie są tworzone w weekendy). Jeśli posiadasz więcej taśm, możesz wydłużyć cykl tworzenia kopii. Metoda taka jest używana w wielu dużych organizacjach, ponieważ charakteryzuje się dobrym wyważeniem pomiędzy kosztami, szybkością i bezpieczeństwem oraz możliwościami odzyskania danych.

Zależnie od potrzeb można wykonywać kopie pełne lub częściowe. Dobrym rozwiązaniem jest tworzenie jednej pełnej kopii systemu po utworzeniu około pięciu kopii częściowych. Można na przykład robić pełne kopie systemu w poniedziałki, w pozostałe dni tygodnia ograniczając się do wykonywania kopii katalogu `/home`. Wyjątki powinieneś robić wówczas, gdy wprowadzasz jakieś zmiany do konfiguracji systemu – wtedy należy zrobić jego pełną kopię. Warto również prowadzić dokładną dokumentację tworzonych kopii zapasowych – ten temat omówimy za chwilę.

Rozszerzeniem przedstawionego wyżej harmonogramu wykonywania kopii zapasowych może być dodanie oprócz cyklu dziennego również cyklu dwutygodniowego. Posiadane nośniki należy podzielić wówczas na dwie grupy. Jeśli na przykład posiadasz czternaście taśm, możesz używać dziesięciu z nich do codziennego tworzenia kopii zapasowych, dokładnie tak samo, jak w poprzednim przykładzie. Opisz je na przykład „Dzień 1” ... „Dzień 10”. Pozostałe cztery taśmy zostaną wykorzystane do utworzenia cyklu dwutygodniowego – można je opisać „Tydzień 1” ... „Tydzień 4”.

Używając tak zmodyfikowanego harmonogramu wykonywania kopii zapasowych, należy wykonywać codzienne kopie tak samo, jak w poprzednim przykładzie, ale po dojściu do końca cyklu codziennego (czyli po zapisaniu kopii na taśmę z etykietą „Dzień 10”) należy wykonać kopię zapasową na kolejnym nośniku wchodzący w skład cyklu dwutygodniowego. Pełny cykl powinien wtedy wyglądać tak: „Dzień 1” ... „Dzień 10” „Tydzień 1” „Dzień 1” ... „Dzień 10” „Tydzień 2” „Dzień 1” itd.

Taki harmonogram ma dość znaczną przewagę nad omówionym wcześniej codziennym cyklicznym wykonywaniem kopii zapasowych. Po zakończeniu całego cyklu, posiadasz dziesięć wykonywanych codziennie kopii, pozwalających wrócić do stanu systemu sprzed dwóch tygodni. Posiadasz również taśmy, których używałeś co dwa tygodnie, dzięki którym masz również szansę na odzyskanie pliku usuniętego osiem tygodni wcześniej. Daje to możliwość odzyskania plików, których uszkodzenie czy usunięcie nie zostało wykryte od razu. Dołączając następne taśmy można wydłużyć cykl dwutygodniowy albo dodać cykl 40-dniowy.

Inwentaryzowanie zapisanych danych

Wielu administratorów zaczyna swoją karierę robiąc grzecznie kopie zapasowe, ale gdy przychodzi im odzyskać jakiś konkretny plik, nie mają pojęcia, gdzie go szukać. Niektórzy radzą sobie z tym w ten sposób, że naklejają na taśmy kartki, na których zapisana jest data wykonania kopii i jej zawartość. Oznacza to jednak, że szukając konkretnego pliku musisz przerzucić cały stos taśm, co może być dość męczące, szczególnie jeśli

taśm jest dużo. Z tego powodu dobrze jest przechowywać gdzieś dokładny spis tego, co na której taśmie się znajduje (odnosi się to do wszystkich kopii zapasowych, zarówno wykonywanych w systemie Linux, jak i DOS czy Windows).

Wykonanie każdej kopii zapasowej powinno wiązać się z uaktualnieniem takiego spisu. Nie musi on być szczególnie złożony ani długi. Wystarczy kilka kolumn danych w notesie czy też przechowywanych w postaci pliku tekstowego na dysku twardym (taki plik należy oczywiście regularnie drukować). Minimalny zestaw informacji, które musisz posiadać, to:

- υ data utworzenia kopii;
- υ nazwa taśmy, na której kopia się znajduje (na przykład „Dzień 1”);
- υ nazwa systemu plików, który został skopiowany;
- υ informacja o tym, czy kopia jest kopią pełną, przyrostową czy częściową, i ewentualnie nazwy katalogów, z których pochodziły archiwizowane dane.

Zapisanie tych informacji trwa tylko kilka sekund, a może zaoszczędzić naprawdę sporo czasu. W większych systemach należy również zanotować:

- υ kto wykonał kopię;
- υ czy kopia była wykonana automatycznie, czy ręcznie;
- υ miejsce, w którym przechowywana jest taśma.

Dzięki zapisywaniu dat można z łatwością przywrócić określona wersję pliku – jeśli na przykład użytkownik wie, że usunął plik tydzień temu, na podstawie daty możesz ustalić, na której z taśm znajduje się ostatnia wersja tego pliku.

Zapisane informacje o kopiiach zapasowych powinny być dla wygody przechowywane w pobliżu systemu, ale niektórzy administratorzy wolą, by znajdowały się one tam, gdzie przechowywane są nośniki. Czasem również – na wypadek katastrofy – administratorzy przechowują kopię danych o taśmach gdzieś indziej.

Używanie programu tar do tworzenia kopii zapasowych

Do tworzenia kopii zapasowych zwykle używa się programu `tar` (ang. *tape archiver*). Potrafi on utworzyć pojedynczy, duży plik, zawierający dane z wszystkich archiwizowanych plików, wraz z danymi o ich położeniu, prawach dostępu itd. (podobnie jak robi to program PKZIP w systemie DOS). Potrafi obsłużyć tylko ten typ archiwów, który sam tworzy.

Używanie tego polecenia jest dość uciążliwe ze względu na dużą liczbę opcji, ale na szczęście w codziennej pracy trzeba używać tylko kilku ich kombinacji. Ogólnie rzecz biorąc, składnia polecenia `tar` ma postać:

```
tar czynność opcje pliki
```

Argument `pliki` określa nazwy plików lub katalogów, które należy zarchiwizować lub odtworzyć. Zwykle jest to cały podsystem plików (jak np. `/home`) lub też pojedynczy plik (np. `/home/reksio/wazny_plik`).

Pole `czynność` określa działanie programu `tar` i może przyjmować jedną z następujących wartości:

- c** utworzenie nowego archiwum,
- r** dołączenie informacji do istniejącego archiwum,
- t** wyświetlenie listy plików zapisanych w archiwum,
- u** dodanie plików do archiwum, o ile nie są już zarchiwizowane,
- x** przywracanie plików zapisanych w archiwum.

Opcje pozwalają kontrolować w pewnym zakresie sposób postępowania z archiwum. Dostępne wartości to:

- A** zapobiega zapisywaniu bezwzględnych ścieżek dostępu do plików;
- b** pozwala podać wielkość bloku (1 – 20);
- e** zapobiega dzieleniu archiwów na kilka nośników;
- f** pozwala podać nazwę pliku lub urządzenia, na którym zostanie zapisane archiwum;
- F** pozwala podać nazwę pliku zawierającego inne opcje;
- k** pozwala podać wielkość archiwum (w kilobajtach);
- l** wyświetla komunikaty o błędach w razie znalezienia „wiszących” dowiązań (nie prowadzących do żadnych istniejących plików ani katalogów);
- m** nie przywraca czasu modyfikacji;
- n** oznacza, że archiwum nie zapisane jest na taśmie;
- p** przywraca pliki wraz z informacjami o prawach dostępu;
- v** wyświetla komunikaty o aktualnie przetwarzanych plikach;
- w** wymaga potwierdzenia przed podjęciem akcji.

Program `tar` w większości przypadków używa bezwzględnych ścieżek dostępu do plików (chyba że zostanie podana opcja `A`).

Kilka przykładów pomoże Ci zrozumieć użycie tego programu i jego opcji. Jeśli używasz urządzenia `/dev/tape` i chcesz sporządzić pełną kopię zapasową systemu na nośniku o pojemności większej niż rozmiar systemu plików, możesz wydać polecenie:

```
tar cf /dev/tape /
```

Utworzono zostanie nowe archiwum (opcja `c`) na nośniku w urządzeniu `/dev/tape` (opcja `f`). Dotychczasowa zawartość nośnika zostanie usunięta – i to bez żadnego ostrzeżenia, więc upewnij się, że nie usuwasz przez przypadek jakichś ważnych danych. Jeśli dodatkowo podana zostanie opcja `v`, program `tar` będzie wyświetlał nazwę i rozmiar pliku, który jest aktualnie zapisywany.

Jeśli chcesz przywrócić pliki zapisane za pomocą polecenia przedstawionego w powyższym przykładzie, wydaj polecenie:

```
tar xf /dev/tape
```

Spowoduje ono przywrócenie wszystkich zapisanych plików, ponieważ nie został wyszczególniony żaden specyficzny katalog ani plik. Jeśli chciałbyś odzyskać tylko pojedynczy plik, wydaj polecenie:

```
tar xf /def/tape /home/reksio/duzy_plik
```

które spowoduje odtworzenie tylko jednego pliku, o nazwie `/home/reksio/duzy_plik`.

Czasem potrzebna jest lista wszystkich plików zapisanych w danym archiwum. Możesz ją otrzymać wydając polecenie:

```
tar tvf /dev/tape
```

Polecenie to wykorzystuje opcję `v`, powodującą wyświetlanie nazw przetwarzanych plików. Ponieważ otrzymana lista może być dość długa, wygodnie jest skierować ją do pliku.

Teraz, kiedy zapoznałeś się już z podstawowymi możliwościami programu `tar`, pora na kilka bardziej praktycznych przykładów. Większość napędów taśmowych wymaga przy zapisie danych określenia wielkości bloku (przy odczycie parametr ten ustawiany jest automatycznie; oznacza on ilość danych, jaką można jednorazowo przesłać do urządzenia). Parametr ten określany jest za pomocą opcji `b`. Przykładowe polecenie:

```
tar cvfb /dev/tape 20 /home
```

tworzy nowe archiwum zapisane na nośniku w urządzeniu `/dev/tape`, przy czym wielkość bloku zapisywanych danych wynosi 20. Jest to wartość prawidłowa dla większości napędów taśmowych, dlatego można przyjąć ją jako wartość domyślną, chyba że akurat Twój napęd wymaga podania innej wartości. W przypadku użycia dyskietek i dysków twardych również należy podać odpowiednią wartość. Zauważ, że argumenty występujące po opcjach muszą pojawić się w takiej samej kolejności, jak odpowiadające im opcje: ponieważ opcja `f` wystąpiła przed `b`, nazwa urządzenia musi wystąpić przed wielkością bloku.

Często zdarza się, że nośnik nie może pomieścić całego systemu plików – w takim przypadku do utworzenia kopii potrzebna będzie więcej niż jedna taśma. Aby poinformować program `tar` o pojemności kasety, należy użyć opcji `k`. Jej argumentem jest właśnie wielkość nośnika w kilobajtach. Polecenie

```
tar cvbfk 20 /dev/tape 122880 /usr
```

mówią programowi `tar`, że powinien zapisać dane (w blokach o wielkości 20) na nośnik w urządzeniu `/dev/tape`, który może pomieścić 122880 kB danych. Po przekroczeniu tego rozmiaru zostaniesz poproszony o wymianę nośnika. Tu również można zauważyć, że porządek parametrów musi odpowiadać porządkowi opcji.

Użycie dyskietek jako nośnika danych nieco komplikuje problem, ponieważ wymagana wielkość bloku danych jest zwykle inna. Tworzone archiwia przeważnie nie mieszczą się w całości na dyskietce, dlatego należy użyć opcji `k` i podać pojemność nośnika. Przykładowo, jeśli chcesz wykonać kopię zapasową zawartości katalogu `/home/reksio` na dyskietkach o pojemności 1.2 MB, powinieneś wydać polecenie:

```
tar cufk /def/fd0 1200 /home/reksio
```

Podczas tworzenia archiwów na dyskietkach warto podać opcję `n`, która informuje program `tar`, że nie współpracuje aktualnie z napędem taśmowym – przyspiesza to nieco jego działanie.

Podsumowanie

W tym rozdziale przedstawiliśmy tylko najbardziej podstawowe zagadnienia dotyczące tworzenia kopii zapasowych. Jeśli chcesz, aby wyniki Twojej pracy były bezpieczne, musisz wykonywać kopie regularnie i notować, co znalazło się na której taśmie. Choć program `tar` jest na początku nieco nieprzyjemny w użyciu, dość łatwo nauczyć się jego obsługi (można również napisać kilka skryptów pozwalających na wykonanie najczęstszych czynności – *przyp. tłum.*).

Obecnie powstaje wiele skryptów i programów obsługiwanych za pomocą menu, ułatwiających dość znacząco tworzenie kopii zapasowych. Nie są one dołączane do dystrybucji Linuxa, ale są dostępne w każdym z węzłów FTP czy BBS oferujących oprogramowanie dla tego systemu.

Automatyzowanie tworzenia kopii zapasowych i innych codziennych zadań omówione jest w rozdziale 46. „`cron` i `at`”.

W rozdziale 47. „Konfigurowanie węzła internetowego” omawiamy podstawowe problemy związane z konfiguracją sieci i oprogramowania tak, byś mógł stać się pełnoprawnym członkiem Internetu.

O aplikacjach, których można używać w systemie Linux, przeczytać możesz w rozdziałach od 62. „`Adabas-D` i inne bazy danych” do 64., „`Program Lone Tar`”.

Rozdział 46.

cron i at

Tim Parker

W tym rozdziale:

- υ Używanie programu cron
- υ Program at

Automatyzowanie zadań administracyjnych jest jednym z warunków utrzymywania systemu w dobrej kondycji. Jeśli wszystkie zadania będą wykonywać się same, bez Twojej interwencji, w tle, administrowanie stanie się samą przyjemnością. Z tego właśnie powodu powstały programy cron i at. Oba pozwalają na uruchamianie programów o zadanej godzinie, bez interwencji użytkownika czy administratora.

Używanie programu cron

Program cron (jego nazwa jest skrótem angielskiego słowa *chronograph*) pozwala na automatyczne wykonywanie poleceń przez system o określonej godzinie. W tym celu jest on ładowany do pamięci jako program rezydentny podczas uruchamiania systemu (zwykle z jednego ze skryptów rc). Podczas działania odczytuje on dane o programach, które ma wykonać, i porach ich uruchomienia z pliku crontab.

Za każdym razem, gdy dane o dniu i godzinie wykonania programu zgadzają się z czasem systemowym, wykonywane jest odpowiednie polecenie. Jeśli następnego dnia czas wykonania zadania znów zgadza się z czasem systemowym, odpowiedni program ponownie zostanie wykonany. Jest to idealne wręcz rozwiązanie do takich zadań, jak tworzenie codziennych kopii zapasowych, aktualizacja baz danych czy „sprzątanie” w systemie plików (usuwanie zbędnych plików tymczasowych, plików, w których rejestrowane są pewne zdarzenia itd.). Cykl jest powtarzany aż do zakończenia działania programu cron lub wprowadzenia odpowiednich zmian do pliku crontab.

W większości systemów dostęp do programu `cron` ma tylko administrator. Taka sytuacja może zostać zmieniona przez wprowadzenie odpowiednich wartości do dwóch plików: `/usr/lib/cron/cron.allow`, który zawiera identyfikatory użytkowników mogących korzystać z tego polecenia, i `/usr/lib/cron/cron.deny`, z listą użytkowników nie mogących tego robić. W obu plikach identyfikatory użytkowników (odpowiadające tym zapisanym w pliku `/etc/passwd`) powinny być wpisywane po jednym w każdym wierszu.

Jeśli nie istnieje żaden z tych plików, tylko użytkownik `root` może używać programu `cron`. Jeśli chcesz, aby mogli go używać wszyscy użytkownicy, utwórz pusty plik `/usr/lib/cron/cron.deny` (w niektórych systemach `/var/cron.d/cron.deny`). Plik `/usr/lib/cron/cron.deny` zawierający następujące dane:

```
bill  
tom
```

spowoduje, że użytkownicy `bill` i `tom` nie będą mieli możliwości użycia programu `cron`, natomiast możliwość taką będą mieli wszyscy pozostali użytkownicy. Jeśli taką samą zawartość miałby plik `/usr/lib/cron/cron.allow`, tylko użytkownicy `bill` i `tom` (oraz oczywiście `root`) mieliby możliwość korzystania z tego programu.



Możesz się zastanawiać, jaki sens ma używanie programu `cron`, skoro można zawsze odpowiednie polecenie wpisać ręcznie. To prawda, ale po pierwsze musisz o tym pamiętać, a po drugie musisz być na miejscu. Program `cron` umożliwia zautomatyzowanie wszystkich codziennych czynności administracyjnych. Dzięki temu nie musisz codziennie wydawać długiej listy poleceń tworzących kopie zapasowe, usuwających niepotrzebne pliki itd.

Tworzenie pliku crontab

Jeśli chcesz zrzuścić na program `cron` codzienne obowiązki, powinieneś użyć programu `crontab`. Odczytuje on dane o procesach, które należy uruchamiać, z pliku, który najczęściej również nazywa się `crontab`. Program ten potrafi również wyświetlić listę zadań, usunąć ją czy dodać do niej nowe zadania.

Program `crontab` posiada opcję pozwalającą jako plik z danymi wykorzystać plik o dowolnej nazwie. Domyslnie odczytywany jest plik o nazwie `crontab` znajdujący się w katalogu bieżącym.

Instrukcje zawarte w pliku `crontab` mają prostą strukturę, choć przyzwyczajenie się do niej trwa chwilę. Każdemu procesowi odpowiada jeden wiersz o następującym formacie:

```
minuta godzina dzień_miesiąca miesiąc_roku dzień_tygodnia polecenie
```

Oto przykładowe wpisy:

```
20 1 * * * /usr/bin/calendar -  
0 2 * * * /bin/organize_data
```

Każdy wiersz składa się z sześciu pól, rozdzielonych znakiem tabulacji lub spacją. Są to kolejno:

minuta	dostępne wartości: 0 – 59;
godzina	dostępne wartości 0 – 23;
dzień_miesiąca	dostępne wartości 1 – 31;
miesiąc_roku	dostępne wartości 1 – 12;
dzień_tygodnia	dostępne wartości 0 – 6 (0 = niedziela, 1=poniedziałek itd.);
polecenie	program, który ma zostać wykonany o danej godzinie w danym dniu.

Dzięki takiemu formatowi możliwe jest dokładne określenie, kiedy dany proces ma zostać uruchomiony. Aż pięć kategorii określających czas wykonania pozwala bardzo dokładnie określić porę wykonania, nawet jeśli wymagane są terminy nie układające się w żaden regularny schemat.

Ostatnie pole zawiera nazwę skryptu, który ma zostać uruchomiony. Taki skrypt może składać się tylko z jednego polecenia, ale może również być całkiem skomplikowanym programem, zawierającym odwołania do innych skryptów i programów. Ważne jest podawanie pełnych ścieżek dostępu do skryptów (bez względu na to, czy katalogi w których są one zapisane wchodzą w skład ścieżki przeszukiwania), ponieważ `cron` nie dziedziczy środowiska użytkownika, w związku z czym nie będzie potrafił odszukać odpowiednich poleceń. Musisz również posiadać uprawnienia do wykonania określonego programu, `cron` bowiem wykonuje proces tak, jakbyś to Ty go uruchomił.

W polach oznaczających czas mogą znaleźć się pojedyncze liczby z dozwolonego zakresu, lista liczb rozdzielonych przecinkami, dwie liczby rozdzielone minusem (oznaczające zakres, np. 1 – 5) lub gwiazdka, oznaczająca wszystkie dostępne wartości.

Spójrzmy na przykładową zawartość pliku `crontab`:

```
20 1 * * * /usr/bin/calendar -  
0 2 1 * 0 /bin/organize data  
10,30,50 9-18 * * * /bin/setperms
```

Pierwsze polecenie, `/usr/bin/calendar -` (myślnik jest częścią polecenia), będzie wykonane 20 minut po pierwszej w nocy każdego dnia tygodnia we wszystkie tygodnie roku. Gwiazdka oznacza wszystkie dostępne wartości.

O drugiej w nocy w każdą niedzielę (0 w piątej kolumnie) i pierwszego dnia miesiąca (1 w trzeciej kolumnie) wykonane zostanie polecenie `/bin/organize_data`. Oczywiście jeśli pierwszego wypada w niedzielę, polecenie zostanie wykonane tylko raz.

Trzecie polecenie, `/bin/setperms`, wykonywane będzie 10, 30 i 50 minut po każdej godzinie pomiędzy 9 i 18, każdego dnia tygodnia.

Wpisy w pliku `crontab` mogą mieć dowolną kolejność, ale dane o każdym z programów muszą zawierać sześć pól i znajdować się w osobnym wierszu. Jeśli do pliku wkradnie się jakiś błąd, `cron` poinformuje Cię o jego istnieniu pocztą (może to być dość uciążliwe, jeśli pomyliś się w poleceniu, które ma być wykonywane często, ponieważ przy każdej próbie uruchomienia błędnego polecenia `cron` wygeneruje nową informację, co może doprowadzić nawet do przepełnienia skrzynki pocztowej).

Plik zawierający zadania najlepiej przechowywać we własnym katalogu pod nazwą `crontab`, chyba że potrzebujesz kilku jego wersji – wybór używanych nazw zależy wówczas wyłącznie od Ciebie.

Zarządzanie plikami crontab

Po stworzeniu własnego pliku `crontab` powinieneś poinformować program `cron`, że należy go przetworzyć. Jego kopia znajdzie się wówczas w katalogu systemowym, zwykle `/usr/spool/cron/crontabs`. Plik będzie miał nazwę zgodną z identyfikatorem użytkownika, który zamówił wykonanie danych zadań (na przykład plik zawierający dane o poleceniach zleconych przez użytkownika `bill` może nazywać się `/usr/spool/cron/crontabs/bill`).

Aby uaktywnić plik `crontab`, wydaj polecenie `crontab`, podając jako argument wywołania nazwę pliku, w którym zapisane są odpowiednie dane, na przykład tak:

```
crontab crontab
```

Jeśli wcześniej wydałeś już takie polecenie, wszystkie zlecenia są anulowane, a w ich miejscu wstawiane są nowe.



Aby zmodyfikować zamówione zadania, powinieneś edytować plik `crontab` i nakazać jego ponowne przetworzenie. Nigdy nie należy modyfikować bezpośrednio plików w katalogu `/usr/spool/cron/crontabs`.

Jeśli chcesz sprawdzić, jakie programy będą uruchamiane przez program `cron`, możesz użyć polecenia `crontab -l` (ang. *list*):

```
crontab -l
```

Polecenie to powoduje wyświetlenie zawartości pliku o nazwie odpowiadającej Twojemu identyfikatorowi użytkownika, zapisanego w katalogu

```
/usr/spool/cron/crontabs.
```

Jeśli chcesz usunąć swój plik nie zastępując go żadnym innym, użyj opcji `-r` (ang. *remove*):

```
crontab -r
```

Polecenie `crontab` pozwala również edytować plik, który aktualnie jest używany (uruchamiany jest edytor domyślny, którego nazwa jest zdefiniowana przez wartość odpowiedniej zmiennej środowiskowej – zwykle edytor `vi`):

```
crontab -e
```

Po zakończeniu edycji i zapisaniu pliku jest on automatycznie przetwarzany przez program `cron`.

Zmiany w plikach `crontab` są zwykle wprowadzane w życie w ciągu pięciu minut – przynajmniej tak często są odczytywane te pliki (w większości systemów odbywa się to co minutę). Oznacza to również, że wykonanie danego programu może się opóźnić o kilka minut; nie można zakładać, że zostanie on uruchomiony punktualnie. Im większe jest obciążenie systemu, tym większe mogą być opóźnienia.

W niektórych systemach administrator ma możliwość monitorowania wszystkich czynności podejmowanych przez program `cron` – w tym celu należy zmodyfikować plik `/etc/default/cron`. Powinien się w nim znajdować wpis definiujący wartość zmiennej `CRONLOG` – ustaw tę wartość na `YES`, a wszystkie wywołania programów zostaną odnotowane w pliku `/usr/lib/cron/log`. Nie we wszystkich wersjach Linuxa rozwiązanie takie jest dostępne. Powinieneś jednak używać go z rozwiązką, ponieważ taki plik może szybko osiągnąć duże rozmiary.

Bardziej złożone polecenia

Plik `crontab` może zawierać dowolne polecenia poprawnie rozpoznawane przez interpreter poleceń powłoki. Problemem czasem staje się jednak fakt, że niektóre z nich generują jakieś informacje, które są następnie przesyłane pocztą do użytkownika, co może spowodować zablokowanie skrzynki pocztowej. Jeśli przewidujesz, że polecenie uruchamiane przez program `cron` będzie generowało jakieś komunikaty, powinieneś wyjście programu i komunikaty o błędach skierować do pliku lub, jeśli generowane informacje Cię nie interesują, do urządzenia `/dev/null` albo `/dev/zero`, na przykład tak:

```
0 * * * * date > /tmp/test1 2 > /dev/null
```

Powyższe polecenie co godzinę przepisuje aktualną datę do pliku `/tmp/test1`, komunikaty o błędach wysyłając do urządzenia `/dev/null` (co oznacza po prostu ich zignorowanie). Możesz również komunikaty o błędach skierować gdzieś indziej, na przykład:

```
30 1 * * * cat /home/reksio/chapt* > /home/reksio/safe/backup 2> Σ  
/home/reksio/cos_nie_tak
```

Powyższe polecenie powoduje, że pliki o nazwach rozpoczynających się od `chapt` w katalogu `/home/reksio` będą łączone w jeden plik o nazwie `backup` zapisany w katalogu `/home/reksio/safe`, a komunikaty o ewentualnych problemach przesyłane będą do pliku `cos_nie_tak`.

Oto inny przykład użycia programu `cron`, wykorzystujący mechanizm potoków (ang. *pipe*):

```
0 1 * * * sort -u /tmp/userlist | mail -s"dzisiejsi użytkownicy" root
```

Powoduje on wysyłanie do użytkownika `root` poczty zawierającej listę użytkowników, którzy logowali się danego dnia (przy założeniu, że taka lista znajduje się w pliku o nazwie `/tmp/userlist`). Lista ta jest posortowana i wyeliminowane są pozycje powtarzające się (opcja `-u` programu `sort` – ang. *unique*).

Należy pamiętać, że domyślnie do przetwarzania poleceń używany jest interpreter `bash`, możesz więc spotkać się z problemami próbując wykonać polecenia powłoki `csh`.

Program at

Program `at` jest dość podobny do programu `cron`, z tym że zadane polecenie wykonuje się okresowo, ale tylko raz. Jego składnia jest następująca:

```
at czas [data] <plik
```

Większość parametrów może być podana na kilka sposobów. Na przykład czas można podać w postaci pełnej (18:43), podać tylko godzinę (10) bądź użyć modyfikatorów `am` i `pm`.

W polu czasu rozpoznawanych jest również kilka wyrazów, takich jak `noon`, `midnight`, `now`, `next` i `zulu` (konwersja na czas GMT). Niektóre wersje nie rozpoznają wyrazu `now`.

Pole daty jest opcjonalne. Jego zawartość może stanowić angielska nazwa miesiąca i dzień (np. May 10) lub dzień tygodnia (pełna nazwa angielska lub skrócona, trzyliterowa). Można także podać rok, ale rzadko jest to konieczne. Tu również rozpoznawanych jest kilka słów, takich jak `today` i `tomorrow` (choć podawanie słowa `today` jest zbędne, ponieważ polecenia domyślnie wykonywane są w tym dniu, w którym zostały wydane).

Plik wejściowy może zawierać dowolne polecenia. Można również wprowadzić je „ręcznie” – kombinacja klawiszy `Control+D` kończy wprowadzanie polecień z klawiatury.

Załóżmy, że w Twoim katalogu znajduje się skrypt o nazwie `reorg.data` (i masz prawo go wykonać) o następującej zawartości:

```
/home/reksio/setperms
/home/reksio/sort_database
/home/reksio/index_database
/home/reksio/cleanup
```

Chciałbyś zapisane w nim polecenia wykonać o 8:30, ale nie będzie Cię wtedy przy komputerze. Aby rozwiązać ten problem, możesz wydać jedno z polecień:

```
at 20:30 < reorg.data
at 8:30 pm <reorg.data
at 20:30 today <reorg.data
```

Jeśli chciałbyś, żeby skrypt został wykonany w piątek, wydaj jedno z polecień:

```
at 8:30 pm Friday < reorg.data
at 20:30 Fri < reorg.data
```

Niektóre wersje programu `at` są jeszcze sprytniejsze i rozpoznają konstrukcje takie jak:

```
at 0900 Monday next week <reorg.data
```

Po zleceniu wykonania polecenia otrzymasz jego identyfikator:

```
at 6 <zrob_cos
job 54342.a at Wed Aug 31 06:00:00 EDT 1997
```

W tym przypadku identyfikatorem jest `54342.a`. Identyfikator ten określa jednoznacznie zadanie. Jest on wymagany np. do odwołania wydanego polecenia.

Listę wszystkich zleconych zadań można obejrzeć za pomocą polecenia `atq` lub, w innych systemach, opcji `-l` programu `at`:

```
$ at -l
user = reksio job 54342.a at Wed Aug 31 06:00:00 EDT 1997
user = reksio job 54343.a at Wed Aug 31 09:30:00 EDT 1997
Jeśli chcesz anulować zlecenie, wydaj polecenie at z opcją -r i numerem
identyfikacyjnym zadania:
at -r 54343.a
```

Linux nie potwierdzi usunięcia zadania, aby sprawdzić, czy wszystko przebiegło po-myślnie, trzeba jeszcze raz wydać polecenie wyświetlające listę oczekujących na wykonanie poleceń. Anulować można wykonanie tylko tych zadań, których jesteś właścicielem (ograniczenie to nie dotyczy użytkownika `root`). W niektórych wersjach systemu do usuwania zadań z kolejki służy polecenie `atrm`.

Dane o zadaniach programu `at` są przechowywane w katalogu `/usr/spool/cron/atjobs`, w plikach o nazwach odpowiadających identyfikatorom zadań. Podobnie jak miało to miejsce w przypadku polecenia `cron`, do limitowania dostępu do polecenia `at` służą pliki `at.deny` i `at.allow`, przechowywane w katalogu `/usr/lib/cron` lub `/etc/cron.d`. Jeśli chcesz umożliwić wszystkim użytkownikom używanie programu `at`, utwórz pusty plik o nazwie `at.deny`.

Komunikaty generowane przez wykonywane programy i komunikaty o błędach wysypane są pocztą do użytkownika, chyba że zostaną skierowane do pliku. Polecenie `at`, w przeciwieństwie do `cron`, dziedziczy po użytkowniku środowisko, więc nie musisz aż tak martwić się o ścieżki dostępu. Jeśli zajrzesz do plików zapisanych w katalogu `/usr/spool/cron/atjobs`, zobaczyś, że przed właściwym poleceniem znajdują się definicje wszystkich zmiennych środowiskowych używanych w momencie wydawania polecenia `at`.

Podsumowanie

Jak widać, programy `cron` i `at` są łatwe w użyciu. Są one niezastąpione do automatyzacji prac administracyjnych, takich jak porządkowanie baz danych, usuwanie plików, w których rejestrowane są czynności systemu, usuwanie plików tymczasowych czy

tworzenie kopii zapasowych. Nie zrobią za Ciebie wszystkiego, ale możesz kazać im zajmować się wszystkimi powtarzającymi się czynnościami.

Wiele systemów linuxowych posiada pewną liczbę przykładowych plików `cron`, które pomogą Ci zorientować się w możliwościach tkwiących w tym programie. Użyj ich jako punktu wyjścia do tworzenia własnych plików, wyręczających Cię z codziennych obowiązków.

Tworzenie kopii zapasowych, czyli podstawowe zadanie każdego administratora, omówione jest w rozdziale 45. „Kopie zapasowe”.

Jak założyć i skonfigurować węzeł internetowy w oparciu o system Linux, dowiesz się z rozdziału 47. „Konfigurowanie węzła internetowego”.

O aplikacjach, których można używać w systemie Linux, przeczytać możesz w rozdziałach od 62. „Adabas-D i inne bazy danych” do 64. „Program Lone Tar”.

Część siódma

Konfiguracja węzła internetowego

W tej części:

- υ Konfiguracja węzła internetowego
- υ Konfigurowanie FTP i anonimowego FTP
- υ Konfiguracja węzła WAIS
- υ Konfigurowanie usługi Gopher
- υ Konfigurowanie węzła WWW
- υ Skrypty GCI
- υ Podstawy języka HTML
- υ Podstawy języka Java i JavaScript
- υ Tworzenie węzła internetowego

Rozdział 47.

Konfiguracja węzła internetowego

Tim Parker

W tym rozdziale:

- v Podłączanie się do Internetu

System Linux jest doskonale przystosowany do połączenia z siecią i korzystania z usług przez nią oferowanych. Nie chodzi tu tylko o przeglądanie stron WWW czy używanie protokołu FTP do przesyłania plików, ale przede wszystkim o możliwość zastosowania systemu linuxowego jako serwera sieciowego. Przy minimalnym nakładzie pracy poświęconej konfiguracji systemu można udostępniać własne strony WWW lub założyć węzeł FTP czy Gopher. Nie jest potrzebne żadne dodatkowe oprogramowanie, w zupełności wystarczy to, które rozprowadzane jest wraz z dystrybucją systemu na dyskach CD-ROM. W tym rozdziale przyjrzymy się różnym sposobom łączenia się z Internetem. Kilka następnych rozdziałów omawia konfigurowanie systemu linuxowego mającego działać jako serwer najpopularniejszych usług internetowych.

Jeśli zamierzasz używać Internetu tylko biernie, korzystając z usług innych serwerów, nie musisz czytać następnych czterech rozdziałów (choć nadal warto zapoznać się z tym rozdziałem, ponieważ opisujemy w nim metody łączenia się z siecią Internet). Z drugiej strony jednak dzielenie się zasobami własnego systemu z innymi użytkownikami sieci – czy to sieci lokalnej, czy też Internetu – jest chyba najprzyjemniejszym jej aspektem.

Jeśli chcesz, by Twój system linuxowy oferował niektóre z usług internetowych (jak WWW, FTP czy Gopher), ale nie chcesz, by dostęp do nich mieli wszyscy użytkownicy (oprócz co najwyżej kilku przyjaciół), nie powinieneś obawiać się połączenia z Internetem. Musisz jednak skonfigurować oprogramowanie serwera.

Podłączanie się do Internetu

Istnieje wiele metod podłączania się do Internetu. Wybór jednej z nich jest głównie kwestią przyzwyczajenia i tego, z jakich usług zamierzasz korzystać. Choć na rynku działa wiele firm oferujących dostęp do Internetu na bardzo różnych warunkach, zasadniczo istnieją tylko cztery typy połączenia – poniżej przedstawiamy ich skrótnowy opis.

- v Bezpośrednie połączenie z Internetem. Ta metoda wymaga zastosowania dedykowanego komputera (bramki), połączonego z Internetem poprzez szybką linię telefoniczną, na przykład linię T1 (1,544 Mbps) czy ISDN (128 kbps). Połączenie bezpośrednie umożliwia pełny dostęp do wszystkich usług sieciowych, ale zarówno jego instalacja, jak i utrzymanie są dość drogie.
- v Połączenie przez czyjąś bramkę internetową. Wymaga – niestety – zezwolenia na używanie czegoś komputera. Również umożliwia dostęp do wszystkich usług sieciowych.
- v Bezpośredni dostawca Internetu. Ten sposób połączenia polega na wykorzystaniu bramki będącej w posiadaniu jakiejś firmy, do której można się podłączyć, by uzyskać częściowy lub pełny dostęp do usług sieciowych. Komputer po stronie firmy pełni tylko rolę bramki internetowej. Dołączenia się z nim zwykle używa się modemu lub dedykowanych, szybkich łączów telefonicznych. Firma, która jest właścicielem bramki, jest określana jako dostawca Internetu lub usługodawca internetowy (ang. *Internet Service Provider*).
- v Pośredni dostawca Internetu. Ta metoda polega na wykorzystaniu połączeń z takimi sieciami, jak Delphi czy CompuServe, do uzyskania dostępu do niektórych lub wszystkich usług internetowych. Metoda taka nadaje się do wykorzystania tylko wtedy, jeśli mało korzystasz z Internetu, i nie pozwala wykorzystać całego potencjału drzemiącego w systemie Linux.

Jeśli Twój system jest częścią większej instytucji lub dzielisz się kosztami dostępu do sieci z kilkoma przyjaciółmi, pośredni dostawca Internetu najprawdopodobniej nie będzie w stanie zapewnić Ci wydajności niezbędnej do korzystania z takich usług, jak poczta elektroniczna czy FTP. Inną wadą takiego rozwiązania jest to, że większość dostawców pośrednich nie umożliwia posiadania własnej nazwy domenowej.

Rzadko można znaleźć bramki internetowe, z których można byłoby korzystać bez uczestniczenia w kosztach ich utrzymania. Większość firm posiadających bramki internetowe nie zgadza się, by korzystał z nich ktokolwiek obcy.

W takiej sytuacji pozostają właściwie tylko dwa rozwiązania: bezpośrednie połączenie z Internetem lub skorzystanie z oferty bezpośredniego dostawcy Internetu. Wybór jednej z tych metod zwykle wynika z kosztów połączenia. Instalacja własnej bramki internetowej może być droga, ale i tak może okazać się tańsza niż skorzystanie z usług dostawcy, szczególnie jeśli planujesz duży ruch w sieci lub system z założenia ma być bardzo duży.

Jeśli połączenie z Internetem będzie wykorzystywane przez jedną osobę lub niewielką firmę, instalowanie własnej bramki nie ma większego sensu. Wtedy najlepszym rozwiązaniem jest skorzystanie z usług bezpośredniego dostawcy Internetu. Niewielkie firmy prawie zawsze łączą się z siecią właśnie w taki sposób.

Usługi, których potrzebujesz

Przed podjęciem decyzji o typie połączenia z Internetem należy zastanowić się, jakie usługi będą potrzebne. Jeśli potrzebujesz tylko dostępu do poczty elektronicznej, możesz wybrać dowolny typ połączenia, choć koszty mogą być niewspółmiernie duże w porównaniu do wykorzystywanych możliwości.

Na początek zdecyduj, które z poniższych usług są niezbędne, a z których mógłbyś zrezygnować.

- υ **Poczta elektroniczna.** Pozwala na wysyłanie i odbieranie wiadomości od innych użytkowników Internetu.
- υ **Telnet.** Pozwala na logowanie się poprzez Internet do systemów zdalnych.
- υ **FTP.** Umożliwia przenoszenie plików pomiędzy komputerami.
- υ **WWW** (ang. *World Wide Web*), czyli dostęp do witryn, zwykle graficznych, tworzonych za pomocą języka HTML (ang. *Hypertext Markup Language*).
- υ **Usenet**, czyli dostęp do grup dyskusyjnych, w których prowadzone są rozmowy na najprzeróżniejsze tematy.
- υ **Gopher.** System wyszukiwania i pobierania informacji.
- υ **WAIS.** Pozwala na wyszukiwanie i pobieranie dokumentów, obsługiwany za pomocą systemów menu.
- υ **Archie.** Metoda wyszukiwania plików, które należy przesłać.
- υ **IRC** (ang. *Internet Relay Chat*), czyli pogawędki internetowe, przypominające nieco CB Radio.

Każdy system podłączony do Internetu przez bramkę (zarówno w przypadku posiadania własnej bramki, korzystania z użyczonej czy wynajmowanej u większości dostawców Internetu) umożliwia korzystanie ze wszystkich wymienionych powyżej usług. Niektórzy dostawcy dość mocno ograniczają jednak prędkość przesyłania danych, co staje się poważnym problemem szczególnie wtedy, gdy chcesz udostępniać strony WWW, zawierające oprócz tekstu również grafikę. Niektórzy dostawcy limitują też na różne sposoby dostęp do poczty elektronicznej i grup dyskusyjnych, należy więc zasięgnąć dokładnych informacji przed zdecydowaniem się na współpracę z którymś z nich.

Połączenie bezpośrednie za pomocą bramki

Połączenie bezpośrednie (nazywane również dedykowanym) polega na podłączeniu się do sieci poprzez przeznaczony do tego celu komputer nazywany bramką lub routerem IP. Połączenie realizowane jest przy użyciu szybkiego, dedykowanego łączego telefonicznego (zwykle o przepustowości 1.44 Mb na sekundę lub większej). Bramka staje się elementem architektury sieci Internet, w związku z czym musi przez cały czas pozostać włączona. Dostęp do usług internetowych można uzyskać w każdym systemie, który pracuje w tej samej sieci, co bramka internetowa.

Zwykle połączenia dedykowane używane są tam, gdzie występuje duże natężenie ruchu i wymagana jest prędkość transmisji nie spadająca poniżej granicy np. 9600 bodów (choćniczym nadzwyczajnym nie są połączenia światłowodowe, umożliwiające przesyłanie danych z prędkością 45 Mbps). Rzadko zdarza się, by bezpośrednie połączenie z Internetem było własnością indywidualnej osoby czy małej firmy, głównie ze względu na wysokie koszty jego instalacji i utrzymania.

Aby stworzyć system posiadający bezpośrednie połączenie z Internetem, trzeba skontaktować się z NIC (Network Information Center), organizacją, która dostarcza odpowiednich informacji konfiguracyjnych. Całkowite koszty instalacji i utrzymania takiego połączenia są wysokie. Wiążą się one z koniecznością posiadania odpowiedniego sprzętu i oprogramowania, a także poprowadzenia łączego o odpowiednio dużej przepustowości. Najczęściej używane typy łączego to T1 (1,544 Mbps), T4 (4 Mbps) i bardziej dostępny dla zwykłych śmiertelników ISDN (128 kbps).

Połączenie za pomocą czyjeś bramki

Alternatywą dla połączenia bezpośredniego jest wykorzystanie bramki zainstalowanej przez kogoś innego, na przykład jakąś firmę czy szkołę. Mało kto jednak zgadza się na używanie swoich komputerów, również ze względów bezpieczeństwa.

Jeśli jednak jesteś jednym z niewielu szczęśliwców, którym udało się otrzymać zgodę na tego typu połączenie, musisz po prostu połączyć się z bramką i – za jej pośrednictwem – z Internetem. Pod wieloma względami rozwiązanie to nie różni się od skorzystania z usług bezpośredniego dostawcy Internetu. Zwykle w takich sytuacjach uzyskujesz pełny dostęp do wszystkich usług sieciowych (choć niektóre organizacje nakładają pewne ograniczenia).

Usługodawcy internetowi

Usługodawcy internetowi (lub dostawcy Internetu) to firmy, które posiadają bezpośrednie połączenie z Internetem (za pomocą bramki) i pozwalają użytkować go swoim klientom za odpowiednią opłatą (bramka jest zwykle „przezroczysta” dla użytkownika). Ten typ połączenia często nazywany jest połączeniem dial-up i opiera się na protokołach

SLIP (ang. *Serial Line Interface Protocol*) i PPP (ang. *Point-to-Point Protocol*). Niektórzy usługodawcy do obsługi poczty elektronicznej używają protokołu UUCP.

Dostawcy Internetu zwykle pobierają opłatę składającą się z pewnej części stałej oraz części uzależnionej od czasu połączenia lub ilości przesyłanych danych. Połączenie się z dostawcą Internetu jest przeważnie bardzo proste. Większość z nich umożliwia również posiadanie własnej nazwy domenowej.

Największą zaletą połączenia przez bezpośredniego dostawcę Internetu jest fakt, że system zachowuje się tak samo, jakby był podłączony bezpośrednio do sieci. Komunikacja z komputerem-bramką jest obsługiwana przez system operacyjny i jest procesem praktycznie niewidocznym dla użytkownika. Wadą tego rozwiązania jest to, że czasem niektóre usługi dostępne są tylko w ograniczonym zakresie (na przykład niektórzy dostawcy nie pozwalają na przesyłanie plików za pomocą FTP do innych serwerów).

Jeśli bierzesz pod uwagę skorzystanie z usług bezpośredniego dostawcy Internetu, powinieneś zorientować się, jakie usługi są oferowane przez dostawców działających w okolicy, czy wymagany jest jakiś dodatkowy sprzęt, jaki jest system opłat (czy są to opłaty stałe, czy zależne od wykorzystania łączą) i jak przedstawia się problem wsparcia technicznego na wypadek kłopotów.

Podsumowanie

Wybór metody połączenia się z Internetem zależy tylko od Ciebie, ale większość użytkowników indywidualnych decyduje się na skorzystanie z usług bezpośrednich dostawców Internetu. Jest to rozwiązanie zrównoważone, jeśli wziąć pod uwagę koszty i możliwości, szczególnie jeżeli Twój system ma przez większość czasu pozostawać połączony z siecią. Po połączeniu się z Internetem możesz skonfigurować własny serwer – jak to zrobić, dowiesz się czytając cztery następne rozdziały.

Konfigurowanie komputera tak, by działał jako serwer FTP, opisane jest w rozdziale 48. „Konfigurowanie FTP i anonimowego FTP”.

Konfiguracja i używanie usługi Gopher omawia rozdział 50. „Konfigurowanie usługi Gopher”.

Jak skonfigurować serwer WWW i własną stronę główną dowiesz się z rozdziału 51. „Konfigurowanie węzła WWW”.

Tworzenie własnych stron WWW omówione jest w rozdziale 53. „Podstawy języka HTML”.

Rozdział 48.

Konfigurowanie FTP i anonimowego FTP

Tim Parker

W tym rozdziale:

- υ Co to jest FTP?
- υ Konfiguracja FTP
- υ Bezpieczniejsze FTP
- υ Zabezpieczanie anonimowego FTP

Jak myślisz, która z usług udostępnianych dzięki protokołowi TCP/IP i Internetowi jest używana najczęściej? Jeśli odpowiedziałeś, że FTP, to masz rację (jeśli uważałeś, że jest inaczej, może to być dla Ciebie niespodzianka – jest to jednak fakt, choć WWW staje się coraz popularniejsze). Popularność FTP jest łatwa do wytlumaczenia: oprogramowanie do jego obsługi dostarczane jest wraz z każdą wersją UNIX-a i Linuxa. Jest ono łatwe do zainstalowania, skonfigurowania i używania. Pozwala na dostęp do ogromnej ilości danych przy minimalnym wysiłku. Rosnąca popularność stron WWW spowodowała również powstanie nowego interfejsu służącego do obsługi FTP, wbudowanego w przeglądarki WWW. Większość przeglądarek, na przykład Netscape Navigator, pozwala na transfer plików za pomocą protokołu FTP.

Jeśli zamierzasz używać protokołu FTP tylko do ładowania plików z innych systemów, musisz tylko załączyć obsługę tego protokołu w swoim systemie. O wiele bardziej interesujące jest jednak umożliwienie innym korzystania z plików zgromadzonych w Twoim systemie. W tym rozdziale omówimy głównie zagadnienia związane z konfigurowaniem węzła FTP. Rozpoczniemy jednak od krótkiego omówienia protokołu FTP i jego powiązań z TCP/IP. Informacje te powinny ułatwić zrozumienie zasady działania usługi FTP.

Co to jest FTP?

FTP (ang. *File Transfer Protocol*, protokół transmisji plików) to jeden z protokołów rodziny TCP/IP, służący do przesyłania plików pomiędzy komputerami obsługującymi system TCP/IP (choć dostępne są również podobne do FTP programy przeznaczone do współpracy z innymi protokołami). FTP pozwala na przesyłanie plików w obie strony i poruszanie się po drzewie katalogów. Protokół ten nie jest przeznaczony do uruchamiania programów w systemie zdalnym, ale jest chyba najlepszym narzędziem do transmisji plików. Aby można było użyć FTP, na obu końcach połączenia musi być uruchomiony odpowiedni program. Komputer, który nawiązuje połączenie (klient), wywołuje drugi komputer (serwer) i po przesłaniu zestawu informacji potwierdzających połączenie jest ustalane.

Zwykle po uzyskaniu połączenia należy się zalogować, co oznacza, że trzeba być użytkownikiem systemu zdalnego (czyli posiadać własny identyfikator oraz hasło). Ponieważ niemożliwe jest przyznanie osobnego identyfikatora i hasła każdemu, kto chce skorzystać z ogólnodostępnych zasobów, w wielu systemach używa się tzw. anonimowego FTP. Rozwiążanie to pozwala każdemu zalogować się do systemu po podaniu identyfikatora `ftp` lub `anonymous`, bądź to bez podania hasła, bądź podając swój adres e-mailowy.

Używanie FTP

Połączenie się z systemem zdalnym za pomocą FTP jest łatwe. Dostęp do komputera zdalnego poprzez Internet (bezpośrednio lub poprzez dostawcę Internetu) lub sieć lokalną można uzyskać wtedy, gdy komputer ten jest bezpośrednio osiągalny. Jeśli chcesz użyć FTP, uruchom odpowiedni program (nazywany klientem FTP), podając nazwę komputera, z którym chcesz się połączyć. Przykładowo, jeśli chcesz połączyć się z systemem o nazwie `chatton.com`, możesz (zakładając, że masz dostęp do komputera zdalnego przez sieć LAN lub Internet; w przypadku Internetu oznacza to również, że nazwa komputera musi dać się przetłumaczyć na numer IP za pomocą usługi Domain Name Service) wydać polecenie:

```
ftp chatton.com
```

Spowoduje ono uruchomienie klienta FTP i próbę nawiązania połączenia z systemem `chatton.com`.

Po nawiązaniu połączenia (zakładając, że system zdalny pozwala na połączenia FTP) system zdalny oczekuje na podanie identyfikatora użytkownika. Jeśli dozwolone są anonimowe połączenia FTP, zwykle pojawia się odpowiednia informacja. Oto przykład pochodzący z węzła `sunsite.unc.edu`:

```
ftp sunsite.unc.edu
331 Guest login ok, send your complete e-mail address as password.
Enter username (default: anonymous): anonymous
Enter password [tparker@tpci.com]:
|FTP| Open
230-                               WELCOME to UNC and SUN's anonymous ftp server
230-                               University of North Carolina
```

```
230-
230-
230 Guest login ok., access restrictions apply.
FTP>
```

Po zakończeniu procesu logowania wyświetlana jest zachęta `FTP>`. Oznacza to, że system jest gotów do przyjmowania poleceń FTP.

W niektórych systemach dodatkowo wyświetlany jest krótki komunikat zawierający instrukcje dotyczące ładowania plików, ograniczeń nałożonych na Ciebie jako użytkownika anonimowego oraz położenia bardziej przydatnych plików. Przykładowe komunikaty z węzła `sunsite.unc.edu` mają postać:

```
To get a binary file, type: BINARY and then: GET "File.Name" newfilename
To get a text file, type:      ASCII      and then: GET "File.Name"
Σ newfilename
Names MUST match upper, lower case exactly. Use the "quotes" as shown.
To get a directory, type: DIR. To change directory, type: CD "Dir.Name"
To read a short text file, type GET "File.Name" TT
For more, type HELP or see FAQ in gopher.
To quit, type EXIT or Control-Z.

230- If you e-mail to info@sunsite.unc.edu you will be sent help
Σ information
230- about how to use the different services sunsite provides.
230- We use the Wuarchive experimental ftpd. If you "get"
Σ <directory>.tar.Z
230- or <file>.Z it will compress and/or tar it on the fly. Using ".gz"
Σ instead
230- of ".Z" will use the GNU zip (/pub/gnu/gzip*) instead, a superior
230- compression method.
```

Po zalogowaniu się do systemu zdalnego można przeglądać zawartość katalogów i poruszać się po nich używając poleceń znanych z systemu Linux. Na przykład, aby wyświetlić zawartość bieżącego katalogu, należy wydać polecenie `ls` (niektóre wersje FTP obsługują również jego DOS-owy odpowiednik, `dir`). Do zmiany katalogu bieżącego służy polecenie `cd`. Jeśli chcesz przejść do katalogu nadzielnego (czyli tego, w którym zawarty jest katalog bieżący), wydaj polecenie `cd ...`. Jak widać, polecenia są takie same, jak te służące do poruszania się w obrębie lokalnego systemu plików; należy jednak pamiętać, że dotyczą one systemu zdalnego.

Podczas używania FTP nie są dostępne żadne skróty klawiaturowe (takie jak dokańczanie nazw plików po naciśnięciu klawisza *Tab*). Oznacza to, że trzeba wpisywać nazwy plików w pełnej i poprawnej postaci. Jeśli się pomyliš, otrzymasz komunikat o błędzie i będziesz musiał spróbować jeszcze raz. Na szczęście, jeśli używasz systemu X Window, możesz wycinać i wklejać tekst, który wpisałeś wcześniej.

Głównym zadaniem FTP jest przesyłanie plików, więc trzeba wiedzieć zarówno w jaki sposób można załadować plik z systemu zdalnego, jak i wysłać plik do tego systemu. Kiedy znajdziesz w systemie zdalnym jakiś plik, który chciałbyś załadować do swojego komputera, użyj polecenia `get`, którego argumentem jest nazwa tegoż pliku. Przykładowe polecenie:

```
get "soundcard_driver"
```

spowoduje załadowanie pliku o nazwie `soundcard_driver` znajdującego się w katalogu bieżącym komputera zdalnego do katalogu bieżącego Twojego systemu. Po wydaniu polecenia `get` system zdalny przesyła dane do Twojego komputera, wyświetlając krótkie podsumowanie po zakończeniu tej operacji. W trakcie przesyłania danych nie jest wyświetlany żaden wskaźnik postępu, więc w przypadku większych plików powinieneś uzbroić się w cierpliwość (większość wersji programu `ftp` udostępnia opcję `hash`, która powoduje wyświetlenie znaku # po przesłaniu każdego kilobajta informacji; mimo tego jednak nie jest wyświetlany czas pozostały do zakończenia transferu). Oto przykład:

```
FTP> get "file1.txt"
200 PORT command successful.
150 BINARY data connection for FILE1.TXT (27534 bytes)
226 BINARY Transfer complete.
27534 byte received in 2.35 seconds (12Kbytes/s).
```

Jeśli chcesz przesyłać pliki w przeciwnym kierunku (z Twojego systemu do komputera zdalnego, zakładając że posiadasz prawo zapisu w systemie plików komputera zdalnego), powinieneś użyć polecenia `put` w taki sam sposób, jak polecenia `get`. Polecenie

```
put "comments"
```

powoduje przesyłanie pliku o nazwie `comments` znajdującego się w katalogu bieżącym Twojego systemu do katalogu bieżącego w systemie zdalnym (można również bezpośrednio podać odpowiednie ścieżki dostępu).

Polecenia `get` (załaj, pobierz) i `put` (wyślij) mają punkt odniesienia w komputerze lokalnym, co oznacza, że wydając polecenie `get`, każesz oprogramowaniu pobrać plik z serwera i umieścić go w swoim komputerze, natomiast polecienniem `put` wysydasz pliki ze swojego komputera do serwera (mamy tu sytuację dokładnie odwrotną, niż w przypadku programu `telnet`, w którym polecenia są odniesione do komputera zdalnego). Ważne jest, by dobrze zapamiętać, które polecenie działa w którą stronę, w przeciwnym przypadku może zdarzyć się, że niechcący pozbędziesz się jakichś ważnych danych.

Cudzysłów otaczający nazwy plików w większości wersji `FTP` nie jest wymagany, ale zabezpiecza przed złym interpretowaniem polecenia przez serwer, więc stosowanie go jest dobrą praktyką.

Niektóre wersje `ftp` udostępniają polecenia umożliwiające przesyłanie większej liczby plików – polecenia `mput` i `mget`. Polecenia `get` i `put` powodują jednorazowe przesyłanie tylko jednego pliku, którego nazwa musi być podana w całości (bez symboli wieloznacznych). Polecenia `mget` i `mput` pozwalają na użycie symboli wieloznacznych, takich jak * czy ?. Jeśli na przykład chcesz załadować wszystkie pliki z rozszerzeniem `.doc` znajdujące się w katalogu bieżącym, powinieneś wydać polecenie

```
mget *.doc
```

Musisz sam sprawdzić, czy polecenia `mget` i `mput` działają w Twojej wersji `ftp` (niektóre wersje zamiast tego umożliwiają po prostu użycie symboli wieloznacznych w argumentach polecień `get` i `put` – powinieneś więc sprawdzić również taką możliwość).

FTP pozwala na przesyłanie plików w kilku trybach, zwykle zależnych od systemu. Większość systemów (w tym Linux) obsługuje tylko dwa tryby: `binary` (binarny) oraz `ASCII` (pliki tekstowe). Niektóre stacje robocze obsługują jeszcze standard EBCDIC, a wiele sieci lokalnych posiada własne formaty, pozwalające na przyspieszenie transmisji (mogą one używać 32 - lub 64 - bitowych słów).

Różnica pomiędzy trybem binarnym i ASCII jest prosta. Tryb tekstowy umożliwia przesyłanie plików ASCII, w których wiersze są rozdzielone powrotem karetki i znakiem nowego wiersza, podczas gdy tryb binarny pozwala przesyłać dane bez żadnego sformatowania. Tryb binarny jest szybszy od tekstu, a pozwala również na prawidłową transmisję plików tekstowych. FTP nie pozwala na przesyłanie praw dostępu do plików, ponieważ nie zostało to ujęte w protokole.

Linux udostępnia dwa tryby transmisji danych: ASCII i binarny. Niektóre systemy łączą się pomiędzy nimi automatycznie po rozpoznaniu pliku binarnego, ale nie należy zakładać, że jest tak zawsze. Dla pewności zawsze warto ustawić odpowiedni tryb samodzielnie. Domyslnie większość wersji FTP uruchamia się w trybie ASCII (choć istnieją i takie, które uruchamiają się w trybie binarnym).



Upewnij się, że używasz właściwego trybu

Jedną z bardziej nieprzyjemnych pomyłek zdarzających się niedoswiadczeniom użytkownikom FTP jest użycie trybu tekstu do przesyłania plików binarnych. Tryb transmisji nie ma znaczenia w przypadku przesyłania plików tekstowych, ale jeśli chcesz uniknąć ładowania niepotrzebnych śmieci, przed rozpoczęciem ładowania plików binarnych powinieneś upewnić się, że załączyleś tryb binarny.

Aby załączyć tryb binarny (wymagany do ładowania plików wykonywalnych oraz plików z danymi zawierającymi specjalne kody, generowanych przez arkusze kalkulacyjne, niektóre edytory tekstu czy programy graficzne), wydaj polecenie

`binary`

Do trybu tekstu możesz wrócić wydając polecenie `ascii`. Ponieważ jednak węzły FTP zwykle odwiedza się po to, by załadować jakieś archiwum, nowe wersje programów itp., warto zawsze używać trybu binarnego. Pliki binarne przesłane w trybie ASCII nie będą się nadawały do użycia.

Tryb ASCII pozwala tylko na przesyłanie znaków wchodzących w skład standardu ASCII (kodowanych za pomocą siedmiu bitów), w plikach binarnych natomiast mogą występować również kody nie należące do tego standardu. Przesłanie pliku ASCII w trybie `binary` nie narusza jego zawartości.

Aby opuścić program `ftp`, wydaj polecenie `quit` albo `exit`. Obydwa te polecenia powodują zamknięcie sesji w systemie zdalnym i zakończenie działania programu `ftp` w systemie lokalnym. Poniższa lista przedstawia inne, często używane polecenia dostępne w programie FTP:

ascii	włączenie tekstowego trybu przesyłania danych;
binary	włączenie binarnego trybu przesyłania danych;
cd	zmiana katalogu bieżącego na serwerze;
close	zamknięcie połączenia;
del	usunięcie pliku na serwerze;
dir	wyświetlenie zawartości bieżącego katalogu serwera;
get	pobranie pliku z serwera;
hash	wyświetlanie znaków # po załadowaniu każdego bloku danych;
help	wyświetlenie listy dostępnych poleceń;
lcd	zmiana katalogu bieżącego w systemie lokalnym;
mget	pobranie kilku plików z serwera;
mput	przesłanie kilku plików do serwera;
open	nawiązanie połączenia z serwerem;
put	wysłanie pliku do serwera;
pwd	wyświetlenie nazwy katalogu bieżącego na serwerze;
quote	bezpośrednie podanie polecenia protokołu FTP;
quit	zakończenie sesji FTP.

W większości wersji FTP w poleceniach rozróżniane są małe i wielkie litery, wpisanie więc polecenia wielkimi literami spowoduje wyświetlenie komunikatu o błędzie. W niektórych wersjach w takim przypadku nastąpi automatyczna konwersja na małe litery. Ponieważ jednak w systemie Linux prawie we wszystkich zastosowaniach używa się małych liter, dobrym pomysłem będzie używanie ich i tu.

Powiązania pomiędzy FTP a TCP/IP

Protokół FTP wykorzystuje dwa kanały TCP: port 20 jest używany do przesyłania danych, a port 21 do przesyłania poleceń. Oba te porty muszą funkcjonować, jeśli FTP ma działać poprawnie. Wykorzystanie dwóch kanałów odróżnia FTP od innych programów do przesyłania plików. Dzięki temu możliwe jest równoczesne przesyłanie poleceń i danych. FTP działa na pierwszym planie i nie umożliwia użycia mechanizmów kolejkowania.

W połączeniu FTP bierze udział program rezydentny, działający przez cały czas po stronie serwera, oraz program nazywany klientem FTP, uruchamiany po stronie nawiązującej połączenie. W systemie Linux program rezydentny dla serwera nazywa się `ftpd`, natomiast program klienta FTP – `ftp`.

Podczas nawiązywania połączenia pomiędzy klientem i serwerem oraz po wydaniu przez użytkownika każdego polecenia, pomiędzy komputerami przesyłane są ciągi poleceń. Polecenia te są dostępne tylko dla protokołu FTP i nazywa się je protokołem wewnętrzny. Protokół wewnętrzny FTP składa się z czteroznakowych poleceń w standardzie ASCII, rozdzielanych znakiem nowego wiersza. Niektóre z poleceń wymagają podania parametrów. Główną zaletą takiego rozwiązania jest możliwość obserwacji i zrozumienia przez użytkownika poleceń, jakie są przesyłane pomiędzy komputerami. Znacznie ułatwia to proces wyszukiwania ewentualnych błędów. Poza tym doświadczony użytkownik może używać połączeń FTP nie wykorzystując programu klienta (innymi słowy, porozumiewać się z programem `ftpd` bez użycia programu `ftp`), ale ma to zastosowanie tylko do wyszukiwania przyczyn ewentualnych problemów (no i ewentualnie popisania się przed kolegami).

Po zalogowaniu się do systemu zdalnego za pomocą FTP nie przenosisz się do tego systemu. Nadal jesteś logicznie po stronie klienta, więc wszystkie instrukcje dotyczące przesyłania plików i poruszania się po drzewie katalogów odnoszą się do komputera lokalnego, a nie do serwera. Po nawiązaniu połączenia należy:

1. zalogować się, podając identyfikator użytkownika i hasło;
2. przejść do katalogu, w którym znajdują się interesujące nas pliki lub do którego zamierzamy wysłać pliki;
3. zdefiniować tryb przesyłania danych;
4. rozpocząć przesyłanie danych (lub też wydać polecenia służące innym celom);
5. po zakończeniu przesyłania danych zamknąć połączenie.

Powyższe kroki należy podjąć dla każdego połączenia.

Po dodaniu do polecenia `ftp` opcji `-d` uruchamiany jest tryb wyszukiwania błędów (ang. *debugging*). W tym trybie wszystkie polecenia przesyłane do portu poleceń są wyświetlane na ekranie. Instrukcje pochodzące od klienta oznaczone są strzałką, natomiast od serwera – trzycyfrową liczbą. Polecenie `PORT` oznacza adres kanału danych, na którym klient oczekuje na odpowiedź od serwera. Jeśli nie jest on podany, domyślnie używany jest port 20. Niestety, w trybie wyszukiwania błędów nie jest możliwe śledzenie postępów w transmisji danych. Przykładowa sesja FTP z załączoną opcją `-d` przedstawiona jest poniżej.

```
$ ftp -d tpci_hpws4
Connected to tpci_hpws4.
220 tpci_hpws4 FTP server (Version 1.7.109.2 Tue Jul 28 23:32:34 GMT
1992) ready.
Name (tpci_hpws4:tparker):
--> USER tparker
```

```
331 Password required for tparker.  
Password:  
---> PASS qwerty5  
230 User tparker logged in.  
---> SYST  
215 UNIX Type: L8  
Remote system type is UNIX.  
---> Type I  
200 Type set to I.  
Using binary mode to transfer files.  
ftp> ls  
---> PORT command successful.  
---> TYPE A  
200 Type set to A.  
---> LIST  
150 Opening ASCII mode data connection for /bin/ls.  
total 4  
-rw-r----- 1 tparker tpc1 2803 Apr 29 10:46 file1  
-rw-rw-r-- 1 tparker tpc1 1264 Apr 14 10:46 file5_draft  
-rwxr----- 2 tparker tpc1 15635 Mar 14 23:23 test_comp_1  
-rw-r----- 1 tparker tpc1 52 Apr 22 12:19 xyzzy  
Transfer complete.  
--->TYPE I  
200 Type set to I.  
ftp> <Ctrl-d>  
$
```

W powyższym przykładzie można zaobserwować, że tryb przesyłania danych jest automatycznie zmieniany z binarnego na tekstowy podczas przesyłania informacji o zawartości katalogu, a następnie jest przywracany tryb binarny (który w tym przypadku jest trybem domyślnym).

Konfiguracja FTP

Jeśli zdecydujesz, że Twój komputer powinien działać jako serwer FTP (udostępniający lub nie anonimowe FTP), musisz odpowiednio skonfigurować system – przede wszystkim uruchamiając rezydentny program obsługi FTP i konfiguruując system plików tak, by uniemożliwić dostęp do plików niepowołanym osobom. Proces konfiguracji rozpoczyna się od wybrania nazwy węzła FTP. Nazwa ta nie jest niezbędna, ale ułatwi innym łączenie się z Twoim systemem (szczególnie w przypadku połączeń anonimowych). Nazwa węzła FTP powinna mieć format:

ftp.nazwa_domeny.typ_domeny

nazwa_domeny jest nazwą domeny, do której przynależy węzeł FTP, zaś typ_domeny to standardowe rozszerzenie DNS. Przykładowa nazwa

ftp(tpci.com

sugeruje, że jest to węzeł obsługujący anonimowe połączenia FTP dla domeny tpc1.com. Używanie nazwy komputera w nazwie węzła FTP, na przykład

ftp.merlin(tpci.com

zwykle nie jest dobrym pomysłem, ponieważ powoduje problemy w sytuacji, gdy chcesz przenieść obsługę FTP na inny system należący do danej domeny. Zamiast tego warto użyć aliasu, który będzie wskazywał na konkretny komputer pełniący funkcje serwera FTP. Takie rozwiązanie nie jest konieczne, jeśli komputer nie wchodzi w skład sieci lokalnej, tylko jest połączony z Internetem za pośrednictwem dostawcy Internetu, ale często okazuje się nieodzowne w większych sieciach. Jeśli używasz systemu DNS, odpowiedni alias może zostać zdefiniowany bardzo łatwo. W takim przypadku wystarczy do bazy danych o aliasach dodać wiersz

```
ftp IN CNAME merlin.tpc.com
```

podstawiając oczywiście nazwę własnego komputera. Wiersz ten skieruje każdego próbującego się połączyć z węzłem `ftp.tpc.com` do komputera `merlin.tpc.com`, działającym jako serwer FTP. Jeśli z jakichś powodów inny komputer przejmie rolę serwera `ftp`, należy zmodyfikować odpowiedni wpis w bazie danych o aliasach, tak by wskazywał na komputer pełniący aktualnie funkcję serwera (zmiana taka nie wchodzi w życie od razu, ponieważ odpowiednie informacje muszą zostać rozesłane do wszystkich baz danych DNS).

Konfigurowanie programu `ftpd`

Program rezydentny `ftpd` musi zostać uruchomiony po stronie serwera FTP. Zwykle jest on uruchamiany nie w jednym z plików `rc`, ale przez program `inetd`, dzięki czemu działa tylko wtedy, gdy jest potrzebny. Jest to najlepsze rozwiązanie w niezbyt obciążonych węzłach FTP. Program `inetd` obserwuje przez cały czas port poleceń TCP (kanal 21) i po nadejściu pakietu danych z żądaniem nawiązania połączenia uruchamiany jest program `ftpd`.

Powinieneś upewnić się, że program `inetd` rzeczywiście może uruchomić `ftpd`, sprawdzając, czy w pliku konfiguracyjnym programu `inetd` (plik ten zwykle nazywa się `/etc/inetd.conf` lub `/etc/inetd.conf`) istnieje wpis podobny do następującego:

```
ftp stream tcp nowait root /usr/etc/ftpd ftpd -l
```

Jeśli takiego wpisu nie ma, należy go dodać. W większości systemów linuxowych taki wpis znajduje się w odpowiednim pliku, ale może być zaznaczony jako komentarz. W takim przypadku wystarczy usunąć symbol komentarza. Wiersz ten informuje program `inetd`, że FTP będzie korzystał z protokołu TCP, i że za każdym razem, gdy wykryte zostanie połączenie z portem FTP, powinien zostać uruchomiony program `ftpd` z opcją `-l`, załączającą rejestrowanie połączeń. Opcje tę można zignorować. Jeśli program `ftpd` znajduje się w innym miejscu niż `/usr/etc/ftpd`, należy oczywiście odpowiednio zmodyfikować ścieżkę dostępu.

Program `ftpd` udostępnia jeszcze kilka innych opcji, pozwalających kontrolować jego działanie – one również mogą zostać podane w pliku `/etc/inetd.conf`. Poniżej podajemy listę najczęściej używanych z nich:

- d uruchamia wysyłanie informacji o podejmowanych czynnościach do procesu `syslog`;
- l aktywuje rejestrowanie danych o sesjach (tylko informacje o udanych i nieudanych połączeniach, bez żadnych informacji dodatkowych); jeśli opcja `-l` zostanie podana dwukrotnie, rejestrowane będą również wszystkie wydane polecenia; jeśli opcja `-l` jest podana trzykrotnie, rejestrowane są także rozmiary plików przesyłanych poleceniami `put` i `get`;
- t ustawia czas, po jakim należy wyłączyć program `ftpd` po zakończeniu sesji (domyślnie: 15 minut); wartość należy podać w sekundach po opcji `-t`;
- T ustawia maksymalny czas połączenia (w sekundach); wartość domyślana: 2 godziny;
- u ustawia wartość zmiennej `umask` dla plików ładowanych do systemu lokalnego (domyślnie: 022); klient może zażądać zmiany tej wartości.

Konta FTP

Jeśli chcesz, by każdy użytkownik korzystający z Twojego węzła FTP posiadał własny identyfikator i hasło, musisz dla każdego z nich utworzyć osobne konto w pliku `/etc/passwd`. Jeśli nie zamierzasz umożliwiać anonimowego dostępu do FTP, nie powinieneś tworzyć konta ogólnodostępnego.

Aby umożliwić anonimowy dostęp do serwera FTP, należy założyć konto dla anonimowego użytkownika (jeśli konto takie jeszcze nie istnieje w systemie, ponieważ w wielu wersjach Linuxa konto takie zakładane jest podczas instalacji, choć dostęp do niego jest zablokowany). Sposób postępowania jest taki sam, jak w przypadku zakładania konta dla zwykłego użytkownika. Identyfikator użytkownika anonimowego powinien mieć postać `anonymous` lub `ftp`, ale można go zdefiniować dowolnie. Katalog domowy takiego użytkownika powinien być ze względów bezpieczeństwa oddzielony od reszty systemu plików. Typowy wpis dla anonimowego użytkownika o identyfikatorze `ftp` w pliku `/etc/passwd` ma postać:

```
ftp:*:400:51:Anonymous FTP access:/usr/ftp:/bin/false
```

Gwiazdka w polu zawierającym hasło powoduje, że dostęp do konta jest zablokowany. Liczbowy identyfikator użytkownika (400) musi oczywiście być niepowtarzalny w obrębie systemu. Ze względów bezpieczeństwa dobrze jest utworzyć dla użytkownika `ftp` osobną grupę (edytując plik `/etc/group`), a następnie przypisać go do tej grupy. Użytkownik `ftp` powinien być jedynym użytkownikiem należącym do tej grupy; dzięki takiemu rozwiązaniu można lepiej skonfigurować prawa dostępu do plików w systemie. Katalogiem domowym w powyższym przykładzie jest `/usr/ftp`, ale możesz wybrać dowolny inny katalog, pod warunkiem że należy on do użytkownika `root` (jest to znów podyktowane względami bezpieczeństwa). Programem uruchamianym po zalogowaniu jest `/bin/false`, co dodatkowo zabezpiecza system przed dostępem do kont i programów nie posiadających dobrego zabezpieczenia hasłem.

Konfigurowanie katalogów

Jak przekonasz się czytając następny podrozdział - „Ustawianie praw dostępu”, możesz spróbować wyizolować z systemu plików podsystem dostępny dla anonimowego użytkownika FTP. Dzięki temu nie będzie mógł on wydostać się poza katalog `/usr/ftp` (czy jakkolwiek inny katalog, który zostanie użyty jako katalog domowy użytkownika `ftp`). W tym celu należy stworzyć jakby miniaturę systemu plików, zachowując standardowe nazwy katalogów i kopując do nich podstawowe pliki, których użytkownik `ftp` może potrzebować.

Proces konfigurowania katalogów, które będą niezbędne anonimowemu użytkownikowi FTP, jest prosty, wymaga tylko utworzenia pewnej liczby katalogów i skopiowania do nich odpowiednich plików. Poniżej przedstawiamy opis takiej procedury.

1. Utwórz podkatalog `bin` (na przykład `/usr/ftp/bin`) i skopiuj do niego polecenie `ls`, które będzie potrzebne użytkownikom do wyświetlania informacji o zawartości katalogów.
2. Utwórz podkatalog `etc` (na przykład `/usr/ftp/etc`) i skopiuj do niego plik `passwd` (`/etc/passwd`) oraz `group` (`/etc/group`). Za chwilę zajmiemy się modyfikowaniem zawartości tych plików.
3. Utwórz podkatalog `lib` (na przykład `/usr/ftp/lib`) i skopiuj do niego pliki `/lib/ld.so` i `/lib/ld.so.X` (gdzie X jest numerem wersji pliku `libc`). Pliki te są używane przez program `ls`. Ten krok jest wymagany tylko wtedy, gdy wersja programu `ls` dostępna w systemie używa biblioteki `libc`; w większości przypadków nie jest to potrzebne.
4. Utwórz podkatalog `pub` (na przykład `/usr/ftp/pub`), w którym przechowywane będą pliki udostępniane dla użytkowników. Katalogowi temu przyjrzymy się dokładniej za chwilę.
5. Utwórz podkatalog `dev` (na przykład `/usr/ftp/dev`) i za pomocą polecenia `mknod` skopiuj do niego plik `/dev/zero`. Skopiowany plik musi posiadać taki sam numer główny i poboczny urządzenia jak plik oryginalny. Jest on używany przez bibliotekę `ld.so` (i co za tym idzie, polecenie `ls`). Ten krok jest niezbędny tylko w przypadku, gdy dostępna w systemie wersja programu `ls` wymaga pliku `/lib/ld.so`.

Do katalogu `~ftp/etc` trafiają kopie plików `/etc/passwd` i `/etc/group`. Należy usunąć z nich wszystkie wpisy oprócz tych dotyczących użytkownika `ftp` (zwykle tylko `anonymous` i `bin`), a w miejsce haseł wstawić gwiazdki.

W katalogu `~ftp/pub` zwykle przechowywane są pliki, które mają być udostępniane anonimowym użytkownikom FTP. Można w nim utworzyć podkatalogi, tak aby zachować przejrzysty układ plików. Można również stworzyć katalog, do którego użytkownicy będą mogli nadsyłać własne pliki (ang. *upload directory*); w takim katalogu użytkownik `ftp` musi mieć prawo zapisu.



Jeśli zezwolisz użytkownikom zdalnym na korzystanie z Twojego systemu, zwróć uwagę na fakt, że zabezpieczenia powinny być legalne - dlatego powinieneś uprzedzić każdego użytkownika, że jego działania w Twoim systemie mogą być obserwowane. W dzisiejszych czasach trzeba zabezpieczać się na każdym kroku!

Ustawianie praw dostępu

Polecenie `chroot` pomaga chronić system przed nieautoryzowanym dostępem do plików. Umożliwia ono potraktowanie jednego z podkatalogów jako katalogu głównego. Przykładowo, polecenie `chroot` jest zwykle wydawane w przypadku użytkownika `ftp`, więc wszystkie polecenia związane z systemem plików będą odniesione do jego katalogu domowego. Innymi słowy, polecenie `cd /bin` nie spowoduje przejścia do katalogu `/bin`, ale do `/usr/ftp/bin` (o ile katalogiem domowym użytkownika `ftp` jest `/usr/ftp`). Takie rozwiązanie ułatwia ograniczanie dostępu do jakichkolwiek plików znajdujących się poza podsystemem zapisanym w katalogu domowym użytkownika.

Jeśli zdecydujesz się na utworzenie katalogu, w którym anonimowy użytkownik będzie miał możliwość zapisu, powinieneś nadać mu prawo do wykonywania i zapisu, ale nie do odczytu – dzięki temu nikt nie będzie mógł załadować plików pozostawionych przez kogoś w Twoim systemie.

Wszystkie pozostałe podkatalogi katalogu `/usr/ftp` powinny mieć ustawione prawa dostępu tak, by nikt nie miał prawa zapisu. Upewnij się, że anonimowy użytkownik może odczytywać z nich dane (właścicielem katalogów i plików powinien być użytkownik `root`, grupa `root` lub `system`) – dla wszystkich plików powinien być możliwy tylko odczyt. Katalogi, do których anonimowy użytkownik powinien mieć możliwość wejścia czy uzyskania informacji o ich zawartości (jak katalog `pub`), powinny mieć również ustawione prawo do wykonywania. Przestrzeganie tych wskazówek zapewnia stosunkowo wysoki poziom bezpieczeństwa. Prawo do odczytu i wykonywania dla katalogu możesz nadać poleceniem:

```
chmod 555 nazwa_katalogu
```

Jedynym wyjątkiem jest katalog, do którego anonimowy użytkownik może przesyłać dane, który może mieć ustawione prawo do zapisu.

Testowanie systemu

Zanim pozwolisz komukolwiek korzystać z węzła FTP, powinieneś zalogować się sam i spróbować uzyskać dostęp do plików, do których nie powinieneś mieć dostępu, wydostać się poza katalog `~ftp`, usunąć pliki, których nie powinieneś mówić usunąć itp. Warto poświęcić na to kilka minut. Upewnij się, że wszystko jest zapięte na ostatni guzik. Jeśli pozostawisz jakieś luki, znajdzie je i wykorzysta ktoś inny.

Często przydatne jest założenie specjalnej skrzynki pocztowej dla administratora systemu FTP, dzięki czemu użytkownicy innych systemów, którzy potrzebują pomocy czy informacji, będą mogli wysłać do Ciebie pocztę. W tym celu powinieneś utworzyć w pliku `/etc/aliases` nowy alias (na przykład `ftp-admin`) i uruchomić polecenie `newaliases`, aby wprowadzić zmiany w życie.

Nie będziemy wchodzić w szczegóły organizowania drzewa podkatalogów katalogu domowego użytkownika `ftp`, podamy tylko kilka porad. Na początek należy zdecydować, gdzie będą przechowywane katalogi FTP i zorganizować je w jakiś rozsądny sposób. Przykładowo, jeśli chcesz udostępnić kilka napisanych przez siebie programów, powinieneś umieścić każdy z nich w osobnym katalogu. Plik `README` w każdym katalogu powinien zawierać objaśnienie co do jego zawartości. Główny plik `README` albo `INSTRUCTIONS` w katalogu `~ftp` może wyjaśniać szczegóły konfiguracji Twojego węzła i skrótnie przedstawiać jego zawartość.

Bezpieczniejsze FTP

System FTP przedstawiony w poprzednich podrozdziałach, rozprowadzany praktycznie z każdą wersją Linuxa, wymaga nieco pracy jeśli chcesz, by był bezpieczny. Niestety, nie jest on zabezpieczony przed bardziej doświadczonymi hakerami. Na szczęście istnieje alternatywa dla tych, którym bardzo zależy na bezpieczeństwie systemu; jest nią WU FTP. Jest to system FTP opracowany w Uniwersytecie Waszyngtońskim, który do standardowego systemu FTP dodaje takie cechy, jak:

- υ dokładniejsza kontrola identyfikatorów użytkowników i identyfikatorów grup,
- υ poprawione śledzenie przesyłania plików do i z systemu,
- υ automatyczne zamknięcie systemu,
- υ automatyczna kompresja i dekompresja plików.

Jeśli uważasz, że powyższe cechy to właśnie to, czego potrzebujesz, możesz uzyskać kopię kodu źródłowego WU FTP w kilku węzłach FTP, na przykład w węźle `wuarchive.wustl.edu`. Poszukaj tam pliku `/packages/wuarchive-ftpd/wu-ftpd-X.X.tar.Z` (gdzie `X.X` jest numerem wersji). Otrzymany kod musi zostać skompilowany w systemie lokalnym.

Szczegóły konfiguracji systemu WU FTP są ustalane poprzez definiowanie odpowiednich zmiennych środowiskowych – z pomocą przyjdzie Ci dokumentacja rozprowadzana razem z kodem źródłowym. Konfigurowanie WU FTP jest o wiele bardziej złożone, niż zwykłego systemu FTP, a dodatkowe zabezpieczenia, choć przydatne, dla wielu węzłów FTP (założonych na przykład w domu czy pracy) mogą być zupełnie niepotrzebne (o ile w systemie nie są przechowywane szczególnie ważne informacje).

Zabezpieczanie anonimowego FTP

Anonimowe FTP jest szybkie i łatwe w użyciu, ale źle skonfigurowane może stanowić ogromną lukę w bezpieczeństwie systemu. Poniżej przedstawiamy kilka kroków, które należy koniecznie podjąć podczas konfigurowania anonimowego FTP.

1. Utwórz konto o identyfikatorze `ftp`. W tym celu ręcznie zmodyfikuj zawartość pliku `/etc/passwd`, a w miejsce hasła wpisz gwiazdkę. Dzięki temu nikt nie będzie mógł dostać się do systemu przez konto `ftp`.
2. Jeśli podczas zakładania konta nie został utworzony katalog domowy użytkownika `ftp`, załącz go (może to być na przykład katalog `/usr/ftp`); katalog ten powinien być używany tylko przez użytkownika `ftp`.
3. Właścicielem katalogu domowego użytkownika `ftp` powinien być `root`, więc wydaj polecenie:

```
chown root /usr/ftp
```

4. Zabezpiecz katalog domowy użytkownika `ftp` przed zapisem wydając polecenie:

```
chmod ugo-w /usr/ftp
```

5. Utwórz w katalogu `ftp` podkatalog `bin`:

```
mkdir ~ftp/bin
```

6. Właścicielem podkatalogu `bin` również powinien być `root`; należy także zabezpieczyć go przed zapisem:

```
chown root ~ftp/bin  
chmod ugo-w ~ftp/bin
```

7. Skopiuj do podkatalogu `bin` polecenie `ls` (i inne polecenia, które chcesz udostępnić anonimowym użytkownikom):

```
cp /bin/ls ~ftp/bin
```

8. Utwórz podkatalog `etc`; powinien on być zabezpieczony przed zapisem, a jego właścicielem powinien być użytkownik `root`:

```
mkdir ~ftp/etc  
chown root ~ftp/etc  
chmod ugo-w ~ftp/etc
```

9. Skopiuj do katalogu `~ftp/etc` pliki `/etc/passwd` i `/etc/group`. Następnie z utworzonych kopii usuń wszystkie wpisy oprócz tych dotyczących użytkownika `ftp` (a przynajmniej usuń wszystkie hasła innych użytkowników i zastąp je gwiazdką).

- 10.** Utwórz podkatalog `~ftp/pub/incoming`; jego właścicielem powinien być użytkownik `root`. Następnie wydaj polecenie zezwalające użytkownikowi `ftp` na zapis do tego katalogu:

```
mkdir ~ftp/pub/incoming  
chown root ~ftp/pub/incoming  
chmod ugo+w ~ftp/pub/incoming
```

- 11.** Wszystkie pliki, które chcesz udostępnić, umieść w podkatalogu `pub` katalogu domowego użytkownika `ftp`. Dzięki temu anonimowi użytkownicy FTP będą mogli je pobrać. Zezwolenie użytkownikom na zapisywanie plików w tym katalogu nie jest pożądane, powinieneś więc odebrać wszystkim prawo zapisu albo często sprawdzać, czy nikt nie uszkodził plików.

Jeśli przy konfigurowaniu anonimowego dostępu do FTP przestrzegałeś powyższych wskazówek (dosłosowując je do potrzeb swojego systemu), Twój węzeł FTP jest względnie bezpieczny.

Podsumowanie

Informacje zawarte w tym rozdziale pozwalają skonfigurować system jako węzeł FTP zarówno dla ograniczonej liczby użytkowników, jak i z anonimowym dostępem do zasobów. Choć sam proces konfiguracji jest dość prosty, powinieneś bardzo dokładnie sprawdzić, czy wszystkie prawa dostępu są ustawione prawidłowo. Po uruchomieniu węzła FTP warto poinformować o jego istnieniu innych użytkowników sieci lokalnej czy Internetu, podając również krótki opis zasobów udostępnianych w Twoim systemie.

Ustalanie właścicieli plików i praw dostępu do nich, konieczne przy przesyłaniu plików za pomocą protokołu FTP, omówione jest w rozdziale 9. „Prawa dostępu do plików i katalogów”.

Jeśli chcesz dowiedzieć się, jak można napisać prosty program w języku powłoki, pozwalający na przykład na przesyłanie plików po wydaniu pojedynczego polecenia, przeczytaj rozdział 14. „Programowanie w języku powłoki”.

Konfiguracja systemu linuxowego do pracy w sieci lokalnej omówiona jest w rozdziale 37. „Praca w sieci”.

Rozdział 49.

Konfiguracja węzła WAIS

Tim Parker

W tym rozdziale:

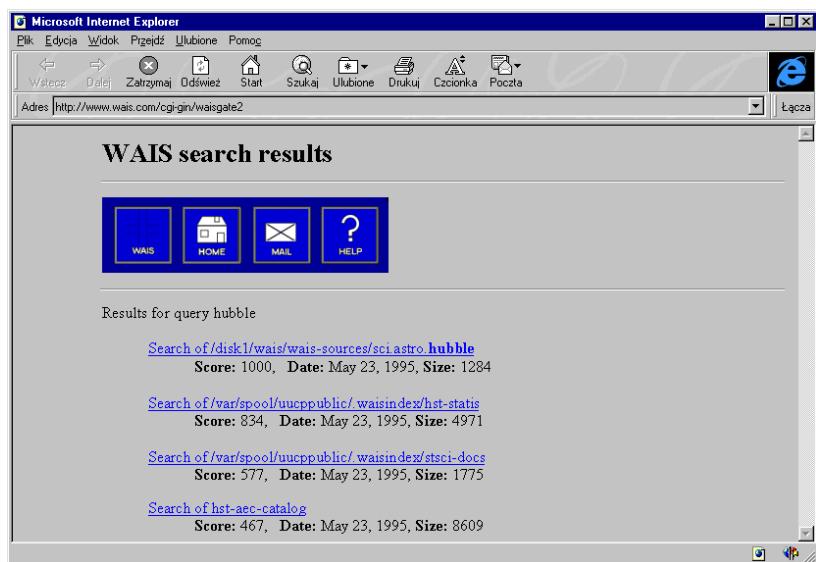
- υ Kompilowanie i instalacja programu freeWAIS
- υ Konfigurowanie programu freeWAIS
- υ Uruchamianie serwera freeWAIS
- υ Tworzenie własnych indeksów dla programu WAIS

WAIS (ang. *Wide Area Information Service*) to narzędzie obsługiwane za pośrednictwem systemu menu, pozwalające na wyszukiwanie słów kluczowych w bazach danych o dokumentach dostępnych w systemie. Program WAIS został opracowany w firmie Thinking Machines, ale jak tylko zdobył popularność, zajęła się nim specjalnie w tym celu założona firma WAIS Incorporated. Jego darmowa wersja, znana pod nazwą freeWAIS została udostępniona instytucji Clearinghouse for Networking Information Discovery and Retrieval (CNDIR) - i jest to wersja najczęściej spotykana w systemach linuxowych.

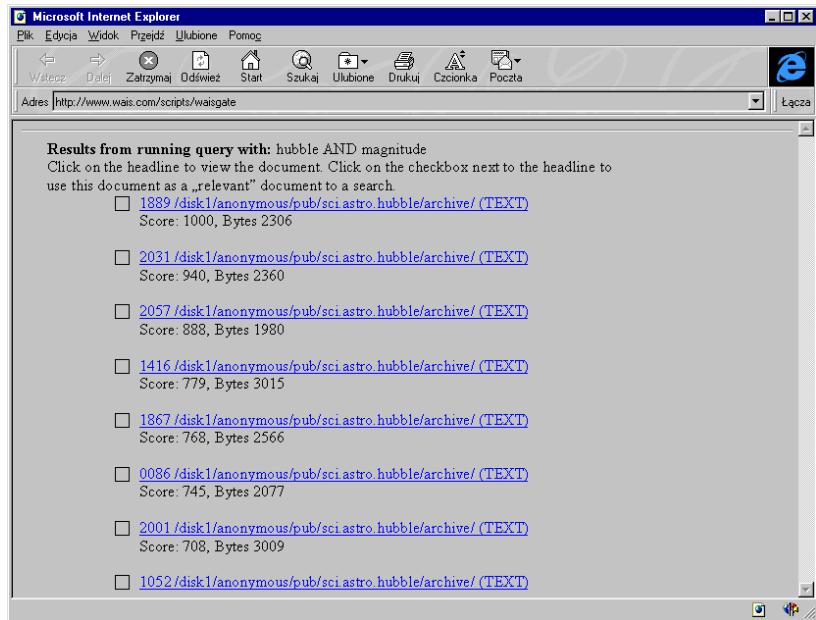
WAIS pozwala na wyszukiwanie wprowadzonych przez użytkownika słów kluczowych czy nawet zdań w bazach danych. Typowy wygląd okienka tego programu przedstawia rysunek 49.1 (wyniki wyszukiwania pochodzą z pierwotnego serwera WAIS, dostępnego pod adresem <http://www.wais.com>; można tam również znaleźć wiele przykładów demonstrujących możliwości tego programu). W przykładzie przedstawionym na rysunku próbowaliśmy znaleźć słowa kluczowe „hubble” (amerykański astronom, jego imieniem nazwano jeden z największych teleskopów; wielkość liter w programie WAIS nie ma zwykle znaczenia) i „magnitude” (jasność gwiazdowa). Po przeszukaniu wszystkich dostępnych indeksów baz danych, WAIS przedstawia wyniki poszukiwań – odpowiednie okienko przedstawione jest na rysunku 49.2.

Rysunek 49.1.

W programie WAIS można użyć prostych i bardziej złożonych kryteriów wyszukiwania

**Rysunek 49.2.**

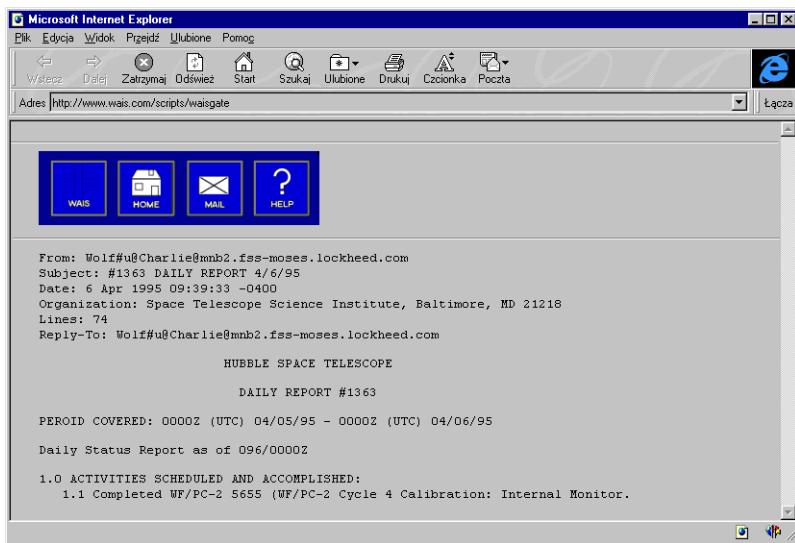
WAIS wyświetla znalezione dokumenty wraz z punktacją



Dane generowane przez program WAIS, wyświetlane w przeglądarce WAIS (jak na rysunkach) lub przeglądarce WWW, zawierają listę wszystkich dokumentów spełniających kryteria wyszukiwania, wraz z punktacją od 0 do 1000, pozwalającą użytkownikowi ocenić, jak dobrze każdy z dokumentów spełnia zadane kryteria (im lepiej, tym więcej punktów).

Rysunek 49.3.

Wybranie dowolnego ze znalezionych dokumentów pozwala obejrzeć jego zawartość



Po zakończeniu wyszukiwania można zdefiniować nowe kryteria lub przejrzeć znalezione dokumenty. Na rysunku 49.3 przedstawione jest okno przeglądarki WWW zawierające jeden ze znalezionych dokumentów. WAIS potrafi obsłużyć wiele formatów plików, włączając w to dokumenty tekstowe, pliki audio, grafiki w formacie GIF i JPEG, a także pliki binarne.

Najczęściej używaną w systemach linuxowych wersją programu WAIS jest `freeWAIS`. W tym rozdziale pokażemy, jak można skonfigurować komputer linuxowy jako serwer WAIS. Warto rozważyć umożliwienie korzystania z takiej usługi szczególnie wtedy, gdy chcesz udostępnić innym użytkownikom większą ilość informacji – mogą to być informacje o produktach, o Twoim hobby czy dane praktyczne każdego innego typu, o ile chcesz udostępnić je innym użytkownikom sieci lokalnej lub Internetu.

Pakiet `freeWAIS` składa się z trzech części: programu indeksującego dokumenty, serwera WAIS oraz programu klienta. Program indeksujący służy do generowania baz danych, zawierających indeksy, składające się ze słów kluczowych i tabeli ich wystąpień w dokumentach. Serwer zapewnia obsługę zapytań użytkownika na podstawie plików z indeksami. Program klienta umożliwia użytkownikowi dostęp do systemu WAIS – jest to zwykle przeglądarka WAIS lub WWW. Przeglądarki WWW mają tę przewagę nad przeglądarkami WAIS, że oprócz stron WAIS potrafią również wyświetlać dokumenty HTML.

Następcą programu WAIS jest program ZDIST (kompatybilny ze swym poprzednikiem). Zachowuje się on bardzo podobnie do programu WAIS; wszystkie zmiany są opisane w dokumentacji. ZDIST ma kilka nowych cech, jest również nieco mniejszy i szybszy niż `freeWAIS`. Ponieważ jest jednak wciąż w fazie testów, skoncentrujemy się na programie `freeWAIS`.

Kompilowanie i instalacja programu freeWAIS

Program `freeWAIS` jest często dodawany do pełnych dystrybucji Linuxa, ale jest również dostępny w wielu węzłach FTP i BBS. Można go także załadować łącząc się z adresem <http://ls6-www.informatik.uni-dortmund.de/ir/projects/freeWAIS-sf/index.html>. W węźle tym dostępne są zarówno pliki wykonywalne przeznaczone dla różnych komputerów, jak i kod źródłowy, który może być dostosowany do potrzeb wielu systemów.

Jednym z plików wchodzących w skład dystrybucji tego programu jest plik `Makefile`, ułatwiający komplikację programu, który powinien znaleźć się w katalogu docelowym. Jeśli komplujesz kod źródłowy samodzielnie, powinieneś upewnić się, że wszystkie zmienne definiowane w pliku `Makefile` mają poprawne wartości. W większości przypadków wskazują one na standardowe programy użytkowe – wtedy nie trzeba ich modyfikować. Poniżej przedstawiamy kilka zmiennych, których modyfikacja może okazać się niezbędna:

- | | |
|-----------------|---|
| CC | nazwa dostępnego w systemie kompilatora C (zwykle <code>cc</code> lub <code>gcc</code>); |
| CURSELIB | wartość odpowiadająca wersji biblioteki <code>curses</code> używanej w systemie; |
| TOP | pełna ścieżka dostępu do katalogu zawierającego kod źródłowy programu WAIS. |

Znaczniki wyszczególnione w zmiennej `CFLAGS` pozwalają na określenie zachowania kompilatora. Dostępnych jest wiele znaczników – większość z nich jest opisana w dokumentacji rozprowadzanej wraz z kodem źródłowym. W większości systemów linuxowych wystarcza pozostawienie wartości domyślnych. Warto jednak wspomnieć o znaczeniu kilku z nich, ponieważ możesz chcieć zmienić ich wartość. Najistotniejsze są znaczniki dotyczące programu indeksującego – w praktyce przydatne mogą być dwa z nich:

- | | |
|--------------------|---|
| -DBIO | pozwala indeksować symbole i terminy biologiczne – znacznik ten powinien być używany tylko wtedy, gdy w Twoim węźle udostępniane są materiały z dziedziny biologii; |
| -DBBOOLEANS | pozwala używać operatorów logicznych AND (iloczyn) i NOT (negacja), dzięki czemu możliwe jest bardziej precyzyjne określanie kryteriów wyszukiwania. |

Znacznik `-DBBOOLEANS` umożliwia używanie operatorów logicznych przy podawaniu kryteriów wyszukiwania. Jeśli na przykład szukasz słów kluczowych „zielony las”, WAIS domyślnie będzie szukał każdego ze słów oddzielnie, niezależnie również przyznając punkty. Jeżeli przy komplikacji aktywny będzie znacznik `-DBBOOLEANS`, będzie można poszukać dokumentów, w których oba słowa występują jednocześnie, używając operatora AND (iloczyn logiczny).

Oto kilka innych znaczników, które mogą być przydatne do określania zachowania się systemu WAIS jako całości:

-DBIGINDEX	powinien być ustawiany tylko wtedy, jeśli w węźle udostępnianych jest dużo (tysiące) dokumentów;
-DLITERAL	pozwala na wyszukiwanie całych tekstów, zamiast wyszukiwania wystąpień pojedynczych wyrazów wchodzących w skład tekstu;
-DPARTIALWORD	umożliwia używanie gwiazdki jako symbolu wieloznacznego (na przykład <code>auto*</code>);
-DRELEVANCE_FEEDBACK	jeśli znacznik ten jest ustawiony na <code>ON</code> , użytkownik może wykorzystać wyniki poprzedniego wyszukiwania jako nowe kryteria wyszukiwania – taka możliwość jest dość przydatna w praktyce.

Zawartość pakietu `freeWAIS` jest rozrzucona po wielu katalogach; przeznaczenie większości z nich jest oczywiste (w katalogu `bin` znajdują się pliki wykonywalne, w katalogu `man` – strony `man` itd.). Katalogi używane przez program `freeWAIS` w domyślnej konfiguracji to:

bin	pliki wykonywalne;
config.c	pliki źródłowe w języku C, służące do konfiguracji;
doc	pliki zawierające dokumentację i odpowiedzi na często zadawane pytania;
include	pliki nagłówkowe, używane przez kompilator;
lib	pliki bibliotek;
man	strony <code>man</code> ;
Src	kod źródłowy programu <code>freeWAIS</code> ;
Wais-Sources	katalog serwerów internetowych;
Wais-Test	przykładowe skrypty programu indeksującego.

Po wprowadzeniu odpowiednich poprawek do pliku konfiguracyjnego można skompilować program `freeWAIS`, wydając polecenie:

```
make linux
```

Domyślnie program `make` generuje dwa programy klienta: `swais` i `waisq`. Jeśli chcesz skompilować wersję przeznaczoną do pracy w systemie X Window o nazwie `xwais` (przydatną, jeśli chcesz umożliwić dostęp z terminali X lub konsol), usuń z pliku `Makefile` znacznik komentarza w wierszu, który kończy się tekstem `makex`.

Konfigurowanie programu freeWAIS

Po prawidłowym skompilowaniu i zainstalowaniu programu `freeWAIS` można przystąpić do indeksowania zgromadzonych w systemie dokumentów. Zwykle w tym celu tworzony jest nowy katalog, o nazwie `waisindex`. Często umieszcza się go w katalogu głównym (`/waisindex`), ale wielu administratorów woli przechowywać go w jakimś innym, wydzielonym miejscu (na przykład `/usr/wais/waisindex`). Należy wziąć pod uwagę fakt, że jeśli pliki z indeksami będą trudne do zlokalizowania, użytkownicy mogą mieć problemy z ich odszukaniem.

Katalog `wais-test`, tworzony podczas instalacji programu `freeWAIS`, zawiera skrypt o nazwie `test.waisindex`, który umożliwia automatyczne wygenerowanie czterech plików indeksowych. Są one używane do sprawdzenia, czy program `freeWAIS` działa poprawnie, demonstrują również różne możliwości indeksowania i wyszukiwania. Wspomniane pliki to:

- | | |
|-------------------|---|
| test-BOOL | indeks trzech przykładowych dokumentów, używający wyrażeń logicznych i synonimów; |
| test-COMP | indeks demonstrujący obsługę skompresowanych plików źródłowych; |
| test-Docs | indeks plików znajdujących się w katalogu <code>doc</code> , demonstrujący możliwość rekursywnego przeszukiwania katalogów; |
| test-Multi | indeks grafik w formacie GIF z możliwością obsługi wielu dokumentów. |

Ostatni z czterech plików indeksowych jest poprawnie obsługiwany tylko w wersji opartej na interfejsie X, ale pozostałe trzy powinny działać poprawnie z każdą wersją przeglądarki.

Po upewnieniu się, że system indeksowania działa poprawnie, i że wszystkie składniki pakietu `freeWAIS` zostały prawidłowo zainstalowane, należy utworzyć indeks dokumentów dostępnych w systemie. Można to zrobić na dwa sposoby, używając polecenia `waisindex` z opcją `-t` i jednym z poniższych słów kluczowych:

- | | |
|-----------------|--|
| one_line | indeksowanie każdego wiersza dokumentu, pozwalające na dostarczanie dokładniejszej informacji co do miejsca wystąpienia szukanego słowa; |
| text | indeksowanie całych dokumentów, przez co w przeglądarce pojawia się tylko nazwa dokumentu, w którym występuje szukany wyraz. |

Polecenie `waisindex` pozwala również na podanie nazwy pliku z indeksami, który należy wygenerować (opcja `-d`, po której następuje nazwa pliku) oraz katalogów lub plików, które mają zostać poindeksowane. Przykładowo, jeśli chcesz utworzyć indeks dla plików znajdujących się w katalogu `/usr/sales/sales-lit` i zachować go w pliku o nazwie `sales`, indeksując każdy wiersz dokumentu osobno, powinieneś wydać polecenie:

```
waisindex -d sales -t one_line /usr/sales/sales_lit
```

Ponieważ nie jest podana pełna ścieżka dostępu do generowanego pliku indeksu, zostanie on utworzony w katalogu bieżącym.

Po uruchomieniu serwera WAIS (patrz podrozdział „Uruchamianie serwera freeWAIS”), możesz przetestować pliki indeksów wydając polecenie `waissearch`. Aby na przykład znaleźć w pliku z indeksami tekst „WAIS”, wydaj polecenie:

```
waissearch -p 210 -d plik_indeksowy WAIS
```

Opcja `-p` pozwala na podanie numeru portu TCP (domyślnie 210), natomiast opcja `-d` służy do poinformowania programu `waissearch` o położeniu pliku indeksowego. Jeśli przeszukiwanie zakończy się pomyślnie (tzn. zostanie odnaleziony co najmniej jeden plik spełniający zadane kryteria), wyświetlona zostanie informacja o liczbie zwróconych rekordów i punktacji przyznanej każdemu ze znalezionych dokumentów. Jeśli nie otrzymałeś żadnych komunikatów lub otrzymałeś komunikat o błędzie, powinieneś sprawdzić, czy do plików konfiguracyjnych wprowadzone zostały prawidłowe informacje i czy plik indeksowy został wygenerowany poprawnie.

Ostatnim krokiem, wymaganym dla zapewnienia dostępu do pliku indeksowego użytkownikom Internetu, jest wydanie polecenia:

```
waisindex -export -register nazwy_plikow
```

gdzie `nazwy_plikow` to nazwy plików indeksowych. Polecenie to powoduje zarejestrowanie indeksów w katalogu serwerów przechowywanym w systemach `cndir.org` oraz `quake.think.com`. Połączenie z tymi serwerami następuje automatycznie po podaniu opcji `-register`. Powyższe polecenie należy wydać tylko wtedy, gdy wszyscy użytkownicy Internetu mają mieć dostęp do zasobów zgromadzonych w systemie (wkrótce przyjrzymy się poleceniu `waisindex` nieco bardziej szczegółowo).

Jeśli chcesz, by klienci mogli łączyć się z Twoim systemem `freeWAIS` za pomocą przeglądarki WWW takich jak Netscape czy Mosaic, musisz również wydać polecenie:

```
waisindex -d WWW -T HTML -contents -export /usr/resources/*.html
```

Zamiast `/usr/resources/` powinieneś podstawić ścieżkę dostępu do plików HTML zebranych w systemie. Powyższe polecenie pozwala klientom WAIS na wyszukiwanie słów kluczowych również w dokumentach HTML.

Jeśli chcesz, możesz skonfigurować system `freeWAIS` tak, by dostęp do niego był możliwy tylko z pewnych domen. Odpowiada za to wpis w pliku `ir.h`, o następującej postaci:

```
#define SERVSECURITYFILE "SERV_SEC"
```

Kopię istniejącego pliku `SERV_SEC` lub plik stworzony własnoręcznie musisz umieścić w tym samym katalogu, w którym znajdują się pliki indeksowe. Jeśli WAIS nie odnайдzie pliku `SERV_SEC`, pozwoli na dostęp z dowolnej domeny (nazwę tego pliku można zmienić, pod warunkiem, że będzie ona taka sama jak nazwa podana w cudzysłowie w pliku `ir.h`).

Każdy wpis w pliku `SERV_SEC` definiuje domenę, z której dostęp ma być możliwy. Wpisy te mają format:

```
domena [adres IP]
```

W każdym wierszu musi znajdować się nazwa domenowa komputera, któremu chcesz umożliwić dostęp; opcjonalnie można również podać jego adres IP. Jeśli adres IP i nazwa domenowa nie zgadzają się, nie ma to znaczenia, ponieważ WAIS przyznaje dostęp nawet wtedy, gdy zgadza się tylko nazwa lub tylko adres IP. Oto przykładowa zawartość pliku SERV_SEC:

```
chatton.com  
roy.sailing.org  
bighost.bignet.com
```

Każdy z wymienionych powyżej serwerów ma dostęp do systemu WAIS, natomiast żądania nadchodzące z dowolnych innych systemów nie zostaną obsłużone.

Właścicielem pliku SERV_SEC powinien być użytkownik, który uruchamia program freeWAIS (ze względu na bezpieczeństwo nie powinien być to użytkownik root), natomiast prawo jego modyfikacji powinien mieć tylko administrator.

Zmienna DATASECURITYFILE spełnia podobną rolę, jak zmienna SERVSECURITYFILE: pozwala zdefiniować nazwę pliku służącego do kontrolowania dostępu do baz danych. Wiersz definiujący tę zmienną w pliku ir.h ma postać

```
#define DATASECURITYFILE "DATA_SEC"
```

W pliku DATA_SEC znajduje się lista wszystkich baz danych wraz z informacją o tym, które systemy mogą mieć do nich dostęp. Również ten plik powinien zostać umieszczony w tym samym katalogu co pliki indeksowe. Wpisy w pliku DATA_SEC mają format:

```
baza_danych domena [adres_IP]
```

baza_danych to nazwa bazy danych, do której odnosi się dany wpis, pola domena i adres_IP mają takie samo znaczenie jak w pliku SERV_SEC. Poniżej przedstawiamy przykładową zawartość pliku DATA_SEC:

```
primary chatton.com  
primary bignet.org  
primary roy.sailing.org  
sailing roy.sailing.org
```

W powyższym przykładzie trzy systemy otrzymały prawo dostępu do bazy danych przechowywanej w pliku o nazwie primary, ale tylko system roy.sailing.org ma dostęp do bazy danych o nazwie sailing. Jeśli chcesz zezwolić wszystkim systemom (wyszczególnionym w pliku SERV_SEC) na korzystanie z którejś z baz danych, możesz zamiast nazwy domenowej i adresu IP wpisać gwiazdki. Przykładowo, by zezwolić wszystkim mającym dostęp do systemu WAIS na korzystanie z bazy danych primary, natomiast tylko systemowi roy.sailing.org na korzystanie z bazy o nazwie sailing, powinieneś do pliku DATA_SEC wprowadzić dane:

```
primary * *  
sailing roy.sailing.org
```

Należy zachować ostrożność przy wprowadzaniu informacji do plików SERV_SEC i DATA_SEC, ponieważ pomyłka może spowodować przyznanie dostępu innym niż zamierzone systemom. Przykładowo, jeśli jako adres IP podasz 155.12, wszystkie systemy o adresach zaczynających się od 155.12 (np. 15.121, 155.122 itp.) będą miały dostęp do baz danych WAIS. Aby tego uniknąć, powinieneś podawać adresy IP bezpośrednio.

Uruchamianie serwera freeWAIS

Podobnie jak miało to miejsce w przypadku serwera FTP, program obsługujący system **freeWAIS** może być uruchamiany z jednego ze skryptów **rc** przy starcie systemu, ale może również być wywoływany przez program **inetd**, gdy ktoś będzie próbował skorzystać z usługi WAIS. Jeśli chcesz, by system **freeWAIS** był uruchamiany z pliku **rc**, powinieneś wpisać do niego następujące polecenie:

```
waisserwer -u id_uzytkownika -p 210 -l 10 -d /usr/wais/wais_index
```

Opcja **-u** pozwala określić, który z użytkowników ma być właścicielem procesu **waisserwer** (**id_uzytkownika** musi być prawidłowym identyfikatorem użytkownika, z odpowiednim wpisem w pliku **/etc/passwd**). Za pomocą opcji **-p** można podać numer portu, którego należy używać (zwykle 210), natomiast opcja **-d** służy do informowania programu **waisserwer** o położeniu plików indeksowych. Jeśli chcesz rejestrować informacje o przebiegu sesji do pliku, użyj opcji **-e** z nazwą pliku, do którego będą zapisywane informacje.

Program **waisserwer** nie powinien być uruchamiany przez użytkownika **root** – uniemożliwi to wykorzystanie niedociągnięć tego programu przez hakerów. Jeśli właścicielem procesu jest zwykły użytkownik (na przykład **wais**), w najgorszym przypadku zagrożone będą tylko pliki, których jest on właścicielem.

Port 210 jest standardowo używany w Internecie do komunikacji z systemem WAIS. Inny numer portu można podać wtedy, gdy system konfigurowany jest wyłącznie z myślą o lokalnym dostępie do danych. Jeśli numer używanego portu jest mniejszy od 1023, usługa WAIS musi być uruchamiana i zarządzana przez użytkownika **root**. W przeciwnym przypadku może być ona obsłużona również przez zwykłego użytkownika. Jeśli chcesz używać portu 210, nie musisz podawać w wierszu poleceń jego numeru, ale nadal należy podać opcję **-p**.

Aby uruchamaniem programu **waisserwer** sterował proces **inetd**, w pliku **/etc/services** musi znajdować się odpowiedni wpis, na przykład:

```
z3955      210/tcp      #WAIS
```

210 to numer portu, który jest używany przez system WAIS, natomiast **tcp** to nazwa wykorzystywanego protokołu. Po zmodyfikowaniu pliku **/etc/services** należy również poinformować program **inetd** o tym, że ma uruchamiać program **waisserwer** po wykryciu żądania przesyłanego do portu 210 (lub innego używanego w Twoim systemie); w tym celu w pliku **inetd.conf** musi znaleźć się wiersz (odpowiednie opcje są takie same, jak w przypadku wywoływania programu z wiersza poleceń):

```
z3955      stream tcp      nowait root/usr/local/bin/waisserwer/waisserwer.d -  
u id_uzytkownika -d /usr/wais/wais_index
```

Zamiast programu **waisserwer** należy uruchomić program **waisserwer.d**, który jest przeznaczony do współpracy z programem **inetd**. Również w tym przypadku można użyć opcji **-e** do załączenia rejestrowania informacji o przebiegu sesji do pliku.

Tworzenie własnych indeksów dla programu WAIS

Jeśli skonfigurowałeś serwer `freeWAIS` i wszystko wydaje się działać prawidłowo, przyszła pora na dostarczenie systemowi WAIS jakichś użytecznych danych. Zwykle ich źródłem są dokumenty tekstowe, ale indeksować można pliki dowolnego typu. Najważniejszym krokiem jest utworzenie indeksu za pomocą programu `waisindex`. Program ten czasem zachowuje się niezgodnie z oczekiwaniami, ale odrobiną praktyki i kilka prób pozwolą Ci go opanować.

Program `waisindex` przegląda dane zawarte we wszystkich plikach, które mają zostać poindeksowane. Na ich podstawie generowanych jest zwykle siedem różnych plików indeksowych (zależnie od zawartości plików i wydanego polecenia). Każdy z tych plików zawiera listę słów występujących w dokumentach. Pliki te są następnie łączone w jedną bazę danych, często nazywaną źródłem (lub źródłem WAIS, ang. *WAIS source*). Kiedy program klienta WAIS zamawia wyszukiwanie, podany przez użytkownika tekst jest porównywany ze źródłem i wyświetlane są wyniki porównania wraz z analizą dokładności trafień (podawaną w postaci punktacji).



Użycie plików indeksowych pozwala na znaczące skrócenie czasu wyszukiwania, ponieważ słowa kluczowe zostały wyszukane wcześniej. Ilość danych zawartych w plikach indeksowych może być jednak dość duża, dlatego powinieneś zapewnić serwerowi WAIS odpowiednio dużo miejsca na dysku (zwykle wystarcza dwa razy tyle, ile zajmuje samo źródło).

Pliki indeksowe programu WAIS

Pliki indeksowe programu WAIS nie nadają się (może poza dwoma wyjątkami) do bezpośredniego przeglądania przez użytkownika. Zwykle program `waisindex` generuje siedem takich plików, choć ich liczba może być inna w zależności od wymagań. Każdy z plików indeksowych ma nazwę składającą się z dwóch członów: pierwszy z nich jest wspólny (można go podać w wierszu poleceń przy wywoływaniu programu `waisindex`; jeśli nie zostanie podany, domyślnie przyjmowana jest nazwa `index`), drugim jest rozszerzenie nazwy pliku, wskazujące na jego zastosowanie. Poniżej przedstawiamy domyślne nazwy plików indeksowych i ich zastosowanie.

- | | |
|------------------|--|
| index.doc | Dokument zawierający tabelę, na którą składają się: nazwa pliku, nagłówek (tytuł) pliku, pozycje pierwszych i ostatnich znaków wpisu, długość dokumentu, liczba wierszy dokumentu, oraz data i czas jego utworzenia. |
|------------------|--|

index.dct	Plik zawierający słownik, czyli listę wszystkich słów (bez powtórzeń) w indeksowanych plikach, wraz z odniesieniami do odpowiednich danych w pliku <code>index.inv</code> .
index.fn	Plik, który zawiera tabelę z nazwami plików, datami utworzenia indeksów, oraz danymi o typach plików.
index.hl	Plik zawierający tabelę nagłówków (tytułów) indeksowanych dokumentów. Nagłówki te są wyświetlane wraz z wynikami wyszukiwania.
index.inv	Plik zawierający tabelę wiążącą każde słowo we wszystkich dokumentach z wskaźnikami do tych dokumentów i wagą danego słowa (określoną przez jego odległość od początku pliku, liczbę wystąpień i procentowączęstość występowania).
index.src	Plik opisu źródła, który zawiera informacje o indeksowanych dokumentach, takie jak nazwa serwera, jego numer IP, numer portu używanego przez WAIS, nazwa pliku źródła, informacja o ewentualnym koszcie usługi, nagłówek, opis pliku źródła oraz adres e-mailowy administratora. Plik ten można edytować za pomocą dowolnego edytora tekstów. Wkrótce omówimy go nieco bardziej szczegółowo.
index.status	Plik statusu, zawierający informacje definiowane przez użytkownika.

Plik opisu źródła (który jest zwykłym plikiem ASCII) jest co pewien czas odczytywany przez program `waisindex`, który sprawdza w ten sposób, czy źródło zostało zmodyfikowane. Jeśli zmiany są znaczące, `waisindex` aktualizuje przechowywane wewnętrznie informacje. Oto przykładowy plik opisu źródła:

```
(:source
:version 2
:ip-address "147.120.0.10"
:ip-name "wizard.tpc.com"
:tcp-port 210
:database-name "Linux stuff"
:cost 0.00
:cost-unit free
:maintainer "wais_help@tpci.com"
:subjects "Everything you need to know about Linux"
:description "If you need to know something about Linux, it's here."
```

Zawartość tego pliku należy zmodyfikować podczas konfigurowania systemu `freeWAIIS`, ponieważ dane domyślne nie są użyteczne.

Polecenie `waisindex`

Polecenie `waisindex` umożliwia podanie wielu różnych opcji – kilka z nich omówiliśmy we wcześniejszej części tego rozdziału. Oto lista najważniejszych opcji tego programu:

- a** umożliwia dołączenie nowych danych do istniejącego pliku indeksowego (opcja używana do aktualniania indeksów zamiast ich ponownego tworzenia za każdym razem, gdy udostępniany jest nowy dokument);
- contents** powoduje indeksowanie zawartości pliku (zachowanie domyślne);
- d** pozwala określić pierwszy człon nazwy plików indeksowych (na przykład podanie parametrów `-d /usr/wais/nic` spowoduje, że wszystkie pliki indeksowe znajdą się w katalogu `/usr/wais`, a ich nazwy będą się zaczynać od słowa `nic`);
- e** pozwala podać nazwę pliku, w którym rejestrowane będą informacje o błędach (domyślnie jest to `stderr` – czyli standardowe urządzenie rejestrujące komunikaty o błędach, zwykle konsola; można również podać opcję `-s`, powodującą wysyłanie komunikatów do urządzenia `/dev/null`);
- export** powoduje dołączenie do pliku opisu nazwy serwera i numeru portu TCP, co ułatwia dostęp poprzez Internet;
- l** pozwala określić, jakie informacje mają być rejestrowane; dostępne wartości to:
 - 0** nie są rejestrowane żadne informacje,
 - 1** rejestrowanie najważniejszych ostrzeżeń i informacji o błędach,
 - 5** rejestrowanie mniej istotnych błędów i ostrzeżeń oraz nazw plików indeksowych,
 - 10** rejestrowanie wszystkich informacji;
- M** umożliwia dołączanie plików różnych typów;
- mem** ogranicza zużycie pamięci podczas indeksowania (im wyższa liczba zostanie podana, tym szybszy będzie proces indeksowania, ale potrzebował będzie również więcej pamięci);
- nocontents** zapobiega indeksowaniu zawartości dokumentu (indeksowana jest wtedy tylko jego nazwa i nagłówek);
- nopairs** powoduje, że sąsiednie wyrazy napisane wielkimi literami nie są indeksowane wspólnie;
- npos** powoduje ignorowanie pozycji słowa kluczowego w dokumencie przy obliczaniu punktacji;
- pairs** powoduje, że sąsiednie wyrazy napisane wielkimi literami będą indeksowane jako jedna całość;

-pos	powoduje obliczanie punktacji na podstawie położenia słów kluczowych (jeśli słowa położone są blisko, punktacja jest większa);
-r	rekursywne indeksowanie dokumentów w podkatalogach;
-register	rejestracja indeksu WAIS w internetowej bazie WAIS Directory of Services;
-stdin	powoduje użycie nazwy pliku wprowadzonej z klawiatury, zamiast z wiersza poleceń;
-stop	pozwala na podanie nazwy pliku zawierającego słowa zbyt popularne, by je indeksować, zwykle zdefiniowanej w pliku <code>/src/ir/stoplist.c</code> ;
-t	znacznik typu danych w plikach;
-T	ustawienie żądanego typu danych.

Jeśli program `waisindex` nie otrzyma informacji o typie danych zapisanych w plikach, może nie być w stanie wygenerować poprawnego indeksu. Listę obsługiwanych przez `freeWAIS` typów plików możesz zobaczyć wpisując polecenie `waisindex` bez argumentów:

```
waisindex
```

Zwykle używa się tylko kilku spośród tych typów. Poniżej przedstawiamy listę najpopularniejszych z nich.

filename	To samo, co zwykły plik tekstowy (<code>text</code>), ale jako nagłówek używana jest nazwa pliku.
first_line	To samo, co zwykły plik tekstowy (<code>text</code>), ale jako nagłówek używany jest pierwszy wiersz pliku.
ftp	Pliki zawierające kody FTP, które mogą zostać wykorzystane przez użytkowników do ładowania informacji z innych komputerów.
GIF	Grafika w formacie GIF (jeden obrazek odpowiada jednemu plikowi). Jako nagłówek używana jest nazwa pliku.
mail lub rmail	Indeksowanie zawartości skrzynki pocztowej <code> mbox</code> – jedna wiadomość odpowiada jednemu dokumentowi.
mail_digest	Standardowa poczta elektroniczna – indeksowanie poszczególnych wiadomości. Jako nagłówek używane jest pole tematu (ang. <code>subject</code>).
netnews	Artykuły pochodzące z grup dyskusyjnych, indeksowane pojedynczo. Jako nagłówek używane jest pole tematu.

one_line	Indywidualne indeksowanie każdego zdania w dokumencie.
PICT	Grafika w formacie PICT (jeden obrazek odpowiada jednemu plikowi). Jako nagłówek używana jest nazwa pliku.
ps	Plik postscriptowy (jeden dokument odpowiada jednemu plikowi).
text	Indeksowanie pliku jako jednego dokumentu. Jako nagłówek używana jest ścieżka dostępu do pliku.
TIFF	Grafika w formacie TIFF (jeden obrazek odpowiada jednemu plikowi). Jako nagłówek używana jest nazwa pliku.

Aby poinformować program `waisindex` o tym, jakiego typu pliki ma indeksować, należy podać opcję `-t`, po której następuje nazwa odpowiedniego typu plików. Przykładowo, jeśli indeksujesz standardowe pliki ASCII, możesz wydać polecenie:

```
waisindex -t text -r /usr/waisdata/*
```

Polecenie to spowoduje utworzenie indeksu dla wszystkich plików w katalogu `/usr/waisdata` i jego podkatalogach (opcja `-r`), przy założeniu, że są one plikami ASCII.



Po zakończeniu procesu indeksowania wszelkie zmiany w dokumencie nie są uwzględniane w indeksie aż do ponownego przeprowadzenia pełnego indeksowania. Użycie opcji `-a` nie aktualnia istniejących w indeksie danych. Zamiast tego powinieneś więc przeprowadzić proces indeksowania od nowa - warto robić to regularnie.

Inne możliwości systemu WAIS

Usługa `freeWAIS` udostępniana w systemie może zostać wzbogacona na wiele sposobów. W tym podrozdziale – który na pewno nie wyczerpuje tematu – przedstawimy dwa proste w implementacji sposoby uatrakcyjnienia węzła WAIS.

Na początek założmy, że udostępniasz animacje, grafiki czy pliki dźwiękowe na jakiś określony temat – na przykład materiały o instrumentach muzycznych – i posiadasz kilka dokumentów o skrzypcach. Możesz wówczas chcieć udostępnić próbkę dźwięku skrzypiec, animację na ich temat czy zdjęcie skrzypiec Stradivariusza. W tym celu wszystkie pliki zawierające te dane powinny mieć nazwy o wspólnym rdzeniu, różniące się tylko rozszerzeniem. Przykładowo, jeśli główny dokument o skrzypcach nazywa się `skrzypce.txt`, w katalogach WAIS mogą znajdować się następujące pliki:

- skrzypce.TEXT** dokument opisujący skrzypce,
- skrzypce.TIFF** zdjęcie skrzypiec Stradivariusza,
- skrzypce.MPEG** animacja przedstawiająca proces budowania skrzypiec,

skrzypce.MIDI plik MIDI zawierający jakiś słynny utwór na skrzypce.

Wszystkie powyższe pliki powinny mieć taki sam rdzeń nazwy (**skrzypce**), ale różnić się typem (rozpoznawanym przez program **waisindex**). Następnie należy powiązać pliki multimedialne z dokumentem tekstowym. Można to zrobić wydając polecenie:

```
waisindex -d skrzypce -M TEXT,TIFF,MPEG,MIDI -export  
Σ /usr/waisdata/skrzypce/*
```

Spowoduje ono, że program **waisindex** obsłuży wszystkie cztery typy plików. Użytkownik szukający słowa kluczowego „**skrzypce**” otrzyma informację o wszystkich czterech plikach, a przeglądarka pozwoli mu obejrzeć czy odsłuchać odpowiednie pliki.

Innym często spotykanym rozwiązaniem jest umożliwienie odnalezienia danego dokumentu po podaniu kilku różnych słów kluczowych – mogą to być synonimy, na przykład znalezienie dokumentu o oferowanych przez firmę przewodach połączeniowych powinno być możliwe zarówno po podaniu słowa „**kable**” jak i „**przewody**”. Do tego celu przeznaczony jest plik o nazwie **SOURCE.syn**, odczytywany automatycznie podczas pracy programu wyszukującego dane. Wiersze tego pliku mają format:

```
słowo synonim [synonim ...]
```

słowo to ciąg znaków, który będzie użyty do wyszukania informacji w bazie danych, natomiast **synonim** to inny ciąg znaków, po podaniu którego użytkownik powinien otrzymać takie same rezultaty. Przykładowo, jeśli w Twoim węźle WAIS udostępniasz informacje o zwierzętach domowych, zawartość pliku **SOURCE.syn** może być następująca:

```
pies wilczur owczarek chart buldog terier  
ptak papuga papużka nimfa kanarek  
ryba skalar gupik sum pielęgnica głonojad
```

Plik zawierający synonimy przydaje się, gdy użytkownicy stosują różne określenia mając na myśli ten sam temat. Możesz łatwo sprawdzić, czy jest on potrzebny w Twoim systemie – wystarczy, że na jakiś czas ustawisz poziom rejestracji zdarzeń programu **waisindex** na 10, a następnie przejrzyś wygenerowany plik – od razu zorientujesz się, jakie wyrażenia są stosowane przez użytkowników korzystających z Twojego węzła, dzięki czemu będziesz mógł dostosować odpowiednio dane zapisane w pliku **SOURCE.syn**. Rejestracja zdarzeń nie powinna być załączona przez zbyt długi czas, ponieważ pliki zawierające rejestrowane dane szybko stają się bardzo duże.

Podsumowanie

Po uruchomieniu i skonfigurowaniu serwera WAIS, pora na utworzenie własnych indeksów i udostępnienie ich innym użytkownikom. WAIS jest systemem łatwym w konfiguracji, a jednocześnie umożliwia efektywne udostępnianie dokumentów zgromadzonych w systemie. Alternatywą dla systemu WAIS, często używaną w systemach tekstowych, jest usługa Gopher, która zostanie omówiona w następnym rozdziale.

O tym, jak w systemie linuxowym skonfigurować serwer WWW, możesz przeczytać w rozdziale 51. „Konfiguracja węzła internetowego”.

W rozdziale 53. „Podstawy języka HTML” omówimy tworzenie stron WWW.

Jak używać języka Java, aby uatrakcyjnić swoją stronę główną, dowiesz się w rozdziale 54. „Java i JavaScript”.

Rozdział 50.

Konfigurowanie

usługi Gopher

Tim Parker

W tym rozdziale:

- υ Gopher i Linux
- υ Konfigurowanie usługi Gopher
- υ Katalogi systemu Gopher
- υ Uruchamianie usługi Gopher
- υ Daj światu znać o sobie

Gopher to jedna z bardziej przydatnych usług internetowych, używana zarówno przez początkujących, jak i doświadczonych użytkowników. Jest to oparty na interfejsie tekstowym system wyszukiwania plików, który umożliwia użytkownikowi odszukanie interesujących danych za pomocą hierarchicznego systemu menu. Konfiguracja węzła Gopher sprowadza się do uruchomienia oprogramowania obsługi serwera i utworzenia pewnej liczby katalogów zawierających pliki indeksowane przez system Gopher.

Działanie systemu Gopher polega na tym, że program klienta (uruchamiany przez użytkownika) łączy się z serwerem usługi Gopher i otrzymuje informacje o plikach dostępnych w Internecie (albo w sieci lokalnej, jeśli działanie serwera jest ograniczone tylko do takiego obszaru). Pod koniec roku 1995 w Internecie działało około 6000 serwerów usługi Gopher, dostępnych dla każdego, kto posiadał odpowiedni program klienta. Choć liczba ta nieco się zmniejszyła (głównie z powodu rozwoju WWW), w sieci wciąż działa mnóstwo węzłów udostępniających usługę Gopher. Serwery te zawierają informacje o ponad 10 milionach elementów, wśród których są najróżniejsze pliki, od dokumentów tekstowych do filmów, dźwięków czy grafik i różnego rodzaju plików wykonywalnych. Gopher pozwala użytkownikowi na przeglądanie i manipulo-

wanie listami plików, dzięki czemu możliwe jest łatwe odnalezienie interesujących materiałów.

Jeśli chcesz łączyć się z serwerem usługi Gopher (osobiście, lub też chce to robić któryś z użytkowników Twojego systemu), będziesz potrzebował oprogramowania klienta Gopher. Niektóre programy tego typu rozprowadzane są wraz z dystrybucjami Linuxa, inne dostępne są w węzłach FTP, BBS i z innych źródeł. Jeśli nie chcesz pozwolić użytkownikom swojego systemu na uruchamianie klienta Gopher, możesz za pomocą programu Telnet połączyć się z serwerem umożliwiającym anonimowemu użytkownikowi dostęp do klienta Gopher. Większość pakietów umożliwiających dostęp do usługi Gopher pozwala również korzystać z systemu WAIS, FTP, a także w pewnym stopniu współpracuje z przeglądarkami WWW.

W tym rozdziale omówimy konfigurowanie serwera usługi Gopher, pozwalającego użytkownikom na dostęp do danych o plikach zgromadzonych w Twoim systemie. Nie będziemy omawiać problemów związanych z utworzeniem odpowiedniej (ułatwiającej poszukiwania) struktury udostępnianych danych, skoncentrujemy się tylko na konfiguracji oprogramowania.

Gopher i Linux

Obecnie dostępne są dwie wersje systemu Gopher przeznaczone do pracy w systemie Linux: Gopher i Gopher+ (Gopher Plus). Gopher jest systemem dostępnym za darmo, natomiast Gopher+ to produkt komercyjny. Różnią się one funkcjonalnością – jeśli zależy Ci na możliwościach, jakie daje Gopher+, możesz rozważyć zakup tego programu. Gopher+ przewyższa system Gopher pod kilkoma względami:

- υ udostępnia rozszerzone informacje o pliku,
- υ umożliwia odczytanie opisu pliku,
- υ pozwala na jednocześnie ładowanie kilku wersji pliku (na przykład dokumentacji w formacie ASCII i postscriptowym),
- υ umożliwia ładowanie plików w oparciu o kryteria wyszukiwania definiowane przez użytkownika.

Gopher+ jest kompatybilny z systemem Gopher, ale Gopher nie potrafi korzystać z możliwości udostępnianych przez system Gopher+. Oba systemy współpracują z przeglądarkami WWW. Koszt zainstalowania systemu Gopher+ wynosi od 100 do 500 dolarów, zależnie od przeznaczenia węzła.

Wersje systemu Gopher przeznaczone dla Linuxa pochodzą z dwóch źródeł. Uniwersytet w Minnesocie opracował systemy Gopher i Gopher+, a system Gopher dostępny jest również jako GN Public License Gopher. Najnowsza dostępna za darmo wersja systemu Gopher opracowana w UM ma numer 1.3 (wersja 2.13 jest darmowa tylko dla instytucji edukacyjnych), ale uniwersytet ten porzucił prace nad doskonaleniem wersji darmowej,

koncentrując się na komercyjnym produkcie Gopher+. GN Public License Gopher zawiera również usługę WWW; niestety obecnie nie jest jeszcze w pełni funkcjonalny.

Gopher bazuje na jednym z protokołów rodziny TCP/IP, nazywanym (co za niespodzianka) protokołem Gopher. Jest to dość prosty protokół oparty na seriach zapytań i odpowiedzi, zaimplementowany ze zwróceniem uwagi na prędkość działania. Gopher przesyła informacje o plikach (nazywaną plikiem menu systemu Gopher) w następującym formacie:

```
<typ> <nazwa_widoku> <selektor> <serwer> <port>
```

Poszczególne pola mają następujące znaczenie:

typ	jednoznakowy identyfikator typu pliku (za chwilę przedstawiemy listę wszystkich wartości, które mogą zostać użyte w tym polu);
nazwa_widoku	nazwa menu lub widoku, po której następuje znak tabulacji;
selektor	unikalny (w obrębie serwera) identyfikator dokumentu, zwykle oparty na nazwie pliku, po którym następuje znak tabulacji;
serwer	nazwa serwera, w którym przechowywany jest dokument, po której również musi wystąpić znak tabulacji;
port	numer portu, przez który należy łączyć się z serwerem. Po numerze portu (zwykle 70) przesyłane są znaki powrotu karetki i nowego wiersza.

Wersja Gopher+ przesyła jeszcze kilka dodatkowych danych, takich jak identyfikator administratora odpowiedzialnego za obsługę systemu Gopher, krótki opis typu dokumentu (na przykład `text`), język, w jakim jest napisany dokument, datę ostatniej modyfikacji i rozmiar w bajtach.

Kiedy użytkownik zdecyduje się na pobranie któregoś z plików, do nawiązania połączenia wykorzystywane są informacje o nazwie serwera i numerze portu, natomiast selektor pozwala na zidentyfikowanie pliku, który ma zostać przesłany.

Gopher obsługuje kilkanaście typów plików, oznaczanych jednoznakowymi kodami. Oto ich lista:

- 0** plik tekstowy;
- 1** katalog;
- 2** serwer książki telefonicznej CSO (nazwa serwera określa komputer, z którym należy się połączyć, natomiast selektor w tym przypadku pozostawiany jest pusty);
- 3** błąd;

- 4 plik typu BinHex systemu Macintosh;
- 5 archiwum binarne systemu DOS;
- 6 archiwum UNIX-owe, zakodowane programem uuencode;
- 7 serwer przeszukiwania indeksu;
- 8 wskaźnik do tekstowej sesji programu Telnet (nazwa serwera określa komputer, z którym należy się połączyć, a selektor zawiera identyfikator użytkownika);
- 9 plik binarny;
- g grafika w formacie GIF;
- h dokument HTML;
- l plik grafiki;
- i tekst wyświetlany w menu, pełniący funkcję separatora;
- M dokument poczty w standardzie MIME;
- P plik dokumentacji w formacie PDF firmy Adobe;
- s dźwięk;
- T wskaźnik do sesji programu Telnet 3270 (nazwa serwera określa komputer, z którym należy się połączyć, a selektor zawiera identyfikator użytkownika);

System Gopher używa kilku innych programów, niezbędnych do prawidłowego działania; są to:

- v tn3270 lub podobny emulator 3270 – używany przy połączeniach Telnet 3270;
- v program komunikacyjny kermit lub zmodem, używany do ładowania plików; odpowiednie pliki wykonywalne zwykle nazywają się kermit, sz, sb i sx;
- v program graficzny – jeśli zezwolisz na wyświetlanie grafiki, niezbędny będzie program do jej obsługi, taki jak na przykład xv.

Powyższe wymagania można dostosować do własnych potrzeb, jeśli system działa tylko w sieci lokalnej, ale jeśli usługa ma być ogólnodostępna, wymienione programy powinny znaleźć się w Twoim systemie.

Konfigurowanie usługi Gopher

Instalowanie i konfiguracja systemu Gopher (lub Gopher+) wymaga zwykle ustawienia pewnych opcji konfiguracyjnych jeszcze przed przystąpieniem do komplikowania oprogramowania (które zwykle jest rozprowadzane w postaci kodu źródłowego) i modyfikacji kilku standardowych plików. Konfiguracja systemu Gopher+ przebiega prawie dokładnie tak samo – trzeba tylko podać kilka dodatkowych informacji. Ponieważ w systemach linuxowych częściej używa się systemu Gopher, a nie Gopher+, właśnie na nim się skoncentrujemy.

W tym podrоздiale nie będziemy używać pełnych ścieżek dostępu do plików, ponieważ punkt zainstalowania oprogramowania obsługującego system Gopher nie ma żadnego znaczenia, o ile ścieżki przeszukiwania są odpowiednio skonfigurowane. Poza tym nie wykłarał się żaden standard co do rozmieszczenia katalogów, więc Ty również masz wolną rękę przy ich tworzeniu.

Plik gopherd.conf

Parametry konfiguracyjne systemu Gopher (i Gopher+) zapisane są w pliku `gopherd.conf`, odczytywanym przez program rezydentny `gopherd`. Wartości domyślne tych parametrów zwykle wymagają niewielkich modyfikacji, ale większość z nich polega tylko na usunięciu lub dodaniu symbolu komentarza.

Pierwszym krokiem konfiguracji jest utworzenie aliasu dla usługi Gopher w Twoim systemie. W pliku `gopherd.conf` powinien znajdować się wpis o następującej postaci:

```
hostalias: tpci
```

Alias ten jest potrzebny, aby serwer Gopher mógł zostać odnaleziony w danym systemie. Nie powinien on wskazywać na komputer bezpośrednio, dzięki czemu łatwiej będzie wprowadzać zmiany w konfiguracji. Najlepszym rozwiązaniem jest utworzenie aliasu i powiązanie go z fizycznym komputerem za pomocą DNS. Jeśli Twój komputer nie działa w sieci, musisz użyć albo aliasu powiązanego z nazwą Twojego komputera, albo bezpośrednio nazwy komputera.

Możliwe jest również określenie maksymalnej dopuszczalnej liczby jednoczesnych połączeń Gopher. Takie rozwiązanie okazuje się czasem niezbędne, aby uniknąć nadmiernego obciążania systemu. Liczba jednoczesnych połączeń zwykle zdefiniowana jest w pliku w katalogu określonym przez wartość zmiennej `PIDS_Directory`. Odpowiedni wiersz w pliku `gopherd.conf` jest zwykle zaznaczony jako komentarz, ponieważ wcześniejsze wersje systemu nie obsługiwały poprawnie takiego ograniczenia. Jeśli chcesz skorzystać z tej możliwości, powinieneś upewnić się, że katalog wskazywany przez zmienną `PIDS_Directory` istnieje i zawiera odpowiednie dla Twojej wersji systemu Gopher pliki, a następnie usunąć symbol komentarza z wiersza

```
#PIDS_Directory: /pids
```

Lepszym sposobem kontrolowania obciążenia systemu jest użycie słowa kluczowego `MaxConnections`, które pozwala określić liczbę obsługiwanych równocześnie klientów.

Aby dobrać odpowiednią wartość, zapewniającą równowagę pomiędzy obciążeniem systemu i wygodą użytkowników, musisz nieco eksperymentować – dobrym punktem wyjścia dla systemów opartych o procesory 80486 czy Pentium jest ok. 15 – 25 użytkowników. Jeśli jednak równocześnie w systemie działa serwer WWW, liczba ta powinna być nieco mniejsza, co pozwoli ograniczyć zużycie zasobów systemu. Odpowiedni wpis w pliku konfiguracyjnym ma postać:

```
MaxConnections: 15
```

Jeśli liczba użytkowników zostanie przekroczona, przy próbie połączenia generowany będzie komunikat o błędzie. Można również skorzystać z możliwości oferowanych przez programy dekodujące. Używane są one wtedy, gdy użytkownik zleca załadowanie pliku, dodając do jego nazwy rozszerzenie (takie jak `.z`, `.zip` czy `.gz`), które powoduje przesyłanie danych w postaci skompresowanej. Dekoder rozpoznaje przekazane przez użytkownika rozszerzenie i wywołuje odpowiedni program użytkowy, dzięki czemu dane są przesyłane w żądanym formacie. W większości plików `gopherd.conf` znajdują się następujące wpisy:

```
decoder: .Z /usr/ucb/zcat
decoder: .gz /usr/gnu/bin/zcat
#decoder: .adpcm /usr/openwin/bin/adpcm_dec
#decoder: .z /usr/gnu/bin/zcat
```

Wpisy dotyczące dwóch ostatnich dekoderów są zaznaczone jako komentarz, ale można usunąć symbol `#`, jeśli chcesz umożliwić korzystanie z plików w formatach `.adpcm` i `.z` poprzez usługę Gopher. Możesz również dodać obsługę innych formatów plików, wpisując dane o odpowiednich dekoderach (nazwę wraz z pełną ścieżką dostępu).

Należy również ustawić czas, przez jaki pliki mają być buforowane. Odpowiedni wpis znajduje się w wierszu rozpoczynającym się od słowa kluczowego `Cachetime`; rozsądna wartością jest ok. 180 sekund. W pliku `gopherd.conf` powinien więc znajdować się wpis:

```
Cachetime: 180
```

Podając odpowiednie dane w pliku `gopherd.conf` można również ograniczyć dostęp do niektórych plików w systemie, używając słowa kluczowego `ignore`. Zwykle znajdują się tam wpisy takie jak:

```
ignore: lib
ignore: bin
ignore: etc
ignore: dev
```

Wszystkie pliki o rozszerzeniach `lib`, `bin`, `etc` i `dev` będą przez system Gopher ignorowane. Jeśli istnieje potrzeba ograniczenia dostępu do innego typu plików, na przykład jeśli w księgowości używa się plików o rozszerzeniu `.acct`, które nie powinny być one widoczne dla klientów Gopher, powinieneś dodać wpis:

```
ignore: acct
```

Słowo kluczowe `ignore` pozwala ukryć pliki tylko na podstawie ich rozszerzenia – takie rozwiązanie bywa czasem niewystarczające. Na szczęście po słowie kluczowym `ignore_patt` (ang. *ignore pattern*, ignoruj według wzorca) możliwe jest użycie symboli wieloznacznych. Przykładowo, aby uniemożliwić klientom Gopher dostęp do plików o nazwach rozpoczynających się od liter `usr`, powinieneś do pliku `gopherd.conf` dodać wpis:

```
ignore_patt: ^usr$
```

Plik gopherdlocal.conf

W pliku `gopherdlocal.conf` należy dokonać co najmniej dwóch modyfikacji, dzięki którym system przestanie generować komunikaty o błędach, które po pewnym czasie zaczynają być denerwujące. W pliku tym domyślnie znajdują się między innymi dwa wpisy:

```
Admin: blank  
AdminEmail: blank
```

Jeśli ich nie zmodyfikujesz, dostosowując do swojego systemu, Gopher będzie zgłaszał komunikaty o różnych (czasem dość dziwnych) błędach. Po słowie kluczowym `Admin` podaje się zwykle imię i nazwisko administratora, czasem również numer telefonu. Po słowie `AdminEmail` – adres e-mailowy administratora. Oto przykładowe wartości:

```
Admin: Yvonne Chow, 555-1212  
AdminEmail: ychow@chatton.com
```

W pliku `gopherdlocal.conf` należy również uzupełnić dane następujące po słowie kluczowym `Abstract` – jest to krótki opis tego, co można znaleźć w Twoim systemie. Jeśli nie zmienisz wartości domyślnej, użytkownicy będą otrzymywać informację, że powinni zażądać uzupełnienia opisu, powinieneś więc zrobić to jak najszybciej. Jeśli opis ma być dłuższy niż jeden wiersz tekstu, na końcu każdego wiersza powinien znaleźć się znak \, który pozwala na kontynuację tekstu w nowym wierszu. Oto przykładowy opis zawartości serwera:

```
Abstract: This server provides sound and graphics files \  
collected by the administrator on a recent trip to Outer \  
Mongolia.
```

Informacje dostarczane użytkownikom o Twoim serwerze zawierają również takie dane, jak nazwa węzła, nazwa organizacji zajmującej się jego prowadzeniem, lokalizacja geograficzna komputera, szerokość i długość geograficzna oraz strefa czasowa. Nie trzeba podawać tych informacji, ale dzięki nim użytkownicy mogą zorientować się, z kim mają do czynienia. Oto przykładowe dane:

```
Site: Explore_Mongolia  
Org: Mongolia_Tourist Bureau  
Loc: North Bay, Ontario, Canada  
Geog: blank  
TZ: EDT
```

Tekst `blank` w polu `Geog` oznacza, że wartość nie została podana – mało który administrator zna dokładne współrzędne geograficzne miejsca, w którym znajduje się komputer.

Można również podać, w jakim języku dostępna jest większość dokumentów w systemie – ułatwi to użytkownikom korzystanie z usługi Gopher. Jeśli jest to na przykład amerykańska odmiana języka angielskiego, w pliku `gopherdlocal.conf` powinien znaleźć się wiersz:

```
Language: En_US
```

Po słowie kluczowym `BummerMsg` należy wpisać treść informacji, która będzie podawana użytkownikowi próbującemu połączyć się w sytuacji, gdy obsługiwana jest maksymalna liczba użytkowników, lub gdy połączenie powoduje błąd. Oto przykładowy wpis:

```
BummerMsg: Sorry, we have exceeded the number of permissible users.
```

Można oczywiście wpisać tu dowolny tekst, pamiętaj jednak o tym, że nigdy nie wiadomo, kto będzie go czytał.

Ostatnim krokiem konfigurowania pliku `gopherdlocal.conf` jest podanie informacji o ograniczeniach praw dostępu do serwera. Służy do tego słowo kluczowe `access`, po którym należy podać dane w następującym formacie:

```
access: nazwa_komputera prawa_dostępu liczba_użytkowników
```

`nazwa_komputera` to nazwa domenowa lub numer IP komputera, z którego następuje połączenie, pole `prawa_dostępu` pozwala określić prawa dostępu obowiązujące dla użytkowników łączących się z tego systemu, a `liczba_użytkowników` jest maksymalną liczbą użytkowników, którzy mogą jednocześnie korzystać z usługi Gopher.

Pole określające prawa dostępu może zawierać dowolną kombinację czterech podanych niżej wyrazów; każdy z nich może być poprzedzony wykryznikiem, co oznacza brak określonego prawa.

browse Pozwala na przeglądanie zawartości katalogów. Jeśli użytkownik nie posiada tego prawa, ma dostęp do poszczególnych plików, ale nie może uzyskać informacji o zawartości katalogu.

ftp Pozwala, by serwer zachowywał się jako bramka dla usługi FTP.

read Pozwala na dostęp do plików. Jeśli użytkownik nie posiada tego prawa, przy próbie dostępu otrzyma informację określoną po słowie kluczowym `BummerMsg`.

search Pozwala na dostęp do indeksów. Jeśli użytkownik nie posiada tego prawa, dostęp do indeksów nie jest możliwy. Słowo to jest używane głównie w systemie Gopher+.

Jeśli na przykład chcesz zezwolić na pełny dostęp do serwera Gopher maksymalnie 10 użytkownikom z sieci `chatton.com`, powinieneś dodać do pliku `gopherdlocal.conf` wiersz:

```
access: chatton.com browse ftp read search 10
```

Pomiędzy każdymi dwoma wyrazami, nie wyłączając specyfikacji praw dostępu, powinna wystąpić co najmniej jedna spacja. Oto inny przykład:

```
access: bignet.org !browse !ftp read search 3
```

Taki wpis pozwala na odczyt plików i dostęp do indeksów co najwyżej trzem użytkownikom sieci `bignet.org`, jednocześnie nie zezwalając na korzystanie z bramki FTP i przeglądanie zawartości katalogów.

Jeśli zamiast nazw domenowych używasz numerów IP, możesz użyć niepełnego numeru, aby zapewnić dostęp całym podsieciom. Zakładając, że numerem sieci `bignet.org` jest `147.12`, można umożliwić dostęp z niej dodając do pliku `gopherdlocal.conf` wiersz:

```
access: 147.12. !browse !ftp read search 3
```

Ważne jest, by po częściowym numerze IP wpisać kropkę. W przeciwnym przypadku, dostęp do usługi Gopher otrzymaliby również użytkownicy sieci o numerach od `147.120` do `147.129` (ponieważ podane cyfry początkowe pasowałyby również do tych numerów).

Jeśli chcesz umożliwić dostęp z jednego konkretnego komputera, możesz podać jego pełną nazwę, na przykład jeśli chcesz udostępnić usługę Gopher przyjacielowi administrującemu systemem `darkstar`, możesz do pliku `gopherdlocal.conf` dodać wiersz:

```
access: darkstar.nazwa.domeny browse ftp read search 1
```

Większość serwerów Gopher jest dostępna dla wszystkich użytkowników, więc możliwe jest również globalne określenie praw dostępu dla wszystkich systemów, dla których nie określono tych praw wprost. Można to zrobić na przykład tak:

```
access: default !browse !ftp read search 15
```

Powyższy wpis pozwala każdemu użytkownikowi korzystać z usługi Gopher, dając mu prawo do odczytu plików i przeglądania indeksów, ale zabraniając używania bramki FTP i przeglądania zawartości katalogów.

Konfigurowanie pliku Makefile

Aby proces komplikacji mógł przebiec poprawnie, należy odpowiednio zmodyfikować zawartość plików `Makefile.conf` oraz `conf.h`. W wielu wersjach systemu Gopher dla Linuxa domyślne wartości wszystkich potrzebnych parametrów są prawidłowe, ale mimo wszystko warto je przejrzeć, aby uniknąć problemów.

Plik `Makefile.conf` (używany przez plik `Makefile` przy tworzeniu plików wykonawczych) jest dość długi, więc należy poruszać się po nim ostrożnie, by uniknąć przypadkowych zmian. Obszarami, które należy sprawdzić, są przede wszystkim definicje ścieżek dostępu do poszczególnych katalogów oraz ustawienia serwera i klienta – znaną one omówione bardziej szczegółowo.

Jedną z opcji, które możesz chcieć zmienić, jest opcja ułatwiająca wyszukiwanie błędów, w większości systemów domyślnie załączona. Jest ona przydatna w trakcie uruchamiania systemu, ale gdy wszystko działa poprawnie, powinieneś ponownie skompilować kod źródłowy wyłączając tę opcję, dzięki czemu nie będą generowane niepotrzebne już informacje, a program będzie mniejszy i szybszy. Aby wyłączyć tę opcję, wstaw symbol komentarza na początku wiersza, w którym definiowana jest wartość DEBUGGING:

```
#DEBUGGING = -DDEBUGGING
```

W większości systemów w tym wierszu domyślnie nie ma symbolu komentarza.

Definicje ścieżek dostępu do katalogów zebrane są zwykle w bloku składającym się z pięciu do siedmiu wpisów, zależnie od liczby dołączonych stron man. Oto typowy blok definicji katalogów:

```
PREFIX = /usr/local
CLIENTDIR = $(PREFIX)/bin
CLIENTLIB = $(PREFIX)/lib
SERVERDIR = $(PREFIX)/etc

MAN1DIR = $(PREFIX)/man/man1
MAN5DIR = $(PREFIX)/man/man5
MAN8DIR = $(PREFIX)/man/man8
```

Zwykle modyfikacji wymaga tylko wartość zmiennej PREFIX, określająca katalog, w którym system Gopher ma zostać zainstalowany. Domyślną wartością tej zmiennej jest przeważnie /usr/local, ale można podać dowolnie wybraną ścieżkę dostępu (na przykład /usr/gopher). Pozostałe zmienne definiują podkatalogi, w których znajdują się poszczególne pliki systemu Gopher i zwykle nie trzeba modyfikować ich wartości, choć oczywiście można dostosować je do swoich upodobań, na przykład umieszczając wszystkie pliki w jednym katalogu. Znaczenie poszczególnych zmiennych jest następujące:

- CLIENTDIR** katalog, w którym przechowywane będzie oprogramowanie klienta Gopher;
- CLIENTLIB** katalog, w którym znajdzie się plik pomocy przeznaczony dla użytkownika (gopher.hlp);
- SERVERDIR** katalog, w którym znajdzie się serwer gopherd i plik konfiguracyjny (gopherd.conf);
- MAN1DIR** katalog, w którym znajdują się strony man dotyczące programu klienta Gopher;
- MAN8DIR** katalog, w którym znajdują się strony man dotyczące serwera gopherd.

Jeżeli w systemie ma działać poprawnie również klient Gopher, należy odpowiednio zmodyfikować wartość zmiennej CLIENTOPTS w pliku Makefile.config. Oto opcje, które można podać:

- DNOMAIL nie pozwala użytkownikom zdalnym wysyłać plików pocztą,

-DAUTOEXITONU pozwala na zakończenie klienta Gopher za pomocą nie tylko polecenia `q`, ale również `u`.

Jeśli chcesz użyć którejś z tych opcji (lub obu), dopisz je w wierszu definiującym zmienną

```
CLIENTOPTS:  
CLIENTOPTS = -DNOMAIL -DAUTOEXITONU
```

Należy również ustawić wartości czterech zmiennych zawierających dane o serwerze: nazwę serwera, numer portu używanego do połączeń Gopher, lokalizację plików z danymi i znaczniki.

Nazwę domeny, do której należy serwer, należy podać jako wartość zmiennej `DOMAIN`, pozostawiając poprzedzającą ją kropkę:

```
DOMAIN = .tpci.com
```

Nie trzeba definiować wartości tej zmiennej, jeśli dostępna w systemie wersja polecenia `hostname` zwraca pełny adres domenowy komputera. W takim przypadku należy pozostawić pole wartości puste.

Zmienna `SERVERPORT` określa numer portu, który powinien być obsłużony przez usługę Gopher – zwykle jest to port TCP o numerze 70. Odpowiedni wpis ma więc postać:

```
SERVERPORT = 70
```

Jeśli nie chcesz, by usługa Gopher była ogólnodostępna, możesz zmienić tę wartość. Jeżeli jednak chcesz udostępnić ją użytkownikom Internetu (nawet jeśli tylko niektórym), powinieneś użyć portu o numerze 70. W przypadku konfigurowania usługi Gopher do użytku w sieci lokalnej, można wybrać dowolny numer od 1024 do 9999, ale należy upewnić się, że wszystkie programy klientów Gopher używają tego samego numeru portu.

Lokalizacja plików z danymi oferowanymi przez serwer Gopher określona jest przez wartość zmiennej `SERVERDATA`. Domyślnie jej definicja ma postać:

```
SERVERDATA = /gopher-data
```

Powinieneś dostosować tę wartość tak, by wskazywała na rzeczywiste położenie plików w systemie.

Wartością zmiennej `SERVEROPTS` może być zestaw znaczników modyfikujących zachowanie się usługi Gopher. Typowa definicja tej zmiennej ma postać:

```
SERVEROPTS = -DSETPROCTITLE -DCAPFILES # -DBIO -DDL
```

Część wiersza znajdującego się po symbolu komentarza (#) jest ignorowana, możesz więc dostosować opcje do własnych potrzeb zmieniając położenie tego symbolu (o ile porządek opcji pozwala na zastosowanie tak prostego rozwiązania). Dozwolone są następujące znaczniki:

-DADD_DATE_AND_TIME powoduje dodawanie do tytułów dokumentów daty i czasu ich utworzenia;

-DBIO	używany tylko z wersją systemu WAIS opracowaną przez Dona Gilberta (<code>wais8b5</code>);
-DDL	umożliwia współpracę z programem obsługi baz danych <code>dl</code> (system <code>dl</code> musi znajdować się w katalogu zdefiniowanym w zmiennej <code>DLPATH</code> , a zmienna <code>DLOBJS</code> musi wskazywać na położenie plików <code>getdesc.o</code> i <code>enddesc.o</code>);
-DCAPFILES	powoduje, że system jest kompatybilny z wersją używającą katalogu <code>cap</code> ;
-DLOADRESTRICT	umożliwia ograniczanie dostępu w zależności od liczby jednocześnie nawiązanych połączeń (możliwość ta omówiona jest w następnym podrozdziale);
-DSETPROCTITLE	pozwala zdecydować, jaką postać będzie miała nazwa procesu wyświetlaną przez program <code>ps</code> (tylko w systemach opartych na BSD-UNIX).

W pliku `conf.h` znajdują się inne, wykorzystywane w fazie komplikacji informacje konfigurujące usługę Gopher. Najważniejsze z nich dotyczą liczby odnośników zwracanych użytkownikowi po zakończeniu wyszukiwania oraz ilości czasu, jaką należy odczekać przez uznaniem połączenia za przerwane. Zwykle wartości te definiowane są w końcowej części pliku `conf.h`.

Zmienna `WAISMAXHITS` określa maksymalną liczbę odnośników, które może zwrócić w odpowiedzi na zapytanie baza danych WAIS; zwykle wartość ta wynosi około 40. Odpowiedni wpis w pliku `conf.h` ma więc postać:

```
#define WAISMAXHITS 40
```

Zwróć uwagę, że symbol `#` na początku wiersza nie oznacza tym razem komentarza, ponieważ plik `conf.h` jest napisany w języku C. Jest on częścią dyrektywy preprocesora i nie powinien być usuwany. W definicji zmiennej nie występuje również znak równości.

Zmienna `MAXLOAD` jest używana tylko wtedy, gdy częścią wartości zmiennej `SERVEROPTS` (definiowanej w pliku `Makefile.config`) jest znaczek `-DLOADRESTRICT`. Określa ona maksymalne średnie obciążenie serwera, przy którym połączenia Gopher będą jeszcze obsługiwane (wartość ta może być zmieniona przez podanie odpowiedniej opcji w wierszu poleceń). Zwykle jej definicja ma postać:

```
#define MAXLOAD 10.0
```

Zmienne `READTIMEOUT` i `WRITETIMEOUT` określają ilość czasu, jaką należy odczekać przed uznaniem, że połączenie zostało przerwane. Domyślne wartości są zwykle odpowiednie. Wiersze definiujące te zmienne mają postać:

```
#define READTIMEOUT (1*60)
#define WRITETIMEOUT (3*60)
```

Konfiguracja programu klienta Gopher jest prosta. Rozpocząć należy od zdefiniowania serwerów Gopher, z którymi komputer lokalny będzie się łączył – służą do tego zmienne `CLIENT_HOST1` i `CLIENT_HOST2`. Klient Gopher przy uruchomieniu sam wybiera jeden z tych serwerów (o ile oba są zdefiniowane). Odpowiednie wpisy w pliku `conf.h` mogą mieć postać:

```
#define CLIENT1_HOST "serwer_gopher.tpc.com"
#define CLIENT2_HOST "innny_serwer_gopher.tpc.com"
```

Numery portów, których należy użyć do łączenia się z serwerami Gopher definiowane są w wierszach:

```
#define CLIENT1_PORT 70
#define CLIENT2_PORT 70
```

Jeśli usługa Gopher ma być dostępna tylko w sieci lokalnej i numery portów zostały zmienione (na przykład po to, by zabezpieczyć się przed dostępem poprzez Internet), należy podstawić odpowiednie wartości. Jeżeli używany jest tylko jeden serwer, wartość zmiennej `CLIENT2_PORT` powinna wynosić zero.

Język używany przez program klienta Gopher można wybrać spośród kilku obsługiwanych – jest on określony przez wartość zmiennej `DEFAULT_LANG`. Domyslnie jest to język amerykański, ustawiany poleceniem:

```
#define DEFAULT_LANG "En_US"
```

Definicje innych języków znajdują się poniżej tego polecenia, zaznaczone jako komentarze – jeśli chcesz użyć jednego z nich, usuń znacznik komentarza w odpowiednim wierszu i wstaw go w wierszu definiującym język amerykański.

Po wprowadzeniu wszystkich informacji konfiguracyjnych można uruchomić proces komplikacji programu klienta i serwera wydając polecenia:

```
make client
make server
```

Można również skompilować oba programy naraz, wydając polecenie `make` bez argumentów. Programy i dane muszą następnie zostać zainstalowane – w tym celu należy wydać polecenie:

```
make install
```

WAIS i Gopher

Klienci Gopher mają możliwość używania indeksów generowanych przez system WAIS do wyszukiwania interesujących ich plików, ale serwer musi być w tym celu odpowiednio skonfigurowany. System WAIS został omówiony w rozdziale 49. „Konfiguracja węzła WAIS”, więc założymy tu, że zainstalowałeś go już i posiadasz odpowiednie pliki indeksowe, które chcesz udostępnić dla systemu Gopher.

Jeśli chcesz udostępnić usługi systemu WAIS poprzez system Gopher, możliwe, że niezbędna będzie drobna modyfikacja plików źródłowych systemu WAIS. Poszukaj w plikach źródłowych wiersza:

```
if (gLastAnd) printf("search_word: boolean 'and' scored\n:");
```

Wiersz ten należy zaznaczyć jako komentarz, jeśli system WAIS ma współpracować z systemem Gopher. Wprowadź więc odpowiednie symbole na początek i koniec wiersza:

```
/* if (gLastAnd) printf("search_word: boolean 'and' scored\n:"); */
```

Jeśli wiersz ten jest już zaznaczony jako komentarz lub po prostu go nie ma, nie musisz wprowadzać żadnych zmian. Po wprowadzeniu poprawki należy ponownie skompilować system WAIS, wchodząc do katalogu z jego kodem źródłowym i uruchamiając polecenie `make` (które wykona polecenia zapisane w pliku `makefile` znajdującym się w tym katalogu).

Następnie odszukaj w pliku `Makefile.config` znajdującym się w katalogu z kodem źródłowym systemu Gopher definicję zmiennej `WAISTYPE`. Powinna mieć ona postać:

```
WAISTYPE = # -DFREEWAIS_0_4
```

Na koniec należy połączyć usługi WAIS i Gopher. Zakładając, że katalog, w którym znajduje się kod źródłowy systemu Gopher to `/usr/gopher/source`, a kod źródłowy systemu WAIS znajduje się w katalogu `/usr/wais/source`, powinieneś wydać polecenia:

```
cd /usr/gopher/source
ln -s /usr/wais/source/include ./ir
ln -s /usr/wais/source/client/ui .
ln -s /usr/wais/source/bin .
```

Po ponownym skompilowaniu systemu Gopher, powstaną połączenia pomiędzy systemami freeWAIS i Gopher umożliwiające ich współpracę.

Katalogi systemu Gopher

Zakładanie katalogów dla systemu Gopher nie sprawia większych kłopotów; przeważnie nazwy katalogów odpowiadają ogólnie przyjętym konwencjom. Zanim zaczniesz, powinieneś zdecydować, które pliki i dokumenty zamierzasz udostępnić użytkownikom systemu Gopher i stworzyć krótki opis dla każdego z udostępnianych dokumentów (jeśli nie wiesz, co znajduje się w których z dokumentów, powinieneś się z nim zapoznać lub polecić streszczenie go autorowi). Dla uproszczenia założymy, że wszystkie udostępniane dokumenty znajdują się w jednym, wspólnym katalogu.

Zacznij od przejścia do głównego katalogu z dokumentami systemu Gopher (jeśli taki katalog nie istnieje w systemie, powinieneś go założyć). Katalog ten nie powinien znajdować się w tym samym miejscu, co kod źródłowy czy pliki konfiguracyjne – wybierz jakiś katalog o łatwo dającej się zidentyfikować nazwie. Przykładowo, aby utworzyć katalog `/usr/gopher/data`, użyj standardowego polecenia `mkdir`:

```
mkdir /usr/gopher/data
```

Przejdź do katalogu systemu Gopher i skopuj do niego wszystkie dokumenty, które chcesz udostępnić. Następnie zmień ich nazwy tak, by jak najlepiej oddawały zawartość pliku (ponieważ w systemie zwykle używa się zwięzłych, mało mówiących nazw) – ich długość nie powinna przekraczać 80 znaków. Przykładowo, jeśli chcesz udostępnić plik q1.sales, powinieneś zmienić jego nazwę na bardziej opisową, na przykład Company_Sales_1997_Q1, dzięki czemu użytkownicy będą mogli nieco łatwiej zidentyfikować jego zawartość.

Można również ułatwić identyfikację plików w inny sposób: tworząc podkatalog .cap w głównym katalogu z dokumentami systemu Gopher. Dla każdego udostępnianego pliku należy utworzyć w katalogu .cap plik o takiej samej nazwie, ale zawierający dwa wiersze tekstu: jeden z opisem pliku, a drugi z jego numerem. Zakładając, że udostępniasz plik o nazwie q1.sales w katalogu /usr/gopher/data, w katalogu /usr/gopher/data/.cap powinieneś utworzyć plik o nazwie q1.sales, o następującej zawartości:

```
Name=Company Sales for the First Quarter, 1997
Numb=1
```

Pozycja Name może zawierać spacje czy inne symbole, ponieważ jest wyświetlana jako zwykły tekst. Pole Numb zawiera numer, pod którym dany plik będzie wyświetlany w menu programu klienta Gopher. Przykładowo założmy, że oprócz wspomnianego wcześniej pliku udostępniasz jeszcze pliki q2.sales i q3.sales, a odpowiadające im pliki w katalogu .cap mają następującą zawartość:

```
$ cat q1.sales
Name=Company Sales for the First Quarter, 1997
Numb=1
$ cat q2.sales
Name=Company Sales for the Second Quarter, 1997
Numb=2
$ cat q3.sales
Name=Company Sales for the Third Quarter, 1997
Numb=3
```

W menu programu klienta Gopher dane o tych plikach będą wyglądały następująco:

1. Company Sales for the First Quarter, 1997
2. Company Sales for the Second Quarter, 1997
3. Company Sales for the Third Quarter, 1997

Porządek nazw plików w katalogu .cap nie ma znaczenia, ale numery w polu Numb nie powinny się powtarzać.

Alternatywą dla katalogu .cap (dzięki któremu możliwe jest łatwe udostępnianie nowych plików) jest użycie jednego pliku z opisami wszystkich udostępnianych dokumentów. Plik ten powinien znaleźć się w głównym katalogu z dokumentami systemu Gopher i powinien nazywać się .names. Dla trzech plików z powyższego przykładu jego zawartość wyglądałaby następująco:

```
$ cd /usr/gopher/data
$ cat .names
#My Gopher main .names file

Path=./q1.sales
Name=Company Sales for the First Quarter, 1997
Numb=1

Path=./q2.sales
Name=Company Sales for the Second Quarter, 1997
Numb=2

Path=./q3.sales
Name=Company Sales for the Third Quarter, 1997
Numb=3
```

Jak widać, format wpisów w tym pliku jest taki sam, jak w plikach w katalogu `.cap`, dodatkowo podane są tylko nazwy dokumentów (które nie były potrzebne w katalogu `.cap`, ponieważ nazwy dokumentów były takie same, jak nazwy plików z opisami). Niewątpliwą zaletą użycia pliku `.names` jest łatwość reorganizacji wpisów menu, ponieważ w tym celu wystarczy zmodyfikować zawartość tylko jednego pliku. Poza tym w pliku `.names` można umieścić krótkie streszczenie każdego z dokumentów. Oto przykładowy wpis:

```
Path=/gopher
Name=How to Set up A Gopher Service
Numb=16
Abstract=This document shows the steps you need to take to set up a Gopher
service.
```

Można również zdefiniować pozycję menu prowadzącą do innego menu, bądź do całkiem innego serwera. Jest to możliwe dzięki dowiązaniom, które definiowane są w pliku dowiązań, o nazwie `.link`. Plik `.link` zawiera wpisy składające się z pięciu pól w formacie takim, jak w poniższym przykładzie:

```
Name=More Sales Info
Type=1
Port=70
Path=/usr/gopher/data/more_sales
Host=wizard.tpc.com
```

Zawartość pola `Name` określa tekst, który będzie wyświetlany w programie klienta Gopher, i może zawierać dowolny opis. Pole `Type` zawiera znacznik typu dokumentu, na który wskazuje dowiązanie. Dozwolone są następujące wartości:

- 0** plik tekstowy,
- 1** katalog,
- 2** serwer nazw CSO,
- 7** pełny indeks tekstowy,
- 8** sesja programu Telnet,

- 9** plik binarny,
- h** dokument w formacie HTML,
- I** plik zawierający grafikę,
- M** plik MIME,
- s** plik dźwiękowy.

Są to te same typy plików, które zostały przedstawione w liście typów plików obsługiwanych przez system Gopher, choć ta lista jest nieco krótsza.

Pole `Port` określa port, którego należy użyć przy łączaniu się z systemem zdalnym (jeśli dowiązanie prowadzi do systemu zdalnego), natomiast pole `Path` zawiera ścieżkę dostępu do pliku w systemie lokalnym lub zdalnym. Pole `Host` zawiera nazwę serwera, na którym znajduje się dany plik. Jeśli dowiązanie do innego serwera ma prowadzić poprzez usługę FTP lub WAIS, w polu `Path` musisz podać również nazwę usługi i odpowiednie argumenty. Przykładowo, jeśli w menu systemu Gopher ma znaleźć się pozycja odsyłająca użytkownika do pliku na innym serwerze poprzez FTP, odpowiedni wpis w pliku `.link` powinien mieć postać:

```
Name=More Sales Info
Type=1
Port=+
Path=ftp:chatton.bigcat.com@/usr/gopher/cats
Host=+
```

Symbol + w polach `Port` i `Host` oznacza, że serwer FTP powinien zwrócić odpowiednie wartości używając domyślnych numerów portów (na przykład portu 21 dla FTP). Dowiązanie do katalogu WAIS ma postać:

```
Name=More Sales Info
Type=7
Port=+
Path=waisrc:/usr/wais/data
Host=+
```

Można również w menu systemu Gopher umieścić pozycję powodującą uruchomienie jakiegoś programu. W tym celu w polu `Path` należy wpisać polecenie `exec`:

```
Path=exec:"argumenty" : nazwa_programu
```

`nazwa_programu` jest nazwą pliku wykonywalnego, natomiast `argumenty` zostaną przekazane do programu tak, jakby zostały podane w wierszu polecień. Jeśli program ma być wywoływany bez argumentów, pozostaw cudzysłów pusty. Format ten jest może nieco dziwny, ale bywa przydatny.

Uruchamianie usługi Gopher

Usługę Gopher można uruchomić z jednego z plików `rc`, z wiersza poleceń bądź też za pomocą programu rezydentnego `inetd`. Odpowiednie polecenie (wpisane w wierszu poleceń lub w pliku `rc`) może mieć postać:

```
/usr/local/etc/gopherd /usr/gopher/gopher-data 70
```

Jego argumentami są ścieżka dostępu do katalogu zawierającego dane oraz numer portu, którego należy używać do połączeń.

Do programu `gopherd` można również przekazać kilka opcji modyfikujących jego zachowanie – większość z nich ma swoje odpowiedniki w plikach konfiguracyjnych. Oto dostępne opcje:

- C wyłącza buforowanie katalogów;
- c wyłącza ograniczenia `chroot`;
- D włącza generowanie informacji diagnostycznych;
- I informuje program `gopherd`, że będzie wywoływany przez program `inetd`;
- L pozwala określić maksymalne średnie obciążenie systemu (wymaga podania wartości liczbowej);
- l pozwala podać nazwę pliku, do którego rejestrowane będą informacje o połączeniach (jego nazwę należy podać po opcji -l);
- o pozwala na użycie pliku konfiguracyjnego o nazwie innej niż `gopherd.conf` (nazwę tę należy podać po opcji -o);
- u pozwala ustalić identyfikator użytkownika, który będzie właścicielem programu `gopherd` (po tej opcji musi być podany prawidłowy identyfikator).

Aby poprawić bezpieczeństwo systemu, powinieneś - używając polecenia `chroot` - stworzyć wyizolowany podsystem plików dla programu Gopher (jak trzeba to zrobić, pokazaliśmy w rozdziale 48 „Konfigurowanie anonimowego węzła FTP”). Użycie opcji `-c` jest mniej bezpieczne, niż uruchamianie programu `gopherd` z aktywnym programem `chroot`. Poza tym należy stosować opcję `-u` tak, by właścicielem procesu był zwykły użytkownik, a nie administrator. Takie rozwiązanie zabezpiecza przed wykorzystaniem przez hakerów ewentualnych błędów w programie.

Jeśli chcesz, aby `gopherd` był uruchamiany przez program rezydentny `inetd` za każdym razem, gdy nastąpi próba połączenia, powinieneś zmodyfikować zawartość plików `/etc/services` i `/etc/inetd.conf`, dodając do nich wpisy dla programu `gopherd`. Odpowiedni wiersz w pliku `/etc/services` zwykle ma postać:

```
gopher 70/tcp
```

Wpis w pliku `/etc/inetd.conf` może natomiast wyglądać tak:

```
gopher stream tcp nowait root /usr/local/etc/gopherd gopherd -I -u  
Σ id_użytkownika
```

`id_użytkownika` to identyfikator użytkownika, który będzie właścicielem procesu `gopherd` (warto założyć do tego celu osobne konto, o standardowych prawach dostępu).

Po zainstalowaniu i skonfigurowaniu procesu serwera Gopher przyszedł czas na jego przetestowanie. Najpierw jednak potrzebny będzie program klienta Gopher. Po uruchomieniu go i połączeniu się z serwerem Gopher (przez podanie nazwy serwera) powinna zostać wyświetlona zawartość głównego katalogu z danymi systemu Gopher. Innym sposobem przetestowania systemu jest użycie programu Telnet. Połącz się za jego pomocą z portem obsługiwany przez system Gopher, wydając polecenie:

```
telnet gopher 70
```

Jeśli połączenie zostanie nawiązane poprawnie, na ekranie wyświetlona zostanie zawartość systemu Gopher.

Jeszcze inną metodą sprawdzenia, czy Gopher został zainstalowany prawidłowo, jest użycie polecenia `gopherls`, którego argumentem jest nazwa katalogu z danymi systemu Gopher. Polecenie to może więc mieć postać:

```
gopherls /usr/wais/gopher/data
```

Przydaje się ono również do testowania katalogów dodawanych do systemu Gopher już w trakcie jego działania.

Daj światu znać o sobie

Poświęciłeś sporo czasu na konfigurację usługi Gopher, więc warto poinformować innych użytkowników Internetu o tym fakcie (oczywiście powinieneś zrobić to tylko wtedy, gdy chcesz udostępnić swoje dokumenty wszystkim użytkownikom sieci i masz pewność, że system działa prawidłowo – nie podejmuj opisanych niżej kroków jeśli chcesz udostępnić usługę Gopher tylko w sieci lokalnej).

Jeśli chcesz, by dane o Twoim serwerze Gopher znalazły się w głównej liście katalogów Gopher, wyślij pocztą elektroniczną wiadomość na adres:

```
gopher@boombox.miro.umn.edu
```

W treści wiadomości powinna znaleźć się pełna nazwa usługi (tak, jak pojawia się ona w menu głównym), nazwa i numer IP serwera, numer portu używanego do połączeń Gopher (jeśli usługa ma być ogólnodostępna, powinien być to port TCP o numerze 70), adres e-mailowy administratora systemu Gopher i krótki opis tego, co można znaleźć w systemie. Jeśli chcesz, możesz również podać tekst określający ścieżkę dostępu do katalogu z danymi, ale nie jest to konieczne, ponieważ większość systemów Gopher

uruchamia się w katalogu głównym (choć możliwość taka może być przydatna, jeśli udostępniasz kilka różnych systemów menu do różnych celów).

Podsumowanie

Teraz usługa Gopher jest już gotowa do użycia. Trzeba jeszcze wprowadzić dane o udostępnianych plikach, ale to wykracza poza zakres materiału omawiany w tym rozdziale. Jeśli chcesz, zajrzyj do jakiejś dobrej książki na temat Internetu albo systemu Gopher – dowiesz się z niej więcej o katalogach, plikach i elementach menu systemu Gopher. Gopher jest wygodnym sposobem udostępniania informacji, a choć proces konfiguracji jest dość długi, po jego przeprowadzeniu system Gopher zwykle pracuje bardzo dobrze.

Jeśli chcesz w swoim systemie linuxowym skonfigurować serwer WWW, zajrzyj do rozdziału 51. „Konfiguracja węzła WWW”.

Tworzenie stron w języku HTML omówione jest w rozdziale 53. „Podstawy języka HTML”.

Jak używać języka Java, aby uatrakcyjnić swoją stronę główną, dowiesz się w rozdziale 54. „Java i JavaScript”.

Program `make` i pliki `makefile` omówione są w rozdziale 56. „Zarządzanie kodem źródłowym”.

Rozdział 51.

Konfigurowanie

węzła WWW

Tim Parker

W tym rozdziale:

- υ Oprogramowanie serwera WWW
- υ Apache

Nie ma chyba nikogo, kto nie słyszałby jeszcze o stronach WWW (ang. *World Wide Web*). Jest to najpopularniejszy aspekt Internetu. Właśnie z powodu popularności WWW coraz więcej użytkowników sieci decyduje się na założenie własnych serwerów WWW i utworzenie swoich stron głównych. Obecnie dostępnych jest coraz więcej wyrafinowanych pakietów oprogramowania działających jako serwery WWW dla różnych systemów operacyjnych. Linux, oparty na systemie UNIX, również zawiera oprogramowanie potrzebne do założenia własnego serwera WWW.

Do skonfigurowania węzła WWW nie jest potrzebne żadne wymyślne oprogramowanie – wystarczy trochę czasu i odpowiednie dane konfiguracyjne. O tym właśnie traktuje ten rozdział. Przyjrzymy się bliżej zagadniению konfiguracji serwera WWW w systemie linuxowym, przeznaczonego dla przyjaciół, użytkowników sieci lokalnej czy też dla całego Internetu.

Jednym z najważniejszych powodów popularności stron WWW, oprócz możliwości multimedialnych, są hiperłącza. Pozwalają one przenosić się jednym kliknięciem z jednego dokumentu do drugiego, z serwera na serwer, od grafiki do filmów i tak dalej. Instrukcje powodujące odpowiednie przejścia są zapisane w kodzie stron WWW.

Strony WWW obsługiwane są przez dwie warstwy oprogramowania: oprogramowanie klienta WWW (nazywane często przeglądarką WWW) i oprogramowanie serwera. Bardzo popularne są programy klientów WWW takie jak Netscape czy Mosaic, ale dostępne są również inne pakiety, niektóre przeznaczone do pracy wyłącznie pod kontrolą Linuxa czy X Window.

Oprogramowanie serwera WWW

Dla systemu Linux dostępne są trzy główne wersje serwera WWW, pochodzące odpowiednio z organizacji NCSA, CERN i Plexus. Najłatwiejszy w użyciu jest system opracowany przez NCSA; organizacja ta opracowała również przeglądarkę Mosaic. Jest on niewielki i szybki, może działać jako program rezydentny, może też być uruchamiany przez proces `inetd`; poza tym zapewnia dość wysoki poziom bezpieczeństwa. W tym rozdziale omówimy proces konfiguracji pakietu pochodzącego właśnie z NCSA, choć z łatwością można również użyć dwóch pozostałych pakietów (dostosowując odpowiednio dane konfiguracyjne). W wielu węzłach FTP i BBS dostępny jest również dość popularny serwer WWW o nazwie Apache – zajmiemy się nim w następnym podrozdziale, po omówieniu bardziej tradycyjnych serwerów WWW.



Oprogramowanie serwera WWW w trzech wymienionych wcześniej wersjach dostępne jest poprzez anonimowe FTP lub strony WWW pod następującymi adresami:

CERN: `ftp://ftp.w3.org/pub/httpd` (**FTP**)

NCSA: `ftp.ncsa.uiuc.edu/Web/httpd/Wnix/ncsa_httpd`

`http://hoohoo.ncsa.uiuc.edu` (**WWW**)

Plexus: `http://www.sandelman.ocunix.on.ca/server/doc/plexus.html`

Oprogramowanie pochodzące z NCSA dostępne jest zarówno w postaci kodu źródłowego, jak i w wersji skompilowanej. Użycie wersji skompilowanej jest znacznie prostsze, ponieważ nie trzeba konfigurować i kompilować kodu źródłowego. Pliki wykonywalne przeważnie dostępne są w postaci zarchiwizowanej programem `tar` i skompresowanej, więc należy je rozpakować. Często spotykane są również gotowe do użycia wersje dostarczane na płytach CD-ROM. Jeśli posiadasz skompresowaną wersję programu serwera WWW, powinieneś, trzymając się wskazówek zawartych w pliku `README` lub podobnym, rozpakować pliki do odpowiednich katalogów.

Rozpakowywanie plików z oprogramowaniem serwera WWW

Jeśli oprogramowanie w postaci kodu źródłowego lub skompilowane otrzymałeś z węzła FTP czy BBS, prawdopodobnie będziesz je musiał najpierw rozpakować (za pomocą programów `tar` i `unzip` – powinieneś jednak sprawdzić to w dołączonych plikach `README`, w przeciwnym przypadku może okazać się, że robisz to niepotrzebnie). Zwykle należy utworzyć osobny katalog dla oprogramowania WWW, przejść do niego i rozpakować otrzymane pliki, na przykład wydając polecenie

```
zcat httpd_X.X_XXX.tar.Z | tar xvf -
```

W nazwie tych plików często znajduje się, oprócz numeru wersji, nazwa platformy, dla której przeznaczone jest oprogramowanie – na przykład `httpd_1.5_linux.tar.z`. Jako argument polecenia `zcat` powinieneś podać pełną nazwę otrzymanego pliku. Instrukcje dotyczące instalacji czasem dostarczane są w postaci osobnego pliku `tar`, na przykład `Install.tar.z`, który powinieneś również zdobyć i rozpakować polecienniem

```
zcat Install.tar.z
```

Podczas wydawania powyższego polecenia katalogiem bieżącym powinien być katalog docelowy – inaczej będziesz musiał przenosić ręcznie sporą liczbę plików. Pliki z oprogramowaniem serwera WWW można umieścić w dowolnym miejscu, ale dobrym pomysłem jest utworzenie dla nich odrębnego katalogu, na przykład `/usr/web` czy `/var/web`, dzięki czemu łatwiej będzie kontrolować prawa dostępu do tych plików.

Po zdekompresowaniu odpowiednich plików i umieszczeniu bibliotek we właściwych katalogach, warto przyjrzeć się podkatalogom utworzonym automatycznie. Wśród nich powinny znaleźć się następujące podkatalogi:

- cgi-bin** standardowy interfejs pomiędzy stronami WWW a programami i skryptami;
- conf** pliki konfiguracyjne;
- icons** ikony przeznaczone do użycia na stronach głównych;
- src** kod źródłowy i (czasem) pliki wykonywalne;
- support** aplikacje wspierające obsługę WWW.

Kompilowanie oprogramowania serwera WWW

Jeśli nie musisz modyfikować i rekompilować kodu źródłowego dostosowując go do potrzeb Linuxa (ponieważ posiadasz linuxową wersję programu), możesz pominać szczegółowe omówienie konfiguracji znajdującej się w dalszej części tego podrozdziału. Z drugiej strony warto jednak wiedzieć, co znajduje się w kodzie źródłowym, ponieważ dzięki temu można lepiej zrozumieć współdziałanie systemu i oprogramowania serwera. Jeśli posiadasz ogólną wersję kodu źródłowego serwera WWW opracowanego przez NCSA, nie dostosowaną do Twojego systemu, musisz przed komplikacją skonfigurować kod źródłowy.

Rozpocznij od wprowadzenia do pliku `src/Makefile` informacji o systemie. Powinieneś sprawdzić wartości kilku zmiennych:

- | | |
|-------------------|--|
| AUX_CFLAGS | pozwala podać dodatkowe opcje kompilatora – usuń komentarz z wiersza definiującego wersję dla Linuxa (zwykle odpowiednie wiersze są opatrzone komentarzem umożliwiającym ich identyfikację); |
|-------------------|--|

CC	pozwala na podanie nazwy kompilatora języka C (zwykle <code>cc</code> lub <code>gcc</code>);
EXTRA_LIBS	pozwala na dołączenie dodatkowych bibliotek (które nie są wymagane w systemie Linux);
FLAGS	pozwala podać dodatkowe opcje dla konsolidatora (większość wersji linuxowych nie wymaga żadnych dodatkowych opcji).

Należy również przyjrzeć się wartości zmiennej `CFLAGS`, zawierającej listę znaczników kompilatora. Niektóre z nich mogą być już ustawione. Oto dostępne wartości:

-DSECURE_LOGS	zabezpiecza przed zakłócaniem procesu rejestrowania danych, prowadzonego przez serwer WWW, przez skrypty CGI;
-DMAXIMUM_DNS	pozwala na użycie bezpieczniejszego systemu tłumaczenia nazw (kosztem wydajności);
-DMINIMAL_DNS	nie zezwala na zwrotne tłumaczenie nazw, co poprawia wydajność;
-DNO_PASS	zabezpiecza przed uruchamianiem wielu procesów potomnych;
-DPEM_AUTH	uaktywnia schematy autoryzacji PEM/PGP
-DXBITHACK	pozwala na sprawdzanie, czy plik HTML jest wykonywalny;
-O2	załącza optymalizację.

Najprawdopodobniej nie będziesz musiał modyfikować wartości zmiennej `CFLAGS`, ale powinieneś być świadomym jej przeznaczenia. Po sprawdzeniu i ewentualnym poprawieniu danych zapisanych w pliku `src/Makefile` należy skompilować kod źródłowy, wydając polecenie:

```
make linux
```

Jeśli jego wykonanie spowoduje wyświetlenie komunikatów o błędach, powinieneś jeszcze raz dokładnie sprawdzić dane w pliku konfiguracyjnym. Najczęściej przyczyną problemów jest źle zdefiniowana platforma (lub wybranie kilku platform jednocześnie).

Konfiguracja oprogramowania serwera WWW

Po zainstalowaniu kodu źródłowego w odpowiednich katalogach i skompilowaniu go dla Linuksa należy skonfigurować system. Proces ten powinieneś rozpocząć od pliku `httpd.conf-dist`. Skopiuj go do pliku o nazwie `httpd.conf` – jest to nazwa pliku konfiguracyjnego programu `httpd`. Zanim zaczniesz go modyfikować, powinieneś zdecydować,

czy chcesz uruchamiać program `httpd` jako program rezydentny, czy też ma on być uruchamiany przez proces `inetd`. Jeśli przewidujesz częste użycie tego programu, powinieneś wybrać pierwszą wersję. W przeciwnym przypadku obie możliwości są równie dobre.

W pliku `httpd.conf` zdefiniowanych jest kilka wartości, które wymagają sprawdzenia lub wprowadzenia jakichś danych. Każda z definicji zmiennych ma składnię:

zmienna wartość

Pomiędzy nazwą zmiennej a wartością nie pojawia się znak równości ani żaden inny specjalny symbol. Oto kilka przykładowych wierszy:

```
FancyIndexing on
HeaderName Header
ReadmeName README
```

Ścieżki dostępu do plików i katalogów zwykle podawane są nie jako pełne ścieżki dostępu, ale względem katalogu serwera WWW. Poniżej podajemy zmienne, których wartości należy dostosować do wymagań systemu.

AccessConfig	Lokalizacja pliku konfiguracyjnego <code>access.conf</code> – domyślnie jest to <code>conf/access.conf</code> . W tym miejscu można użyć zarówno pełnej, jak i względnej ścieżki dostępu.
AgentLog	Nazwa pliku, w którym rejestrowane będą dane o typie i wersji przeglądarki WWW używanej do połączenia się z serwrem. Domyślana wartość to <code>logs/agent_log</code> .
ErrorLog	Nazwa pliku, do którego zapisywane będą dane o błędach. Domyślną wartością jest <code>logs/error_log</code> .
Group	Identyfikator grupy będącej właścicielem procesu serwera (używany tylko, jeśli serwer działa jako program rezydentny). Może to być zarówno nazwa, jak i identyfikator numeryczny grupy, z tym, że należy poprzedzić go symbolem <code>#</code> . Domyślana wartością jest <code>#-1</code> .
MaxServers	Maksymalna dozwolona liczba procesów potomnych.
PidFile	Nazwa pliku, w którym przechowywane będą identyfikatory procesów wszystkich kopii programu <code>httpd</code> . Domyślana wartość to <code>logs/httpd.pid</code> . Wartość ta jest używana tylko wtedy, gdy serwer działa jako program rezydentny.
Port	Numer portu, pod którym należy oczekiwac na połączenia programów klientów WWW. Domyślnie jest to port 80. Jeśli nie chcesz, by serwer WWW był ogólnodostępny, zmień tę wartość.
ResourceConfig	Ścieżka dostępu do pliku <code>srm.conf</code> , zwykle <code>conf/srm.conf</code> .

ServerAdmin	Adres e-mailowy administratora.
ServerName	Pełna nazwa domenowa serwera.
ServerRoot	Ścieżka dostępu do katalogu, powyżej którego użytkownicy nie będą mogli się dostać (zwykle jest to katalog główny serwera WWW lub katalog <code>/usr/local/etc/httpd</code>).
ServerType	Typ serwera: <code>standalone</code> (samodzielny program rezydentny) lub <code>inetd</code> (uruchamiany przez proces <code>inetd</code>).
StartServers	Liczba procesów serwera, które uruchamiane są przy uruchamianiu programu rezydentnego.
TimeOut	Czas (w sekundach), przez jaki należy czekać na odpowiedź klienta przed przerwaniem połączenia (domyślnie zmienna ta ma wartość 1800 i należy ją zmniejszyć).
TransferLog	Ścieżka dostępu do pliku, w którym rejestrowane będą dane o dostępie do plików. Domyślną wartością jest <code>logs/access_log</code> .
TypesConfig	Ścieżka dostępu do pliku konfiguracyjnego MIME. Domyślnie jest to <code>conf/mime.conf</code> .
User	Identyfikator użytkownika będącego właścicielem procesu serwera WWW (używany tylko wtedy, gdy serwer działa jako program rezydentny). Można tu podać identyfikator tekstowy lub liczbowy, ale poprzedzony symbolem <code>#</code> . Domyślna wartość to <code>#-1</code> .

Należy również sprawdzić dane o zasobach serwera, zapisane w pliku `srm.conf`. Oto lista zmiennych, których wartości mogą wymagać modyfikacji.

AccessFileName	Zawiera nazwę pliku z danymi o prawach dostępu (domyślnie <code>.htaccess</code>).
AddDescription	Pozwala wprowadzić opis typu pliku, na przykład <code>AddDescription PostScript file *.ps</code> . Dozwolone jest wprowadzenie większej liczby takich wpisów.
AddEncoding	Pozwala podać, że określony typ plików jest w jakiś sposób zakodowany, na przykład <code>AddEncoding compress z</code> . Podobnie jak poprzednio, dozwolone jest wprowadzenie większej liczby takich wpisów.
AddIcon	Pozwala na podanie nazwy ikony wyświetlanej jako symbol pliku każdego z typów.
AddIconType	Pozwala na określenie wyświetlanej ikony na podstawie typu MIME.

AddType	Pozwala wyłączyć rozpoznawanie typu MIME dla plików o określonym rozszerzeniu.
Alias	Pozwala na zastąpienie ścieżki dostępu krótszą nazwą, na przykład <code>Alias /usr/www/data</code> .
DefaultType	Definiuje domyślny typ MIME – zwykle <code>text/plain</code> .
DefaultIcon	Określa, która ikona ma być używana domyślnie, jeśli aktywna jest opcja <code>FancyIndexing</code> (domyślnie jest to ikona zapisana w pliku <code>icons/unknown.xbm</code>).
DirectoryIndex	Pozwala określić nazwę pliku, który należy zwrócić jeśli użytkownik nie zażądał żadnego konkretnego pliku. Domyślnie jest to plik <code>index.html</code> .
DocumentRoot	Zawiera pełną ścieżkę dostępu do katalogu zawierającego dokumenty HTML. Domyślna wartość to <code>/usr/local/etc/httpd/htdocs</code> .
FancyIndexing	Pozwala na dodanie do listy indeksowanych plików ikon i nazw plików. Domyślną wartością jest <code>on</code> (załączone). Opcja ta jest potrzebna dla zapewnienia kompatybilności z pierwszą wersją HTTP.
HeaderName	Zawiera nazwę pliku wyświetlany na początku listy indeksowanych plików. Domyślnie jest to plik o nazwie <code>Header</code> .
IndexOptions	Zawiera parametry indeksowania (między innymi <code>FancyIndexing</code> , <code>IconsAreLinks</code> , <code>ScanHTMLTitles</code> , <code>SuppressLastModified</code> , <code>SupressSize</code> i <code>SuppressDescription</code>).
ReadmeName	Zawiera nazwę pliku wyświetlany jako stopka wraz z danymi o zawartości katalogów. Domyślnie jest to plik o nazwie <code>README</code> .
Redirect	Odwzorowuje ścieżkę dostępu na inny adres URL.
ScriptAlias	Pozwala na zastąpienie ścieżki dostępu krótszą nazwą, podobnie jak polecenie <code>Alias</code> . Polecenie <code>ScriptAlias</code> używane jest w skryptach.
UserDir	Zawiera nazwę katalogu, który może być użyty przez użytkowników do dostępu do <code>httpd</code> . Domyślnie jest to katalog <code>public_html</code> . Przeważnie należy wpisać tu nazwę katalogu zawierającego stronę główną użytkownika. Można również użyć wartości <code>DISABLED</code> (wyłączone).

Trzecim plikiem, którego zawartość należy przejrzeć i ewentualnie zmodyfikować, jest plik `access.conf-dist`, definiujący usługi dostępne dla przeglądarek WWW. Zwykle dostępne są wszystkie usługi, możesz jednak chcieć zmodyfikować ten plik dla zapewnienia

wyższego poziomu bezpieczeństwa lub po to, by wyłączyć usługi nie obsługiwane przez Twój węzeł WWW. Format pliku `conf-dist` jest nieco inny niż format dwóch poprzednich plików. Używane są w nim polecenia dzielące cały plik na sekcje, w których znajdują się dyrektywy. Ogólnie format pojedynczego wpisu jest następujący:

```
<Directory nazwa_katalogu>
...
</Directory>
```

Wszystko, co znajduje się pomiędzy ogranicznikami `<Directory>` i `</Directory>`, to dyrektywy. Sprawa komplikuje się jeszcze bardziej, ponieważ w obrębie pliku może wystąpić kilka rodzajów dyrektyw. Najlepszym sposobem na dostosowanie pliku `access.conf-dist` w przypadku typowej instalacji serwera WWW jest trzymanie się poniższych rad.

1. Znajdź dyrektywę `Options` i usuń opcję `Indexes`. Dzięki temu użytkownicy nie będą mogli przeglądać zawartości katalogu `httpd`. Inne opcje podawane po dyrektywie `Options` omówione zostaną za chwilę.
2. Znajdź pierwszą dyrektywę `Directory` i sprawdź ścieżkę dostępu do katalogu ze skryptami `cgi`. Domyślnie jest to wartość `/usr/local/etc/httpd/cgi-bin`.
3. Znajdź definicję zmiennej `AllowOverride` i ustaw jej wartość na `None` (żadne), dzięki czemu użytkownicy nie będą mogli zmieniać ustawień. Domyślną wartością jest `All` (wszystkie). Dostępne wartości zostaną krótko omówione w dalszej części tego podrozdziału.
4. Znajdź dyrektywę `Limit` i nadaj jej żądaną wartość.

Dyrektyna `Limit` pozwala ograniczyć dostęp do serwera. Poniżej przedstawiamy listę dostępnych wartości.

allow	Pozwala na dostęp do serwera WWW z systemu o nazwie podanej po słowie <code>allow</code> .
deny	Nie pozwala na dostęp do serwera WWW z systemu o nazwie podanej po słowie <code>deny</code> .
order	Określa porządek, w jakim odczytywane będą dyrektywy <code>allow</code> i <code>deny</code> (zwykle <code>deny, allow</code> , ale może również być odwrotnie).
require	Umożliwia żądanie autoryzacji poprzez plik użytkownika, określony na podstawie wartości zmiennej <code>AuthUserFile</code> .

Dyrektyna `Option` zawiera kilka wartości o różnym przeznaczeniu. Domyślnie zawiera ona wpisy:

```
Options Indexes FollowSymlinks
```

Wpis `Indexes` powinien zostać usunięty już wcześniej. Wszystkie wpisy odnoszą się do katalogu, w którym znajduje się pole `Options`. Dostępne wartości to:

All	Włączenie wszystkich opcji.
ExecCGI	Umożliwienie wykonywania skryptów CGI z określonego katalogu.
FollowSymLinks	Umożliwienie użycia przez program <code>httpd</code> dowiązań symbolicznych.
Includes	Włączenie obsługi przez serwer plików <code>include</code> .
IncludesNoExec	Włączenie obsługi przez serwer plików <code>include</code> , ale przy wyłączeniu opcji <code>exec</code> .
Indexes	Umożliwienie użytkownikom przeglądania indeksów generowanych przez serwer (nie dotyczy indeksów skompilowanych wcześniej).
None	Wyłączenie wszystkich opcji.
SymLinksIfOwnerMatch	Umożliwienie użycia przez program <code>httpd</code> dowiązań symbolicznych pod warunkiem, że identyfikator właściciela dowiązania jest taki sam jak identyfikator właściciela pliku, do którego prowadzi dowiązanie.

Zmienna `AllowOverride` domyślnie ma wartość `All`, którą należy zmienić; dla większości systemów zalecana jest wartość `none`. Poniżej przedstawiamy wszystkie dostępne wartości.

All	Umożliwienie kontroli dostępu przez pliki konfiguracyjne znajdujące się w każdym katalogu.
AuthConfig	Załączanie obsługi kilku procedur zabezpieczających. Dostępne wartości to: <code>AuthName</code> (weryfikacja nazwy katalogu), <code>AuthType</code> (weryfikacja typu katalogu, choć dozwolony jest tylko jeden typ – <code>Basic</code>), <code>AuthUserFile</code> (określenie nazwy pliku zawierającego identyfikatory i hasła użytkowników) i <code>AuthGroupFile</code> (określenie nazwy pliku zawierającego nazwy grup).
FileInfo	Uaktywnienie dyrektyw <code>AddType</code> i <code>AddEncoding</code> .
Limit	Uaktywnienie dyrektywy <code>Limit</code> .
None	Wyłączenie kontroli dostępu poprzez pliki konfiguracyjne.
Options	Uaktywnienie dyrektywy <code>Options</code> .

Teraz w plikach konfiguracyjnych powinny już znajdować się prawidłowe dane. Choć składnia tych plików nie jest zbyt intuicyjna, przyjrzenie się wartościom domyślnym pozwala łatwo zorientować się, jaki format powinny mieć wprowadzane modyfikacje. Można przystąpić do uruchomienia oprogramowania serwera WWW.

Uruchamianie serwera WWW

Po zakończeniu procesu konfiguracji przyszła pora na wypróbowanie oprogramowania serwera WWW. Dane o tym, czy ma ono działać jako program rezydentny, czy też będzie uruchamiane przez proces `inetd`, zapisane zostały w odpowiednich plikach konfiguracyjnych. Sposób uruchamiania oprogramowania jest nieco inny w obu tych przypadkach, ale w wierszu poleceń zawsze można użyć następujących opcji:

- d ścieżka dostępu do katalogu głównego serwera (używana tylko wtedy, jeśli wartość domyślna nie jest prawidłowa),
- f nazwa pliku konfiguracyjnego, o ile nie ma to być domyślny plik `httpd.conf`,
- v wyświetlenie numeru wersji.

Jeśli do uruchamiania oprogramowania serwera WWW chcesz używać procesu `inetd`, musisz do pliku `/etc/services` dopisać wiersz

```
http      port/tcp
```

zwalniający programowi `inetd` na uruchamianie serwera WWW (port to numer używanego portu, zwykle 80).

Następnie do pliku `/etc/inetd.conf` należy dodać polecenie pozwalające na uruchomienie programu `httpd`:

```
httpd stream tcp nowait nobody /usr/web/httpd
```

Ostatnia pozycja określa ścieżkę dostępu do programu serwera WWW. Po zrestartowaniu procesu `inetd` (można to zrobić używając polecenia `kill` i uruchamiając go ponownie lub restartując system) usługa powinna być już dostępna przez port o numerze określonym w pliku `/etc/services`.

Jeśli oprogramowanie serwera WWW ma działać jako program rezydentny, można uruchomić je bezpośrednio z wiersza poleceń:

```
httpd &
```

Lepiej jednak dodać odpowiednie polecenia do jednego z plików `rc` przetwarzanych podczas uruchamiania systemu. Odpowiedni wpis ma zwykle postać:

```
#start httpd
if [ -x /usr/web/httpd ]
then
  /usr/web/httpd
fi
```

Powinieneś oczywiście dostosować ścieżkę dostępu do programu `httpd` do wymagań swojego systemu. Po ponownym uruchomieniu systemu oprogramowanie serwera powinno obsługiwać port o numerze domyślnym.

Aby przetestować działanie serwera WWW, uruchom dowolną przeglądarkę stron WWW i w polu URL wprowadź adres

```
http://nazwa_serwera
```

gdzie `nazwa_serwera` to nazwa Twojego serwera WWW. Możesz to zrobić zarówno przez Internet, jak i z dowolnego komputera w sieci lokalnej (jeśli jesteś podłączony do sieci lokalnej). Jeżeli wyświetlona zostanie zawartość pliku `index.html` znajdującego się w katalogu głównym serwera WWW, wszystko jest w porządku. W przeciwnym przypadku poszukaj informacji o problemie w plikach, do których rejestrowane są błędy i przejrzyj jeszcze raz pliki konfiguracyjne.

Jeśli nie zainstalowałeś jeszcze żadnej przeglądarki internetowej, możesz za pomocą programu `telnet` sprawdzić, czy serwer WWW działa prawidłowo. Wydaj następujące polecenie, podstawiając nazwę swojego serwera i numer portu, jeśli jest inny niż 80:

```
telnet www.wizard(tpci.com 80
```

O ile serwer działa prawidłowo, powinny zostać wyświetlane następujące informacje:

```
Connected to wizard(tpci.com
Escape character is '^].
HEAD/HTTP/1.0
HTTP/1.0 200 OK
```

Powinny również pojawić się informacje o dacie i zawartości serwera. Prawdopodobnie nie będziesz miał dostępu do żadnego pliku, ale dzięki temu wiesz, że serwer działa prawidłowo.

Apache

Apache to pakiet oprogramowania zawierający serwer WWW. Staje się on coraz popularniejszy, głównie dzięki jego elastyczności i istnieniu wersji przeznaczonych dla wielu platform sprzętowych i systemów operacyjnych. Apache jest oparty na serwerze Mosaic, o którym wspomniano już wcześniej w tym rozdziale, ale zawiera dość sporo nowego kodu. Został napisany jako program darmowy i jest ogólnodostępny. W sieci Internet działa spora grupa użytkowników służących pomocą w razie problemów, a w księgarniach dostępnych jest również kilka książek na jego temat.



Podczas pisania tej książki najnowsza wersja serwera Apache miała numer 1.2. Dostępnych jest również wiele starszych wersji, ale w wersji 1.2 wprowadzono na tyle dużo istotnych zmian, że warto ją załadować z któregoś z węzłów FTP lub też z węzła <http://www.apache.org>, poświęconego temu programowi.

Użycie programu make z pakietem Apache

Serwer Apache rozpowszechniany jest czasem tylko w postaci nie skompilowanego kodu źródłowego. W niektórych dystrybucjach Linuxa dostarczanych na dyskach CD-ROM dostępne są również pliki wykonywalne, co pozwala na pominięcie etapu komplikacji. Jeśli jednak załadowałeś oprogramowanie Apache z węzła FTP czy poprzez stronę WWW, najprawdopodobniej będziesz musiał je skompilować sam. Nie jest to trudne, ponieważ cały proces komplikacji został zautomatyzowany za pomocą programu użytkownika `make`. Program `make` zostanie omówiony dokładniej w rozdziale 56. „Zarządzanie kodem źródłowym”, ale w tym podrozdziale nie musisz przejmować się szczegółami jego działania.

Najpierw upewnijmy się, że oprogramowanie Apache jest gotowe do zainstalowania. Jeśli pobrałeś je z Internetu, posiadasz prawdopodobnie pojedyńczy, skompresowany plik (o nazwie `apache.tar` i z rozszerzeniem `.Z` lub `.gz`), który należy skopiować do katalogu tymczasowego (o dowolnej nazwie) i rozpakować wydając jedno z poleceń:

```
uncompress apache.tar.Z  
gunzip apache.tar.gz
```

zależnie od rozszerzenia posiadanej pliku (programy do kompresji i dekompresji plików zostały omówione w części drugiej „Poznawanie Linuxa” i części czwartej „Linux dla administratorów”).



Jeśli używasz wersji programu `tar` rozpowszechnianej na licencji GNU (która jest dodawana do większości systemów linuxowych), możesz za jednym zamachem zdekompresować plik i wydobyć pliki z archiwum `tar` wydając polecenie:

```
tar zxvf apache.tar.gz
```

Po zdekompresowaniu plik będzie miał nazwę `apache.tar` (tę samą, którą miał przed dekomprezją, ale bez rozszerzenia `.Z` czy `.gz`; poniżej założymy, że jest to nazwa `apache.tar` – jeśli w Twoim przypadku jest inaczej, powinieneś podstawić odpowiednią nazwę). Plik `tar` zawiera wszystkie pliki wymagane przez Apache – należy je z niego wydobyć wydając polecenie:

```
tar xvf apache.tar
```

Spowoduje ono utworzenie sporej liczby plików, o nazwach składających się w większości z małych liter, choć nazwy kilku z nich, na przykład pliku `README`, składają się z również wielkich liter. W pliku `README` znajdują się wskazówki dotyczące komplikowania serwera Apache po skonfigurowaniu systemu. Prawidłowe informacje konfiguracyjne, dotyczące oprogramowania i sprzętu dostępnego w Twoim systemie, powinny znaleźć się w pliku `Configuration`, który można edytować dowolnym edytorem tekstów zapisującym pliki w standardzie ASCII.

Plik `Configuration` ma dość dziwną składnię, umożliwiającą bardzo dokładne skonfigurowanie systemu; niuanse konfiguracji są jednak zbyt skomplikowane, by wyjaśniać je w rozdziale takim jak ten. Na szczęście większością drobiazgów nigdy nie będziesz musiał się przejmować. Plik `Configuration` jest używany przez skrypt o nazwie `Configure`, który generuje plik `Makefile`, używany z kolei przez program `make` do skompilowania kodu źródłowego. W pliku `Configuration` znajdują się zasadniczo dane czterech typów:

- υ **komentarze** – wszystkie wiersze rozpoczynające się od symbolu #,
- υ **polecenia programu make** – wiersze nie rozpoczynające się niczym szczególnym,
- υ **moduły** – grupy wierszy rozpoczynające się słowem kluczowym `Module`,
- υ **reguły** – wiersze rozpoczynające się słowem `Rule`.



Nie powinieneś edytować pliku `Makefile` bezpośrednio. Wszystkie wprowadzone modyfikacje i tak zostaną utracone po uruchomieniu skryptu `Configure`. Plik `Makefile` jest generowany automatycznie przez skrypt `Configure` na podstawie informacji zawartych w pliku `Configuration`.

Praktycznie we wszystkich przypadkach jedyne modyfikacje pliku `Configuration`, których trzeba dokonać, polegają na usunięciu bądź wstawieniu symbolu komentarza w odpowiednich wierszach. Wpisy dotyczące modułów i reguł można pominąć, chyba że zamierzasz zagłębić się w szczegóły konfiguracji systemu. Jeśli spojrzesz na zawartość pliku `Configuration`, zauważysz, że większość odniesień do sekcji `Module` jest zaznaczona jako komentarz – można usunąć odpowiednie symbole komentarza, dzięki czemu podczas generowania pliku `Makefile` moduły będą również brane pod uwagę. Standartowa wersja serwera Apache dla Linuxa zawiera plik `Configuration`, w którym wszystkie opcje mają prawidłowe wartości – możesz jednak przejrzeć ten plik, aby zorientować się, jakie opcje są dostępne.



Choć usuwanie symboli komentarza poprzedzających większość sekcji pliku `Configuration` nie wymaga żadnego wysiłku, trzy z sekcji powinny pozostać w formie komentarza. Moduł `cern_meta_module` używany jest dla zapewnienia kompatybilności ze starą wersją serwera CERN. Moduł `dld_module` pozwala na obsługę dynamicznego dołączania kodu i nie jest używany w systemie Linux. Trzeci moduł, `msql_auth_module`, umożliwia obsługę dużych plików z hasłami użytkowników za pomocą mechanizmów SQL i jest bezużyteczny, o ile nie jest zainstalowany system Minerva SQL.

Aby uruchomić proces generowania pliku `Makefile` na podstawie danych zapisanych w pliku `Configuration`, należy uruchomić skrypt `Configure` (zauważ, że polecenie to rozpoczyna się od wielkiej litery – musi być wpisane dokładnie tak, jak pojawia się na wydruku zawartości katalogu). W tym celu upewnij się, że jesteś zalogowany jako `root` i po prostu wpisz nazwę skryptu:

```
Configure
```

Na ekranie pojawi się kilka komunikatów, najprawdopodobniej takiej postaci:

```
Using Configuration as config file
Configured for FreeBSD platform
```

Informacje wyświetlane w drugim wierszu mogą być nieco inne, zależnie od wersji systemu Apache. Na koniec należy uruchomić program użytkowy `make` (który wymaga, by zainstalowany był kompilator języka C i kilka programów użytkowych):

```
make
```

Domyślnie, program `make` przetwarza plik o nazwie `Makefile` znajdujący się w katalogu bieżącym. Następnie wyświetlonych zostanie kilka komunikatów generowanych przez kompilator (o ile został on zainstalowany – w przeciwnym przypadku pojawią się komunikaty o błędach) i program `make`, po czym program ten zakończy działanie. Wygenerowany zostanie plik wykonywalny o nazwie `httpd`, który zawiera program rezydentny do obsługi serwera HTTP. Jeśli spróbujesz go uruchomić, prawdopodobnie otrzymasz komunikat, że brakuje jakiegoś pliku – program `httpd` próbuje bowiem znaleźć plik konfiguracyjny (który zwykle nazywa się `httpd.conf`).

Edycja pliku konfiguracyjnego

Po samodzielnym skompilowaniu serwera Apache (albo jeśli otrzymałeś gotową, skompilowaną wersję) należy wprowadzić odpowiednie informacje do pliku konfiguracyjnego, który zwykle nazywa się `httpd.conf`. W katalogu z systemem Apache powinien znajdująć się podkatalog o nazwie `conf`, a w nim trzy pliki konfiguracyjne o nazwach `srm.conf-dist`, `access.conf-dist` i `httpd.conf-dist`. Należy skopiować je, zmieniając nazwy plików w następujący sposób:

```
cp srm.conf-dist srm.conf
cp access.conf-dist access.conf
cp httpd.conf-dist httpd.conf
```

Otwórz plik `httpd.conf` za pomocą dowolnego edytora tekstu ASCII. Plik ten zawiera ogólne informacje o serwerze, takie jak numer portu, którego należy użyć, identyfikator użytkownika, który ma być właścicielem procesu itp. Większość tych informacji nie wymaga korekty, ale jeśli chcesz, możesz dostosować je do własnych wymagań. Następnie otwórz plik `srm.conf`, zawierający dane o drzewie dokumentów dla stron głównych i specjalne instrukcje HTML. Również ten plik nie wymaga modyfikacji. Ostatni z plików konfiguracyjnych, `access.conf`, pozwala zdefiniować podstawowe poziomy dostępu do systemu – możesz dostosować go do własnych wymagań lub pozostawić wartości domyślne.



Serwer Apache dostarczany jest z trzema plikami konfiguracyjnymi, ale wszystkie zawarte w nich dane można umieścić w jednym pliku o nazwie `httpd.conf`. Przy konfigurowaniu systemu łatwiej jest jednak edytować trzy mniejsze pliki. Jeśli chcesz, aby Apache zignorował pliki `access.conf` i `srm.conf`, dodaj do pliku `httpd.conf` następujące polecenia:

```
AccessConfig /dev/null  
ResourceConfig /dev/null  
i usuń pliki access.conf i srm.conf.
```

Aby uruchomić program `httpd` z przygotowanymi plikami konfiguracyjnymi, wydaj polecenie:

```
httpd -f sciezka/httpd.conf
```

`sciezka` to pełna ścieżka dostępu do katalogu, w którym znajduje się plik konfiguracyjny. Opcja `-f` pozwala na użycie pliku konfiguracyjnego o dowolnej nazwie. Powyższe polecenie spowoduje uruchomienie programu rezydentnego `httpd` – możesz sprawdzić poleceniem `ps`, że faktycznie pozostaje on w pamięci.

Opcje serwera Apache `httpd`

Jedną z dostępnych opcji serwera `httpd` w wersji Apache poznałeś przed chwilą. Poza nią istnieje jeszcze tylko kilka innych opcji, oto ich lista:

- d** pozwala podać główny katalog dokumentów,
- f** umożliwia podanie nazwy pliku konfiguracyjnego serwera,
- h** pozwala wyświetlić listę obowiązujących dyrektyw (pochodzących z pliku `Configuration`),
- v** podaje numer wersji,
- X** załącza wyświetlanie informacji przydatnych w czasie uruchamiania systemu.

W nowych wersjach serwera Apache pojawia się zwykle kilka innych opcji, ale większość z nich powoduje tylko drobne zmiany w zachowaniu lub też ułatwia uruchamianie systemu. Pełną listę opcji możesz otrzymać wydając polecenie:

```
httpd -?
```

lub też przeglądając strony `man` czy dokumentację dostępną w węźle `apache.org` w katalogu `/docs`.

Konfigurowanie serwera Apache w niewielkim węźle WWW

Aby nie komplikować niepotrzebnie zagadnienia, założymy że chcesz używać Apache jako serwera WWW dla sieci lokalnej. Komputery w tej sieci będą łączyć się z serwerem za pomocą przeglądarki internetowych. Podany poniżej prosty przykład pokazuje, w jaki sposób można skonfigurować serwer Apache w prostej sieci; można z łatwością zmodyfikować konfigurację tak, by udostępniać strony WWW w sieci Internet.

Rozpoczniemy od skonfigurowania głównego katalogu z dokumentami. Założmy, że serwer ma działać w systemie o nazwie `darkstar`. Według konwencji używanej przez Apache komputer ten nazywany będzie `site.darkstar`. Pliki konfiguracyjne znajdą się w podkatalogu `conf` katalogu `site.darkstar` (na przykład `/usr/web/site.darkstar/conf`).

Katalog główny dokumentów można zdefiniować za pomocą opcji `-d` polecenia `httpd`:

```
httpd -d /usr/web/site.darkstar
```

Wszystkie dokumenty udostępniane w węźle WWW powinny znajdować się w katalogu głównym dokumentów lub w którymś z jego podkatalogów – w dokumentach HTML muszą wówczas znaleźć się odpowiednie odnośniki.

Możliwe jest również dostosowanie wielu innych aspektów działania serwera Apache, ale jest to proces dość czasochłonny i jego wyjaśnienie zajęłoby pewnie ze sto stron. Jeśli chcesz dowiedzieć się więcej o systemie Apache, zapoznaj się z dokumentacją dostępną w węźle apache.org albo zaopatrz się w jedną z książek poświęconych temu tematowi.

Podsumowanie

Utworzenie własnej strony głównej wymaga użycia jednego z narzędzi generujących kod w języku HTML albo napisania jej samodzielnie za pomocą dowolnego edytora tekstu. Omówienie języka HTML wykracza poza zakres tej książki, ale dostępnych jest wiele innych publikacji na ten temat. Język HTML jest łatwy do opanowania. Posiadając informacje podane w tym rozdziale, powinieneś być w stanie założyć własną stronę główną i pozwolić przeglądać ją wszystkim użytkownikom Internetu.

Zależnie od tego, jakich informacji poszukujesz, możesz teraz przejść do innych rozdziałów.

Pisanie skryptów CGI dla stron WWW omówione jest w rozdziale 52. „Skrypty CGI”.

Język HTML, używany do tworzenia stron WWW, przedstawiamy w rozdziale 53. „Podstawy języka HTML”.

Java, język przeznaczony głównie do rozszerzania funkcjonalności stron WWW, omówiony jest w rozdziale 54. „Java i JavaScript”.

Rozdział 52.

Skrypty CGI

Tim Parker

W tym rozdziale:

- υ Co to jest CGI?
- υ GCI i HTML
- υ CGI i Perl

Przy omawianiu zagadnień związanych ze stronami WWW często używa się terminu GCI (ang. Common Gateway Interface, wspólny interfejs bramkowy). Na pewno nie uda się nam w tym rozdziale wyjaśnić wszystkich zagadnień związanych z tym interfejsem, postaramy się jednak przedstawić jego zastosowania i przybliżyć sposoby użycia.

Jeśli zaangażujesz się w tworzenie bardziej złożonych stron WWW (językom HTML i Java przyjrzymy się w jednym z następnych rozdziałów), w końcu na pewno zechcesz skorzystać z możliwości oferowanych przez mechanizmy CGI – pozwalają one w znacznym stopniu rozszerzyć funkcjonalność stron WWW. Dlatego przyjrzymy się temu interfejsowi nieco bardziej szczegółowo.

Co to jest CGI?

Rozwiniecie skrótu CGI – Common Gate Interface, wspólny interfejs bramkowy – nie ułatwia zrozumienia, do czego w rzeczywistości służy ten interfejs. Jego nazwa może być nieco myląca. Zasadniczo CGI pozwala na uruchamianie aplikacji za pośrednictwem stron WWW. Można na przykład na stronie WWW umieścić przycisk uruchamiający program, którego zadaniem jest wyświetlanie danych statystycznych o częstotliwości odwiedzania Twojego węzła WWW. Po kliknięciu na tym przycisku zostanie uruchomiony program wykonujący odpowiednie obliczenia. CGI jest interfejsem pomiędzy takim programem a kodem w języku HTML; pozwala on na przesyłanie danych w obie strony pomiędzy kodem HTML i aplikacjami, nie muszącymi wcale stanowić części strony WWW.

CGI wykonuje jeszcze inne operacje, ale są one zwykle ukryte przed użytkownikiem. Programy CGI nie muszą w zasadzie być uruchamiane za pośrednictwem stron WWW, choć rzadko uruchamia się je w inny sposób, głównie ze względu na fakt, że do prawidłowego działania wymagają one dość specyficznego i dość trudnego do emulacji środowiska.

Co to oznacza w praktyce? Kiedy uruchamiasz stronę WWW napisaną w języku HTML, serwer ustawia wartości wielu zmiennych środowiskowych. Są one używane do kontrolowania programów i komunikowania się z nimi, jak również do wielu innych celów. Kiedy użytkownik kliką na przycisku uruchamiającym program zewnętrzny, wartości tych zmiennych środowiskowych są wykorzystywane do przesyłania odpowiednich danych (takich jak na przykład identyfikator użytkownika, który uruchomił program, czas systemowy itp.). Uruchamiany program odsyła przetworzone informacje z powrotem do serwera WWW również za pośrednictwem odpowiednich zmiennych środowiskowych.

Mówiąc o programowaniu skryptów CGI mamy na myśli pisanie programów wykorzystujących interfejs pomiędzy stronami w języku HTML i innymi aplikacjami. CGI jest właśnie tym interfejsem.

Można by zapytać: no i co z tego? Możliwości oferowane przez język HTML są dość ograniczone. CGI pozwala obejść te ograniczenia i umieszczać na stronach WWW obiekty zachowujące się w zasadzie w dowolny sposób. Jeśli na przykład chcesz na stronie WWW umożliwić obliczenie parametrów statystycznych danych dostarczanych przez użytkownika, bardzo wygodnie będzie zrobić to za pomocą interfejsu CGI, przekazując wprowadzone dane jako wartość odpowiednich zmiennych środowiskowych i zwracając gotowe wyniki do serwera WWW. Nawet na najprostszej stronie WWW można zamieścić mnóstwo elementów, które mogłyby zostać wykonane za pomocą interfejsu CGI – właśnie dlatego jest on tak popularny.

Mechanizmy obsługi CGI są zwykle wbudowane w serwery WWW, choć nie muszą one istnieć we wszystkich programach tego typu. Prawie wszystkie dostępne obecnie serwery WWW (za wyjątkiem bardzo wczesnych wersji i kilku wersji okrojonych), na przykład oferowane przez NCSA, Netscape czy CERN, obsługują interfejs CGI.

CGI i HTML

Aby uruchomić aplikację CGI ze strony WWW, musisz wysłać odpowiednie żądanie do serwera WWW. Żądanie to realizowane jest za pośrednictwem odpowiedniej metody, odpowiadającej za wywoływanie programów CGI (przez metodę rozumie się odpowiednią procedurę lub funkcję). Protokół HTTP (*HyperText Markup Language*) zawiera wiele metod, a to, którą z nich należy wywołać do uruchomienia programu CGI zależy od typu danych, jakie mają zostać przekazane do tegoż programu. Do tego zagadnienia wróćmy za chwilę, najpierw pokażemy, w jaki sposób można osadzić kod CGI w dokumentach HTML.

Język HTML generalnie opiera się na zastosowaniu znaczników (ang. *tags*) – dokładniej problem ten omówimy w następnym rozdziale. Do uruchomienia programu CGI również używa się jednego ze znaczników, podając nazwę programu, który ma zostać wykonany, i tekst, który zostanie wyświetlony na stronie WWW po przetworzeniu kodu w języku HTML. Przykładowo, poniższy znacznik spowoduje wyświetlenie na stronie WWW tekstu *Kliknij tu, aby wyświetlić informacje statystyczne*:

```
<a href="licz_stat"> Kliknij tu, aby wyświetlić informacje statystyczne </a>
```

Kiedy użytkownik kliknie na wyświetlonym tekście, uruchomiony zostanie program o nazwie *licz_stat* (znaczniki *<a>* i ** definiują zakotwiczenie – ang. *anchor* – tekst zawarty pomiędzy nimi jest traktowany jako odnośnik do jakiegoś innego obiektu; rozmieszczenie tego typu znaczników decyduje o ostatecznej formie strony WWW).

Jak przekonasz się czytając następny rozdział, poświęcony językowi HTML, można w ten sposób uruchomić również program dostępny w innym systemie, podając jego nazwę domenową. Przykładowo, poniższy fragment kodu w języku HTML definiuje odnośnik, po kliknięciu którego uruchamiany jest skrypt zapisany w systemie www.tpci.com bez względu na to, w jakim systemie zapisana jest sama strona WWW:

```
<a href="http://www.tpci.com/stats.cgi"> Wyświetl informacje statystyczne </a>
```

Jeśli użytkownik wybierze tekst *Wyświetl informacje statystyczne*, zostanie nawiązane połączenie z serwerem www.tpci.com i uruchomiony zostanie odpowiedni program, nawet jeśli serwer ten znajduje się w innym kraju (pod warunkiem, że da się nawiązać połączenie).

Do uruchamiania programów CGI zwykle wykorzystuje się trzy metody: *GET*, *HEAD* i *POST* (wszystkie one są elementami języka HTTP). Każdej z tych metod używa się w nieco inny sposób, więc przyjrzymy się im z osobna.

Metoda *GET* używana jest wówczas, gdy aplikacja CGI spodziewa się przesłania danych poprzez zmienną środowiskową *QUERY_STRING*. Aplikacja odczytuje i dekoduje wartość tej zmiennej, interpretując ją odpowiednio według potrzeb. Metoda *GET* jest używana zwykle wtedy, gdy aplikacja powinna otrzymać jakieś dane, ale nie musi zwracać żadnych wyników.

Metoda *HEAD* jest bardzo podobna do metody *GET*, z tym że serwer transmituje do klienta tylko nagłówki HTTP. Informacje wchodzące w skład głównej części wiadomości są ignorowane. Taka metoda może przydać się na przykład do przesyłania identyfikatora użytkownika.

Metoda *POST* jest o wiele bardziej elastyczna i pozwala użyć standardowego urządzenia wejściowego (*stdin*) do odbierania danych. Wartość zmiennej środowiskowej o nazwie *CONTENT_LENGTH* pozwala aplikacji zorientować się, jakiej liczby danych ma się spodziewać. Metoda *POST* została opracowana po to, by umożliwić zmianę zachowania się serwera, ale wielu programistów używa jej do wszelkich zadań, ponieważ pozwala ona uniknąć obcinania adresów URL, co może zdarzyć się podczas wykorzystywania metody *GET*.

Interfejs CGI korzysta z wielu zmiennych środowiskowych, są one omówione ze szczególnymi w każdej książce poświęconej programowaniu z wykorzystaniem tego interfejsu. Opisywanie tych zmiennych w naszej książce bez podania przykładów ich zastosowania byłoby stratą czasu.

CGI i Perl

Jeśli zainteresuje Cię programowanie z wykorzystaniem CGI, szybko zorientujesz się, że większość tego typu aplikacji tworzona jest za pomocą języka Perl (który został omówiony w rozdziale 28 „Perl”). Takie programy mogą być pisane w dowolnym języku (wielu projektantów stron WWW tworzy programy CGI w takich językach, jak np. C, C++ czy Visual Basic, ponieważ języki te są im lepiej znane), ale Perl zdaje się być ulubionym językiem twórców stron WWW pracujących w systemach UNIX-owych.

Przyczyna takiej popularności języka Perl staje się oczywista w momencie, gdy nauczysz się nim posługiwać. Jest on językiem bardzo prostym, a jednocześnie potężnym i wydajnym. Jest również przenośny, dzięki czemu programy używające interfejsu CGI mogą praktycznie bez przeróbek działać w różnych systemach operacyjnych.

Na stronach WWW można znaleźć mnóstwo skryptów w języku Perl. Wpisanie odpowiedniego zapytania w którejś z wyszukiwarek, na przykład Altavista, powoduje znalezienie setek dokumentów zawierających przykładowe skrypty, czekające tylko na to, aby je załadował i przestudiował. Jednym z najczęściej używanych skryptów Perl jest skrypt obsługujący książkę gości (ang. *GuestBook*). Pozwala on użytkownikowi na wpisanie się do książki gości i pozostawienie komentarza dotyczącego odwiedzanej strony WWW. Zwykle skrypt taki rejestruje dane o imieniu i nazwisku użytkownika, jego adres e-mail, pochodzenie (zwykle tylko miasto czy nawet kraj) i komentarze. Jest to dobry sposób na zorientowanie się, co też użytkownicy sądzą o udostępnianych przez Ciebie stronach WWW.

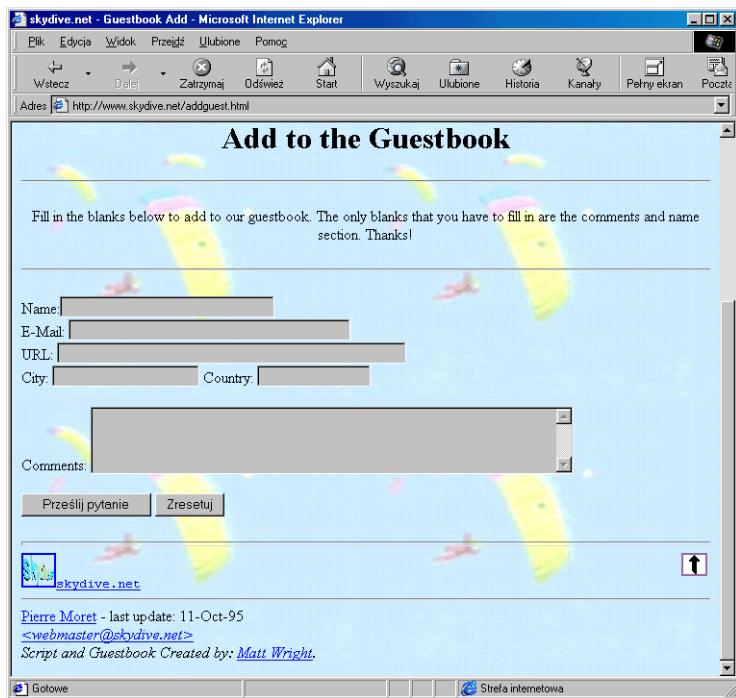
Kiedy program obsługujący książkę gości zostanie uruchomiony, wyświetla na ekranie formularz, który należy wypełnić, a następnie aktualnia bazę danych serwera w oparciu o wprowadzone dane. W Sieci dostępnych jest bardzo wiele wersji tego programu, na rysunku 52.1 przedstawiamy wyniki działania jednej z nich.

Każdy ze skryptów obsługujący książki gości nieco inaczej prezentuje się w oknie przeglądarki, ale zazwyczaj ma on postać zbliżoną do tego przedstawionego na rysunku 52.1. Informacje wprowadzone przez użytkownika są zapisywane w bazie danych po stronie serwera, dzięki czemu administrator systemu ma do nich dostęp.

Rysunek 52.2 przedstawia inną stronę WWW, zawierającą menu pozwalające na uruchomienie kilku przykładowych skryptów CGI. Uruchomienie skryptu dostarczającego informacji o domenie (*domain name lookup*) powoduje przesłanie serii standardowych żądań HTTP pomiędzy serwerem i klientem i ostatecznie wyświetlenie danych przedstawionych na rysunku 52.3. Jak widać, dane te przedstawione są za pomocą czcionki o standardowych parametrach, ponieważ skrypt nie stara się sformatować ich w żaden określony sposób. Takie rozwiązanie jest wystarczające w przypadku prostych aplikacji CGI.

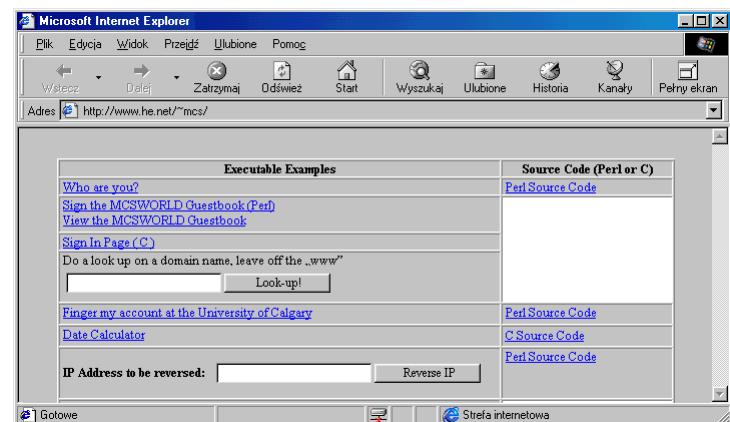
Rysunek 52.1.

Przykładowy program obsługi ksiązki gości pobiera dane od użytkownika i zapisuje je w bazie danych



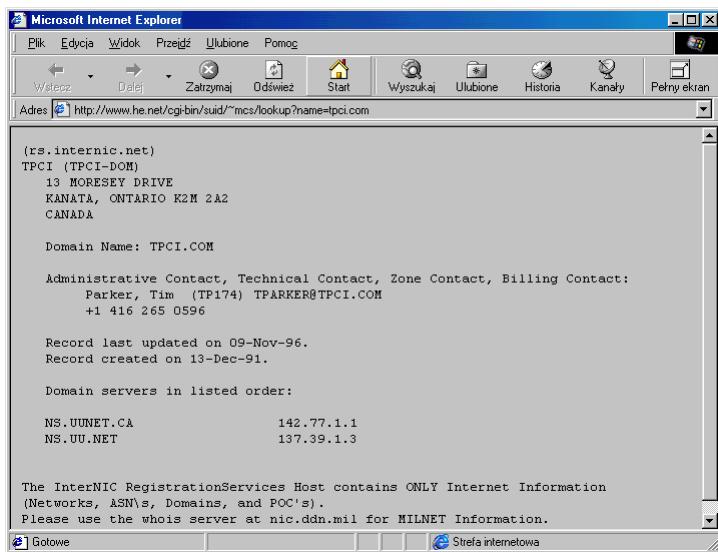
Rysunek 52.2.

Strona WWW zawierająca odwołania do kilku przykładowych aplikacji CGI w języku Perl i C



Tworzenie skryptów CGI w języku Perl nie jest skomplikowane. Przykładowy skrypt (wywoływany po wybraniu tekstu Who Are You?) zamieszczony na stronie przedstawionej na rysunku 52.2 przesyła informację za pośrednictwem zgłoszenia HTTP. Kod tego skryptu w języku Perl wyświetlony za pomocą przeglądarki Netscape przedstawiony jest na rysunku 52.4. Jak widać, składa się on zaledwie z kilku wierszy. Każdy, kto potrafi programować w języku Perl, jest w stanie szybko stworzyć tego typu aplikację CGI.

Rysunek 52.3.
Rezultat wykonania skryptu CGI napisanego w języku Perl, dostarczającego informacje o domenie



Rysunek 52.4.
Kod źródłowy aplikacji wywoływanej po uruchomieniu programu Who Are You? (w języku Perl)

```
#!/usr/bin/perl

$remote_address = $ENV{'REMOTE_ADDR'};
$remote_host = $ENV{'REMOTE_HOST'};
$referral_address = $ENV{'HTTP_REFERER'};

@subnet_numbers = split ('./.', $remote_address);
$packed_address = pack ("C4", @subnet_numbers);
($remote_host) = gethostbyaddr ($packed_address, 2)

print "Content-type: text/html", "\n\n";
print "<pre>";
print "Remote Address: ", $remote_address, "\n";
print "Remote Host: ", $remote_host, "\n";
print "You came from: ", $referral_address, "\n\n";
print "date", "\n";
print "This server: ", 'uptime', "\n";
print "</pre>";

exit(0);
```

Podsumowanie

Używanie interfejsu CGI jest bardzo proste, szczególnie w programach w języku Perl, i może dać Twoim aplikacjom zupełnie nowe możliwości. Kiedy będziesz już potrafił tworzyć własne strony w języku HTML (tego zagadnienia nie omawiamy w tej książce ze względu na ograniczoną ilość miejsca), możesz spróbować stworzyć kilka aplikacji

wykorzystujących interfejs CGI, które ożywią strony WWW. Kilka języków programowania omówiliśmy we wcześniejszych rozdziałach, które być może pominąłeś.

Programowanie w języku Perl, idealnie nadającym się do tworzenia skryptów CGI, omówione zostało w rozdziale 28. „Perl”.

Jeśli chcesz do tworzenia aplikacji wykorzystujących interfejs CGI używać języka C, przeczytaj rozdział 26. „Programowanie w języku C”.

Z rozdziału 45. „Kopie zapasowe” dowiesz się, w jaki sposób należy wykonywać kopie zapasowe systemu plików tak, aby stworzone przez Ciebie skrypty CGI były bezpieczne.

Rozdział 53.

Podstawy języka HTML

Tim Parker

W tym rozdziale:

- υ Programy do tworzenia dokumentów HTML
- υ Konserwacja dokumentów HTML
- υ Podstawy języka HTML

Utrzymywanie serwera WWW w sytuacji, gdy nie udostępniamy żadnych stron WWW, nie ma większego sensu, więc trzeba skonfigurować go tak, by inni użytkownicy sieci mieli możliwość przeglądania naszych zasobów. Do tego celu używane są adresy w formacie URL (ang. *Uniform Resource Locator*, jednolity lokator zasobów) wskazujące na poszczególne pliki. Użytkownik chcący uzyskać dostęp do danego pliku (zasobu) musi znać tylko jego URL. Jeśli nie prowadzisz zbyt rozbudowanej strony głównej, sprawia jest prosta: każdy, kto łączy się z Twoim systemem za pomocą przeglądarki WWW, zobaczy stronę o nazwie `index.html`, zapisaną w głównym katalogu z dokumentami HTML, lub też informacje o zawartości tegoż katalogu w przypadku, gdy plik o nazwie `index.html` nie istnieje. Nie jest to jednak zbyt atrakcyjne rozwiązanie, dlatego wielu użytkowników chce udostępniać bardziej wyszukane strony główne. Jeśli chcesz utworzyć własną stronę WWW, musisz nauczyć się posługiwać językiem HTML (HyperText Markup Language).

Strona główna pełni zwykle funkcję głównego menu. Możliwe, że niektórzy użytkownicy nigdy jej nie zobaczą, wchodząc bezpośrednio do podkatalogów systemu lub też uzykując dostęp do plików za pośrednictwem hiperłączy umieszczonego na jakichś innych stronach WWW. Mimo tego, wielu użytkowników woli rozpocząć przeglądanie zasobów od strony głównej. Strona główna ma zwykle nazwę `index.html` i znajduje się w głównym katalogu z dokumentami HTML.

Tworzenie dokumentu w języku HTML nie jest zbyt trudne. Język ten opiera się na znacznikach (ang. *tags*), określających w jaki sposób mają być traktowane poszczególne fragmenty dokumentu (na przykład jako nagłówki, jako główny tekst dokumentu, jako wykres itp.). Najtrudniejszym aspektem języka HTML jest prawidłowe umieszczanie wszystkich tych znaczników. Jego składnia jest bardzo ścisła – jeśli chcesz uzyskać dobre rezultaty, musisz się jej dokładnie trzymać.

Kiedyś wszystkie dokumenty HTML tworzone były za pomocą prostych edytorów tekstów. W miarę, jak strony WWW zdobywały coraz większą popularność, zaczęły powstawać edytory przystosowane do języka HTML, potrafiące rozpoznać i odpowiednio obsługiwać jego składnię. Wkrótce powstały dziesiątki edytorów HTML, filtrów i innych programów użytkowych wspierających i ułatwiających projektowanie stron WWW. Edytory HTML dostępne są dla wielu systemów operacyjnych.

Programy do tworzenia dokumentów HTML

Dokumenty HTML można tworzyć na wiele sposobów. Można używać do tego celu zwykłych edytorów tekstów, można również użyć narzędzi przeznaczonych specjalnie do pracy z tym językiem. Wybór jednej z tych metod zależy od indywidualnych preferencji. Nie bez znaczenia jest również doświadczenie w tworzeniu stron WWW i to, czy akurat masz dostęp do odpowiednich narzędzi. Główną zaletą wspomnianych programów narzędziowych jest fakt, że potrafią one sprawdzić poprawność składniową dokumentu HTML. Są również zwykle nieco łatwiejsze w użyciu niż edytory nie dostosowane do obsługi języka HTML. Z drugiej strony, programiści mające większe doświadczenie w tworzeniu stron WWW często używają edytora tekstów, do którego są przyzwyczajeni, posługując się co najwyżej filtrem lub programem kontrolującym poprawność składniową dokumentu.



Jeśli szukasz edytora lub filtra HTML, powinieneś odwiedzić węzeł CERN. Zajrzyj pod adres <http://info.cern.ch/WWW/Tools> i obejrzyj zawartość pliku `Overview.html`. Warto również zajrzeć do węzła NCSA, w którym pod adresem [http://www.ncsa.uiuc.edu/SDG/ Software/Mosaic/Docs](http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs) znajduje się dokument `faq-software.html`, zawierający aktualną listę dostępnych programów tego typu.

Do tworzenia stron WWW można użyć dowolnego edytora tekstów, nie wyłączając takich programów, jak `vi` czy `emacs`. Każdy edytor pozwala na wstawienie do tekstu dowolnego znacznika HTML, ale znaczniki te są traktowane na równi z pozostałym tekstem. Proste edytory nie potrafią sprawdzić, czy w dokumencie nie występują jakieś błędy składniowe, ponieważ nie „rozumieją” języka HTML. Dla edytora `emacs` i niektórych innych edytorów dostępne są rozszerzenia umożliwiające kontrolę składni w oparciu o proste szablony, ale nie wymuszają one poprawnej składni.

Jeśli chcesz używać jednego z prostych edytorów tekstów, musisz bardzo uważnie sprawdzać składnię swojego dokumentu. Najłatwiej robić to importując dokument do jednego z edytorów HTML umożliwiającego sprawdzanie składni. Można również uruchomić przeglądarkę WWW i sprawdzić, czy dokument przedstawia się zgodnie z oczekiwaniami.

Programy narzędziowe przeznaczone specjalnie do tworzenia dokumentów HTML pod kontrolą systemu Linux są dostępne w różnych zakątkach Internetu, ale nie są one aż tak popularne, jak analogiczne aplikacje przeznaczone do pracy w systemie DOS i Windows (dla tych systemów dostępne są dziesiątki programów tego typu). Jeśli w Twoim systemie zainstalowany jest DOS czy Windows, możesz po prostu wykorzystać odpowiednie oprogramowanie, a następnie przenieść stworzoną stronę WWW do systemu linuxowego. Dla systemu Windows dostępnych jest kilka popularnych edytorów HTML, takich jak HTML Assistant, HTMLed czy HoTMetaL (a także polski Pajaczek – *przyp. tłum.*). Niektóre z tych programów, na przykład HoTMetaL, działają również w systemie X, czyli pod kontrolą Linuxa. Programy tego typu albo opierają się w całości na interfejsie graficznym i technologii przeciagnij-i-upuść, albo bazują na edycji pliku źródłowego w trybie tekstowym. Prawie wszystkie jednak posiadają mechanizmy umożliwiające kontrolę składniową tworzonego dokumentu.

Tworzenie stron WWW w systemie Windows

Jak wspomnialiśmy wcześniej, programów przeznaczonych do tworzenia dokumentów HTML pracujących pod kontrolą Windows jest o wiele więcej niż linuxowych. Jeśli posiadasz oba te systemy, możesz z łatwością tworzyć strony WWW pod Windows, a następnie przenosić je do systemu Linux. Narzędzia do tworzenia dokumentów HTML w systemach DOS i Windows można podzielić na trzy kategorie: programy typu WYSIWYG (ang. *What You See Is What You Get* – To, co widzisz, jest tym, co otrzymasz), pozwalające na bezpośrednią obserwację efektów pracy, programy nie oferujące takiej możliwości oraz filtry umożliwiające konwersję do formatu HTML dokumentów innego typu (na przykład tworzonych przez programy Word for Windows czy WordPerfect).

Prawdopodobnie najpopularniejszym edytorem HTML dla systemu Windows jest program HTML Assistant. Jest to edytor tekstów wzbogacony o kilka dodatkowych mechanizmów i interfejs graficzny pozwalający łatwo wstawiać najpopularniejsze znaczniki języka HTML. Możliwa jest także równoczesna edycja kilku dokumentów, jak również przesunięcie i kopiowanie fragmentów tekstu pomiędzy nimi.



Jeśli chcesz załadować program HTML Assistant za pomocą FTP, zaloguj się jako anonimowy użytkownik do systemu `ftp.cuhk.hk` i poszukaj w katalogu `/pub/www/windows/util` pliku o nazwie `htmlasst.zip`.

Program HTML Assistant ma jedno dość poważne ograniczenie: rozmiar tworzonych za jego pomocą plików nie może przekraczać 32 kB. W związku z tym nie nadaje się on do tworzenia większych stron WWW, chyba że podzielone zostaną one na kilka części, a następnie połączone za pomocą innego edytora. Przy próbie edycji pliku o rozmiarze większym niż 32 kB lub utworzenia pliku zawierającego większą ilość danych wyświetlany jest komunikat `Out of Memory` (brak pamięci) i może się zdarzyć, że efekty pracy zostaną utracone.

HTML Assistant posiada dobrej jakości pomoc dostępną podczas pracy, a także inne przydatne możliwości, takie jak na przykład możliwość automatycznego ładowania edytowanego dokumentu do przeglądarki, dzięki czemu można na bieżąco śledzić efekty swojej pracy. Przełączanie się pomiędzy edytowanym kodem a wynikiem jego sformatowania nie stanowi żadnego problemu, co pozwala obejść niektóre z niedogodności edytowania dokumentu w trybie tekstowym.

HTML Assistant ułatwia również tworzenie odnośników hipertekstowych, przechowując listę ostatnio używanych adresów w formacie URL i pozwalając na łatwe przechodzenie pomiędzy nimi.

Innym programem służącym do tworzenia stron WWW jest HTMLEd, również działający pod kontrolą systemu Microsoft Windows. Jest to szybki edytor, obsługujący znaki międzynarodowe i potrafiący zapisywać i odczytywać pliki zarówno w formacie DOS-owym, jak i UNIX-owym. Użyteczna jest możliwość konwersji adresów URL przechowywanych w pliku MOSAIC.INI do dokumentu HTML z zachowaniem jego oryginalnej struktury. Względnie prosta (w porównaniu z edytorem HTML Assistant) jest również konwersja plików tekstowych do formatu HTML.



Kopię programu HTMLEd można załadować za pośrednictwem anonimowego FTP z serwera [ftp.cuhk.hk](ftp://ftp.cuhk.hk) - program ten znajduje się w pliku o nazwie `htmed10.zip` (lub nieco innej, zależnie od wersji programu) w katalogu `/pub/www/windows/util`.

Program HTMLEd nie jest obsługiwany za pomocą wielowierszowego paska z przyciskami odpowiadającymi poszczególnym znacznikom HTML, jak miało to miejsce w przypadku programu HTML Assistant. Zamiast tego poszczególne znaczniki wybiera się z rozwijanego menu lub paska narzędzi, który można również dostosować do własnych potrzeb. Można również utworzyć kilka pasków narzędzi, osadzając w nich poszczególne znaczniki reprezentowane przez ikony. Paski narzędzi mogą być swobodnie rozmieszczane w obrębie okna aplikacji. Aktualna wersja programu HTMLEd nie zawiera pomocy dostępnej podczas pracy, ale w przyszłych wersjach pomoc taka ma zostać wprowadzona.

HTMLEd dość sprytnie wstawia znaczniki do dokumentu, zapobiegając umieszczeniu wielu znaczników w jednym wierszu tam, gdzie nie jest to dozwolone. Pozwala to unikać najczęściej występujących błędów składniowych dokumentów HTML. Podobnie jak program HTML Assistant, HTMLEd udostępnia przycisk pozwalający obejrzeć efekty pracy bezpośrednio w przeglądarce kodu HTML.

Program HoTMetaL firmy Softquad pracuje w trybie zbliżonym do WYSIWYG, w związku z czym jest on prawdopodobnie najatrakcyjniejszy z punktu widzenia użytkowników nie mających doświadczenia w tworzeniu dokumentów HTML. HoTMetaL pracuje pod kontrolą systemu Windows i w niektórych systemach UNIX-owych. Wersja o nieco ograniczonej funkcjonalności (ale wciąż o całkiem sporych możliwościach) jest dostępna za darmo, natomiast pełną wersję komercyjną o nazwie HoTMetaL Pro można zakupić od firmy Softquad.



Kopię programu HoTMetaL można załadować w większości węzłów rozprowadzających oprogramowanie użytkowe, takich jak NCSA czy CERN. Zajrzyj na przykład na stronę <http://info.cern.ch/hypertext/www/Tools/HoTMetaL.html>

HoTMetaL to edytor zintegrowany z przeglądarką WWW. Nie działa on w pełni w trybie WYSIWYG, ale pozwala dobrze ocenić widok końcowy tworzonego dokumentu. HoTMetaL prawie że wymusza użycie prawidłowej składni języka HTML, sprawdzając czy na końcu i początku dokumentu występują odpowiednie znaczniki i pozwalając wstawiać znaczniki tylko tam, gdzie są one dozwolone (pozwala to oszczędzić sporo czasu nowym użytkownikom języka HTML).

HoTMetaL potrafi również sprawdzić składnię dokumentu i poinformować o występujących problemach ze znacznikami (ta możliwość przydaje się szczególnie przy importowaniu dokumentów stworzonych za pomocą innych edytorów, ponieważ HoTMetaL po prostu nie pozwoli na wstawienie nieprawidłowego znacznika). Niestety, sygnalizacja błędów nie działa doskonale – nie zawsze wiersz, w którym wskazywane jest wystąpienie błędu, jest faktycznie tym powodującym problemem.

HoTMetaL ma również kilka innych wad. Obrazki GIF wstawiane do dokumentu nie są widoczne w oknie tego programu (aby je obejrzeć, trzeba uruchomić przeglądarkę stron WWW). Poza tym niektóre dokumenty nie odpowiadające dokładnie składni języka HTML nie mogą zostać załadowane do edytora, co może być problemem w przypadku modyfikowania starszych dokumentów HTML.

Rozwiązaniem alternatywnym do zastosowania dedykowanego edytora HTML jest wzbogacenie edytora pracującego w trybie WYSIWYG o możliwość obsługi kodu HTML. Najczęściej tego typu rozwiązania używane są z takimi programami, jak Word for Windows, WordPerfect czy Word for DOS. Dostępnych jest kilka takich rozszerzeń, różniących się swoją złożonością. Większość z nich działa w systemie Windows, choć kilka przeniesiono również na platformy linuxowe.

Główną zaletą użycia tego typu rozwiązań jest fakt, że przy tworzeniu dokumentów HTML można posługiwać się ulubionym i dobrze znanym edytorem korzystając z oferowanego przez niego trybu WYSIWYG. Choć programy te nie są w stanie przedstawić dokumentu dokładnie tak, jak pojawi się on w przeglądarce WWW, mogą zrobić to na tyle precyzyjnie, by umożliwić uniknięcie większości poważniejszych problemów.

Dla programu Microsoft Word for Windows dostępny jest szablon o nazwie CU_HTML, który umożliwia tworzenie stron WWW niemalże w trybie WYSIWYG. Pod względem wyglądu CU_HTML nie różni się zasadniczo od programu Word w zwykłej postaci, dodając tylko nowy pasek narzędzi i pozycję rozwijanego menu. CU_HTML udostępnia zestaw predefiniowanych stylów i pasek narzędzi umożliwiający łatwy dostęp do najczęściej wykorzystywanych funkcji. Upraszczając również wstawianie odnośników hipertekstowych i inne zadania spędzające sen z powiek nowym użytkownikom języka HTML. W wielu przypadkach wykorzystywane są odpowiednie okna dialogowe, znacznie ułatwiające obsługę programu.

Najważniejszą wadą szablonu CU_HTML jest fakt, że nie umożliwia on edycji istniejących dokumentów HTML nie zapisanych w formacie programu Word. Przy tworzeniu dokumentu HTML za pomocą szablonu CU_HTML zapisywane są dwie wersje dokumentu, jedna w formacie HTML, a druga w formacie .DOC. Dokument nie może być edytowany, jeśli nie są dostępne obie wersje pliku. Istniejący dokument można zainportować, ale niestety powoduje to utratę wszystkich znaczników.

ANT_HTML to podobny do CU_HTML szablon rozszerzający funkcjonalność programu Word for Windows. Pod niektórymi względami szablon ten jest lepszy od CU_HTML, pod innymi zaś nieco mu ustępuje. Zawiera lepszy system pomocy i wygodniejszy pasik narzędzi, potrafi również automatycznie otwierać i zamazywać znaczniki w zależności od potrzeb.

Tworzenie dokumentów HTML w systemie Linux

Wśród użytkowników Linuxa popularność zdobył program narzędziowy o nazwie tkWWW, oparty na języku `Tcl` i jego rozszerzeniu pozwalającym na korzystanie z interfejsu X o nazwie `Tk`. Jest to kombinacja przeglądarki WWW i edytora HTML pracującego w trybie zbliżonym do WYSIWYG. Choć program ten pierwotnie przeznaczony był tylko dla systemów UNIX-owych, powstały jego wersje dla innych systemów operacyjnych, w tym Windows i Macintosh.



Program `tkWWW` można załadować poprzez anonimowe FTP z węzła `ftp.aud.alcatel.com` - znajduje się on w katalogu `/tcl`. `Tcl` i `Tk` można załadować z różnych serwerów, w zależności od systemu operacyjnego, ale języki te są dołączone do większości standardowych dystrybucji Linuxa. Jeśli chcesz załadować wersje tych języków z sieci Internet, zacznij poszukiwania od anonimowego węzła `FTP ftp.aud.alcatel.com` - zajrzyj do katalogu `tcl/exten-sion`. Warto również zajrzeć na oficjalną stronę poświęconą językom `Tcl/Tk` pod adresem `http://www.sunscript.com`

Podczas tworzenia strony WWW w trybie edycji programu `tkWWW` można w każdej chwili przełączyć się w tryb przeglądarki i sprawdzić, czy dokument jest sformatowany prawidłowo. Również w trybie edycji większość formatowania jest odpowiednio odwzorowywana na ekranie, ale widoczne są także znaczniki. Takie rozwiązanie umożliwia szybkie opracowywanie nowych stron WWW.

Niestety program `tkWWW` korzysta z interfejsu udostępnianego przez system Tk, co powoduje że na przeciwnych komputerach działa on dość wolno. Poza tym przeglądarka wbudowana w `tkWWW` również nie jest najwyższych lotów, opiera się na standardowych ramkach oferowanych przez Tk. Mimo tego `tkWWW` jest bardzo atrakcyjnym narzędziem do tworzenia stron WWW – jeśli nie w całości, to przynajmniej w ogólnym zarysie, szczególnie wtedy, gdy znasz język `Tcl`.

Innym rozwiązaniem jest użycie filtra HTML. Program tego typu pozwala utworzyć dokument HTML w oparciu o dokument pochodzący z jakiegoś edytora tekstów (nie wyłączając oczywiście zwykłych plików ASCII). Filtry są przydatne w przypadku, gdy używasz edytora potrafiącego formatować tekst w odpowiedni sposób, jak na przykład Microsoft Word.

Filtry HTML są szczególnie użyteczne, jeśli zamierzasz nadal pracować w swoim edytorze tekstów, a potrzebujesz tylko narzędzia umożliwiającego konwertowanie dokumentów do formatu HTML. Są one dość szybkie i łatwe w obsłudze – zwykle wymagają tylko podania nazwy dokumentu, na podstawie którego ma być wygenerowany plik w formacie HTML. Poziom sprawdzania poprawności składniowej i raportowania zależy od konkretnej implementacji filtra.

Filtry HTML są dostępne dla większości typów dokumentów, a wiele z nich działa bezpośrednio pod kontrolą Linuxa; czasem są one również rozpowszechniane w postaci kodu źródłowego, dającego się bez żadnych modyfikacji skompilować w systemie Linux. Dokumenty programu Word w wersji dla Windows i DOS można przełożyć na język HTML za pomocą omówionych wcześniej szablonów CU_HTML i ANT_HTML. Pojawiają się również samodzielne programy do konwersji dokumentów .DOC. Program o nazwie WPTOHTML pozwala utworzyć dokument HTML na podstawie dokumentu w formacie programu Word Perfect. Jest on w zasadzie zestawem makropoleceń programu Word Perfect w wersji 5.1 i 6.0. Filtra tego można również używać z innymi formatami plików, które mogą być zainportowane do edytora Word Perfect.

Pliki LaTeX i TeX mogą zostać przełożone na język HTML za pomocą kilku różnych programów. Dla systemu Linux dostępnych jest sporo tego typu aplikacji, na przykład LATEXTOHTML, który potrafi obsłużyć nawet równania znajdujące się w wierszu tekstu i odnośniki. Przy konwertowaniu prostszych dokumentów użytkownika może okazać się program VULCANIZE, który jest szybszy, ale nie potrafi radzić sobie z równaniami matematycznymi. Oba te programy są skryptami napisanymi w języku Perl.



Program LATEXTOHTML dostępny jest poprzez anonimowe FTP na serwerze [ftp.tex.ac.uk](ftp://ftp.tex.ac.uk) w katalogu pub/archive/support, pod nazwą latextohtml. Program VULCANIZE natomiast można otrzymać na stronie WWW pod adresem <http://plover.com/vulcanize>.

Program RTFTOHTML pozwala tworzyć dokumenty HTML w oparciu o dokumenty w formacie RTF, obsługiwany przez wiele edytorów tekstów – można więc zapisać dokument ze swojego ulubionego edytora do formatu RTF, a następnie przekonwertować go do formatu HTML za pomocą programu RTFTOHTML.



Program RTFTOHTML jest dostępny na stronie WWW pod adresem <http://kali.usask.ca/r2h68k/html/overview.htm>

Konserwacja dokumentów HTML

Rola autora strony WWW nie sprowadza się tylko do jej utworzenia i udostępnienia calemu świata. Jeśli strona nie ma postaci prostego dokumentu tekstowego, to najprawdopodobniej znajduje się na niej sporo odnośników do innych stron i serwerów. Odnośniki te muszą być regularnie weryfikowane. Co jakiś czas należy również sprawdzać prawidłowość udostępnianych stron, mając na uwadze fakt, że mogą czytać je ludzie z całego świata.

Na szczęście istnieje kilka programów wspomagających sprawdzanie istniejących na stronie WWW hiperłączy. Pozwalają one również na wyszukiwanie innych węzłów czy dokumentów, do których odnośniki mógłbyś chcieć zamieścić na swojej stronie. Programy takie, często nazywane pajakami lub robotami, potrafią samodzielnie przemieszczać się w sieci, tworząc listę odpowiednich adresów (pająki są podobne do programów użytkowych Archie i Veronica, które jednak nie potrafią poruszać się w całym Internecie).

Choć uważa się je za przydatne tylko dla użytkowników stron WWW (można za ich pomocą uzyskać listę węzłów, które warto odwiedzić), pająki i programy pokrewne mogą również przydać się autorom stron WWW, dostarczając informacji o potencjalnie interesujących odnośnikach, które można zamieścić na stronie WWW. Jednym z bardziej znanych programów tego typu jest program o nazwie World Wide Web Worm, w skrócie WWW. Pozwala on na wyszukiwanie słów kluczowych i używanie operatorów logicznych. Potrafi przeszukiwać zarówno tytuły dokumentów, jak i same dokumenty; możliwe jest również użycie kilku innych trybów wyszukiwania (włącznie z wyszukiwaniem wszystkich znanych stron WWW).

Podobnym narzędziem jest program o nazwie WebCrawler, pozwalający dodatkowo na wyszukiwanie dowolnych słów kluczowych w treści dokumentów. Lista znalezionych dokumentów uporządkowana jest od najlepiej spełniających kryteria wyszukiwania.



Program World Wide Web Worm można załadować ze strony <http://guano.cs.colorado.edu/home/mcbryan/WWWguano.html>, natomiast WebCrawler dostępny jest pod adresem <http://www.biotech.washington.edu/WebCrawler/WebCrawler.html>.

W przypadku dokumentów HTML bardzo często zdarza się, że w miarę upływu czasu umieszczone w nich hiperłącza tracą ważność, wskazując na pliki lub serwery które w rzeczywistości już nie istnieją (ponieważ zmieniła się ich lokalizacja albo zostały usunięte). Z tego powodu warto regularnie sprawdzać i aktualniwać wszystkie odnośniki na stronach WWW. Pomoże w tym program o nazwie HTML_ANALYZER, który potrafi sprawdzić, czy wszystkie odnośniki na danej stronie są prawidłowe. Po sprawdzeniu danego dokumentu tworzy on plik tekstowy, do którego zapisuje listę wszystkich adresów, również tych nie będących odnośnikami. Przy następnym uruchomieniu sprawdza, czy odnośniki nadal prowadzą w te same miejsca.

HTML_ANALYZER wykonuje trzy rodzaje testów: sprawdza, czy dokument wskazywany przez odnośnik jest dostępny (ang. *validation*), sprawdza występujące w bazie danych

adresy nie będące hiperłączami (ang. *completeness*) i sprawdza, czy zostały zachowane odpowiednie relacje pomiędzy hiperłączami i wskazywanymi przez nie dokumentami (ang. *consistency*). Wszelkie odchylenia od normy są przedstawiane użytkownikowi.

Użytkownicy programu HTML_ANALYZER muszą dość dobrze znać język HTML, system operacyjny i potrafić używać analizatorów sterowanych z wiersza poleceń. Program ten należy przed użyciem skompilować za pomocą programu narzędziowego `make`. Przed uruchomieniem programu HTML_ANALYZER trzeba utworzyć kilka katalogów, a podczas działania tworzy on kilka plików tymczasowych, które nie są usuwane. Z tych powodów nie jest to program łatwy w obsłudze dla mniej doświadczonych użytkowników.

Podstawy języka HTML

Język HTML (HyperText Markup Language) jest dość prosty do opanowania i używania, a ponieważ ostatnimi czasy pojawiły się nowe wersje tego języka, jest on również dość potężny. Nie będziemy nawet próbować nauczyć Cię języka HTML w tym jednym niewielkim rozdziale – spróbujmy jednak dokonać swego rodzaju przeglądu możliwości przez niego oferowanych i podać podstawowe informacje odnośnie jego używania na przykładzie prościutkich stron WWW.

Jeśli widziałeś kiedykolwiek jakąś stronę WWW, widziałeś wyniki zastosowania języka HTML. HTML to język używany do opisu stron WWW – z jego pomocą definiuje się wygląd stron WWW pojawiających się w przeglądarce po połączeniu się z serwerem. Serwer przesyła instrukcje języka HTML do przeglądarki, która odpowiednio je interpretuje i formatuje wyświetlany dokument. Do wyświetlania dokumentów napisanych w języku HTML zwykle używa się przeglądarki WWW, choć zadanie to mogą również wykonać inne programy. Obecnie dostępna jest cała gama różnorakich przeglądarek WWW, rozpoczynając od najstarszej chyba aplikacji stworzonej w NCSA – Mosaic. Najczęściej używanymi przeglądarkami są Netscape Navigator i Microsoft Explorer. Jednak to, jakiego programu używasz do przeglądania stron WWW, nie ma w większości przypadków większego znaczenia, ponieważ wszystkie one spełniają tę samą funkcję – wyświetlają kod HTML odebrany od serwera. Przeglądarka WWW prawie zawsze działa jako klient, wysyłając żądania przesłania odpowiednich danych do serwera.

Język HTML jest oparty na innym języku o nazwie SGML (Standard Generalized Markup Language), używanym do opisywania struktury dokumentu i ułatwiającym przenoszenie dokumentów pomiędzy różnymi edytorami tekstów. Język HTML nie określa sposobu, w jaki strona będzie wyświetlana – nie jest to język opisu strony, taki jak na przykład PostScript. Język HTML opisuje strukturę dokumentu. Pozwala wyznaczyć fragment tekstu pełniący rolę nagłówka, zdefiniować treść dokumentu, określić położenie obrazków w tekście. Nie zawiera jednak dokładnych informacji o wyglądzie strony – za to odpowiedzialna jest przeglądarka WWW.

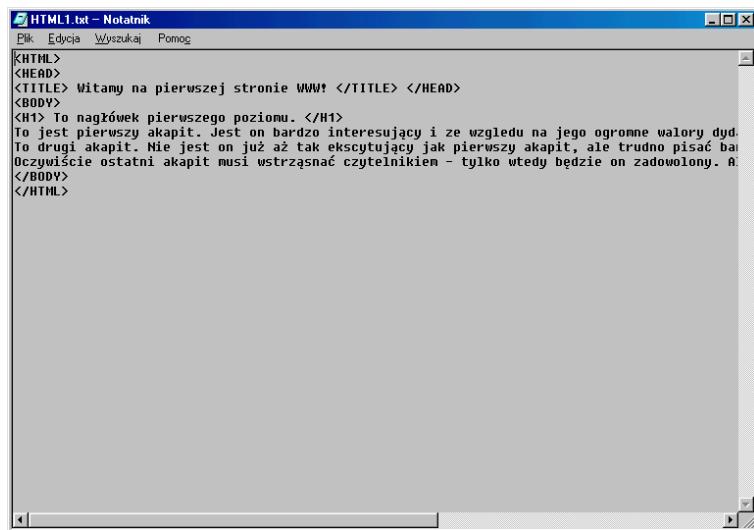
Dlaczego warto używać języka HTML? Przede wszystkim dlatego, że jest to mały język, dzięki czemu możliwe jest szybkie przesyłanie jego poleceń za pośrednictwem sieci. Choć ze względu na swój niewielki rozmiar jest on nieco ograniczony, nowsze wersje

coraz bardziej rozszerzają jego możliwości. Inną bardzo ważną zaletą tego języka (z której nie wszyscy zdają sobie sprawę) jest jego niezależność od urządzenia. Nie ma znaczenia, jakiego systemu używasz do przeglądania stron WWW – przeglądarka potrafi przetworzyć plik HTML i wyświetlić go w danym systemie. To przeglądarka jest zależna od urządzenia, natomiast tworząc strony WWW, nie musisz zastanawiać się nad tym, w jakich systemach będą one przeglądane.

Jak wygląda język HTML?

Jak za chwilę się przekonasz, kod w języku HTML jest bardzo prosty. W większości składa się on z zestawu znaczników (ang. *tags*), opisujących początek i koniec elementu struktury dokumentu (takiego jak na przykład nagłówek, akapit, obrazek, tabela itp.). Każdy z elementów powinien być otoczony znacznikiem rozpoczętym (otwierającym) i kończącym (zamykającym). Przykładowy kod źródłowy dokumentu HTML przedstawiony jest na rysunku 53.1. Nie przejmuj się, jeśli na razie nie wszystko rozumiesz – w dalszej części tego rozdziału przyjrzymy się wszystkim widocznym na rysunku znacznikom (zrzuty ekranu przedstawione w tym rozdziale zostały wykonane w systemie Windows).

Rysunek 53.1.
Kod źródłowy
przykładowego
dokumentu
w języku HTML



The screenshot shows a Microsoft Notepad window titled "HTML1.txt - Notatnik". The content of the file is as follows:

```
<HTML>
<HEAD>
<TITLE> Witamy na pierwszej stronie WWW! </TITLE> </HEAD>
<BODY>
<H1> To nagłówek pierwszego poziomu. </H1>
To jest pierwszy akapit. Jest on bardzo interesujący i ze względu na jego ogromne walory dyd.
To drugi akapit. Nie jest on już aż tak ekscytujący jak pierwszy akapit, ale trudno pisać ba.
Oczywiście ostatni akapit musi wstrząsnąć czytelnikiem - tylko wtedy będzie on zadowolony. A.
</BODY>
</HTML>
```

Zanim zaczniemy, wyjaśnijmy kilka ważnych kwestii dotyczących znaczników. W ich nazwach nie są rozróżniane małe i wielkie litery i prawie zawsze występują one parami: znacznik otwierający i znacznik zamykający. Najczęściej spotykanym błędem składniowym w dokumentach HTML jest nieprawidłowo zamknięty lub nie zamknięty znacznik. W wielu przypadkach strona z takim błędem będzie wyświetlana poprawnie, ale w innych okolicznościach może to spowodować poważne problemy z jej sformatowaniem. Ten typ błędów można wyeliminować przez dokładne sprawdzanie kodu źródłowego.



Nie wszystkie znaczniki występują w parach. Istnieje kilka znaczników „pojedynczych”, istnieją również znaczniki będące swego rodzaju pojemnikami (ang. *containers*) przechowującymi dodatkowe informacje. Tego typu znaczniki nie posiadają swych odpowiedników zamkujących.

Znaczniki zapisuje się w pojedynczych nawiasach trójkątnych. Nawiasy te informują przeglądarkę o tym, że ich zawartość należy traktować jako instrukcję języka HTML. Fragmenty kodu w języku HTML mają więc zwykle postać:

```
<nazwa_znacznika_otwierającego> tekst tekst tekst  
<nazwa_znacznika_zamykającego>
```

Tego typu para znaczników określa, w jaki sposób traktowany będzie tekst zawarty pomiędzy nimi. Jeśli na przykład są to znaczniki początku i końca nagłówka, tekst zapisany pomiędzy nimi w przeglądarce będzie odpowiednio wyróżniony, np. wytłuszczonej czy powiększonej.

W jaki sposób tworzy się kod w języku HTML? Można to robić na kilka sposobów. Najprostszym z nich jest użycie dowolnego edytora ASCII. Upewnij się, że nie zapisujesz dokumentu HTML w jakimś specjalnym formacie, na przykład jako dokument programu Word, ponieważ przeglądarki WWW radzą sobie tylko i wyłącznie z plikami w formacie ASCII. Dostępne są również wyspecjalizowane edytory pozwalające na wstawianie znaczników po wybraniu ich z menu i śledzenie na bieżąco efektów formatowania tworzonego dokumentu, które są bardzo przydatne przy pisaniu większych stron WWW, ale dla większości użytkowników prosty edytor tekstów w zupełności wystarczy do poznania podstawowych zasad używania języka HTML.

Początek dokumentu HTML

Dokument HTML powinien rozpoczynać się od znacznika informującego, że jest on napisany właśnie w tym języku. Znacznik ten ma postać `<HTML>` i na jego podstawie przeglądarka internetowa może zorientować się, że ma do czynienia z początkiem dokumentu w języku HTML. Oto fragment kodu źródłowej strony WWW:

```
<HTML>  
<HEAD>  
<TITLE> To jest moja strona WWW! </TITLE> </HEAD>  
<BODY>  
<H1> To pierwszy nagłówek mojej strony głównej. </H1>  
To fragment tekstu zamieszczonego na mojej stronie głównej. Mam nadzieję,  
że przypadła Ci ona do gustu.  
</BODY>  
</HTML>
```

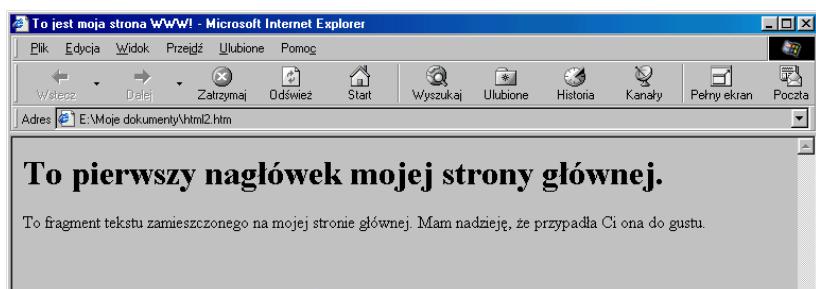
Pierwszy i ostatni znacznik – czyli `<HTML>` i `</HTML>` – wyznacza początek i koniec kodu w języku HTML. Ukośnik wchodzący w skład drugiego ze znaczników symbolizuje koniec elementu strukturalnego. Znaczniki `<HEAD>` i `</HEAD>` oznaczają część wstępna pliku; zwykle umieszcza się pomiędzy nimi tytuł strony i słowa kluczowe, które

re będą wyświetlane w wyszukiwarkach internetowych (pomiędzy tymi znacznikami dozwolone jest użycie tylko kilku innych znaczników). Znaczniki `<TITLE>` i `</TITLE>` pozwalają zdefiniować tytuł dokumentu. Pomiędzy znacznikami `<BODY>` i `</BODY>` umieszcza się treść dokumentu. Znaczniki `<H1>` i `</H1>` pozwalają wstawić do dokumentu nagłówek pierwszego poziomu.

Powyższy kod może zostać zinterpretowany przez dowolną przeglądarkę internetową – efekt sformatowania dokumentu przedstawia rysunek 53.2. Jak widać, w oknie głównym wyświetlany jest tylko tekst umieszczony pomiędzy znacznikami określającymi treść dokumentu, nie jest natomiast wyświetlany tekst określający jego tytuł – on pojawi się na pasku tytułowym okienka przeglądarki i spełnia funkcję identyfikacyjną.

Rysunek 53.2.

Przykładowy dokument HTML wyświetlony za pomocą przeglądarki Internet Explorer



Przedstawiony powyżej przykładowy plik składa się z kilku wierszy tekstu – taki podział nie jest jednak konieczny i ma na celu tylko zwiększenie czytelności kodu. Jeśli chcesz, możesz wszystkie instrukcje umieścić w jednym, długim wierszu, ponieważ w języku HTML znaki białe są domyślnie ignorowane. Jednak ze względu na łatwość wyszukiwania ewentualnych usterek i czytelność kodu warto organizować kod w przeszysty sposób.

Znacznik `<TITLE>` powinien zawsze występować w kontekście znacznika nagłówka (`<HEAD>` i `</HEAD>`); używa się go do definiowania tytułu strony, który powinien opisywać jej zawartość. Strona może posiadać tylko jeden tytuł. Pomiędzy znacznikami początku i końca nagłówka nie mogą występować żadne inne znaczniki. Tytuł strony powinien być w miarę krótki, a jednocześnie dawać jasne informacje o zawartości strony, dzięki czemu użytkownicy przeglądający dany dokument będą mogli go łatwo zidentyfikować.

Znaczniki `<BODY>` i `</BODY>` określają początek i koniec treści dokumentu. Zwykle w dokumencie występuje dokładnie jedna taka para. Wszystkie instrukcje określające zawartość strony, tekst, odnośniki, grafiki itd. powinny znaleźć się pomiędzy tymi znacznikami.

W języku HTML zdefiniowanych jest kilka poziomów nagłówków, dzięki czemu możliwe jest tworzenie rozdziałów, podrozdziałów itd. W przedstawionym wcześniej przykładzie użyliśmy nagłówka definiowanego przez znacznik `<H1>`, czyli nagłówka pierwszego, najwyższego poziomu. Jeśli struktura dokumentu jest wielopoziomowa, warto zastosować kilka rodzajów nagłówków. Oto przykładowy dokument:

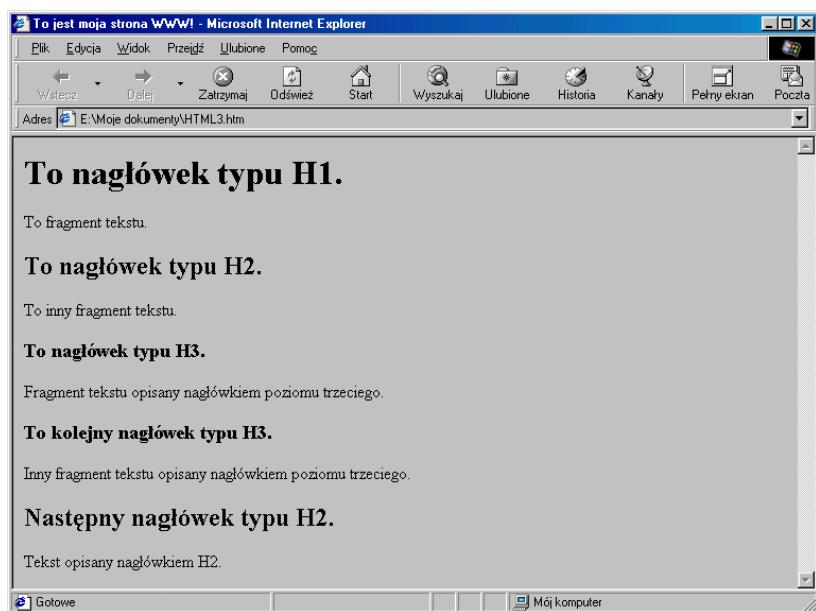
```

<HTML>
<HEAD>
<TITLE> To jest moja strona WWW! </TITLE> </HEAD>
<BODY>
<H1> To nagłówek typu H1. </H1>
To fragment tekstu.
<H2> To nagłówek typu H2. </H2>
To inny fragment tekstu.
<H3> To nagłówek typu H3. </H3>
Fragment tekstu opisany nagłówkiem poziomu trzeciego.
<H3> To kolejny nagłówek typu H3. </H3>
Inny fragment tekstu opisany nagłówkiem poziomu trzeciego.
<H2> Następny nagłówek typu H2. </H2>
Tekst opisany nagłówkiem H2.
</BODY>
</HTML>

```

Rysunek 53.3 przedstawia efekt wyświetlania powyższego dokumentu w przeglądarce WWW. Jak widać, poszczególne nagłówki różnią się nieco od siebie – nagłówki wyższego poziomu są zwykle nieco większe i bardziej się wyróżniają. Dzięki temu możliwe jest podzielenie zawartości strony WWW na jednostki logiczne i przydzielenie każdej z nich nagłówka odpowiedniego poziomu. Nagłówków można używać tak, jak byłyby one stosowane w książce: tekst opisany nagłówkiem pierwszego poziomu (`<H1>`) może zawierać kilka fragmentów opisanych nagłówkami drugiego poziomu (`<H2>`), zawierających z kolei fragmenty opisane nagłówkami trzeciego poziomu (`<H3>`) itd. Choć nie ma żadnych formalnych ograniczeń co do używania i mieszania poziomów nagłówków (nic nie stoi na przeszkodzie, aby na przykład używać tylko nagłówków trzeciego poziomu), zdrowy rozsądek zwykle pomaga w określeniu prawidłowej struktury tworzonego dokumentu.

Rysunek 53.3.
Nagłówki różnych poziomów prezentują się nieco odmiennie



Akapity

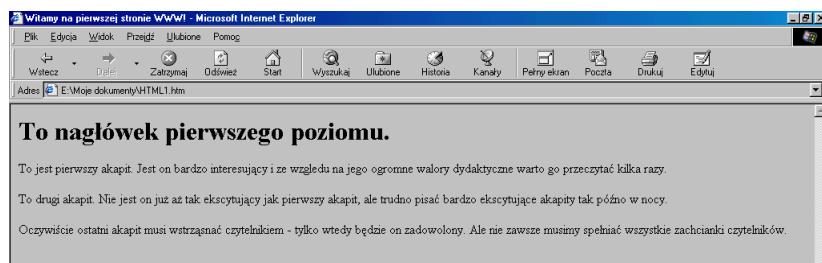
A co z akapitami? Istnieje kilka sposobów uzyskania prawidłowego podzielenia tekstu na akapity, ponieważ reguły zmieniały się w kolejnych wersjach języka HTML. Najprostszym podejściem jest jednak użycie znaczników `<P>` i `</P>` (ang. *paragraph*), wyznaczających zakres tekstu wchodzącego w skład pojedynczego akapitu. Oto przykładowy dokument w języku HTML zawierający trzy akapity:

```
<HTML>
<HEAD>
<TITLE> Witamy na pierwszej stronie WWW! </TITLE> </HEAD>
<BODY>
<H1> To nagłówek pierwszego poziomu. </H1>
<P> To jest pierwszy akapit. Jest on bardzo interesujący i ze względu na jego ogromne walory dydaktyczne warto go przeczytać kilka razy. </P>
<P> To drugi akapit. Nie jest on już aż tak ekscytujący jak pierwszy akapit, ale trudno pisać bardzo ekscytujące akapity tak późno w nocy. </P>
<P> Oczywiście ostatni akapit musi wstrząsnąć czytelnikiem - tylko wtedy będzie on zadowolony. Ale nie zawsze musimy spełniać wszystkie zachcianki czytelników. </P>
</BODY>
</HTML>
```

Efekt przetworzenia tego dokumentu przez przeglądarkę internetową przedstawia rysunek 53.4. Zauważ, że poszczególne akapity są rozdzielone, a pomiędzy nimi zostały wstawione odpowiednie odstępy. Co jednak stało by się, gdyby znaczniki akapitów zostały pominięte? Ponieważ znaki białe – w tym symbole nowego wiersza – są ignorowane przez przeglądarkę stron WWW, cały tekst zostałby wyświetlony jako pojedynczy akapit, tak jak przedstawia to rysunek 53.5. Do rozdzielania poszczególnych akapitów należy więc używać znaczników `<P>` i `</P>`. Pamiętaj o tym, że wstawienie nawet większej liczby pustych wierszy pomiędzy fragmentami tekstu nie spowoduje potraktowania ich jako osobnych akapitów – znaki nowego wiersza zostaną zignorowane i tekst zostanie potraktowany jako pojedynczy akapit.

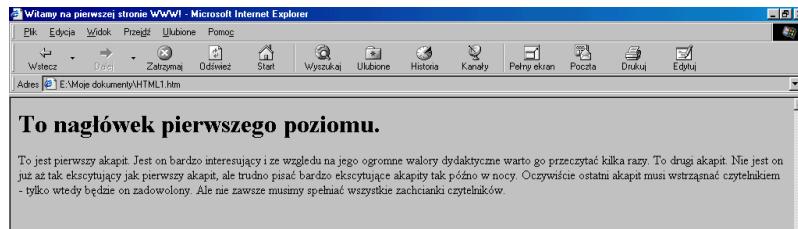
Rysunek 53.4.

Użycie znaczników akapitu powoduje rozdzielenie fragmentów tekstu i wstawienie pomiędzy nie odpowiedniego odstępu



Rysunek 53.5.

Bez znaczników akapitu cały tekst traktowany jest jak pojedynczy akapit



Mówiąc ściślej, użycie znacznika `</P>` do oznaczenia końca akapitu nie jest konieczne, ponieważ znacznik `<P>` powoduje wymuszenie rozpoczęcia nowego akapitu. Znacznik `<P>` jest więc jednym ze znaczników, które nie muszą występować wraz ze swym odpowiednikiem zamykającym (czyli `</P>`). Mimo tego, dobrym zwyczajem jest zamykanie również tego znacznika.

A co z komentarzami w kodzie dokumentów HTML? Przecież może zdarzyć się, że będziesz chciał zapisać w dokumencie informację o jego autorstwie, skomentować fragment kodu itp. Komentarz można wstawić do dokumentu w następujący sposób:

```
<!-- To jest komentarz -->
```

Komentarz powinien być otoczony nawiasami trójkątnymi, jego pierwszym znakiem powinien być symbol wykrzyknika, a na początku i końcu komentarza powinien wystąpić myślnik. Oto przykład kodu w języku HTML zawierającego komentarze:

```
<HTML>
<!-- Autor: TJP, 12/12/95; v1.23 -->
<HEAD>
<TITLE> Witam na mojej stronie WWW! </TITLE> </HEAD>
<BODY>
<H1> To nagłówek pierwszego poziomu. </H1>
<!-- ta sekcja dotyczy znaczników akapitu -->
<P> To jest pierwszy akapit </P>
</BODY>
</HTML>
```

Hiperłącza

Odnośniki pozwalające łatwo przejść do innego fragmentu dokumentu czy innego dokumentu są bardzo ważnym aspektem stron WWW. W języku HTML odnośniki takie tworzy się w bardzo prosty sposób. Rozpoczynają się one od znacznika początku odnośnika `<A>`, a kończą znacznikiem `` (ang. *anchor* – zakotwiczenie; nazwa ta wzięła się od faktu, że odnośnik jest „zakotwiczony” pomiędzy tymi znacznikami).

Znacznik `<A>` różni się nieco od innych przedstawionych wcześniej znaczników, ponieważ pomiędzy nawiasami trójkątnymi może znaleźć się dodatkowy tekst. Oto przykład:

```
<A HREF="strona_2.html">Idź do strony 2</A>
```

W powyższym przykładzie tekst znajdujący się pomiędzy znacznikami zostanie wyświetlony na ekranie, dzięki czemu użytkownik zobaczy tekst Idź do strony 2, zwykle podkreślony czy wyróżniony w jakiś inny sposób wskazujący, że jest on odnośnikiem. Jeśli użytkownik kliknie na takim odnośniku, zostanie odczytana wartość pola `HREF` wchodzącego w skład znacznika `<A>`, po czym do przeglądarki zostanie załadowany dokument o nazwie `strona_2.html`. Skrót `HREF` pochodzi od angielskich słów *Hypertext Reference*; można po nim podać nazwę pliku lub URL, do którego będzie prowadził dany odnośnik.

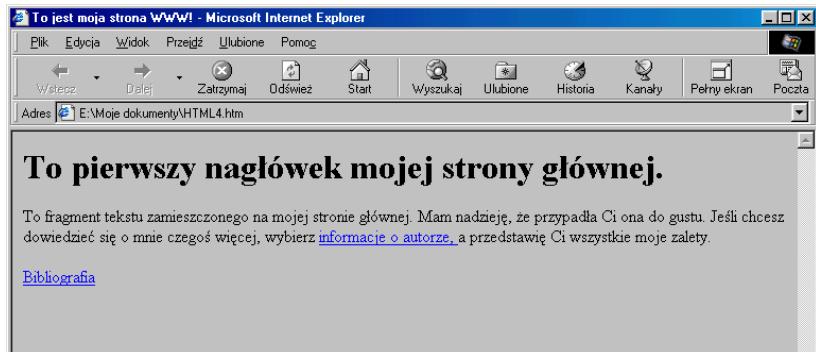
Odnośników można używać zarówno w tekście dokumentu, jak i na przykład jako osobnych pozycji menu. Poniższy dokument HTML prezentuje możliwość wstawienia odnośnika znajdującego się w osobnym akapicie.

```
<HTML>
<HEAD>
<TITLE> To jest moja strona WWW! </TITLE> </HEAD>
<BODY>
<H1> To pierwszy nagłówek mojej strony głównej. </H1>
To fragment tekstu zamieszczonego na mojej stronie głównej. Mam nadzieję,
że przypadła Ci ona do gustu. Jeśli chcesz dowiedzieć się o mnie czegoś
więcej, wybierz <A HREF="o_autorze.html"> informacje o autorze, </A> a
przedstawię Ci wszystkie moje zalety. </P>
<P><A HREF="biblio.html">Bibliografia</A>
</BODY>
</HTML>
```

Rysunek 53.6 przedstawia wygląd powyższego dokumentu po sformatowaniu przez przeglądarkę WWW. Każdy z odnośników jest podkreślony, dzięki czemu łatwo jest go zidentyfikować (w niektórych przeglądarkach zamiast podkreślenia stosowana jest zmiana koloru tekstu lub też wyróżnienie innego typu).

Rysunek 53.6.

Dokument zawierający dwa odnośniki



Kiedy odnośnik prowadzi do innego dokumentu, trzeba poprawnie podać nazwę pliku, w którym jest on zapisany. Można to zrobić na dwa sposoby: podając nazwę pełną lub względową. Nazwa pełna zawiera ścieżkę dostępu do pliku, natomiast nazwa względna jest odniesiona do bieżącego położenia dokumentu. Oto przykład odnośnika do pliku określonego za pomocą nazwy pełnej:

```
A HREF="/usr/tparker/html/home.html">
```

Względne ścieżki dostępu odniesione są do bieżącego położenia dokumentu i mogą zawierać polecenia służące do przemieszczania się w strukturze katalogów, na przykład:

```
<A HREF=". /home.html">  
<A HREF=". . /html/home.html">
```

Odnośniki do innych dokumentów dostępnych za pomocą adresów w formacie URL definiuje się w ten sam sposób – wystarczy tylko zamiast nazwy pliku podać lokator URL. Oto przykładowy odnośnik prowadzący do strony głównej wyszukiwarki Yahoo!:

```
<A HREF="http://www.yahoo.com">Yahoo!</A>
```

W dokumencie można osadzić dowolną liczbę odnośników. Warto dołączać do nich jak najbardziej treściwe komentarze, dzięki czemu użytkownicy nie będą trafiać na strony, których być może wcale nie mieli zamiaru odwiedzać. Jeśli tworzysz odnośniki do innych stron WWW, powinieneś od czasu do czasu sprawdzać, czy nadal są one dostępne. Wiele stron WWW zmienia lokalizację lub też całkowicie kończy działanie, więc odnośniki powinny być kontrolowane, aby zapobiec nieporozumieniom.

Wyliczenia

Język HTML pozwala na użycie kilku różnych formatów list i wyliczeń, na przykład numerowanych, oznaczanych odpowiednimi etykietami lub symbolami graficznymi. Listy otacza się takimi znacznikami, jak na przykład `` i `` (ang. *ordered list*, lista uporządkowana) czy `<MENU>` i `</MENU>`. Każda pozycja listy powinna rozpoczynać się od znacznika `` lub też innego znacznika o podobnej funkcji. Dostępnych jest również kilka specjalnych znaczników służących na przykład do formatowania słowników, ale nie będziemy ich tu omawiać.

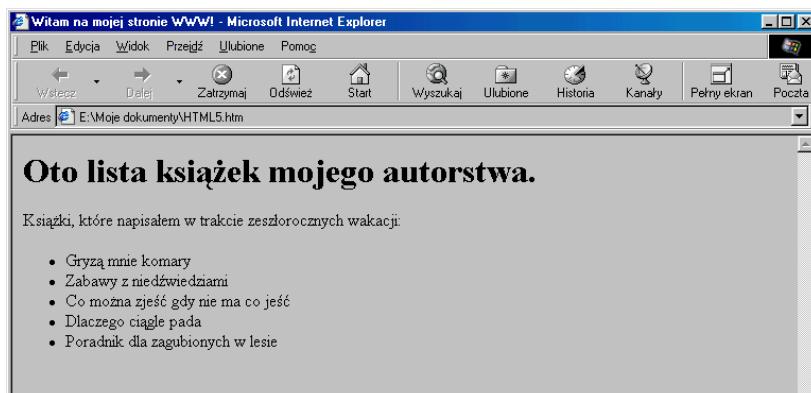
Oto przykład prostej listy definiowanej za pomocą znaczników `` i `` (ang. *unordered list*, lista nieuporządkowana):

```
<HTML>
<HEAD>
<TITLE> Witam na mojej stronie WWW! </TITLE> </HEAD>
<BODY>
<H1> Oto lista książek mojego autorstwa. </H1>
Książki, które napisałem w trakcie zeszłorocznych wakacji:
<UL>
<LI> Gryzą mnie komary
<LI> Zabawy z niedźwiedziami
<LI> Co można zjeść gdy nie ma co jeść
<LI> Dlaczego ciągle pada
<LI> Poradnik dla zagubionych w lesie
</UL>
</BODY>
</HTML>
```

Lista nieuporządkowana jest podobna do uporządkowanej, z tym że jej elementy nie są numerowane. Efekt sformatowania powyższego kodu przedstawiony jest na rysunku 53.7 – można zauważać, że poszczególne elementy listy oznaczone są symbolami graficznymi, tworzącymi pionową linię.

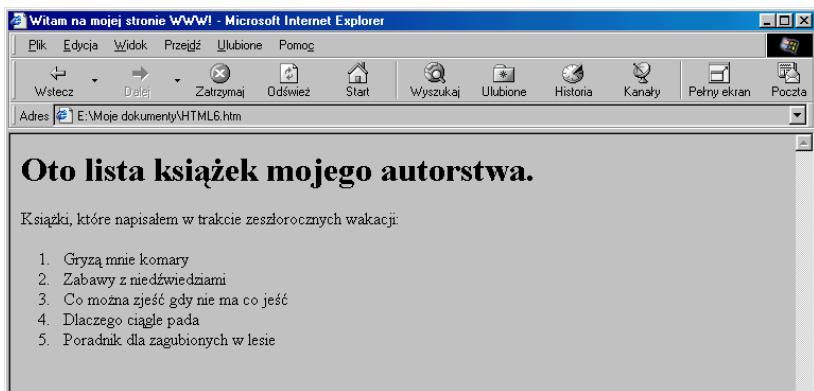
Rysunek 53.7.

Nieuporządkowana lista w dokumencie HTML



Ten sam dokument może zostać sformatowany w postaci listy uporządkowanej – służą do tego znaczniki `` i ``. Elementy takiej listy są poprzedzone odpowiednimi numerami, jak widać na rysunku 53.8. Pozostała część dokumentu nie została zmodyfikowana – zamiast znaczników `` i `` wstawiono odpowiednio `` i ``.

Rysunek 53.8.
*Dokument HTML
może również
zawierać listę
numerowaną*



Zmiana kroju pisma

W języku HTML zdefiniowano również zestaw znaczników służących do modyfikowania wyglądu znaków. Pozwalają one zarówno na bezpośrednie określenie kroju znaków (na przykład wymuszenie zastosowania kursywy, wytłuszczenia itp.), jak i na logiczne wyróżnienie fragmentu tekstu (na przykład dla zaakcentowania jakiegoś wyrazu, zaznaczenia kodu programu czy tekstu innego typu). Wymuszanie kroju pisma zwykle nie jest najlepszym pomysłem, ponieważ niektóre przeglądarki mogą wyświetlać tekst w sposób inny, niż zamierzony. Mimo tego, można używać tej możliwości, jeśli wiadomo, że dokument będzie wykorzystywany tylko przez użytkowników dysponujących przeglądarkami formatującymi tekst w sposób zgodny z zamierzeniami.

O wiele lepszym pomysłem jest zastosowanie znaczników pozwalających na logiczne wyróżnienie fragmentu tekstu, ponieważ w takim przypadku przeglądarka może samodzielnie zdecydować jaki kój pisma będzie najodpowiedniejszy dla danej platformy. Z tego powodu tym właśnie znacznikom przyjrzymy się nieco bliżej. Język HTML udostępnia osiem powszechnie wykorzystywanych znaczników pozwalających logicznie wyróżniać fragmenty tekstu:

- υ `<CITE>` – cytat
- υ `<CODE>` – fragment kodu źródłowego (zwykle czcionka Courier)
- υ `<DFN>` – wyróżnienie definicji
- υ `` – zaakcentowanie fragmentu tekstu (zwykle wyróżnienie kursywą)
- υ `<KBD>` – fragment tekstu, który użytkownik powinien wprowadzić z klawiatury
- υ `<SAMP>` – tekst przykładowy, formatowany podobnie jak `<CODE>`

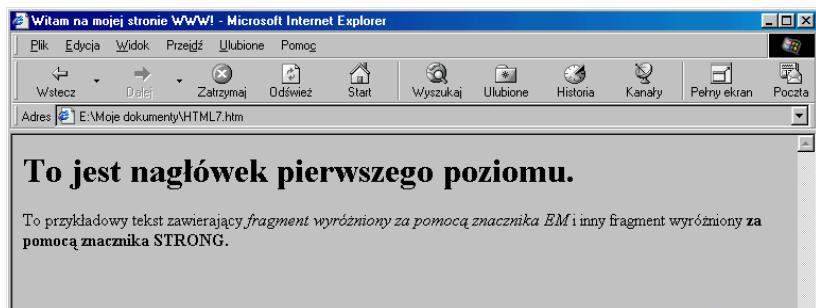
- υ <**STRONG**> – tekst wyróżniony (zwykle wytłuszczone)
- υ <**VAR**> – nazwa zmiennej, wyświetlna kursywą lub podkreślona (zwykle w kontekście znacznika <CODE>)

Poniższy kod źródłowy dokumentu przedstawia zastosowanie niektórych przedstawionych wyżej stylów, a rezultat otwarcia tegoż dokumentu w przeglądarce stron WWW przedstawia rysunek 53.9.

```
<HTML>
<HEAD>
<TITLE> Witam na mojej stronie WWW! </TITLE> </HEAD>
<BODY>
<H1> To jest nagłówek pierwszego poziomu. </H1>
<P> To przykładowy tekst zawierający <EM> fragment wyróżniony za pomocą
znacznika EM </EM> i inny fragment wyróżniony <STRONG> za pomocą
znacznika STRONG. </STRONG> </P>
</BODY>
</HTML>
```

Rysunek 53.9.

Zastosowanie
znaczników
pozwalających
na logiczne
wyróżnianie
fragmentów
tekstu



Jak widać, przedstawiona na rysunku przeglądarka (Internet Explorer) interpretuje znacznik <**EM**> wyróżniając fragment tekstu kursywą, a znacznik <**STRONG**> – wytłuszczeniem. Większość przeglądarek interpretuje te znaczniki właśnie w taki sposób, ale inne znaczniki mogą być różnie interpretowane w różnych przeglądarkach.

Jeśli chcesz wymusić określony krój pisma, możesz zastosować znaczniki <**B**> i </**B**> (wytluszczenie, ang. *bold*), <**I**> i </**I**> (kursywa, ang. *italics*) oraz <**TT**> i </**TT**> (czcionka maszyny do pisania, używana zwykle do prezentacji kodu źródłowego).

Inne znaczniki

Przy tworzeniu stron WWW przydaje się również kilka innych znaczników. Jednym z nich jest znacznik <**PRE**>, umożliwiający poinformowanie przeglądarki, że fragmentu tekstu został sformatowany wcześniej (ang. *preformatted*). W tekście zawartym pomiędzy znacznikami <**PRE**> i </**PRE**> znaki białe nie są ignorowane. Pozwala to na ręczne

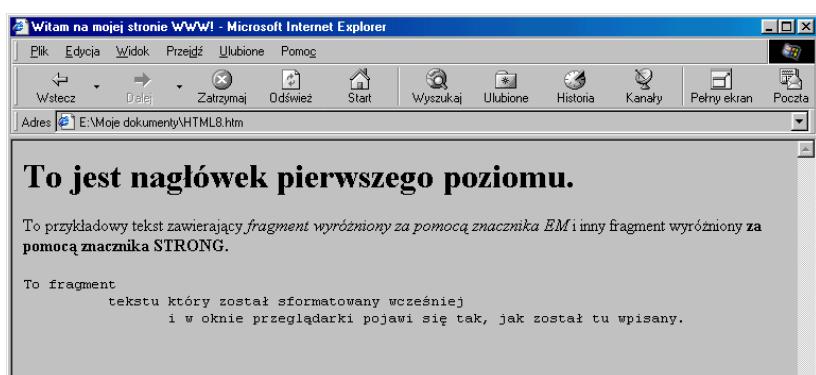
sformatowanie tabeli czy tekstu innego typu dokładnie w wymagany sposób. Oto przykładowy dokument zawierający te znaczniki:

```
<HTML>
<HEAD>
<TITLE> Witam na mojej stronie WWW! </TITLE> </HEAD>
<BODY>
<H1> To jest nagłówek pierwszego poziomu. </H1>
<P> To przykładowy tekst zawierający <EM> fragment wyróżniony za pomocą
znacznika EM </EM> i inny fragment wyróżniony <STRONG> za pomocą
znacznika STRONG. </STRONG> </P>
<PRE>
To fragment
    tekstu który został sformatowany wcześniej
        i w oknie przeglądarki pojawi się tak, jak został tu wpisany.
</PRE>
</BODY>
</HTML>
```

Jak widać na rysunku 53.10, we fragmencie otoczonym znacznikami `<PRE>` zachowane zostały nadmiarowe spacje i znaki nowego wiersza, a nawet zachowana została czcionka użyta do napisania kodu źródłowego (Courier).

Rysunek 53.10.

Znacznik `<PRE>`
pozwala formatować
fragment tekstu
ręcznie



Innym prostym, ale bardzo użytecznym znacznikiem jest znacznik `<HR>`, pozwalający wstawić do dokumentu poziomą linię. Spróbujmy wzbogacić o ten element przedstawiony powyżej przykładowy dokument:

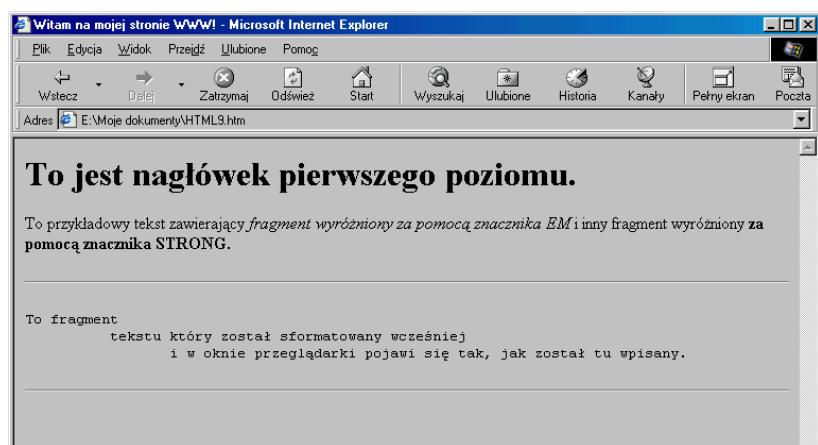
```
<HTML>
<HEAD>
<TITLE> Witam na mojej stronie WWW! </TITLE> </HEAD>
<BODY>
<H1> To jest nagłówek pierwszego poziomu. </H1>
<P> To przykładowy tekst zawierający <EM> fragment wyróżniony za pomocą
znacznika EM </EM> i inny fragment wyróżniony <STRONG> za pomocą
znacznika STRONG. </STRONG> </P>
<HR>
<PRE>
To fragment
    tekstu który został sformatowany wcześniej
        i w oknie przeglądarki pojawi się tak, jak został tu wpisany.
```

```
</PRE>
<HR>
</BODY>
</HTML>
```

Jak widać na rysunku 53.11, na stronie pojawiły się dwie poziome linie. Ich forma może różnić się w zależności od wykorzystywanej przeglądarki, ale ogólna idea pozostaje taka sama: jest to pozioma linia rozdzielająca fragmenty dokumentu.

Rysunek 53.11.

Znacznik *<HR>* służy do podziału strony za pomocą poziomej linii



Podsumowanie

Jeśli nie chcesz używać generatora ani konwertera kodu, możesz tworzyć strony WWW pisząc je bezpośrednio w języku HTML. Język ten nie jest trudny od strony „programistycznej”, dlatego można go opanować w bardzo krótkim czasie. Do programowania w języku HTML nie jest potrzebne doświadczenie z innymi językami programowania.

Założenie własnej strony głównej wymaga posłużenia się jakimś programem narzędziowym umożliwiającym tworzenie dokumentów HTML lub napisania strony WWW bezpośrednio w języku HTML za pomocą dowolnego edytora tekstu. Dokładne omówienie języka HTML wykracza poza zakres materiału omawianego w tej książce, ale na rynku dostępnych jest obecnie sporo książek na ten temat. Język HTML jest bardzo prosty do opanowania. Opierając się na informacjach podanych w tym rozdziale, powinieneś być w stanie założyć własną stronę główną i udostępnić ją wszystkim użytkownikom Internetu.

Jeśli chcesz dowiedzieć się czegoś więcej o języku Java, za pomocą którego można uatrakcyjnić strony WWW, przejdź do rozdziału 54. „Podstawy języka Java i JavaScript”.

Rozdział 56. „Zarządzanie kodem źródłowym” zawiera informacje, które pozwolą Ci używać systemów zarządzania kodem źródłowym i innych narzędzi tego typu.

Inne aplikacje dla systemu Linux omówione są w rozdziałach od 62. „Adabas-D i inne bazy danych” do 64. „Program Lone-Tar”.

Rozdział 54.

Podstawy języków Java i JavaScript

Tim Parker

W tym rozdziale:

- υ Co będzie Ci potrzebne?
- υ Język Java
- υ JavaScript i HTML

Zanim rozpoczniemy ten rozdział, wyjaśnijmy jedną rzecz: celem tego rozdziału nie jest nauczyć Cię programowania w języku Java. Temat ten wymagałby o wiele obszerniejszego omówienia. W tym rozdziale chcemy przedstawić koncepcję tego języka, jego zastosowania i niektóre aspekty posługiwania się nim.

Co to jest Java? Jest to język programowania opracowany przez firmę Sun Microsystems. W komunikatach dla prasy był on przedstawiany jako język „prosty, obiektowy, łatwy do zastosowania w systemach rozproszonych, interpretowany, mocny, bezpieczny, niezależny od architektury, przenośny, wydajny, wielowątkowy i dynamiczny”. Co to tak naprawdę oznacza? Zaczniemy od tego, że język Java został zaprojektowany jako język obiektowy prostszy w użyciu niż C++ czy Smalltalk, które są językami bardzo rozległymi i dość skomplikowanymi. Dzięki temu, że język Java jest prosty, jest on również mniej podatny na błędy niż inne języki programowania. To właśnie oznacza prostota i moc tego języka. Niewielki rozmiar przyczynia się również do wzrostu wydajności.

Java jest językiem interpretowanym, co oznacza, że każdy z wierszy kodu źródłowego jest kolejno odczytywany i wykonywany przez interpreter języka Java, nie jest natomiast tworzony nadający się do bezpośredniego uruchomienia plik wykonywalny. W zasadzie sprawa jest nieco uproszczona, ponieważ programy w języku Java są jakby połowicznie skompilowane (ang. *pseudo-compiled*) i są rozprowadzane w postaci binarnych plików pośrednich – rozszerzenie nazw plików zawierających tego typu kod ma postać `.class`. Takie rozwiązanie może być nieco wolniejsze niż użycie w pełni skompilowanych plików, ale dzięki użyciu języka niezależnego od platformy sprzętowej (co oznacza, że w kodzie programu nie występują żadne instrukcje specyficzne dla danego sprzętu) kod źródłowy

w języku Java może działać w dowolnym systemie wyposażonym w interpreter tego języka. O tym właśnie mówią przymiotniki „niezależny od architektury” oraz „przenośny”. Łatwość zastosowania w systemach rozproszonych wynika bezpośrednio z przenośności i niezależności od architektury – kod źródłowy w języku Java, który ma zostać wykonany, może z łatwością być przesyłany z jednego systemu do innego za pośrednictwem sieci. Dzięki temu serwery mogą wysyłać kod programu do klientów, tworząc system rozproszony (program w języku Java działa po stronie klienta i odsyła wyniki działania do serwera).

Ponieważ programy w języku Java mogą zostać uruchomione praktycznie w każdym systemie operacyjnym, mogą wykorzystywać zalety tychże systemów – na przykład możliwość zastosowania wielowątkowości w systemach UNIX-owych. Choć sam język Java może być uważany za wielowątkowy, w praktyce wszystko zależy od systemu operacyjnego, w którym wykonywany jest program. Pozostało jeszcze bezpieczeństwo tego języka – było ono jednym z celów przyświecających jego twórcom. Potrzebna była metoda bezpiecznego przesyłania danych pomiędzy klientem i serwerem, więc język Java został zaprojektowany tak, by sprostać tym wymaganiom.

Aby zapewnić odpowiednie oprogramowanie po stronie klienta i serwera, odpowiedni komponent języka Java jest dodawany do aplikacji innego typu, głównie do przeglądarek internetowych. Zarówno Microsoft Internet Explorer, jak i Netscape Navigator zawierają interpreter języka Java (w postaci wtyczki, ang. *plug-in*). Kiedy wykryta zostanie transmisja kodu w języku Java, uruchamiany jest interpreter tego języka obsługujący nadchodzące dane.

JavaScript został wprowadzony na rynek nieco później niż sam język Java. Jest on wbudowany w większość przeglądarek internetowych obsługujących język Java. Poza nazwami, JavaScript i Java nie mają ze sobą wiele wspólnego. Wiele osób uważa, że JavaScript to okrojona wersja języka Java, ale jest to pogląd błędny. JavaScript to swego rodzaju rozszerzenie języka HTML pozwalające na tworzenie interaktywnych stron WWW w systemach o architekturze klient-serwer.

JavaScript ma wiele zastosowań decydujących o jego atrakcyjności – na przykład z jego pomocą można określić, co w danej chwili robi użytkownik. Kiedy opuszcza on stronę WWW albo wybiera któryś z przycisków, klient JavaScript może obsługiwać generowane komunikaty i uruchomić odpowiednie procedury. JavaScript nadaje się również znakomicie do zadań porządkowych i obsługi niektórych aspektów programowania wykraczających poza zakres języka HTML, na przykład obsługi łańcuchów znaków.

Co będzie Ci potrzebne?

Jeśli chcesz tworzyć programy w języku Java, będziesz potrzebował pakietu JDK (Java Development Kit). Zawiera on komplet programów niezbędnych do tworzenia, komplikowania i testowania aplikacji Javy. Oprócz JDK potrzebna będzie również przeglądarka WWW obsługująca język Java, dzięki której będziesz mógł oglądać efekty swojej pracy. Interpreter języka Java jest zaimplementowany w nowszych wersjach przeglądarek Internet Explorer, Netscape Navigator i innych. Firma Sun opracowała również własną przeglądarkę obsługującą ten język, o nazwie HotJava, dostępną na stronach WWW firmy Sun.



Pakiet JDK jest dostępny na wielu stronach WWW i w węzłach FTP. Szukanie możesz rozpocząć od strony WWW firmy Sun, pod adresem <http://java.sun.com>, choć w większości węzłów oferujących oprogramowanie dla Linuxa (na przykład www.blackdown.org) dostępne są odnośniki prowadzące również do innych serwerów oferujących pakiet JDK. Jeśli potrzebna Ci jest przeglądarka WWW obsługująca język Java, zatrzymaj się na stronie firm Netscape lub Microsoft lub na stronie firmy Sun (przeglądarka HotJava). Informacji o innych przeglądarkach i narzędziach wspomagających tworzenie programów w języku Java szukaj na stronach WWW poświęconych systemowi Linux.

Jeśli chcesz załadować JDK za pośrednictwem FTP, połącz się z węzłem java.sun.com i przejdź do katalogu /pub, w którym znajdziesz potrzebne pliki. Niestety - jeśli spodziewasz się, że firma Sun wspiera JDK dla Linuxa, będziesz rozczarowany - odmawiają oni wszelkiego rodzaju pomocy.

Na serwerze firmy Sun dostępne są również inne materiały przydatne przy tworzeniu aplikacji w języku Java, takie jak przykładowe programy, porady czy dokumenty zawierające odpowiedzi na często zadawane pytania. Dokument dostępny pod adresem <http://java.sun.com/tutorial/java/index.html> zawiera opis języka i jest doskonałym punktem wyjścia do jego nauki. Pakiet Java Development Kit jest darmowy, pod warunkiem, że nie jest wykorzystywany w celach komercyjnych; jeśli jednak zamierzasz publikować strony WWW oparte na języku Java, możesz potrzebować licencji. Szczegółowe informacje rozpowszechniane są wraz z JDK.

Jeśli instalujesz JDK, upewnij się, że zainstalowane pliki wykonywalne znajdują się w katalogach wchodzących w skład ścieżki przeszukiwania. Najlepiej utwórz dowiązania do nich w katalogu /usr/bin. Początkujący programiści najczęściej popełniają błędy związane z używaniem klas i z tym, że nie przestrzegają właściwej wielkości liter. W języku Java, podobnie jak w systemach UNIX-owych, wielkość liter ma znaczenie, należy więc zwrócić uwagę na ten aspekt tworzenia kodu źródłowego. Rozwiązań dotyczących użycia klas pochodzą z języka C++ i trzeba je opanować. Nie sposób nauczyć się języka Java bez dobrego podręcznika – na szczęście na rynku jest obecnie sporo książek na ten temat.

Jeśli chcesz używać języka JavaScript, będziesz potrzebował przeglądarki potrafiącej go obsługiwać (takiej jak na przykład Netscape Navigator w wersji 3.0 lub wyższej), dowolnego edytora tekstów potrafiącego zapisywać pliki w formacie ASCII i połączenia TCP/IP pozwalającego komunikować się z innymi komputerami w sieci lokalnej czy Internecie.

Język Java

Z punktu widzenia programisty Java jest bardzo okrojonym obiektowym językiem programowania. Język Java jest językiem obiektowym głównie dlatego, że obiektowe meto-

dy programowania sprawdzają się przy tworzeniu aplikacji opartych o architekturę klient-serwer. Dzięki temu uniknięto tworzenia struktur charakterystycznych dla języków proceduralnych, powodujących znaczne zwiększenie objętości programów i spowolnienie ich działania. Również dzięki programowaniu obiektowemu można obejść wiele problemów, które wymagałyby tworzenia odpowiednich procedurinicjalizacyjnych. Pozwala to w jeszcze większym stopniu zmniejszyć i przyspieszyć programy w języku Java. Opanowanie tego języka nie jest szczególnie trudne, prawdopodobnie jednak łatwiej będzie Ci zrozumieć panujące w nim reguły, jeśli znasz już jakiś język obiektowy, na przykład C++, ponieważ Java nie ma wiele wspólnego z takimi językami proceduralnymi, jak C czy BASIC.

Java nie jest do końca językiem obiektowym. Język Smalltalk, dla przykładu, uważany jest za język czysto obiektowy, ponieważ każdy jego aspekt opiera się na zastosowaniu obiektów i komunikatów, a wszystkie dane są reprezentowane przez obiekty. Język Java nie jest aż tak „ortodoksyjny”, dzięki czemu tworzone programy mogą być mniejsze. W języku Java proste typy danych dostępne również w języku C (liczby całkowite, rzeczywiste i znaki) są zaimplementowane w sposób nieobiektowy, ale wszystkie inne rozwiązania mają charakter obiektowy. Zaletą takiego rozwiązania jest fakt, że pozwala ono tworzyć mniejsze i szybsze programy.

Jak wspomniano wcześniej, Java jest językiem interpretowanym. Program w tym języku tworzy się tak samo jak w innych językach programowania, następnie przetwarza się go za pomocą swego rodzaju translatora, tworzącego binarny plik pośredni. Plik taki nazywa się klasą i nie nadaje się do bezpośredniego przeglądania, ale może być przesyłany szybciej, niż kod źródłowy, na podstawie którego został on wytworzony. Kiedy przeglądarka zawierająca interpreter Javy odbiera plik klasy, uruchamia interpreter, który dekoduje poszczególne instrukcje i wykonuje je. W pewnym sensie Java jest zarówno językiem kompliwowanym, jak i interpretowanym, podobnie jak wiele wcześniejszych języków programowania – programy w nich napisane były rozprowadzane w postaci pseudokodu wymagającego do poprawnego działania odpowiednich bibliotek czasu wykonania (kiedyś w taki sposób działało wiele kompilatorów języka Pascal). Program klienta, który dekoduje plik klasy, przeprowadza tłumaczenie ogólnych instrukcji zapisanych w tym pliku na instrukcje specyficzne dla sprzętu i systemu operacyjnego, w którym program ma zostać uruchomiony.

Oswojenie się z programowaniem w języku Java może zabrać dłuższą chwilę. Programy mogą często wydawać się zlepkiem instrukcji z języków C i C++, dlatego doświadczenie w programowaniu w obu tych językach będzie dość pomocne.

Mimo tego, Java jest językiem dość prostym i może być w krótkim czasie opanowana nawet przez użytkowników nie posiadających żadnego doświadczenia programistycznego. Najtrudniejszym aspektem zwykle okazuje się zrozumienie reguł rządzących programowaniem obiektowym i przyzwyczajenie się do nieco dziwnie wyglądającej składni (znanej z języków C i C++). Oto przykład króciutkiego programu w języku Java:

```
// AhojApp.java
class AhojApp {
    public static void main (String args[]) {
        System.out.println("Ahoj, przygodo!");
```

```
    }  
}
```

Powyższy program (nazwany AhojApp) definiuje tylko jedną metodę o nazwie `main`, która nie zwraca żadnej wartości (`void`), a jej treścią jest pojedyncza instrukcja powodująca wyświetlenie komunikatu `Ahoj, przygodo.`. Pierwszy wiersz powyższego przykładu jest komentarzem. Jeśli chcesz skompilować powyższy kod źródłowy do pliku klasy, powinieneś uruchomić kompilator `javac`

```
javac AhojApp.java
```

który wygeneruje plik o nazwie `AhojApp.class`, nadający się do uruchomienia w dowolnej przeglądarce WWW obsługującej język Java lub też za pomocą programu klienta uruchamianego z wiersza poleceń:

```
java AhojApp
```

Powyższe polecenie nakazuje interpreterowi załadowanie pliku o nazwie `AhojApp.class` i uruchomienie go. Ten applet musi być uruchamiany z wiersza poleceń, jeśli chcesz zobaczyć rezultaty jego działania.

Jak się pewno domyślasz, język Java jest o wiele bardziej złożony niż można by sądzić na podstawie tego przykładu, ale jeśli masz jakieś doświadczenie w programowaniu, zauważysz mnóstwo cech wspólnych z innymi językami (szczególnie takimi jak C i C++). Język Java jest dość bogaty i aby go opanować w pełni być może będziesz potrzebował nawet kilku tygodni, ale mimo wszystko dla większości użytkowników jest on łatwiejszy niż inne języki programowania.

JavaScript i HTML

Polecenia języka JavaScript są osadzane w dokumentach HTML (tworzenie dokumentów w języku HTML omówiliśmy dokładniej w rozdziale 53. „Podstawy języka HTML”) przez otoczenie ich znacznikami `<SCRIPT>` i `</SCRIPT>`. Mówiąc najbardziej ogólnie, składnia programu w języku JavaScript osadzonego w dokumencie HTML jest następująca:

```
<SCRIPT language="JavaScript">  
    polecenia języka JavaScript  
</SCRIPT>
```

Pole `language` znacznika `<SCRIPT>` jest opcjonalne, ale warto je wstawić by mieć pewność, że program zostanie prawidłowo rozpoznany przez przeglądarkę. Jeśli chcesz załadować skrypt w języku JavaScript podając jego adres URL, również ten adres powinieneś umieścić wewnątrz znacznika `<SCRIPT>`:

```
<SCRIPT language="JavaScript" src="http://www.jakis.adres.com">
```

Jeśli źródło JavaScript jest osadzone w pliku HTML, można pominąć pole `src`. Oto bardzo prosty przykładowy skrypt osadzony w dokumencie HTML (który również ze względu na przejrzystość przykładu został ograniczony do niezbędnego minimum):

```
<HTML>
<HEAD>
...
<SCRIPT language="JavaScript">
    alert("Witaj na mojej stronie WWW!");
</SCRIPT>
</HEAD>
</HTML>
```

Funkcja `alert()` języka JavaScript powoduje wyświetlenie okienka dialogowego zawierającego prócz tekstu pictogram przedstawiający symbol wykrywnika. Jest ona używana do przyciągnięcia uwagi użytkownika w sytuacjach, gdy podejmowana akcja ma kluczowe znaczenie, jest nieprawidłowa lub może spowodować problemy. Jak widać, funkcje w języku JavaScript wykorzystywane są podobnie jak w języku C. Możliwe jest również umieszczenie treści funkcji w osobnym pliku i odwołanie się do niej w dokumencie HTML.

Jeśli wywołujesz z dokumentu HTML funkcję zapisaną w jakimś innym pliku, powinieneś do jego nazwy dodać rozszerzenie `.js`. Warto trzymać się tej konwencji, ponieważ niektóre aplikacje (również MIME) potrafią poprawnie rozpoznać i obsłużyć pliki z rozszerzeniem `.js`.

Niestety, nie mamy do dość miejsca na to, aby zagłębiać się w szczegóły programowania w języku JavaScript – na szczęście na rynku dostępnych jest obecnie wiele książek poświęconych temu tematowi.

Podsumowanie

W tym rozdziale przedstawiliśmy języki Java i JavaScript, oba dostępne dla systemów linuxowych - zarówno w wersji dla serwera, jak i dla klienta. Programowanie w tych językach jest łatwiejsze dla użytkowników mających pewne doświadczenie; ich zastosowanie pozwala znacznie wzbogacić strony WWW i inne dokumenty HTML. Zresztą spróbuj sam – przecież potrzebne oprogramowanie jest darmowe.

Jeśli chcesz dowiedzieć się więcej o języku Perl, doskonale nadającym się do tworzenia skryptów CGI, zajrzyj do rozdziału 28. „Perl”.

Rozdział 26. „Programowanie w języku C” przybliży Ci nieco język C, który również może być wykorzystany do tworzenia skryptów CGI.

Język HTML jest omówiony dokładniej w rozdziale 53. „Podstawy języka HTML”.

Jeśli chcesz zabezpieczyć się przed możliwością utraty wszystkich napisanych przez Ciebie skryptów CGI, zajrzyj do rozdziału 45. „Kopie zapasowe”.

Rozdział 55.

Projektowanie

spójnych stron WWW

Tim Parker

W tym rozdziale:

- v Dostępność systemu
- v Utrzymywanie porządku na stronach WWW

W tej części książki pokazaliśmy już, w jaki sposób można skonfigurować system linuxowy, aby działał jako serwer usług internetowych, takich jak FTP, Gopher, Wais czy WWW. W poprzedniej części omówiliśmy również konfigurowanie poczty i grup dyskusyjnych. Większość użytkowników łączących się z Internetem chce jednak korzystać z WWW, bądź to łącząc się z innymi serwerami, bądź też udostępniając własne strony WWW.

Internet bardzo się rozrosł w ciągu kilku ostatnich lat, co spowodowało, że wielu użytkowników postanowiło udostępnić w tej sieci informacje dotyczące interesujących ich tematów. Wiele stron WWW jest jednak źle zorganizowanych, co powoduje, że ładują się one bardzo powoli. W tym rozdziale przyjrzymy się kilku podstawowym zasadom, których należy przestrzegać przy tworzeniu własnej strony WWW w systemie linuxowym. Tematowi prawidłowego projektowania stron WWW poświęcone są całe książki, nie mniej postaramy się tu króciutko naświetlić najważniejsze aspekty tego zagadnienia.

Nie ma żadnego znaczenia, czy zakładasz stronę WWW po to, by pochwalić się reszcie świata faktem, że zainstalowałeś Linuxa, czy też zamierzasz poświęcić ją swojemu ulubionemu programowi telewizyjnemu. Każda strona WWW, która będzie udostępniana użytkownikom Sieci, powinna spełniać kilka podstawowych warunków dotyczących działania serwera i wyglądu samej strony.

Dostępność systemu

Jednym z najbardziej drażniących problemów występujących podczas korzystania z Sieci jest fakt, że niektóre strony WWW ładują się potwornie długo (a nawet pojawiają się komunikaty mówiące o tym, że danej strony w ogóle nie można odnaleźć). Długie czekanie na załadowanie strony bardzo skutecznie zniechęca użytkowników do jej odwiedzania. W grę wchodzą dwa czynniki: dostępność systemu i czas oczekiwania na odpowiedź.

System jest dostępny wtedy, gdy strony WWW są dostępne w Internecie. Jeśli wyłączysz swój serwer w czasie, gdy z niego nie korzystasz, nikt inny również nie będzie miał do niego dostępu. Każdy, kto będzie próbował obejrzeć Twoją stronę WWW, otrzyma komunikat o nieprawidłowym adresie URL. Większość użytkowników w takiej sytuacji po prostu usunie odnośnik do Twojej strony ze swojego zbioru adresów i już nigdy jej nie odwiedzi.

Rozwiązywanie nasuwa się samo: serwer powinien pozostawać włączony 24 godziny na dobę, co najwyżej za wyjątkiem krótkich przerw poświęconych na konserwację systemu. Nawet jeśli nie spodziewasz się, aby ktokolwiek chciał oglądać Twój stronę o czwartej nad ranem, musisz pamiętać o fakcie, że Sieć ma zasięg ogólnoszczególny i dla kogoś w innej strefie czasowej jest właśnie późne popołudnie. Zastosowanie systemu UPS (Uninterruptible Power Supply, system podtrzymywania zasilania) pozwoli również uniezależnić się w pewnym stopniu od krótkotrwałych awarii zasilania czy wahania napięcia sieciowego – warto więc w niego zainwestować.

Innym problemem jest czas oczekiwania na odpowiedź. Zależy on głównie od dwóch czynników: prędkości komputera i prędkości połączenia z siecią. Jeśli używasz modemu o maksymalnej prędkości przesyłu 1200 bodów, bez względu na to, jak szybka będzie reszta Twojego sprzętu, połączenia będą kuleć z powodu zbyt małej wydajności łącza. Nie oznacza to wcale, że musisz od razu zainwestować w łącza ISDN czy T1 – zwykły modem asynchroniczny o prędkości 28.8 kbps lub większej powinien w zupełności wystarczyć do obsługi większości rzadko odwiedzanych serwerów WWW. Jeśli jednak zaczyna odwiedzać Cię coraz więcej użytkowników i połączenia są realizowane coraz wolniej, powinieneś rozważyć zakup szybszego łącza. Mimo wszystko zdecydowana większość stron WWW jest odwiedzana bardzo rzadko.

Jeśli połączenie z Internetem jest wystarczająco szybkie, prędkość komputera zwykle nie odgrywa decydującej roli. Komputer klasy 80486 może wysyłać dane do modemu o wiele szybciej, niż ten potrafi transmitować je poprzez linię telefoniczną, nie ma więc większego znaczenia, czy używasz komputera Pentium, czy nie, chyba że prowadzisz bardzo popularną stronę WWW – wówczas wszystkie elementy muszą być bardzo szybkie.

Utrzymywanie porządku na stronach WWW

Jednym z największych błędów popełnianych przez użytkowników udostępniających swoje pierwsze strony WWW jest ich zbytnie komplikowanie. Oczywiście, jeśli stworzyłeś jakąś zwalającą z nóg animację w języku Java i chcesz, by wszyscy ją obejrzeli, wszystko jest w porządku, ale pamiętaj o tym, że każdy, kto połączy się z Twoją stroną, będzie musiał załadować cały kod. Może to zabrać sporo czasu, szczególnie za pośrednictwem modemu asynchronicznego. Jeśli zaprojektowałeś wspaniale wyglądającą w Twoim systemie stronę z purpurowym tekstem na zielonym tle, musisz pamiętać o tym, że w innym systemie może ona być zupełnie nieczytelna, ponieważ nie zawsze kolory zostaną oddane prawidłowo. Wszystkie te wskazówki trzeba mieć na uwadze przy projektowaniu stron WWW. Podstawową regułą jest, że strony WWW powinny być możliwie najprostsze.

Umieszczaj najważniejsze informacje na początku strony

Przy projektowaniu stron WWW bardzo ważne jest, aby wszystko, co próbujesz przekazać, było jasne i zrozumiałe. Jeśli tworzysz stronę prezentującą wyroby Twojej firmy, jej nazwa i rodzaj działalności powinny pojawić się w widocznym miejscu, tak, by były dostrzegalne na pierwszy rzut oka. Pamiętaj o tym, że większość ludzi czyta tekst od góry do dołu, więc istotne informacje umieszczaj na początku dokumentu. Ta sama zasada obowiązuje przy drukowaniu ogłoszeń i artykułów prasowych: nagłówki i ważne informacje prezentowane są w górnej części strony, czyli tam, gdzie przeciętny czytelnik spojrzy najpierw, po to, aby przyciągnąć jego uwagę i sprowokować do przeczytania reszty informacji. Większość czytelników zaczyna czytanie od początku strony, przerywając je po natknięciu się na coś, co ich zniechęci – wynika stąd, że zaprojektowanie prostej i niezaśmieconej strony daje większe szanse na to, że użytkownik dobrnie do jej końca.

Dla przykładu rozważmy sytuację, w której chcesz udostępnić użytkownikom jakieś oprogramowanie dla systemów linuxowych. Jeśli odnośnik do tego oprogramowania umieścisz na końcu swojej strony WWW, większość użytkowników prawdopodobnie go przegapi, szczególnie jeśli zauważenie go będzie wymagało przewinięcia zawartości strony. Choć wiele stron WWW jest dość długich, tak naprawdę mało kto ma ochotę przewijać je do samego końca. Z tego powodu zwykle dobrym pomysłem jest ograniczenie konieczności przewijania strony do niezbędnego minimum; idealną sytuacją byłoby całkowite wyeliminowanie przewijania. Umieszczenie odnośników do oferowanego oprogramowania na początku strony znacznie zwiększa szansę na to, że ktoś pokusi się o jego załadowanie – o wiele łatwiej kliknąć na wyróżniającym się odnośniku niż przewijać zawartość strony szukając go gdzieś na dole czy też pomiędzy innymi elementami.

Dzielenie dokumentu na wiele stron

Jeśli treść, którą chcesz przekazać, nie mieści się w obrębie jednej strony, jak najbardziej wskazane jest dołączenie kolejnej strony i umieszczenie na każdej z nich odpowiednich odnośników. Należy jednak zwrócić uwagę na to, by użytkownik poszukujący konkretnych informacji nie musiał przechodzić przez zbyt wiele poziomów odnośników. Jeśli na przykład chcesz za pośrednictwem sieci sprzedać swoje wyroby lub usługi, nie jest najlepszym pomysłem zmuszenie czytelnika do przejścia przez sześć czy siedem poziomów odnośników, zanim będzie mógł dowiedzieć się czegoś konkretnego o Twojej ofercie. Przeciętny użytkownik po prostu nie jest aż tak cierpliwy. Bardziej przemyślane zaplanowanie strony i kilka eksperymentów pozwoli stworzyć stronę dobrze zaprojektowaną i wygodną w użyciu. Pamiętaj o tym, by wszystkie informacje organizować w możliwie logiczny sposób, dzięki czemu odwiedzający będą mogli szybko oswoić się ze strukturą Twojej strony.



Istnieje kilka prostych sposobów przyciągnięcia uwagi czytającego do pewnych elementów strony WWW. Jednym z nich jest umieszczenie za pomocą znacznika <HR> poziomej linii w odpowiednim miejscu strony. Linia taka jest definiowana przez zaledwie kilka znaków, więc nie wprowadza w zasadzie żadnego opóźnienia przy przesyłaniu danych. Mimo tego, powinieneś utrzymywać liczbę takich linii na rozsądny poziomie, ponieważ ich nadmiar może przetoczyć użytkownika.

Ikony

Ikony to doskonały sposób zwracenia uwagi czytelnika na najważniejsze elementy Twojej strony głównej. Niestety, wielu projektantów stron WWW przesadza i umieszcza na stronie mnóstwo porozrzucanych bezładnie ikon. Nie daje to zamierzonego efektu – rozprasza uwagę czytelnika nie pozwalając ocenić, które z ikon są istotne, a które nie.

Niewielkie pictogramy nadają się również do wyróżniania kolejnych elementów wyliczeń, pod warunkiem że są to krótkie wyliczenia, które można łatwo ogarnąć wzrokiem. Również tu trzeba zdecydować się na coś prostego, a jednocześnie przyciągającego wzrok. Użycie innego pictogramu dla każdego elementu listy nie jest najlepszym pomysłem. Najprostszy z takich pictogramów może mieć postać kulki, ale niektóre przeglądarki automatycznie dodają tego typu pictogram przed elementami list.



Atrakcyjny pictogram wskazujący na istotne elementy strony może naprawdę ją ożywić. Biblioteki klipartów są pełne obrazków przedstawiających różnego rodzaju strzałki czy dlonie z wyciągniętym palcem wskazującym (bywają też inne palce, ale one średnio nadają się one do zamieszczenia na stronie WWW), które można zastosować do zaakcentowania ważnych fragmentów tekstu. Tego typu

elementy przyciągają wzrok czytającego.

Potencjalne przyczyny problemów warto oznaczyć piktogramem przedstawiającym znak ostrzegający o niebezpieczeństwie (zwykle żółty trójkąt z symbolem wykrzyknika) lub znak stop. One szczególnie zwracają na siebie uwagę, ale należy używać ich z umiarem i tylko tam, gdzie są uzasadnione.

Ikona „Nowość” w najróżniejszych formach jest przydatna do wyróżniania elementów, które pojawiły się na stronie WWW niedawno – dzięki temu osobom regularnie odwiedzającym stronę WWW łatwiej będzie orientować się we wprowadzanych modyfikacjach. Powinieneś jednak upewnić się, że odnosi się ona do elementów, które rzeczywiście są nowe. Usuwaj takie ikony co parę tygodni, aby Twоя strona nie wydawała się archaiczna i zaniedbana.

Prawidłowe stosowanie odnośników

Stosowanie odnośników to jeden z warunków utworzenia dobrej strony WWW, ale powinny one być dobrze przemyślane. Wielu projektantów stron WWW zaczyna od sporządzenia na kartce papieru diagramu obrazującego strukturę połączeń pomiędzy poszczególnymi stronami, dzięki czemu mogą oni mieć pewność, że struktura jest jasna, logiczna i nie zawiera błędów takich jak na przykład przypadkowo utworzona pętla. Jeśli Twoja witryna ma składać się z więcej niż trzech czy czterech stron, powinieneś również pomyśleć o stworzeniu tego typu diagramu.

Wybór tekstu, który będzie reprezentował dany odnośnik, również nie jest sprawą banalną. Przykładowo, poniższy fragment tekstu nie spełnia najlepiej swego zadania:

Kliknij tu by zobaczyć nowe produkty.

O wiele lepiej prezentuje się tekst o następującej postaci:

Tu możesz obejrzeć informacje o naszych najnowszych produktach.

Prawidłowe stosowanie znaczników HTML

Osoby tworzące strony WWW często nieprawidłowo używają również nagłówków. Za ich pomocą można wyróżnić tytuł używając większej czy wytłuszczonej czcionki. Należy jednak stosować je z umiarem i zachowywać odpowiedni porządek. Po znacznikach <H1> i <H2> powinien wystąpić znacznik <H3> i tak dalej, w miarę potrzeb. Nie należy pomijać jednego z poziomów nagłówków, ponieważ niektóre przeglądarki na ich podstawie planują wygląd strony.

Należy również unikać zagnieźdzania znaczników, szczególnie tych powodujących zmianę rodzaju czcionki. Dla przykładu, poniższy fragment kodu:

```
<STRONG> Ahoj, <EM> Przygoda! </EM> </STRONG>
```

może prezentować się poprawnie w jednej przeglądarce, ale inna może nie zinterpretować go zgodnie z zamierzeniami autora. Zawsze zamykaj otwarte znaczniki i unikaj mieszania ich.

Znaczniki powodujące zmianę kroju pisma powinny zawsze znajdować się wewnątrz znaczników akapitu, a nie odwrotnie. Oto przykład prawidłowego umieszczenia znacznika powodującego podkreślenie fragmentu tekstu:

```
<A HREF="Najświeższe wiadomości">
<UL>
    <LI> Wiadomości dzisiejsze
    ...
</UL>
</A>
```

Unikaj używania znacznika `
`, wymuszającego przejście do nowego wiersza. Zamiast tego pozwól przeglądarce rozmieścić tekst według własnego uznania – w przeciwnym przypadku może okazać się, że Twój strona w niektórych sytuacjach wygląda dość dziwnie. Pamiętaj o tym, że fakt, że strona wygląda dobrze w Twoim systemie, nie oznacza, że będzie ona prezentować się tak samo przeglądającym ją użytkownikom korzystającym z innej przeglądarki czy systemu.

Podsumowanie

Przedstawienie wszystkich zasad, których należy się trzymać podczas tworzenia strony WWW na kilku stronach jest niestety niemożliwe. Dostępnych jest mnóstwo dobrych książek dotyczących tego zagadnienia. Weź sobie przedstawione tu wskazówki do serca i zaprojektuj jasną i niezamieconą stronę WWW – w przeciwnym przypadku może się okazać, że konfiguracja serwera WWW była stratą czasu, ponieważ nikt go nie odwiedza.

Jeśli chcesz zabezpieczyć swoje strony HTML, dzięki czemu będziesz mógł je z łatwością odtworzyć w razie problemów, zajrzyj do rozdziału 45. „Kopie zapasowe”.

Rozdział 51. „Konfiguracja serwera WWW” omawia problemy dotyczące konfigurowania i instalowania oprogramowania obsługującego serwer WWW.

Jeśli chcesz dowiedzieć się czegoś więcej o systemach zarządzania kodem źródłowym, dzięki którym nie trzeba przechowywać wielu kopii dokumentów HTML, zajrzyj do rozdziału 56. „Zarządzanie kodem źródłowym”.

Część ósma

Programowanie

dla zaawansowanych

W tej części:

- υ Zarządzanie kodem źródłowym
- υ Jądro systemu
- υ Tworzenie sterowników urządzeń
- υ Projekt WINE
- υ HylaFAX
- υ Gry
- υ Adabas-D i inne bazy danych
- υ StarOffice
- υ Program Lone-Tar firmy Lone Star Software

Rozdział 56.

Zarządzanie kodem źródłowym

Peter MacKinnon i Tim Parker

W tym rozdziale:

- υ Program `make`
- υ RCS
- υ Uzyskiwanie informacji o wersji z pliku RCS
- υ Dostęp do pliku RCS
- υ Porównywanie i łączenie poprawek
- υ Praca z programem `make` i systemem RCS

Duże projekty programistyczne, w których skład wchodzi wiele plików źródłowych tworzonych przez różnych programistów, mogą sprawiać mnóstwo kłopotów, szczególnie jeśli to właśnie Ty jesteś odpowiedzialny za zarządzanie całością projektu. Oto najczęściej występujące problemy:

- υ skąd można mieć pewność, że plik zawierający np. biblioteki wejścia / wyjścia napisane przez innego programistę zawiera najnowszą wersję procedur?
- υ jak poradzić sobie w sytuacji, gdy trzeba przekompilować aplikację, ale nie bardzo wiadomo, które z 50 plików wchodzących w skład kodu źródłowego zostały zmodyfikowane?

Okazuje się, że śledzenie najnowszych wersji plików i szukanie w razie potrzeby wersji sprzed godziny czy też sprzed kilku dni może zająć więcej czasu niż samo programowanie.

Nawet niewielkie aplikacje zwykle tworzone są na podstawie więcej niż jednego pliku zawierającego kod źródłowy. Podczas kompilowania programów w języku C musisz

posługiwać się nie tylko plikami z kodem źródłowym, ale również plikami nagłówkowymi i plikami bibliotek. Na szczęście, dla systemu Linux dostępne jest środowisko programistyczne znakomicie upraszczające większość zadań związanych z zarządzaniem kodem źródłowym.

Program make

Program `make`, będący prawdopodobnie najważniejszym programem użytkowym wykorzystywanym podczas tworzenia oprogramowania, pozwala określić zależności pomiędzy plikami i uaktualniać tylko te pliki, które tego wymagają. Uaktualnianie zwykle sprowadza się do komplikacji odpowiednich plików źródłowych czy też konsolidacji plików pośrednich, ale w skład tego procesu może wchodzić również usuwanie plików tymczasowych. Proces uaktualniania może być powtarzany dziesiątki razy w trakcie tworzenia aplikacji. Zamiast zajmowania się związanymi z nim zadaniami ręcznie, można zlecić je programowi `make`, zyskując więcej czasu na ważniejsze czynności, takie jak tworzenie samego kodu źródłowego czy oglądanie telewizji.

Program `make` generuje odpowiednie polecenia na podstawie pliku opisu nazywanego najczęściej `makefile`. Wygenerowane polecenia są następnie przetwarzane przez interpreter poleceń powłoki. Zawartość pliku `makefile` to w zasadzie zestaw zadań (zwanych regułami), które należy wykonać przy uaktualnianiu projektu. Definiowanie reguł sprowadza się do określenia zależności pomiędzy plikami. W przypadku tworzenia pliku wykonywalnego z kodu w języku C w systemie Linux zwykle oznacza to komplikację plików źródłowych do postaci pośredniej, a następnie konsolidację plików pośrednich (być może wraz z dodatkowymi bibliotekami) do postaci wykonywalnej. Program `make` potrafi również sam określić, które z plików zostały zmodyfikowane i wymagają uaktualnienia (na podstawie czasu ostatniej modyfikacji pliku).



Nazwa `makefile` czy `Makefile` jest faktycznie nazwą pliku, który program `make` spodziewa się znaleźć w katalogu bieżącym. Można zażądać przetwarzania pliku o innej nazwie, ale zgodnie z tradycją (a kto chciałby się spierać z ponad trzydziestoletnią tradycją?) programiści używają pliku o nazwie `makefile`.

Program `make` jest zdecydowanie najlepiej przystosowany do programowania w języku C, ale można również wykorzystywać go przy pracy z innymi językami programowania dla Linuxa, na przykład z asemblerem czy językiem FORTRAN.

Przykładowy plik makefile

Przyjrzyjmy się prostemu przykładowi zastosowania programu `make`.

Polecenie

```
$make cosnowego
```

powoduje, że zostanie utworzona nowa wersja programu `cosnowego`. W naszym przypadku będzie to plik wykonywalny, w związku z czym proces ten będzie składał się z komplikacji i konsolidacji odpowiednich plików. Plik `cosnowego` określa się jako plik docelowy (ang. *target*) programu `make`. Pliki pośrednie konsolidowane do jednego pliku wykonywalnego nazywane są plikami pierwotnymi (ang. *dependents*) pliku `cosnowego`. Pliki zawierające kod źródłowy są kompilowane do postaci pośredniej, są więc również (choć nie wprost) plikami pierwotnymi pliku `cosnowego`.

Oto lista plików, które będą potrzebne do utworzenia programu `cosnowego` (ich zawartość jest dla naszego przykładu nieistotna):

- ✓ dwa pliki zawierające kod źródłowy w języku C, `main.c` i `zrobto.c`;
- ✓ trzy pliki nagłówkowe, `tak.h`, `nie.h` i `moze.h`;
- ✓ jeden plik zawierający bibliotekę funkcji, `/usr/nowe/lib/swiezy.a`;
- ✓ plik zawierający kod źródłowy w asemblerze, `szyciej.s`.

Jak widać, nasz projekt nie jest duży, więc nie byłoby szczególnie trudno ręcznie kompilować i konsolidować odpowiednie pliki. Zamiast tego jednak spróbujmy utworzyć odpowiedni plik `makefile`, pozwalający zautomatyzować te niezbyt fascynujące czynności.

Używając swojego ulubionego edytora tekstów, utwórz plik `makefile` o następującej treści:

```
cosnowego: main.o zrobto.o szybciej.o /usr/nowe/lib/swiezy.a
          cc -o cosnowego main.o zrobto.o szybciej.o /usr/nowe/lib/swiezy.a
main.o: main.c
       cc -c main.c
zrobto.o: zrobto.c
       cc -c zrobto.c
szyciej.o: szyciej.s
       as -o szyciej.o szyciej.s
sprzątaj:
       rm *.o
moze.h: tak.h nie.h
       cp tak.h nie.h /usr/reksio/
```

Format pliku makefile

Załóżmy, że wszystkie wspomniane pliki znajdują się w tym samym katalogu, co plik `makefile`; co zyskaliśmy, tworząc go? Każdy plik `makefile` (więc również ten, który przed chwilą utworzyłeś) składa się z pewnej liczby wpisów. Przykładowy plik składa się z sześciu wpisów. Pierwszy wiersz każdego wpisu definiuje zależności pomiędzy plikami. Pozwala on określić (po dwukropku), jakie pliki są niezbędne do utworzenia pliku (obiektu) docelowego o nazwie zapisanej przed dwukropkiem. Następne wiersze

określają zestaw poleceń, które zostaną wykonane przez program `make`, gdy zajdzie potrzeba uaktualnienia danego pliku docelowego. Pojedynczy wpis ma więc postać:

```
plik docelowy: pliki potrzebne do jego utworzenia  
(TAB) lista poleceń
```

Wiersze określające polecenia, które mają zostać wykonane, muszą rozpoczynać się od znaku tabulacji – jest to jedna z reguł składniowych pliku `makefile`. Wierszy określających polecenia może być więcej, ale każdy z nich jest wykonywany oddzielnie, w swoim własnym środowisku. Wynika stąd, że następujące polecenia:

```
cd gdzieś  
mv *.c gdzieindziej
```

nie zadziałają zgodnie z oczekiwaniemi. Aby zapobiec tego rodzaju sytuacjom, należy polecenia, które mają zostać wykonane wspólnie, zapisywać w jednym wierszu, rozdzielając je średnikami. Można to zrobić na dwa sposoby:

```
plik docelowy: pliki potrzebne do jego utworzenia  
    polecenie1; polecenie2; polecenie3;...
```

albo:

```
plik docelowy: pliki potrzebne do jego utworzenia  
    polecenie1; \  
    polecenie2; \  
    polecenie3;
```

i tak dalej. Jeśli używasz drugiego sposobu, ukośnik musi być ostatnim znakiem wiersza, dzięki czemu zapobiegnie on interpretacji znaku końca wiersza.



Można również określić kilka rodzajów zależności pomiędzy plikami, definiując kilka wpisów dla tej samej nazwy pliku docelowego. Program `make` ma o wiele większe możliwości, niż opisane w tym rozdziale, ale prawdopodobnie nie będą Ci one potrzebne, dopóki nie będziesz pracował z naprawdę dużymi projektami, uwikłanymi w mnóstwo wzajemnych zależności.

Pierwszy wpis naszego pliku `makefile` jest kluczowy dla utworzenia pliku wykonywalnego `cosnowego`. Stwierdza on, że plik `cosnowego` może zostać uaktualniony w przypadku, gdy wszystkie pliki pośrednie i pliki biblioteki, od których jest on zależny, istnieją i którykolwiek z nich jest nowszy niż ostatnia wersja pliku `cosnowego`. Oczywiście, jeśli plik `cosnowego` nie istnieje, również zostaną podjęte wszystkie działania prowadzące do jego utworzenia. Najpierw program `make` sprawdzi, czy któryś z plików pierwotnych wymaganych do utworzenia pliku `cosnowego` wymaga uaktualnienia i, w razie potrzeby, podejmie odpowiednie działania. Proces ten powtarzany jest rekurencyjnie – program `make` sprawdza zależności dla plików docelowych na kolejnych poziomach, zgodnie z danymi zapisanymi w pliku `makefile`.

Ostatni wpis jest dość nietypowy – powoduje on skopiowanie plików `tak.h i nie.h` (zależnych w jakiś sposób od pliku `moze.h`) do katalogu domowego użytkownika `reksio` pod warunkiem, że zostały one zmodyfikowane. Takie rozwiązanie może być przydatne na przykład w przypadku, gdy `reksio` pracuje nad jakimś programem związanym z naszym projektem i powinien zawsze mieć dostęp do najnowszej wersji plików nagłówkowych. Jak widać, program `make` może być wykorzystany do celów innych niż tylko kompilowanie i konsolidacja programów – potrafi on wykonywać różne polecenia w oparciu o zadeklarowane zależności.

W naszym pliku `makefile` zdefiniowany jest również plik docelowy o nazwie `sprzataj`, którego uaktualnienie nie powoduje kompilowania żadnych plików. Plik ten nie wymaga istnienia żadnych plików pierwotnych – dla programu `make` nie stanowi to żadnego problemu. Jeśli w katalogu bieżącym nie ma pliku o nazwie `sprzataj`, program `make` wykona wszystkie polecenia powiązane z tym identyfikatorem – a więc usunie pliki pośrednie (z rozszerzeniem `.o`). Plik `sprzataj`, choć dość nietypowy (ponieważ z założenia nie jest nigdy tworzony), jest traktowany na równi z wszystkimi pozostałymi obiektami docelowymi.

Jeśli wydasz polecenie

```
$ make cosnowego
```

program `make` wygeneruje serię poleceń, które spowodują uaktualnienie wszystkich plików pierwotnych pliku docelowego `cosnowego`, a następnie uaktualni sam ten plik. Wszystkie przetwarzane polecenia są wyświetlane na ekranie. Taki sam efekt da wpisanie polecenia

```
$ make
```

ponieważ program `make` wywołany bez argumentów uaktualnia pierwszy plik docelowy napotkany w pliku `makefile`. Przetwarzane polecenia są wyświetlane na ekranie, a po wystąpieniu ewentualnego błędu działanie programu `make` jest wstrzymywane.

Jeśli wszystkie pliki pierwotne i plik `cosnowego` nie wymagają uaktualnienia, żadne polecenia nie zostaną wykonane i program `make` wygeneruje komunikat:

```
'cosnowego' is up to date
```

Parametrem programu `make` może być dowolna nazwa (lub nazwy) pliku docelowego zdefiniowanego w pliku `makefile`. Poszczególne pliki docelowe są uaktualniane w taki kolejności, w jakiej pojawiły się w wierszu poleceń, z zachowaniem wszystkich reguł określonych w pliku `makefile`. Jeśli podana nazwa pliku docelowego nie zostanie odnaleziona w pliku `makefile`, zostanie wygenerowany komunikat:

```
$ make cosinnego
make: Don't know how to make cosinnego. Stop.
```

Tworzenie kilku wersji programu

Załóżmy, że chcesz utworzyć dwie wersje programu `cosnowego`, w większości oparte na tym samym kodzie źródłowym, ale korzystające z różnych bibliotek. Biblioteki te są zapisane w plikach źródłowych w języku C o nazwach `zrobtoc.c` i `zrobtamto.c`. Zamiast tworzyć oddzielne pliki `makefile` dla obu wersji programu, można po prostu zdefiniować dwie wersje pliku docelowego, tworzone w nieco odmienny sposób. Plik `makefile` miałby wówczas postać (pierwsze dwa wiersze są komentarzem, oznaczonym za pomocą symbolu `#`):

```
# Plik makefile sluzacy do tworzenia
# dwoch wersji programu cosnowego
cosnowego1: main.o zrobtoc.o szybciej.o /usr/nowe/lib/swiezy.a
            cc -o cosnowego main.o zrobtoc.o szybciej.o /usr/nowe/lib/swiezy.a
cosnowego2: main.o zrobtamto.o szybciej.o /usr/nowe/lib/swiezy.a
            cc -o cosnowego main.o zrobtoc.o szybciej.o /usr/nowe/lib/swiezy.a
main.o: main.c
        cc -c main.c
zrobtoc.o: zrobtoc.c
        cc -c zrobtoc.c
zrobtamto.o: zrobtoc.c
        cc -c zrobtamto.c
szybciej.o: szybciej.s
        as -o szybciej.o szybciej.s
sprzataj:
        rm *.o
moze.h: tak.h nie.h
        cp tak.h nie.h /usr/reksio/
```

Dzięki takiemu rozwiązaniu za pomocą jednego pliku `makefile` można utworzyć dwie różne wersje programu. Polecenie

```
$ make cosnowego1
```

powoduje utworzenie wersji wykonywalnej programu, używającej funkcji zapisanych w pliku `zrobtoc.c`. Program korzystający z funkcji zapisanych w pliku `zrobtamto.c` może zostać skompilowany za pomocą polecenia

```
$ make cosnowego2
```

Wymuszanie uaktualnienia

Możliwe jest wymuszenie uaktualnienia pliku docelowego lub też wymuszenie zaniechania jego uaktualnienia. Przykładem sytuacji, w której możesz chcieć zabronić ponownej komplikacji plików, może być skopiowanie plików źródłowych do nowego katalogu. Taka operacja powoduje zmianę czasu ostatniej modyfikacji plików, choć w rzeczywistości nie wymagają one rekompilacji. Jeśli chcesz uaktualnić czas modyfikacji plików docelowych

wych określonych w pliku `makefile`, możesz posłużyć się programem `touch` lub uruchomić program `make` z opcją `-t`.



Jeśli chcesz przetestować swój plik `makefile`, możesz uruchomić program `make` z opcją `-n`. Spowoduje to wyświetlenie wszystkich poleceń, które byłyby wykonane, ale bez ich faktycznego wykonywania. Wyświetlone zostaną również komunikaty o ewentualnych błędach składniowych. Dzięki takiemu rozwiązaniu nie trzeba czekać na zakończenie trwającej czasem dość długiego komplikacji.

Makropolecenia

Program `make` pozwala również na definiowanie makropoleceń, rozwijanych do pełnej postaci przed wykonaniem poleceń zapisanych w pliku `makefile`. Definicje makropoleceń mają następującą postać:

```
nazwa_makra = tekst
```

Pole `tekst` może zawierać na przykład nazwę pliku, katalogu, programu, który należy uruchomić, czy dowolny inny tekst. Może to również być lista plików lub stała napisowa ujęta w podwójny cudzysłów. Oto przykłady definicji makropoleceń:

```
LIBFILES = /usr/nowe/lib/swiezy.a
posrednie = main.o zrobto.o
CC = /usr/bin/cc
wersja = "To jest pierwsza wersja programu cosnowego"
OPTIONS =
```

Zgodnie z powszechnie przyjętą konwencją, nazwy makropoleceń składają się tylko z wielkich liter, choć – jak widać w powyższym przykładzie – nie jest to wymagane. Zauważ, że w definicji makropolecenia `OPTIONS` po znaku równości nie występuje żaden tekst. Oznacza to, że w miejsce jego wystąpienia nie zostanie wstawiony żaden ciąg znaków. Podobnie potraktowane zostanie użycie makropolecenia, które w ogóle nie zostało zdefiniowane.

W definicjach makropoleceń można używać innych makropoleceń, na przykład tak:

```
KSIAZKA_DIR = /usr/book/
MOJE_ROZDZ = ${KSIAZKA_DIR}/reksio/ksiazka
```

Makropolecenia muszą być definiowane przed ich użyciem w wierszu określającym zależności pomiędzy plikami, ale mogą odnosić się do siebie wzajemnie w dowolnym porządku.

Program `make` rozpoznaje również kilka predefiniowanych makropoleceń, ułatwiających użycie najczęściej wykorzystywanych programów. Kompilator języka C jest zdefiniowany w makropoleceniu `CC`, natomiast znaczniki, z którymi będzie on uruchomiony, w makropoleceniu `CFLAGS`.

Wywołanie makropolecenia wymaga otoczenia jego nazwy nawiasami klamrowymi i poprzedzenia nawiasu otwierającego symbolem dolara (\$). Oto plik `makefile` z poprzedniego przykładu, w którym wykorzystano możliwość definiowania makropoleceń:

```
# Pora przecwiczyc makropolecenia
CC = /usr/bin/cc
AS = /usr/bin/as
OBJS = main.o zrobto.o szybciej.o
TN = tak.h nie.h
#
LIB_DIR = /usr/nowe/lib
LIB_FILES = ${LIB_DIR}/swiezy.a
cosnowego: ${OBJS} ${LIB_FILES}
        ${CC} -o cosnowego ${OBJS} ${LIB_FILES}
main.o: main.c
        ${CC} -c main.c
zrobto.o: zrobto.c
        ${CC} -c zrobto.c
szybciej.o: szybciej.s
        ${AS} -o szybciej.o szybciej.s
sprzataj:
        rm *.o
moze.h: ${TN}
        cp tak.h nie.h /usr/reksio/
```

W taki sam sposób jak makropolecenia traktowane są nazwy zmiennych środowiskowych, o ile są one zdefiniowane w tym samym środowisku, w którym został uruchomiony program `make`. Jeśli na przykład w powłoce C zdefiniowana jest zmienna środowiskowa o nazwie `KOPIA`

```
$ setenv KOPIA /usr/nowe/backup
można użyć jej w pliku makefile. Definicja
INNA_KOPIA = ${KOPIA}/zeszly_tydz
```

spowoduje więc, że makropolecenie `INNA_KOPIA` będzie równoważne tekstowi

```
/usr/nowe/backup/zeszly_tydz
```

Możliwe jest dalsze zmniejszenie rozmiaru pliku `makefile`. Nie trzeba na przykład określać położenia kompilatorów języka C i asemblera – są one znane programowi `make`. Można również używać dwóch wewnętrznych makropoleceń o nazwach `$@` i `$?`. Makropolecenie `$@` jest zawsze równoważne nazwie aktualnie przetwarzanego pliku docelowego, natomiast w skład makropolecenia `$?` wchodzą nazwy wszystkich plików pierwotnych nowszych od bieżącego pliku docelowego. Oba te makropolecenia mogą być używane tylko w wierszach określających polecenia, które mają zostać wykonane w celu uaktualnienia pliku. Przykładowy wpis:

```
cosnowego: ${OBJS} ${LIB_FILES}
        ${CC} -o $@ ${OBJS} ${LIB_FILES}
```

jest więc przy wywołaniu polecenia `make cosnowego` równoważny następującemu:

```
/usr/bin/cc -o cosnowego main.o zrobto.o szybciej.o /usr/nowe/lib
/swiezy.a
```

Makropolecenie `$?` ma jeszcze większe możliwości. Można je na przykład zastosować do kopiowania plików `tak.h` i `nie.h` do katalogu domowego użytkownika `reksio` tylko w przypadku, gdy zostały one zmodyfikowane. Zapisane w pliku `makefile` polecenie

```
moze.h: ${TN}  
        cp $? /usr/reksio/
```

będzie więc równoważne poleceniu

```
cp nie.h /usr/reksio/
```

w przypadku, gdy od czasu ostatniej aktualizacji programu `cosnowego` zmodyfikowany zostanie tylko plik `nie.h`; natomiast w przypadku, gdy oba pliki `nie.h` i `tak.h` zostaną zmodyfikowane, zostanie ono rozwinięte do postaci

```
cp tak.h nie.h /usr/reksio/
```

Używając podanych wyżej informacji, można nieco zredukować rozmiar pliku `makefile`:

```
# Nowa wersja pliku makefile  
OBJS = main.o zrobto.o szybciej.o  
TN = tak.h nie.h  
LIB_DIR = /usr/nowe/lib  
LIB_FILES = ${LIB_DIR}/swezy.a  
cosnowego: ${OBJS} ${LIB_FILES}  
        ${CC} -o $@ ${OBJS} ${LIB_FILES}  
main.o: main.c  
        ${CC} -c $?  
zrobto.o: zrobto.c  
        ${CC} -c $?  
szybciej.o: szybciej.s  
        ${AS} -o $@ $?  
sprzataj:  
        rm *.o  
moze.h: ${TN}  
        cp $? /usr/reksio/
```

Reguły przyrostkowe

Jak wspomniano wcześniej w podrozdziale „Makropolecenia”, program `make` nie wymaga określenia reguł dla każdego z plików docelowych z osobna. Ponieważ program ten został zaprojektowany z myślą o uproszczeniu pracy związanej z tworzeniem oprogramowania w systemie Linux, wie on, w jaki sposób działają kompilatory (w szczególności kompilator języka C). Program `make` wie na przykład, że kompilator języka C spodziewa się otrzymania pliku źródłowego w języku C (którego nazwa ma rozszerzenie `.c`) i na jego podstawie generuje plik pośredni z rozszerzeniem `.o`. Tego typu reguła (nazywana regułą przyrostkową) oparta jest na rozszerzeniu nazwy pliku – na jego podstawie podejmowana jest decyzja o tym, jakie polecenia mają zostać wykonane.

Program `make` opiera swoje działanie na wielu zdefiniowanych wewnętrznie regułach przyrostkowych; większość z nich określa sposoby komplikacji kodu źródłowego i konsolidacji plików pośrednich. Oto reguły przyrostkowe stosowane domyślnie w Twoim pliku `makefile`:

```
. SUFFIXES: .o .c .s
.c.o:
    ${CC} ${CFLAGS} -c $<
.s.o:
    ${AS} ${ASFLAGS} -o $@ $<
```

Pierwszy wiersz określa, dla jakich rozszerzeń plików należy zastosować reguły przyrostkowe jeśli żadne reguły nie zostały zdefiniowane bezpośrednio. Drugi wiersz nakazuje programowi `make` uruchomić komplikację plików z rozszerzeniem `.c` w przypadku, gdy odpowiadające im pliki z rozszerzeniem `.o` są nieaktualne. Trzeci pełni funkcję taką, jak drugi, ale dla plików asemblera z rozszerzeniem `.s`. Makropolecenie `$<` ma funkcję zbliżoną do omówionego wcześniej `$?`, ale może być używane tylko w regułach przyrostkowych. Reprezentuje ono nazwę aktualnie opracowywanego pliku docelowego.

Dzięki regułom przyrostkowym zadanie programisty ogranicza się do wprowadzenia do pliku `makefile` nazw wszystkich plików pośrednich; program `make` sam zajmuje się resztą. Jeśli plik `main.o` jest nieaktualny, program `make` automatycznie skompiluje plik `main.c`. Taka sama reguła odnosi się do pliku `szybciej.o`. Po uaktualnieniu plików pośrednich można przystąpić do uaktualnienia pliku `cosnowego`.

Możliwe jest również tworzenie własnych reguł przyrostkowych, dzięki czemu można wymusić podejmowanie innych niż domyślne działań. Jeśli na przykład po skompilowaniu pliki powinny również zostać skopiowane do oddzielnego katalogu, można w pliku `makefile` umieścić następującą regułę:

```
.c.o:
    ${CC} ${CFLAGS} -c $<
    cp $@ kopia
```

Makropolecenie `$@`, jak już wiesz, odpowiada nazwie aktualnie opracowywanego pliku docelowego. W powyższym poleceniu plik docelowy ma rozszerzenie `.o`, natomiast plik pierwotny – `.c`.

Spróbujmy przepisać nasz plik `makefile` (już po raz ostatni), korzystając z reguł przyrostkowych.

```
# Ostatnia wersja
OBJS = main.o zrobto.o szybciej.o
TN = tak.h nie.h
LIB_FILES = /usr/nowe/lib/swiezy.a
cosnowego: ${OBJS} ${LIB_FILES}
    ${CC} -o $@ ${OBJS} ${LIB_FILES}
sprzataj:
    rm *.o
moze.h: ${TN}
    cp $? /usr/reksio/
```

Powyższy plik `makefile` działa dokładnie tak samo, jak pierwsza z przedstawionych wersji – program `cosnowego` można skompilować wydając polecenie

```
$ make cosnowego
```

Można również skompilować jeden z jego fragmentów, na przykład tak:

```
$ make szybciej.o
```

Program `make` został w tym rozdziale omówiony bardzo pobicie. Jeśli chcesz dowiedzieć się czegoś więcej o jego możliwościach, powinieneś przejrzeć dotyczące go strony `man`.

RCS

Innym bardzo ważnym aspektem tworzenia oprogramowania jest zarządzanie kodem źródłowym w trakcie jego ewolucji. Bez względu na rodzaj tworzonej aplikacji, często zdarza się, że jej opracowywanie jest kontynuowane i pojawiają się nowe, poprawione i ulepszone wersje. Przy tworzeniu większych aplikacji pracuje zwykle kilku programistów, co dodatkowo komplikuje zagadnienie zarządzania kodem źródłowym. Jeśli nie zostanie zastosowany żaden program zajmujący się tymi zadaniami, bardzo łatwo zagubić się przy aktualizowaniu wersji plików. Może to prowadzić do sytuacji, w której modyfikacje są tracone albo powtórnie opracowywane przez różnych programistów. Na szczęście w systemie Linux z pomocą przychodzą takie programy, jak RCS (Revision Control System, system kontroli wersji).

RCS to tylko jeden z licznych programów tego typu – istnieje również wiele innych; niektóre z nich są darmowe, inne komercyjne. Pierwszym UNIX-owym programem do zarządzania kodem źródłowym nadającym się do poważniejszych zastosowań był SCCS (Source Code Control System), używany zresztą po dziś dzień. RCS jest systemem o nieco większych możliwościach, implementującym rozwiązania, których brakowało w SCCS. Systemy RCS dostępne są w wersjach dla większości platform, włączając wszystkie wersje UNIX-a, DOS, Windows i Windows NT. Jeśli pracujesz nad projektem tworzonym przez większą liczbę programistów, przyda Ci się wersja pozwalająca na pracę w sieci, o nazwie CVS (Concurrent Versions System). CVS jest dołączany do wielu wersji Linuxa. Polecenia używane w systemie CVS różnią się od poleceń wykorzystywanych podczas pracy z RCS, ale ponieważ większość czytelników nie będzie na razie potrzebowała wersji sieciowej, w tym rozdziale skoncentrujemy się na systemie RCS. Jeśli gryzie Cię ciekawość, z wersjami Linuxa wyposażonymi w system CVS dostarczane są też odpowiednie strony `man`.

Program RCS rejestruje informacje o wersjach plików, kontrolując dostęp do nich. Każdy użytkownik próbujący zmodyfikować określony plik musi zarejestrować się w systemie RCS i podać powód modyfikacji. RCS zapisuje te informacje i dane o wprowadzonych poprawkach w osobnym pliku. Ponieważ poprawki są rejestrowane w pliku innym niż oryginalny, można w razie potrzeby z łatwością wrócić do pierwotnej wersji kodu źródłowego. Takie rozwiązanie pozwala również zaoszczędzić miejsce na dysku, ponieważ nie trzeba wykonywać kopii całych plików. Jest to szczególnie widoczne przy modyfikacjach obejmujących tylko kilka wierszy kodu źródłowego, natomiast mało przydaje się, gdy istnieje tylko kilka znacznie różniących się między sobą wersji programu.

Delta

Zestaw modyfikacji zapisywanych przez system RCS do odpowiedniego pliku nazywany jest deltą. Numer wersji może mieć dwie formy. Pierwsza z nich składa się z numeru wydania i numeru poziomu. Numer wydania zmieniany jest zwykle po wprowadzeniu jakichś znaczących zmian w kodzie źródłowym. Po utworzeniu pliku RCS, otrzymuje on numer 1.1, oznaczający pierwszy poziom pierwszego wydania. RCS będzie następnie przy wprowadzaniu modyfikacji odpowiednio zwiększał numer poziomu (na 1.2, 1.3 itd.). Można również wymusić zmianę numeru wydania programu.

Druga forma numeru wersji programu również zawiera numer wydania i poziom, ale dodatkowo zawiera jeszcze numer kolejny poprawki. Można używać go, jeśli na przykład opracowałeś program, co do którego klient zgłasza zastrzeżenia, ale kolejnych poprawionych wersji nie chcesz numerować tak, jakby były wersjami „oficjalnymi”. Choć następna wersja może również zawierać wprowadzone poprawki, jej wydanie może zostać opóźnione przez prace nad dodaniem nowej funkcjonalności programu. Z tego powodu czasem warto dodać do numeru wersji nowy składnik, zwiększany wraz z wprowadzaniem poprawek. Jeśli na przykład planowałeś wydanie wersji o numerach 3.1, 3.2, 3.3, 3.4 itd., a w którymś momencie zorientowałeś się, że w wersji 3.3 jest błąd, który wymaga wydania nowej wersji, nie zawierającej jeszcze rozwiązań planowanych w wersji 3.4, możesz wydać wersję o numerze 3.3.1.1, następnie 3.3.1.2 itd.



Zwykle każdy nowy numer wersji powinien określać kompletny zestaw poprawek. Oznacza to, że kod w każdej z wersji powinien być przetestowany i powinny zostać z niego usunięte wszystkie błędy. Nie powinieneś zmieniać numeru wersji z 2.0 na 3.0 tylko dlatego, że usunąłeś kilka błędów w wersji 2.0. Główne numery wersji są zmieniane po dodaniu do programów nowych funkcji, a nie po skorygowaniu występujących usterek.



Czy istnieją programy nie zawierające błędów? Na pewno nie są ich pozbawione większe aplikacje, w których niektóre usterki ujawniają się dopiero po scaleniu fragmentów kodu tworzonych przez różnych programistów. Celem programisty powinno być usunięcie wszystkich błędów w tworzonej przez niego części programu. Choć prawdopodobnie nigdy nie uda się wyeliminować wszystkich usterek, można ograniczyć ich liczbę, dzięki czemu łatwiej będzie wykrywać i usuwać pozostałe.

Tworzenie pliku RCS

Załóżmy, że chcesz pracować nad programem, którego kod źródłowy zapisany w pliku `loty.c` ma postać:

```
/* Plik przykładowy dla systemu RCS
#include <stdio.h>
main()
{
    printf("Program najwyższe lotu...\n");
}
```

Pierwszym krokiem powinno być utworzenie katalogu RCS:

```
$ mkdir RCS
```

W tym katalogu będą przechowywane pliki systemu RCS. Następnie należy poinformować ten system, że dany plik ma być nadzorowany. Służy do tego polecenie `ci` (ang. *check-in*):

```
$ ci loty.c
```

Po jego wydaniu należy wprowadzić odpowiedni komentarz dotyczący modyfikacji – wraz z innymi informacjami zostanie on zapisany w pliku o nazwie `loty.c,v` w katalogu RCS. Zawartość oryginalnego pliku zostanie skopiowana do pliku oznaczonego numerem wersji 1.1. Po wydaniu kolejnego polecenia `ci` system RCS usunie kopię roboczą z katalogu RCS.

Odzyskiwanie pliku RCS

Jeśli chcesz uzyskać kopię pliku nadzorowanego przez RCS, powinieneś użyć polecenia `co` (ang. *check-out*). Polecenie to wydane bez żadnych argumentów daje wersję pliku przeznaczoną tylko do odczytu, której nie można edytować. Jeżeli chcesz modyfikować otrzymany plik, powinieneś użyć opcji `-l`:

```
$ co -l loty.c
```

Po zakończeniu modyfikowania pliku można ponownie załączyć nadzorowanie go za pomocą polecenia `ci`. Zostaniesz poproszony o komentarz dotyczący wprowadzonych zmian. Nadzorowany plik zostanie oznaczony numerem wersji 1.2.

Oznaczenia numeru wersji stosowane przez system RCS składają się z numeru wydania, poziomu (ang. *level*) i dwóch składników określających numer poprawki (ang. *branch* i *sequence*). Polecenia RCS operują domyślnie na najświeższej wersji pliku, ale można wymusić użycie innej wersji. Jeśli na przykład najnowsza wersja pliku `loty.c` ma numer 2.7, a modyfikacje, które wprowadziłeś, są na tyle znaczące, że postanowiłeś zmienić numer wydania, powinieneś wydać polecenie `ci` z opcją `-r` i numerem nowego wydania:

```
$ ci -r3 loty.c
```

Powyższe polecenie spowoduje oznaczenie pliku numerem wersji 3.1. Można również otworzyć nową gałąź poprawek rozpoczynającą się na poziomie 2.7 za pomocą polecenia:

```
$ ci -r2.7.1 loty.c
```

Polecenie `rcs` z opcją `-o` służy do usuwania niepotrzebnych, nieaktualnych wersji plików. Aby na przykład usunąć wersję o numerze 2.6, powinieneś wydać polecenie

```
$ rcs -o2.6 loty.c
```

Słowa kluczowe

Częścią pliku nadzorowanego przez system RCS mogą być również słowa kluczowe. Służą one do osadzania w pliku takich informacji, jak dane o autorze, data utworzenia poszczególnych wersji pliku itp. – dane te mogą być odczytane za pomocą polecenia `ident`. Słowa kluczowe należy wprowadzać bezpośrednio do aktualnie opracowywanej kopii pliku. Przy wydawaniu poleceń `ci` i `co` do słów kluczowych systemu RCS dołączane są odpowiednie wartości. Słowa kluczowe należy otaczać z obu stron symbolami dolara:

```
$słowo_kluczowe$
```

Są one przekształcane do postaci:

```
$słowo_kluczowe: wartość $
```

Oto niektóre ze słów kluczowych rozpoznawanych przez system RCS:

- | | |
|---------------------|---|
| \$Author\$ | użytkownika, który jest autorem danej wersji; |
| \$Date\$ | data i czas zarejestrowania wersji; |
| \$Log\$ | wszystkie komentarze odnoszące się do danego pliku; |
| \$Revision\$ | numer wersji pliku. |

Jeśli słowa te byłyby używane w pliku `loty.c`, polecenie `ident` mogłoby zwrócić na przykład takie informacje:

```
$ ident loty.c
$Author: reksio $
$Date: 97/01/15 23:15:32 $
$Log: loty.c,v $
# Revision 1.2 97/01/15 23:15:32 reksio
# Drobne modyfikacje
#
# Revision 1.1 97/01/15 23:10:12 reksio
# Utworzenie pliku loty.c
#
$Revision: 1.2 $
```

Uzyskiwanie informacji o wersji z pliku RCS

Czasem zamiast informacji opartych na słowach kluczowych RCS bardziej potrzebne są ogólniejsze, podsumowujące dane, podawane przez polecenie `rlog` z opcją `-t`:

```
$ rlog -t loty.c
RCS file:      loty.c,v;      Working file:  loty.c
head:          3.2
locks:         reksio:  2.1; strict
access list:   rick tim
symbolic names:
comment leader: " * "
total revisions: 10;
description:
Tak... Program najwyzszego lotu...
=====
```

Pole `head` zawiera informację o numerze ostatniej wersji pliku. W polu `locks` zawarte są identyfikatory użytkowników, którzy modyfikowali plik oraz typ stosowanego za- bezpieczenia (bezpośredni czy pośredni dla właściciela pliku RCS). Pole `access list` zawiera identyfikatory użytkowników, którzy mają prawo tworzenia delt danego pliku – jak je zmienić, dowiesz się w następnym podrozdziale.

Dostęp do plików RCS

Jedną z ważniejszych funkcji systemu RCS jest ograniczanie dostępu do tych plików, które nie powinny być modyfikowane przez danego użytkownika. RCS przechowuje listę użytkowników uprawnionych do modyfikacji każdego pliku z osobna. Początkowo lista ta jest pusta, co oznacza, że wszyscy użytkownicy mogą wprowadzać poprawki do nadzorowanego pliku. Aby umożliwić modyfikowanie pliku tylko określonym użytkownikom lub grupom, należy wydać polecenie `rcs` z opcją `-a`. Przykładowe polecenie

```
$ rcs -arick,tim loty.c
```

spowoduje, że prawo modyfikacji pliku `loty.c` będą mieli tylko użytkownicy o identyfikatorach `tim` i `rick`. Jeśli zmienisz zdanie i uznasz, że `rick` nie powinien jednak wtrącać się do tworzenia Twojego wspaniałego programu, możesz zabrańić mu dostępu do niego za pomocą opcji `-e` programu `rcs`:

```
$ rcs -erick loty.c
```

Jeśli w przyklepie paranoi stwierdzisz, że nikt nie powinien mieć możliwości modyfikowania Twojego programu, możesz po prostu go zablokować (co uniemożliwi wprowadzanie zmian nawet właścicielowi pliku). Aby zablokować modyfikowanie drugiego wydania programu `loty.c`, wydaj polecenie:

```
$ rcs -e -L2 loty.c
```

Właściciel pliku może wprowadzić modyfikacje tylko po jawnym zablokowaniu pliku, zarówno przy wydawaniu polecenia `ci`, jak i `co`.

Porównywanie wersji i łączenie poprawek

System RCS umożliwia porównywanie poszczególnych wersji plików, co pozwala łatwo zorientować się, jakie zmiany zostały wprowadzone. Dzięki takiemu rozwiązaniu można bezpiecznie łączyć poprawki wprowadzane do pliku przez różne osoby. Do wyświetlania różnic pomiędzy poszczególnymi wersjami pliku nadzorowanego przez RCS bądź też pomiędzy którąś z wersji pliku a wersją najnowszą służy polecenie `rcsdiff`.
Polecenie

```
$ rcsdiff -rl.2 -rl.5 loty.c
```

powoduje wyświetlenie różnic pomiędzy wersjami 1.2 i 1.5 pliku `loty.c`, na przykład tak:

```
RCS file: loty.c,v
retrieving revision 1.1
rdiff -r 1.2 -rl.5 loty.c
6a7,8
> /* Choc w zasadzie nie jest szczegolnie ciekawy */
```

Powyższe dane wskazują, że poszczególne wersje różnią się tylko tym, że po wierszu szóstym zostały dodane dwa nowe wiersze (które zostały wyświetcone). Jeśli chcesz sprawdzić, czym pierwotna wersja pliku (określona w polu `head`) różni się od wersji za-rejestrowanej ostatnio, wydaj polecenie:

```
$ rcsdiff loty.c
```

Po stwierdzeniu, że pomiędzy poprawkami wniesionymi przez Ciebie i innych programi-stów nie ma potencjalnych konfliktów, możesz zdecydować się na połączenie poszczegól-nych wersji. Służy do tego polecenie `rcsmerge`. Składnia tego polecenia wymaga podania nazwy jednego lub dwóch plików, określających wersje, które należy połączyć i trzeciej nazwy, określającej nazwę pliku roboczego (w poniższym przykładzie – `loty.c`). Wydanie polecenia

```
$ rcsmerge -rl.3 -rl.6 loty.c
```

powoduje wygenerowanie następujących komunikatów:

```
RCS file: loty.c,v
retrieving version 1.3
retrieving version 1.6
Merging differences between 1.3 and 1.6 into loty.c
```

Jeśli wprowadzone modyfikacje kolidują ze sobą, program `rcsmerge` wstawia do pliku obie wersje, oznaczając odpowiednio ich pochodzenie. Zaistniały konflikt trzeba roz-wiązać ręcznie, edytując powstały plik przed ponownym nakazaniem nadzorowania go przez system RCS.



Porządek, w jakim podawane są wersje plików, które należy połączyć, jest istotny. Jeśli po opcji `-r` najpierw podany zostanie numer późniejszej wersji, wszystkie wprowadzone w niej zmiany (w stosunku do wersji wcześniejszej) zostaną wycofane.

Praca z programem make i systemem RCS

Program `make` jest przystosowany do współpracy z systemem RCS, wraz z którym tworzy kompletne pod wieloma względami środowisko programistyczne. Mimo wszystko jednak używanie obu tych programów równocześnie może stwarzać pewne problemy, szczególnie jeśli nad projektem pracuje kilku programistów. Ujawniają się one szczególnie na etapie testowania programu – wtedy programista potrzebuje stabilnych, nie zmieniających się wersji plików. Testowanie może stwarzać większe trudności natury organizacyjnej niż jakakolwiek inna faza tworzenia projektu. Dla pojedynczego programisty jednak problemy te nie mają żadnego znaczenia, a współpraca programu `make` i systemu RCS znakomicie ułatwia tworzenie aplikacji, szczególnie w systemie Linux.

Aby umożliwić opracowywanie plików RCS przez program `make`, należy w pliku `makefile` zdefiniować regułę przyrostkową dla rozszerzenia `.v`. Pliki z rozszerzeniem `.v` są właśnie plikami systemu RCS. Jeśli pliki nadzorowane przez system RCS mają być komplikowane za pomocą kompilatora języka C, w pliku `makefile` powinny znaleźć się następujące reguły:

```
CO = co
.c, v.o:
    ${CO} $<
    ${CC} ${CFLAGS} -c $*.c
    - rm -f $*.c
```

Makropolecenie `CO` jest równoważne poleceniu `co`. Makropolecenie `$*.c` jest niezbędne, ponieważ program `make` automatycznie usuwa rozszerzenie `.c`. Myślnik poprzedzający polecenie `rm` powoduje, że program `make` będzie kontynuował działanie nawet w przypadku, gdy wykonanie polecenia `rm` zakończy się błędem. Jeśli na przykład system RCS nadzorowałby plik o nazwie `main.c`, program `make` wygenerowałby następujące polecenia:

```
co main.c
cc -O -c main.c
rm -f main.c
```

Podsumowanie

Programy `make` i `RCS` są najważniejszymi aplikacjami służącymi do zarządzania procesem tworzenia oprogramowania. Program `make` potrafi generować polecenia komplikujące,

konsolidujące i służące do wykonywania innych czynności o podobnym charakterze. Dzięki temu, że pozwala on na ustalanie zależności pomiędzy poszczególnymi plikami, możliwe jest komplikowanie tylko tych fragmentów projektu, które wymagają aktualnienia. System RCS to zestaw programów służących do zarządzania kodem źródłowym, dzięki którym kilku programistów może bez kłopotów pracować nad pojedynczym projektem. Zachowuje on dane o poszczególnych modyfikacjach kodu źródłowego.

Inną korzyścią wynikającą ze stosowania systemu RCS jest oszczędność miejsca zajmowanego przez projekt na dysku twardym. System CVS jest rozszerzeniem systemu RCS, pozwalającym na automatyczne łączenie kilku wersji pliku. Dzięki temu możliwa jest równoległa praca kilku programistów nad jednym plikiem źródłowym, choć system CVS nie zwalnia ich od konieczności rozwiązywania występujących konfliktów.

Po zapoznaniu się z informacjami dotyczącymi systemów zarządzania kodem źródłowym, możesz przejść do rozdziałów omawiających poszczególne języki programowania.

Jeśli chcesz zastosować system RCS i program `make` do tworzenia programów w języku C lub C++, przejdź do rozdziału 26. „Programowanie w języku C” lub 27. „Programowanie w języku C++”.

W rozdziale 30. „Inne kompilatory” znajdziesz informacje o innych językach programowania dostępnych dla systemów linuxowych.

Rozdział 31. „Smalltalk/X” omawia opartą na interfejsie graficznym wersję popularnego języka obiektowego Smalltalk.

Rozdział 57.

Jądro systemu

Kamran Husain i Tim Parker

W tym rozdziale:

- υ Uaktualnianie i instalowanie nowych składników jądra systemu
- υ Kompilowanie kodu źródłowego jądra systemu
- υ Dodawanie sterowników urządzeń
- υ Uaktualnianie bibliotek
- υ Kompilator języka C dla systemu Linux

Poza przypadkami takimi jak instalowanie nowej wersji systemu, instalowanie nowych składników sieci (takich jak NFS czy NIS) czy nowych sterowników urządzeń wymagających specjalnego traktowania nie ma konieczności modyfikowania jądra systemu. Szczegółowe informacje na temat instalowania sterowników urządzeń zwykle są prowadzone wraz z oprogramowaniem. Nie jest tak jednak w każdym przypadku, dlatego w tym rozdziale przedstawimy ogólne zasady dotyczące pracy z jądrem systemu.



Na początek powiedzmy wyraźnie, że nie powinieneś modyfikować jądra systemu, jeśli nie wiesz, co chcesz przez to osiągnąć. Jeśli kod źródłowy lub pliki konfiguracyjne zostaną uszkodzone, kernel może być niestabilny, co w najgorszym przypadku może prowadzić do uszkodzenia systemu plików. Uważnie zapoznaj się z przedstawionymi poniżej informacjami i przestrzegaj podanych wskazówek. Modyfikowanie jądra systemu wymaga sporej wiedzy, a w tym rozdziale możemy przedstawić tylko podstawowe informacje.

Obecnie używa się powszechnie kilku różnych wersji Linuxa, nie do końca zgodnych pomiędzy sobą. Z tej przyczyny przedstawione poniżej instrukcje mogą nie działać prawidłowo w Twojej wersji Linuxa. Mimo tego idea pozostaje ta sama, różnice mogą wystąpić tylko w przypadku nazw programów użytkowych i ścieżek dostępu do nich. Do większości wersji Linuxa dołączana jest dokumentacja omawiająca proces kompilowania jądra i zawierająca informacje o położeniu kodu źródłowego i skompilowanych programów.



Zanim podejmiesz jakiekolwiek kroki upewnij się, że posiadasz odpowiedni zestaw dyskietek startowych. Warto również wykonać pełną kopię zapasową systemu plików. Choć proces modyfikowania jądra nie jest trudny, jeśli coś pójdzie nie tak, może okazać się, że nie da się ponownie uruchomić systemu. W takiej sytuacji najprostszym rozwiązaniem jest użycie dyskietek startowych, warto więc posiadać przynajmniej jeden zapasowy ich zestaw.

Ponieważ jądro systemu jest komplikowane za pomocą kompilatora języka C wchodzącego w skład każdej dystrybucji Linuxa, to właśnie na nim skoncentrujemy się w dalszej części tego rozdziału, omawiając jego znaczniki i sposoby stosowania. Nie będzie to oczywiście kompletny opis kompilatora języka C, ale powinien być wystarczający przy podstawowych czynnościach związanych z modyfikowaniem jądra systemu (i kompilowaniem dowolnych programów napisanych w języku C).

Uaktualnianie i instalowanie nowych składników jądra systemu

Linux jest systemem rozwijającym się bardzo dynamicznie. Nowe wersje jądra systemu bądź też innych komponentów dołączanych do jądra są co jakiś czas udostępniane dla użytkowników. Decyzję o tym, czy należy uaktualnić wersję jądra systemu, musisz podjąć sam – zwykle opiera się ona na danych dotyczących błędów usuniętych w nowej wersji czy też oferowanych przez nią nowych możliwościach. Po dołączeniu nowego oprogramowania prawdopodobnie będzie trzeba odnowić wszystkie dowiązania do jądra systemu, chyba że nowe komponenty są sterownikami urządzeń czy niezależnymi programami użytkowymi.

Mimo wszystko, należy jednak zdecydowanie unikać uaktualniania systemu wraz z każdą nową wersją, i to z kilku powodów. Może na przykład zdarzyć się, że zainstalowane oprogramowanie nie jest zgodne z nową wersją programów użytkowych czy jądra systemu. Może również okazać się, że nowa wersja posiada poważny błąd. Tego typu sytuacje mogą powodować nie kończące się problemy. Większość nowych wersji programów nie zachowuje istniejących informacji konfiguracyjnych, więc konieczna jest także ponowna konfiguracja wszystkich pakietów, które zostały zainstalowane na nowo.

Poza tym, nowe wersje programów pojawiają się na tyle często, iż nie jest wykluczone, że więcej czasu spędziłbyś na ich ładowaniu i instalowaniu niż na faktycznym korzystaniu z systemu. Taka sytuacja może być bardzo męcząca. Liczba zmian wprowadzanych w kolejnych wersjach systemu różniących się tylko numerem pobocznym jest zwykle niewielka, dlatego należy przejrzeć dołączone do niej informacje i upewnić się, że faktycznie warto poświęcić czas na jej instalację.

Z praktycznego punktu widzenia dobrze jest aktualizować system raz lub dwa razy do roku, tylko wtedy, gdy nowa wersja zawiera rozwiązania, które znacznie zmienią sposób, w jaki używasz systemu. Posiadanie zawsze najnowszej wersji systemu jest kuszące, ale

trzeba również brać pod uwagę korzyści wynikające z utrzymywania stabilnego i funkcjonalnego systemu operacyjnego.

Jeśli uaktualniasz wersję oprogramowania, powinieneś zdawać sobie sprawę z faktu, że nie musisz uaktualniać wszystkiego. W kilku ostatnich wersjach Linuxa zmieniało się co najwyżej 5% systemu operacyjnego. Zamiast wymiany całego oprogramowania warto więc rozważyć zainstalowanie tylko tych fragmentów, które faktycznie uległy znaczącym zmianom, czyli zwykle jądra systemu, kompilatora, bibliotek i często wykorzystywanych programów użytkowych. Takie rozwiązanie pozwoli uniknąć konieczności ponownego konfigurowania pozostałych części systemu.

Kompilowanie kodu źródłowego jądra systemu

Uaktualnienie, wymiana czy dodanie nowego kodu do jądra systemu jest zwykle procesem dość prostym. Sprawdza się do zdobycia odpowiedniego kodu źródłowego, dostosowania informacji konfiguracyjnych, skompilowania a następnie umieszczenia skompilowanych programów w odpowiednim miejscu tak, by system mógł działać prawidłowo. Proces ten zwykłe jest zautomatyzowany za pomocą skryptu powłoki lub programu instalacyjnego – w niektórych przypadkach wszystko sprowadza się do wydania jednego polecenia.

Kod źródłowy nowych wersji jądra systemu dostępny jest zwykle z dystrybucjami rozprowadzanymi na płytach CD-ROM, w węzłach FTP, grupach dyskusyjnych i w wielu innych miejscach. Większość wersji jądra posiada numer składający się z głównego i pobocznego numeru wersji (dwie pierwsze liczby) oraz numeru poprawki (trzecia liczba), na przykład 1.12.123. W większości węzłów FTP czy BBS równocześnie udostępnianych jest kilka wersji jądra – musisz wziąć to pod uwagę, jeśli chcesz załadować jego najnowszą wersję.

Poprawki do kodu źródłowego jądra systemu zwykle nie zawierają całego kodu. Ich instalacja powoduje nadpisanie fragmentów istniejącego kodu źródłowego. Po jego ponownym skompilowaniu poprawka jest już zainstalowana. Tego typu poprawki udostępniane są dość często.



Podczas instalowania poprawki należy ograniczyć do minimum liczbę otwartych plików i działających procesów oraz aplikacji. Pozwoli to uniknąć problemów związanych z pozostawieniem otwartych plików, co może prowadzić nawet do uszkodzenia tablicy I-node. Ponieważ aby zainstalować poprawkę i tak musisz być zalogowany jako `root`, możesz zakończyć wszystkie niepotrzebne procesy i aplikacje.

W większości przypadków kod źródłowy jest rozpowszechniany w postaci skompresowanego (zwykle programem `gzip`) archiwum programu `tar`. Taki plik należy rozpakować do

katalogu `/usr/src`, w którym zwykle przechowywany jest kod źródłowy programów. W niektórych wersjach Linuxa do tego celu przeznaczony jest inny katalog – powinieneś więc przejrzeć dokumentację rozprowadzaną z posiadaną przez Ciebie wersją Linuxa lub poszukać w katalogu `/usr/src` pliku `README`, w którym powinny znajdować się dokładniejsze informacje.

Często zdarza się, że rozpakowanie pliku do katalogu `/usr/src` powoduje utworzenie podkatalogu `/usr/src/linux`, co może prowadzić do nadpisania obecnej w systemie wersji kodu źródłowego jądra. Przed rozpoczęciem rozpakowywania warto więc zmienić nazwę istniejącego katalogu lub skopiować jego zawartość w bezpieczne miejsce, dzięki czemu w przypadku problemów łatwo będzie wrócić do wersji poprzedniej.

Po rozpakowaniu kodu źródłowego należy utworzyć dwa dowiązania symboliczne (o ile nie zostały one utworzone przez program instalacyjny) do katalogu `/usr/include`. Można to zrobić wydając polecenia:

```
ln -sf /usr/src/linux/include/linux /usr/include/linux
ln -sf /usr/src/linux/include/asm /usr/include/asm
```

Jeśli w Twoim systemie używane są inne ścieżki dostępu, podstaw zamiast `/usr/src/linux` odpowiednią wartość. Jeśli takie dowiązania nie istnieją, poprawne uaktualnienie jądra systemu nie będzie możliwe.

Po rozpakowaniu kodu źródłowego i utworzeniu odpowiednich dowiązań można rozpocząć proces komplikacji. W tym celu niezbędny będzie kompilator `gcc` lub `g++` (kompilator GNU języka C i C++) lub inny zgodny z nimi kompilator. Należy sprawdzić w dokumentacji dołączonej do nowej wersji kodu źródłowego czy używana wersja kompilatora jest odpowiednia, ponieważ zdarza się, że nowe wersje jądra nie dają się skompilować za pomocą starszych wersji kompilatorów.

Następnie należy przejrzeć plik `/usr/src/linux/Makefile` (w niektórych dystrybucjach może on znajdować się w innym katalogu). Znajdź w nim wiersz definiujący zmienną `ROOT_DEV`, określającą urządzenie, na którym zapisany jest główny system plików. Zwykle definicja ta ma postać

```
ROOT_DEV = CURRENT
```

Jeśli wartość zmiennej `ROOT_DEV` jest inna, upewnij się, że jest ona odpowiednia dla Twojego systemu. Jeżeli w pliku `Makefile` nie ma takiej definicji, należy ją dodać.

Proces komplikacji rozpoczyna się po przejściu do katalogu `/usr/src/linux` i wydaniu polecenia

```
make config
```

uruchamiającego program `make`, generujący odpowiednie polecenia komplilujące pliki źródłowe. W niektórych wersjach Linuxa proces ten może wyglądać nieco inaczej – odpowiednich informacji należy szukać w dokumentacji dołączonej do plików źródłowych.

W następnym kroku program `config` zadaje serię pytań dotyczących różnych aspektów konfiguracji jądra, na które należy odpowiedzieć przed rozpoczęciem właściwej komplikacji.

Pytania te dotyczą między innymi typu dysku twardego, procesora, typów używanych partycji czy innych urządzeń dołączonych do komputera, na przykład napędów CD-ROM. Powinieneś odpowiedzieć na nie najlepiej, jak potrafisz. Jeśli masz wątpliwości co do którejś z odpowiedzi, wybierz odpowiedź domyślną albo tę, która wydaje się być najbardziej prawidłowa. W najgorszym przypadku – jeśli system nie będzie działał prawidłowo – będzie trzeba powtórzyć cały proces od początku (oczywiście pod warunkiem, że przygotowałeś dyskietki startowe).

Następnie należy ustalić wszystkie zależności pomiędzy plikami kodu źródłowego. Ten etap często jest pomijany, ale takie postępowanie może powodować mnóstwo problemów. Aby ustalić zależności dla zainstalowanej wersji kodu źródłowego jądra systemu, wydaj polecenie:

```
make dep
```

Jeśli w skład instalowanego oprogramowania nie wchodzi plik zależności (o nazwie `dep`), powinieneś upewnić się, że odpowiednie czynności podejmowane są w czasie wykonywania innych etapów konfiguracji oprogramowania (informacje takie znajdują się w dokumentacji dołączonej do plików źródłowych).

Na koniec pora rozpoczęć komplikację nowego jądra – w tym celu należy wydać polecenie

```
make Image
```

które spowoduje skompilowanie jądra i pozostawienie go w katalogu bieżącym (zwykle `/usr/src/linux`). Jeśli chcesz utworzyć skompresowaną wersję jądra systemu, wydaj polecenie

```
make zImage
```

Nie wszystkie dystrybucje Linuxa pozwalają na kompresowanie jądra systemu w trakcie jego komplikacji.

Ostatnim krokiem jest skopiowanie nowej wersji jądra systemu na dysk, z którego uruchamiany jest system, lub też na dyskietkę startową. Polecenie

```
cp Image /dev/fd0
```

spowoduje umieszczenie kopii jądra systemu na dyskietce. Jeśli chcesz umieścić ją gdzieś indziej, podstaw odpowiednią nazwę urządzenia. Jeżeli natomiast do uruchamiania systemu używany jest program LILO, wówczas aby zainstalować nowe jądro, należy uruchomić program instalacyjny lub program `/usr/lilo/lilo` – dokładniejsze informacje na ten temat przedstawiliśmy w rozdziale 4. „Program LILO”.

Teraz pozostało tylko uruchomić ponownie system i sprawdzić, czy nowe jądro prawidłowo ładuje się do pamięci. Jeśli wystąpią jakieś problemy, należy uruchomić system z dyskietki startowej i powtórzyć cały proces. W dołączonej do kodu źródłowego dokumentacji znajdziesz wskazówki dotyczące rozwiązywania niektórych problemów oraz informacje o ewentualnych zmianach w procesie komplikacji jądra.

Dodawanie sterowników urządzeń

Nowe sterowniki urządzeń można również dołączać do istniejącej wersji jądra bez konieczności jego konfiguracji i komplikowania. Z taką sytuacją można często zetknąć się w przypadkach, gdy do systemu dodane zostanie takie urządzenie, jak karta multiport czy dysk optyczny, do których obsługi niezbedne jest załadowanie odpowiednich sterowników podczas uruchamiania systemu. Może również zdarzyć się, że będziesz chciał zainstalować jakiś rodzaj oprogramowania podnoszącego poziom bezpieczeństwa systemu i wymagającego modyfikacji samego jądra systemu.

Sterowniki wymagające komplikacji jądra przeważnie rozpowszechniane są wraz z odpowiednią dokumentacją. Zwykle otrzymany kod źródłowy należy umieścić w tym samym katalogu, w którym znajduje się kod źródłowy jądra systemu (na przykład `/usr/src`). Aby dodać do jądra systemu nowy fragment kodu, trzeba zmodyfikować plik `Makefile` – ręcznie bądź też za pomocą odpowiedniego skryptu instalacyjnego. W niektórych przypadkach do kodu źródłowego sterowników dodawany jest osobny plik `Makefile`.

Nastepnym krokiem jest skompilowanie jądra wraz z nowymi sterownikami. Proces ten nie różni się od opisanego w poprzednim podrozdziale – skompilowane jądro należy skopiować na dyskietkę startową lub zainstalować za pomocą programu LILO. Zwykle cały proces nie zabiera więcej niż dziesięć minut i raczej nie sprawia kłopotów, chyba że dostawca oprogramowania kiepsko wywiązał się ze swych obowiązków. Przed instalacją upewnij się, że nowe sterowniki będą działać z Twoją wersją jądra systemu – odpowiednie informacje znajdziesz w plikach tekstowych dołączonych do kodu źródłowego oraz w plikach dotyczących zgodności z różnego typu oprogramowaniem, rozpowszechnianych z większością wersji Linuxa.

Uaktualnianie bibliotek

Większość programów przeznaczonych dla systemów linuxowych korzysta z bibliotek współużytkowanych (ang. *shared libraries*; jest to zestaw podprogramów, które mogą być wykorzystywane przez różne aplikacje). Jeśli po uaktualnieniu jądra systemu przy uruchomianiu programów wyświetlany jest komunikat

`Incompatible library version`

oznacza to, że biblioteki również wymagają uaktualnienia. Większość bibliotek jest kompatybilna wstecz, co oznacza, że programy wykorzystujące starsze wersje bibliotek będą działać poprawnie również po zainstalowaniu nowych wersji bibliotek.

Nowe wersje bibliotek pojawiają się rzadziej niż nowe wersje jądra systemu i są rozprowadzane w ten sam sposób. Zwykle do nowych wersji jądra dołączane są pliki tekstowe zawierające informacje o wymaganych bibliotekach i wskazujące miejsca, w których można zaopatrzyć się w ich najnowsze wersje.

Nowe wersje bibliotek rozprowadzane są najczęściej w postaci skompresowanego archiwum programu `tar`, więc proces ich rozpakowywania jest taki sam, jak plików zawierających kod źródłowy jądra systemu, z tym że katalogami docelowymi są zwykle katalogi `/lib`, `/usr/lib` i `/usr/include`. Pliki z rozszerzeniami `.a` i `.aa` trafiają przeważnie do katalogu `/usr/lib`, natomiast pliki obrazów bibliotek współużytkowanych, których nazwy mają postać `libc.so.wersja`, do katalogu `/lib`.

Po zainstalowaniu nowych wersji bibliotek czasem zachodzi konieczność ręcznej modyfikacji dowiązań symbolicznych, tak by prowadziły do najnowszych wersji bibliotek. Przykładowo, jeśli używasz biblioteki `libc.so` w wersji `4.4.1` i uaktualniłeś jej wersję do `4.4.2`, powinieneś zmodyfikować odpowiednie dowiązanie, wydając polecenie:

```
ln -sf /lib/libc.so.4.4.1 /lib/libc.so.4
```

Ostatni argument jest nazwą uaktualnianej biblioteki – nazwy te mogą być różne w zależności od dystrybucji – należy to sprawdzić w dokumentacji.

W taki sam sposób należy zmodyfikować również dowiązanie dla biblioteki `libm.so`. Dowiązań symbolicznych nie należy usuwać, ponieważ programy korzystające z bibliotek współużytkowanych (m.in. program `ls`) nie będą działać poprawnie.

Kompilator języka C dla systemu Linux

W systemie Linux kompilator języka C jest wykorzystywany do komplikowania wszystkich wersji jądra systemu i znacznej większości programów użytkowych. Dla wszystkich wersji Linuxa dostępny jest kompilator GNU C, o nazwie `gcc`. Powstał on w ramach projektu Free Software Foundation, w związku z czym jest darmowy.

Kompilator GNU C rozprowadzany wraz z dystrybucją Slackware jest w pełni funkcjonalnym kompilatorem spełniającym standardy ANSI C. Jeśli znasz już jakiś kompilator języka C z innej platformy, będziesz w stanie opanować `gcc` w bardzo krótkim czasie.

Przy wywoływaniu kompilatora GCC należy podać szereg argumentów, na które składają się różne opcje i przynajmniej jedna nazwa pliku. Najogólniej rzecz ujmując, składnia polecenia uruchamiającego kompilator `gcc` jest następująca:

```
gcc [opcje] [nazwy_plików]
```

Działania określone przez opcje podane w wierszu poleceń zostaną wykonane w odniesieniu do każdego z wyszczególnionych plików. Opcji rozpoznawanych przez kompilator `gcc` jest ponad setka. Większość z nich prawdopodobnie nigdy nie będziesz musiał użyć, ale kilka jest niezbędnych nawet przy najprostszych czynnościach.

Wiele nazw opcji programu `gcc` składa się z więcej niż jednego znaku. Z tego powodu każda z opcji musi zostać poprzedzona oddzielnym myślnikiem. Nie można grupować opcji po wspólnym myślniku, jak ma to miejsce w przypadku większości poleceń systemu Linux. Podane poniżej dwa przykładowe polecenia mają różne znaczenie:

```
gcc -p -g test.c  
gcc -pg test.c
```

Pierwsze z nich nakazuje programowi `gcc` skompilować plik `test.c`, dołączając do pliku wykonywalnego kod pozwalający na wykonanie profilu dynamicznego programu (`-fprofile-generate`) oraz informacje pozwalające na współpracę z debuggerem (`-fprofile-generate`), natomiast drugie poleceń (z opcją `-fprofile-generate`) powoduje skompilowanie pliku `test.c` i dołączenie do niego informacji pozwalających na profilowanie za pomocą polecenia `gprof`.

Kiedy kompilujesz program nie używając żadnych opcji, w bieżącym katalogu tworzony jest plik wykonywalny (o ile komplikacja zakończy się bez błędów) o nazwie `a.out`. Jeśli chcesz, by nazwa tworzonego pliku wykonywalnego była inna, powinieneś użyć opcji `-o`. Przykładowo, aby skompilować program zapisany w pliku `licznik.c` do pliku wykonywalnego o nazwie `licznik` należy wydać polecenie:

```
gcc -o licznik licznik.c
```



Nazwa pliku wyjściowego musi pojawić się bezpośrednio po opcji `-o`. Nie należy wstawiać pomiędzy nie innych opcji.

Inne opcje kompilatora decydują o tym, na jakim etapie należy zakończyć proces kompilacji. Opcja `-c` powoduje zakończenie procesu po utworzeniu pliku pośredniego (domyślnie z rozszerzeniem `.o`), z pominięciem asemblacji i konsolidacji. Jest ona używana dość często, ponieważ umożliwia przyśpieszenie komplikacji złożonych, wieloplikowych programów.

Opcja `-s` powoduje zakończenie komplikacji po wygenerowaniu plików asemblera (domyślnie z rozszerzeniem `.s`). Opcja `-E` powoduje, że kompilator tylko wstępnie przetworzy pliki wejściowe, wykonując zawarte w nich dyrektywy preprocesora. W takim przypadku dane wyjściowe wysypane są na ekran, a nie do pliku.

Kompilator GCC stara się utworzyć kod wynikowy w możliwie najkrótszym czasie w taki sposób, by łatwo było znaleźć w nim ewentualne błędy. Oznacza to, że kolejność operacji pozostaje taka sama jak w pliku źródłowym i nie jest dokonywana żadna optymalizacja. Istnieje wiele opcji, dzięki którym możesz nakazać tworzenie mniejszych czy szybszych wersji programów, kosztem czasu ich komplikacji oraz łatwości wyszukiwania błędów. Najczęściej używa się opcji `-O` oraz `-O2`.

Opcja `-O` powoduje zastosowanie podstawowych technik optymalizacyjnych. Owocuje to zwykle powstaniem szybciej działających wersji programów. Opcja `-O2` powoduje, że wygenerowany zostanie możliwie krótki oraz szybki kod. Czas komplikacji w tym przypadku jest dłuższy, ale za to program wynikowy działa szybciej.

Poza tymi opcjami istnieje jeszcze wiele opcji niskiego poziomu, których można użyć do dalszego przyspieszania działania programu. Są one jednak bardzo specyficzne i powinieneś używać ich tylko wtedy, gdy dokładnie zdajesz sobie sprawę z tego, co powodują, i jakie mogą być ich konsekwencje. Bardziej szczegółowe informacje o tych opcjach znajdziesz na stronach `man`.

Opcje współpracy z debugerem i programem profilującym

Spośród kilku opcji dotyczących wstawiania dodatkowego kodu służącego do określania szybkości wykonywania programu i ułatwiającego uruchamianie i testowanie, najczęściej używane są dwie: `-g` oraz `-pg`.

Opcja `-g` powoduje dołączenie do pliku wykonywalnego informacji dla debugera `gdb`, który często okazuje się niezbędny w procesie wyszukiwania błędów. GCC oferuje coś, czego nie daje większość innych kompilatorów: możliwość łącznego użycia opcji `-g` oraz `-O` (która powoduje wygenerowanie zoptymalizowanej wersji programu). Jest to bardzo przydatne, szczególnie jeśli chcesz testować produkt jak najbardziej zbliżony do wersji końcowej. Powinieneś jednak zdawać sobie sprawę, że część kodu zostanie przez kompilator nieco zmodyfikowana.

Opcja `-pg` pozwala na dołączenie do programu wykonywalnego dodatkowego kodu, który, po uruchomieniu programu, wygeneruje informacje o czasie wykonania poszczególnych sekcji programu. Dane te mogą być przeglądane za pomocą programu `gprof`. Więcej informacji na jego temat znajdziesz w podrozdziale „`gprof`”.

Wyszukiwanie błędów - debugger `gdb`

Wraz z Linuxem rozprowadzany jest program `gdb`, który jest bardzo potężnym debuggerem, służącym do wyszukiwania błędów w programach napisanych w językach C i C++. Umożliwia dostęp do struktur danych i pamięci wykorzystywanej przez program podczas jego działania. Oto jego podstawowe zalety:

- υ pozwala śledzić wartości zmiennych podczas wykonywania programu,
- υ pozwala ustawiać pułapki, które zatrzymują program po osiągnięciu danego wiersza kodu,
- υ pozwala wykonywać program krokowo, wiersz po wierszu.

Przy uruchamianiu programu `gdb` można również podać różne parametry w wierszu poleceń. Zwykle program ten uruchamia się podając nazwę pliku wykonywalnego, w którym chcemy szukać błędów:

```
gdb <nazwa_pliku>
```

W takim przypadku plik wykonywalny zostanie załadowany automatycznie. Można również uruchomić `gdb` w taki sposób, by możliwe było oglądanie zawartości pliku ze zrzutem pamięci (ang. *core dump*) wygenerowanego przez program; można też dołączyć `gdb` do działającego już procesu. Aby obejrzeć listę dostępnych opcji z ich krótkim opisem, zajrzyj na stronę `man` lub uruchom `gdb` z opcją `-h`.

Aby program `gdb` mógł działać poprawnie, do pliku wykonywalnego muszą zostać dołączone przez kompilator informacje o typach i nazwach zmiennych, o mapowaniu kodu na linie programu źródłowego itd., które pozwolą na powiązanie kodu źródłowego

i skompilowanego kodu programu. W tym celu w poleceniu komplikującym plik źródłowy należy podać również opcję `-g`.

Podsumowanie

Kompilacja jądra systemu i uaktualnianie jego wersji przebiega zwykle bezproblemowo, o ile zdajesz sobie sprawę z tego, co robisz. Nie powinieneś obawiać się tego procesu, ale zawsze należy mieć pod ręką zestaw dyskietek startowych. Jeśli tylko dostępne są odpowiednie instrukcje, należy ich ścisłe przestrzegać, ponieważ nowe oprogramowanie ma często specjalne wymagania dotyczące dołączania do jądra systemu czy zastępowania istniejących fragmentów kodu.

Jeśli chcesz dowiedzieć się czegoś więcej o kompilatorze języka C rozprowadzanym z większością dystrybucji Linuxa, przejdź do rozdziału 26. „Programowanie w języku C”.

W części czwartej, w rozdziale 32. „Podstawy administrowania systemem” i kilku następnych, zamieściliśmy informacje pozwalające na prawidłowe skonfigurowanie jądra systemu.

O tym, w jaki sposób korzystać z systemów zarządzania kodem źródłowym, pozwalających na zmniejszenie liczby przechowywanych plików zajmujących niepotrzebnie miejsce na dysku twardym, możesz dowiedzieć się z rozdziału 56. „Zarządzanie kodem źródłowym”.

Rozdział 58.

Tworzenie sterowników urządzeń

Tim Parker

W tym rozdziale:

- υ Sterowniki urządzeń
- υ Przerwania
- υ Struktura sterownika urządzenia systemu Linux
- υ Używanie nowego sterownika

Sterownik urządzenia stanowi interfejs pomiędzy systemem operacyjnym a urządzeniem dołączonym do systemu. Typowy sterownik składa się z pewnej liczby funkcji potrafiących obsługiwać operacje wejścia / wyjścia zleczane przez system. Takie rozwiązanie pozwala na ujednolicenie interfejsu pomiędzy jądrem systemu i urządzeniami.

Nie możemy niestety omówić wszystkich aspektów tworzenia sterowników urządzeń w jednym rozdziale. Na ten temat napisano kilka całkiem pokaźnych książek. Ponieważ sterowniki urządzeń nie są raczej tworzone przez zwykłych użytkowników, a przez utalentowanych programistów, informacje przedstawione poniżej mają jedynie charakter przedstawienia zagadnienia.

Fragmenty kodu prezentowane w tym rozdziale pochodzą z zestawu przykładowych sterowników napisanych w języku C. Sterowniki te są przenośne i choć zostały zaprojektowane dla systemu UNIX, będą działać poprawnie również w Linuxie. Jeśli zdecydujesz się na tworzenie własnych sterowników, podane tu przykłady powinieneś traktować tylko jako wskazówek. Jeżeli zamierzasz zająć się tym zagadnieniem na poważnie, powinieneś zaopatrzyć się w jedną ze specjalistycznych książek poświęconych mu w całości.



Tworzenie sterowników urządzeń jest teoretycznie całkiem proste. Mimo tego przy próbie napisania pierwszego sterownika będziesz zaskoczyły liczbą problemów, z którymi się zetkniesz. Wiele osób uważa tworzenie sterowników za swego rodzaju sztukę, ale tak naprawdę wymaga to jedynie odrobinę praktyki i doświadczenia. Napisanie dobrego sterownika daje ogromną satysfakcję.

Sterowniki urządzeń

System Linux używa sterownika dla każdego urządzenia podłączonego do systemu. Najbardziej podstawowe sterowniki są częścią jądra systemu i są ładowane podczas jego uruchamiania. Dzięki sterownikom urządzenia mogą być traktowane przez system tak samo jak pliki – można je adresować, przekierowywać do nich dane czy używać podczas wykorzystywania mechanizmu potoków, zupełnie tak, jak w przypadku zwykłych plików.

Każde urządzenie podłączone do systemu linuxowego jest opisane w pliku programu sterownika, a niektóre parametry są również zapisane w pliku urządzenia, przechowywanym zwykle w katalogu `/dev`. Jeśli do systemu zostanie dodane nowe urządzenie, należy zainstalować lub też napisać własnoręcznie odpowiedni sterownik. Każdemu urządzeniu musi również odpowiadać plik w katalogu `/dev`. Jeśli warunki te nie zostaną spełnione, urządzenie nie będzie mogło zostać wykorzystane w systemie linuxowym.

Do każdego pliku urządzenia przypisany jest numer urządzenia, pozwalający systemowi operacyjnemu jednoznacznie je zaadresować. W systemie Linux numer urządzenia składa się z dwóch części: numeru głównego, określającego ogólny typ urządzenia, na podstawie którego wybierany jest odpowiedni sterownik, oraz numeru poboczny pozwalającego wyszczególnić każde z urządzeń obsługiwanych przez jeden sterownik. Dla przykładu, kilka dysków twardych w systemie może posiadać taki sam numer główny urządzenia (ponieważ są obsługiwane przez ten sam sterownik), ale każdy z nich musi posiadać inny numer poboczny, jednoznacznie go identyfikujący.

Istnieją dwa podstawowe typy sterowników urządzeń: blokowe i znakowe. Każdy urządzenie w systemie UNIX-owym używa jednego lub obu tych typów sterowników. Najczęściej spotykane są sterowniki blokowe. Pozwalają one na przesyłanie do i z urządzenia danych buforowanych przez jądro systemu (kopijowanych w odpowiednie miejsce w pamięci operacyjnej). Pierwotnie sterowniki takie były wykorzystywane tylko dla dysków twardych, ale szybko znalazły zastosowanie również do obsługi innych urządzeń pamięci masowej, takich jak napędy taśmowe, magnetoptyczne, a nawet modemy synchroniczne i niektóre szybkie drukarki.

Urządzenia pracujące w trybie znakowym różnią się od blokowych pod dwoma zasadniczymi względami. Po pierwsze, wyniki operacji wejścia / wyjścia mogą być przekazywane bezpośrednio do przestrzeni adresowej procesu, z pominięciem buforowania przez jądro systemu. Po drugie, żądania operacji wejścia / wyjścia zwykle przekazywane są bezpośrednio do urządzenia. Przykładami urządzeń znakowych są terminale i drukarki, podobnie jak modemy asynchroniczne i niektóre modele napędów taśmowych.

Urządzenia blokowe opierają swe działanie na funkcji „strategii”, pozwalającej na odczytanie i zapisanie do urządzenia bloku danych. Dla urządzeń znakowych dostępny jest szereg specjalnych funkcji pozwalających na bezpośrednią komunikację, nazywanych `ioctl()`. Urządzenia blokowe czasem również działają jako znakowe, właśnie po to, by możliwe było użycie funkcji `ioctl()`. Dobrym przykładem jest napęd taśmowy, który może współpracować zarówno ze sterownikiem znakowym, jak i blokowym, zależnie od typu zapisywanych danych.

Niezależnie od typu, sterownik urządzenia podejmuje szereg podstawowych kroków za każdym razem, gdy ma nawiązać komunikację z urządzeniem. Najpierw sprawdza on, czy urządzenie jest gotowe i dostępne. Jeśli tak, jest ono „otwierane”, co umożliwia procesowi dostęp do niego. Następnie zwykle wywoływanie są funkcje `read` i `write`, służące odpowiednio do odczytu i zapisywania danych do urządzenia, po czym urządzenie jest „zamykane”, dzięki czemu inne procesy mogą mieć do niego dostęp.

Przerwania

Przerwania to sygnały przesyłane przez urządzenia do systemu operacyjnego, powiadające go o konieczności obsługi danego urządzenia. Przerwania są generowane przy wszystkich operacjach wejścia / wyjścia. System przerwa wykorzystywany w Linuxie jest podobny do DOS-owego, więc jeśli jesteś obeznany z przerwaniami DOS-owymi, znasz już większą część teorii.

Po odebraniu sygnału przerwania system operacyjny wstrzymuje wszystkie wykonywane procesy i obsługuje przerwanie. W większości przypadków przerwania są obsługiwane przez sterownik urządzenia. Przerwania nie powinny w żaden sposób zaburzać działania innych procesów, za wyjątkiem co najwyżej chwilowego ich wstrzymania.

Pewien problem stwarza fakt, że wywołanie przerwania nie powinno prowadzić do zawieszenia działania jądra systemu ani procesu obsługi urządzenia, za wyjątkiem ściśle określonych sytuacji. Przerwania, które nie są prawidłowo obsługiwane lub sprawdzane, mogą prowadzić do zawieszenia sterownika urządzenia, który obsługiwał komunikację z nim.

Obsługa przerwa jest zwykle wstrzymywana tylko na czas wykonywania operacji o istotnym znaczeniu. Obszary kodu sterownika, które nie powinny być przerywane, nazywane są sekcjami krytycznymi lub niewstrzymywalnymi. Zwykle zawieszenie obsługi przerwań na czas realizacji sekcji krytycznych realizowane jest przez podniesienie priorytetu procesora do poziomu równego lub wyższego od priorytetu przerwań. Po zakończeniu wykonywania takiej sekcji priorytet procesora jest obniżany do poprzedniej wartości.

Priorytet przerwań można modyfikować za pomocą funkcji `sp15()`, `sp16()`, `sp17()` i `sp1x()`. Wywołanie jednej z trzech pierwszych funkcji powoduje zablokowanie obsługi przerwań. Funkcja `sp15()` wyłącza obsługę przerwań generowanych przez dyski, drukarki i klawiaturę, funkcja `sp16()` wyłącza zegar systemowy, natomiast `sp17()` wyłącza

obsługę wszystkich przerwań, w tym pochodzących od urządzeń szeregowych. Funkcje te zwracają kod oznaczający wartość poziomu przerwań, do którego można wrócić za pomocą funkcji `splx()`.

Przed wejściem do sekcji krytycznej kodu należy więc wykonać polecenie

```
pierw_poz = spl5();
```

które spowoduje zawieszenie obsługi przerwań aż do momentu wywołania funkcji

```
splx():  
splx(pierw_poz);
```

Poniższy przykład przedstawia wielokrotne zmiany poziomu obsługiwanych przerwań w sterowniku urządzenia.

```
int poz_a, poz_b;  
poz_a = spl5();  
/* kod, który nie powinien być */  
/* przerywany przez dyski */  
poz_b = spl7();  
/* instrukcje, które nie mogą */  
/* być w ogóle przerywane */  
splx(poz_b);  
/* kod końcowy, który nie może */  
/* być przerwany przez dyski */  
splx(poz_a);
```

Ta dość dziwaczna na pierwszy rzut oka żonglerka poziomami przerwań jest niezbędna, aby uniknąć wstrzymania obsługi sterownika urządzenia i jądra systemu, co może prowadzić do nieprawidłowej pracy systemu. Mechanizmy zabezpieczające powinny być uruchamiane na tak krótko, jak tylko jest to możliwe.

Zwykle nie należy używać funkcji `spl6()` i `spl7()`. W niektórych przypadkach wywołanie funkcji `spl6()` może spowodować opóźnienie zegara systemowego, natomiast wywołanie funkcji `spl7()` może spowodować utratę danych przesyłanych przez urządzenia szeregowe, chyba że blokada jest aktywna przez bardzo krótki okres czasu. Mimo wszystko jednak w większości przypadków przed wejściem do sekcji krytycznej wystarcza wywołanie funkcji `spl5()`.

Struktura sterownika urządzenia dla systemu Linux

Kod źródłowy sterownika urządzenia pod względem struktury nie odbiega od kodu zwykłych programów. W systemie Linux sterowniki w zasadzie tworzone są w języku C, choć czasem korzysta się również z asemblera i języka C++.

Pliki nagłówkowe

Typowy sterownik urządzenia zawiera plik nagłówkowy, w którym znajdują się dyrektywy włączające do kodu deklaracje funkcji systemowych, adresów rejestrów urządzeń, definicje zawartości oraz definicje zmiennych globalnych używanych przez sterownik. W większości sterowników wykorzystuje się następujący zestaw plików nagłówkowych:

- param.h** parametry jądra systemu;
- dir.h** parametry katalogów;
- user.h** definicje użytkownika
- tty.h** definicje terminali i bufora `clist`
- buf.h** informacje o buforowaniu

Plik `tty.h` jest wykorzystywany w sterownikach pracujących w trybie znakowym, natomiast plik `buf.h` – w trybie blokowym.

Adresy rejestrów urządzenia, definiowane w jednym z plików nagłówkowych, są zależne od samego urządzenia. Dla urządzeń znakowych odpowiadają one zwykle adresom portów, takich jak port wejścia / wyjścia czy zawierający bity stanu u kontrolne. Polecenia zmieniające stan urządzenia są zdefiniowane jako kody urządzenia.

Oto przykład inicjalizacji rejestrów urządzenia dla sterownika terminalu standardowego (UART):

```
/* definicja rejestrów */
#define RRDATA      0x01 /* odbior */
#define RTDATA      0x02 /* nadawanie */
#define RSTATUS      0x03 /* stan */
#define RCTRL        0x04 /* bity kontrolne */
...itd.
/* definicja rejestrów stanu */
#define SRRDY       0x01 /* dane odebrane */
#define STRDY       0x02 /* nadajnik gotowy */
#define SPERR        0x03 /* błąd parzystosci */
#define SCTS        0x04 /* gotowy do wysłania inf. o stanie */
...itd.
```

Funkcje, które musi udostępniać sterownik, zależą od natury samego urządzenia. Wszystkie sterowniki muszą posiadać funkcje `open()` i `close()`, pozwalające na realizację operacji wejścia / wyjścia.

Otwieranie urządzenia

Podprogram `open()` musi sprawdzać, czy wybrane urządzenie jest prawidłowe, czy proces wywołujący może uzyskać do niego dostęp (czy ma prawo dostępu i czy urządzenie

jest gotowe) oraz inicjalizować urządzenie. Podprogram `open()` jest wywoływany za każdym razem, gdy jakiś proces korzysta z urządzenia.

Poniżej prezentujemy procedurę `open()` obsługującą ogólne urządzenie terminalu o nazwie `td`.

```
tdopen (device, flag)
int device, flag;
{
    /* definicje zmiennych lokalnych oraz szczegoly */
    /* zostaly pominiete */

    /* sprawdz numer urzedzenia /
    if (UNMODEM (device) >= NTDEVS)
    {
        seterror (ENXIO);
        return;
    }

    /* sprawdz, czy urzadzenie jest uzywane */
    /* jezeli tak, to wymus zwolnienie jesli */
    /* uzytkownikiem jest root (suser) */
    tp = &td_tty[UNMODEM(device)];
    adres = td_address[UNMODEM(device)];
    if ( ( tp->t_lflag & XCLUDE ) && !suser() )
    {
        seterror (EBBUSY);
        return;
    }

    /* Jesli urzadzenie nie jest otwarte, zainicjalizuj je */
    /* wywolujac funkcje ttinit() */
    if ( (tp->t_state & (ISOPEN|WOPEN)) == 0 )
    {
        ttinit(tp);
        /* inicjalizacja znacznikow i wywolanie tdpParam() */
        /* aby ustalic parametry linii */
        tdpParam (device);
    }

    /* Jesli uzywany jest modem, sprawdz stan nosnej */
    /* Jesli polaczenie bezposrednie, ustal */
    /* wartosc znacznika wykrywania nosnej */
    /* ustal priorytet przerwan aby zapobiec nadpisania */
    /* czekaj na sygnal wykrycia nosnej */
    /* na potrzeby tego przykladu */
    /* implementacja zostala pominieta */
}
```

Zamykanie urządzenia

Podprogram `close()` używany jest tylko po zakończeniu komunikacji z urządzeniem. Powoduje on wyłączenie obsługi przerwań pochodzących od urządzenia oraz przesłanie odpowiednich informacji kończących jego pracę. Wszystkie wewnętrzne odniesienia do urządzenia są usuwane. Podprogram `close()` w większości przypadków nie jest wymagany, ponieważ urządzenia są traktowane jako dostępne przez cały czas. Wyjątek stanowią dyski wymienne i urządzenia wymagające wyłączności na używanie. Również nie-

które modemy wymagają, aby w procedurze `close()` umieścić kod pozwalający na zwolnienie linii telefonicznej.

Również tu posłużymy się przykładem pochodzącym ze sterownika obsługującego terminal.

```
tdclose (device)
{
    register struct tty *tp;
    tp = &td_tty[UNMODEM(device)];
    (*linesw[tp->t_line].l_close) (tp);
    if (tp->t_cflag & HUPCL)
        tdmodem(device, TURNOFF);
    /* usun znacznik wylacznosci */
    ip->t_lflag &= ~XCLUDE;
}
```

Funkcje strategii

Funkcje strategii (używane tylko przez sterowniki urządzeń blokowych) są wywoływanie z parametrem prowadzącym do nagłówka bufora, do którego jądro systemu składa dane. Nagłówek bufora zawiera informacje dotyczące odczytu czy zapisu danych, wraz z adresem miejsca pamięci, w którym znajdują się dane do wysłania lub mają znaleźć się odbierane dane. Rozmiar bufora jest zwykle ustalany podczas instalacji i wahaj się pomiędzy 512 i 1024 bajtami. Rozmiar ten definiowany jest w pliku `param.h` jako wartość zmiennej `BSIZE`. Rozmiar bloku danych używanego przez urządzenie może być mniejszy od rozmiaru bufora – w takim przypadku sterownik wykona kilka operacji odczytu czy zapisu pod rząd.

Funkcja strategii zostanie przedstawiona na przykładzie prostego sterownika dysku twardego. Nie zamieszczamy samego kodu źródłowego, a jedynie szkieletowe informacje o działaniu funkcji.

```
int hdstrategy (bp)
register struct buf *bp;
{
    /* inicjalizuj dysk i numery partycji */
    /* ustal wartosci zmiennych lokalnych */

    /* sprawdz, czy dysk i partycja sa prawidlowe */
    /* ustal numer cylindra docelowego */
    /* wylacz przerwania */
    /* wstaw zadanie do kolejki */
    /* sprawdz kontroler, jesli nie jest aktywny, uruchom go */
    /* przywroc poprzedni poziom przerwan */
}
```

Funkcje write()

Urządzenia pracujące w trybie znakowym używają funkcji `write()`, która sprawdza, czy jej argumenty są poprawne, a następnie kopiuje dane z obszaru pamięci procesu do bufora sterownika urządzenia. Po skopiowaniu wszystkich danych lub zapełnieniu bufora uruchamiane są procedury wejścia / wyjścia aż do opróżnienia bufora, po czym pro-

ces się powtarza. Dane z pamięci procesu są odczytywane za pomocą prostej funkcji (`cpass`) zwracającej `-1` w przypadku, gdy osiągnięty zostanie koniec obszaru pamięci. Zapisywanie danych do pamięci procesu odbywa się za pośrednictwem funkcji komplementarnej (`passc`). Oto przykładowa funkcja `write()` obsługująca urządzenie terminalu:

```
tdwrite (device)
{
    register struct tty *tp;
    tp = &td_tty[UNMODEM(device)];
    (*linesw[tp->t_line].l_write) (tp);
}
```

Większe ilości danych są obsługiwane przez proces o nazwie `copyio`, który pobiera argumenty określające adres danych źródłowych i miejsca, do którego mają one zostać skopiowane, oraz liczbę bajtów i znacznik stanu.

Funkcje `read()`

Funkcja `read()` dla urządzenia znakowego przesyła dane z urządzenia do pamięci procesu. Jest to operacja analogiczna do działania procedury `write()`. Kod źródłowy tej procedury dla urządzenia terminalu ma następującą postać:

```
tdread (device)
{
    register struct tty *tp;
    tp = &td_tty[UNMODEM(device)];
    (*linesw[tp->t_line].l_read) (tp);
}
```

Jeśli funkcje `read()` i `write()` mają skopiować kilka bajtów, używany jest niewielki bufor, o nazwie `clist`. Używa on szeregu powiązanych list, które z kolei do usuwania i wstawiania znaków do bufora używają funkcji `getc` i `putc`. Nagłówek bufora `clist` zawiera również informację o liczbie przechowywanych znaków.

Podprogramy start i ioctl

Podprogram `start` używany jest zarówno przez urządzenia blokowe, jak i znakowe. Pobiera on dane z kolejki urządzenia i kieruje je kolejno do urządzenia. Urządzenia blokowe do kolejkowania danych używają funkcji strategii, natomiast urządzenia znakowe – prostego bufora `clist`. Podprogram `start` w czasie przesyłania danych do urządzenia automatycznie obsługuje znaczniki mówiące o tym, że urządzenie jest zajęte. Po zakończeniu procesu urządzenie wywołuje podprogram `intr`, który inicjalizuje je ponownie, przygotowując do obsługi następnego procesu.

Podprogram `ioctl()`, używany w sterownikach urządzeń znakowych, pozwala przesyłać do sterownika specjalne instrukcje, umożliwiające na przykład zmianę metody komunikacji pomiędzy sterownikiem a systemem operacyjnym czy wykonanie operacji ściśle zależnych od urządzenia (jak przewinięcie taśmy czy alokacja pamięci).

Funkcję `ioctl()` zilustrujemy przykładem pochodząącym ze sterownika terminal. W tym przypadku do ustalenia parametrów urządzenia wywoływana jest inna funkcja. Nie przedstawiamy jej kodu źródłowego, tylko ogólny zarys pozwalający zorientować się w zasadach działania.

```
tdioctl (device, cmd, arg, mode)
int device;
int cmd;
int mode;
faddr_t arg;
{
    if (tticom(&td_tty[UNMODEM(device)], cmd, arg, mode))
        tdpParam (device);
}

tdparam(device)
{
    /* inicjalizacja zmiennych */
    /* pobierz adres i znaczniki linii, do której */
    /* odnosi się polecenie */
    adr = td_addr[UNMODEM(device)];
    cflag = td_tty[UNMODEM(device)].t_cflag;

    /* sprawdz predkosc; jeśli zero to zwolnij linie */
    /* zmien predkosc */
    /* ustal zarządzanie linia */
    /* obsluz przerwania */
}
```

Używanie nowego sterownika

Proces dodawania do systemu nowego sterownika urządzenia składa się z kilku etapów. Najpierw identyfikowany jest podprogram obsługi przerwania, a następnie punkty wejścia do programu sterownika (na przykład adres procedury `open`) są dodawane do tabeli punktów wejścia sterownika. Cały sterownik jest komplikowany i dołączany do jądra systemu, a następnie umieszczany w katalogu `/dev` (w rozdziale 57. „Jądro systemu” znajdziesz dodatkowe informacje dotyczące dodawania sterowników do jądra systemu). Na koniec system musi zostać zrestartowany, po czym można przetestować działanie nowego sterownika. Oczywiście wprowadzenie modyfikacji do kodu sterownika wymaga powtórzenia całego procesu, więc usuwanie błędów może być dość męczące i prawdziwą sztuką jest osiągnięcie jak najmniejszej liczby restartów.



Przy tworzeniu sterowników urządzeń nie należy używać funkcji `sleep()` i `seterror()` oraz operacji zmiennoprzecinkowych we fragmentach, podczas wykonywania których zawieszona jest obsługa przerwań.

Czas, na jaki wstrzymywane są przerwania, powinien być jak najkrótszy; należy również uważać, by nie uszkodzić danych przechowywanych w buforze. Bardzo ważne jest także wykorzystanie jak najmniejszej liczby komórek stosu.

Można ułatwić sobie proces usuwania błędów ze sterowników urządzeń, umieszczając w kodzie źródłowym wywołania funkcji `printf` czy `getchar` w odniesieniu do innego urządzenia, na przykład konsoli. Takie rozwiązanie pozwala na pozostawienie śladu wykonania sterownika. Jeśli testujesz sterownik jako `root`, możesz również wykorzystać debugger `adb`, pozwalający na obserwację pamięci używanej przez jądro systemu w czasie pracy sterownika. Ostrożne korzystanie z tego programu pozwala na bezpośrednie śledzenie zmian wartości zmiennych czy zawartości komórek pamięci; bądź jednak uważny, ponieważ nieprawidłowe użycie debugera `adb` może prowadzić do załamania się systemu.

Jednym z najczęstszych problemów występujących przy tworzeniu sterowników (oprócz oczywiście błędów w kodzie źródłowym) jest „gubienie” przerwania lub uśpienie urządzenia podczas jego obsługi. Takie sytuacje powodują, że urządzenie się zawiesza. W większości przypadków do sterowników urządzeń dołącza się procedurę ograniczającą czas, przez jaki należy oczekiwany na odpowiedź urządzenia, dzięki czemu unika się zawieszania urządzenia. Jeśli spodziewane przerwanie nie zostanie zgłoszone w określonym czasie, urządzenie jest sprawdzane bezpośrednio, aby upewnić się, że przerwanie nie zostało „zgubione”. Jeśli okaże się, że przerwanie zostało zgubione, może ono zostać zasymulowane przez sterownik. Używanie funkcji `sp1()` w czasie testowania zwykle ułatwia wyizolowanie tego typu problemów.

Sterowniki urządzeń blokowych najczęściej tworzone są w oparciu o przerwania. Jednak coraz więcej programistów używa ostatnio techniki zwanej odpytywaniem, nadającej się do obsługi urządzeń znakowych. Technika ta polega na regularnym sprawdzaniu stanu urządzenia. Sterownik nie czeka więc na zgłoszenie przerwania, co pociąga za sobą nieco większe zużycie czasu procesora. Odpytywanie nie jest właściwym rozwiązaniem dla większości urządzeń, takich jak pamięci masowe, ale w przypadku urządzeń znakowych może przynieść pewne korzyści. Urządzenia szeregowe są zwykle obsługiwane za pomocą odpytywania, co pozwala na zmniejszenie liczby wywołań przerwań.

Terminal połączony z systemem linią o prędkości transmisji 19200 bodów generuje około 1920 przerwań na sekundę, co powoduje wielokrotną obsługę samego przerwania i wchodzenie oraz wychodzenie z funkcji jego obsługi. Jeśli zamiast przerwań zastosowany zostanie mechanizm odpytywania, przedział czasowy pomiędzy kolejnymi zgłoszeniami konieczności obsługi przez procesor może być znacznie zwiększyony, na przykład przez zastosowanie niewielkiego bufora przechowującego dane wysypane do lub odbierane z urządzenia. Urządzenia czasu rzeczywistego również nadają się do obsługi za pomocą mechanizmów odpytywania, ponieważ zmniejsza to znacznie liczbę wywoływanych przerwań. Jeśli chcesz używać tego mechanizmu w swoich sterownikach urządzeń, powinieneś zaopatrzeć się w jedną z książek poświęconych w całości sterownikom, ponieważ jest to temat dość złożony.

Podsumowanie

Większość użytkowników systemu Linux nigdy nie będzie tworzyć własnych sterowników urządzeń, ponieważ przeważnie sterowniki takie są już dostępne. Jednak jeśli posiadasz zupełnie nowe urządzenie lub też sterownik, którego używasz, zawiera jakiś błąd, możesz pokusić się o stworzenie własnego sterownika. Nie jest to zadanie bardzo trudne

(pod warunkiem, że znasz dobrze język C), ale zwykle dużo kłopotów sprawia usuwanie powstających usterek. Programista tworzący sterownik musi cały czas uważać, by nie wpływał on na działanie innych procesów lub urządzeń. Napisanie poprawnie działającego sterownika jest jednak powodem do dumy.

Rozdział 26. „Programowanie w języku C”, omawia zagadnienia związane z językiem C dla systemu Linux, którego można użyć do tworzenia sterowników.

Język Perl, bardzo poręczny i doskonale nadający się do tworzenia krótkich i zaskakująco wydajnych programów, omówiony jest w rozdziale 28. „Perl”.

Inne języki programowania dostępne dla systemu Linux przedstawione są w skrócie w rozdziale 30. „Inne kompilatory”.

Rozdział 59.

Projekt Wine

Robert Pfister

W tym rozdziale:

- υ Obecny stan projektu Wine
- υ Konfiguracja programu Wine
- υ Używanie programu Wine
- υ Zasady działania programu Wine
- υ Gdzie kończy się Wine, a zaczyna MS-Windows?
- υ Ograniczenia programu Wine

Wine to skrót od słów Windows Emulator. Program ten pozwala na uruchamianie aplikacji przeznaczonych dla MS-Windows w środowisku X Window działającym w systemach UNIX-owych. Podobnie jak DOSemu, przy uruchamianiu programów dla Windows Wine wykorzystuje bezpośrednio możliwości oferowane przez architekturę procesorów 386 firmy Intel. Program Wine po prostu tłumaczy wszystkie wywołania funkcji MS-Windows na odpowiadające im wywołania funkcji systemowych UNIX-a i X Window. Podobnie jak w systemie OS/2, programy dla systemu Windows uruchamiane w środowisku Wine mogą korzystać z możliwości oferowanych przez system operacyjny. Wine jest po prostu jednym z wielu procesów użytkownika w systemie linuxowym, więc również jest zabezpieczony przed uszkodzeniem przez inne procesy. W systemie OS/2 ta właściwość nazywana jest po angielsku *crash-protection*, czyli zabezpieczenie przed załamaniem. Ponieważ wielozadaniowość jest podstawą systemu Linux, procesy Wine mogą współistnieć z innymi procesami, nie stwarzając problemów spotykanych przy pracy z systemem MS-Windows.

Obecny stan projektu Wine

Podobnie jak większość programów linuxowych, Wine jest tworzony przez grupę ochotników. Program Wine jest obecnie w fazie wczesnych testów. Tylko niektóre prostsze aplikacje MS Windows działają bez żadnych problemów. Mój ulubiony zestaw gier dla Windows, Pipe Dream firmy Lucas Arts, działa zupełnie zadawalająco pod kontrolą Wine.

Choć da się grać w Pipe Dream i inne proste gry, nie wszystko działa doskonale. Można zauważać pewien spadek prędkości, zdarzają się również problemy z odświeżaniem ekranu.

Firma Sun Soft opracowała produkt dość podobny do Wine, o nazwie WABI, przeznaczony dla UNIX-owych stacji roboczych. WABI jest na rynku od 1994 roku i obsługuje nawet tak złożone aplikacje jak Microsoft Excel czy Lotus Smart Suite dla systemu Windows 3.11. Program WABI nie obsługuje jednak aplikacji systemu Windows 95. Można się spodziewać, że za jakiś czas program Wine będzie również dobrze sobie radził z aplikacjami Windows ogólnego przeznaczenia.

Konfiguracja Wine

Program Wine jest rozprowadzany tylko w postaci kodu źródłowego. Jeśli posiadasz odpowiednie oprogramowanie i nieco cierpliwości, konfigurowanie Wine nie powinno sprawić Ci większych kłopotów – nawet jeśli nie jesteś programistą.

Wymagania systemowe

Aplikacje Wine będą działać z rozsądną prędkością w każdym systemie linuxowym, w którym działa system X Window. Teoretycznie Wine powinien posiadać pewną przewagę nad systemem Windows, który jest zależny od środowiska MS-DOS. Mimo wszystko dane doświadczalne wskazują, że pod kontrolą obecnej wersji Wine aplikacje działają wolniej niż w środowisku Windows.

Aby wykorzystywać program Wine, trzeba posiadać zainstalowaną na partycji dostępnej dla Linuxa wersję systemu MS-Windows 3.1. Wygodnie jest również uruchamiać dostępne aplikacje MS-Windows z tych samych katalogów, w których są one zainstalowane w ich rodzimym środowisku DOS i Windows. Typowy użytkownik Linuxa posiada również zainstalowany system DOS i Windows, więc konfiguracja sprowadza się tylko do udostępnienia odpowiednich katalogów dla systemu Linux. Wersje jądra do 1.1.83 nie obsługują skompresowanych za pomocą programów `stacker` czy `drvspace` plików MS-DOS.



Niektóre programy instalacyjne systemu Linux pozwalają na zainstalowanie partycji DOS-owej w jednym z podkatalogów systemu plików. Jeśli nie skonfigurowałeś partycji w ten sposób, powinieneś dodać do pliku `/etc/fstab` następujący wiersz:

```
/dev/hda1 /c MSDOS defaults
```

`hda1` to nazwa partycji, która zawiera system MS-DOS¹, natomiast `/c` to podkatalog, w którym dostępny będzie tenże system. W tym przykładzie założyliśmy, że katalog `/c` został wcześniej utworzony - jeśli nie, należy to zrobić za pomocą polecenia `mkdir`.

Program Wine jest rozprowadzany w postaci kodu źródłowego, który przed użyciem musi zostać skompilowany. Potrzeba do tego około 10 MB wolnego miejsca na dysku. Sam kod źródłowy zajmuje ok. 3.5 MB. Aby skompilować Wine, będziesz potrzebował:

- υ kompilatora GCC,
- υ biblioteki LibC,
- υ systemu X Window wraz z elementami dla programistów (ang. *development*),
- υ wersji jądra nowszej niż 0.99.13

Skąd można załadować Wine

Nowe wersje systemu Wine opracowywane są średnio raz w tygodniu. Główne węzły FTP oferujące oprogramowanie linuxowe zwykle udostępniają najnowszą wersję Wine. Dla przykładu, w węźle `sunsite.unc.edu` programu Wine należy szukać w katalogu `/pub/Linux/ALPHA/Wine`. Kolejne wersje tego programu opatrzone są numerami opartymi na dacie ich powstania – na przykład wersja Wine-950727 pochodzi z 27 lipca 1995 roku. Nowsze wersje mają oczywiście bardziej aktualne daty. Więcej informacji można znaleźć na stronie WWW pod adresem

<http://daedalus.dra.hmg.gb/gale/wine/wine.html>.

Jak zainstalować Wine

W przeciwieństwie do programu DOSemu, program Wine nie musi być instalowany w żadnym konkretnym katalogu. Dla wygody warto utworzyć dowiązanie symboliczne o nazwie np. `/usr/wine` do katalogu, w którym Wine jest faktycznie zainstalowany (dajmy na to `/usr/src/Wine950122`), wydając polecenie `ln` w następujący sposób:

```
ln -s /usr/src/Wine950122 /usr/wine
```

¹ Nowsze wersje jądra systemu obsługują system plików o nazwie `vfat`, który warto zastosować zamiast podanego w przykładzie systemu `MSDOS`, ponieważ potrafi on poprawnie obsługiwać długie nazwy plików używane w systemie MS-DOS 7.0 (*przyp. tłum.*).

Kod źródłowy programu Wine rozprowadzany jest w postaci spakowanego archiwum programu tar. Aby je rozpakować, należy wydać polecenie o następującej formie:

```
tar -zxvf nazwapliku.tar.gz
```

Jak skonfigurować Wine przed komplikacją

Kod źródłowy programu Wine przed komplikacją wymaga skonfigurowania. Skrypt o nazwie `Configure` pozwala zautomatyzować ten proces, pobierając od użytkownika potrzebne informacje i na ich podstawie tworząc właściwe pliki konfiguracyjne. Konfiguracja Wine składa się z trzech zasadniczych etapów:

1. konfiguracja procesu komplikacji,
2. podanie parametrów czasu wykonania,
3. automatyczna konfiguracja dostosowująca Wine do konkretnego systemu.

Skrypt konfiguracyjny zadaje następujące pytania:

```
Build Wine as emulator or library (E/L) [E] ?  
Short filenames (Y/N) [N] ?  
Use the XPM library (Y/N) [N] ?  
Language [En/De/No] ?  
Global configfile name /usr/local/etc/wine.conf
```

(czy skompilować Wine jako bibliotekę, czy emulator, czy używać krótkich nazw plików, czy wykorzystać bibliotekę XPM, jakiego użyć języka i jaką nazwę ma mieć globalny plik konfiguracyjny?). Bezpiecznie jest zaakceptować sugerowane wartości domyślne wciskając Enter w odpowiedzi na każde z pytań. Podane parametry są zapisywane do globalnego pliku konfiguracyjnego o nazwie `autoconf.h`. Jeśli konieczna okaże się ich modyfikacja, należy ponownie uruchomić skrypt `Configure`, dzięki czemu można uniknąć błędów mogących powstać podczas ręcznej edycji pliku `autoconf.h`.

Wstępna konfiguracja parametrów czasu wykonania za pomocą skryptu `Configure`

Pytania zadawane w tej części procesu konfiguracyjnego odnoszą się do odpowiednich wpisów w globalnym pliku konfiguracyjnym `/usr/local/etc/wine.conf`. Poszczególne pytania są dość oczywiste:

```
Which directory do you want to use as A:  
(który katalog ma być traktowany jako napęd A:)  
Which directory do you want to use as C:  
(który katalog ma być traktowany jako napęd C:)
```

W odpowiedzi na nie należy podać nazwy katalogów, w których są zamontowane DOS-owe dyski A: i C:. Jeśli na przykład partycja zawierająca system MS-Windows zamontowana jest w katalogu `/c`, właśnie ten katalog należy wskazać w odpowiedzi na drugie pytanie.

Jeśli nie planujesz używać stacji dyskietek, nie musisz przejmować się faktem, że napęd A: nie będzie prowadził do prawidłowego katalogu.

Następne pytania dotyczą ścieżek dostępu do katalogów systemowych – katalogu Windows, katalogu System, katalogu, w którym mają być przechowywane pliki tymczasowe i ścieżek przeszukiwania dla programów i bibliotek DLL.

```
Where is the Windows directory 'c:\windows'  
Where is the System directory 'c:\windows\system'  
Where should Windows apps store temp files 'c:\windows\temp'  
Which path should be used to find progs/DLL's  
'c:\windows;c:\windows\system'
```

Podane tu ścieżki dostępu powinny odpowiadać ścieżkom używanym w systemie MS-Windows. Ponieważ domyślnie system ten instalowany jest w katalogu `c:\windows`, podpowiadane odpowiedzi będą w większości przypadków odpowiednie.

Następne pytanie dotyczy ścieżki dostępu do pliku `sysres.dll`, zawierającego zasoby systemu Wine, takie jak obrazki czy okienka dialogowe wyświetlane podczas pracy Wine.

```
Where is sysres.dll '/usr/wine/sysres.dll'
```

Również tu w większości przypadków nie są potrzebne żadne modyfikacje.

Podobnie jak w programie DOSemu, w Wine porty szeregowe i równoległe mogą zostać powiązane z dowolnymi zainstalowanymi w systemie Linux portami podobnego typu. Najprościej jest powiązać porty COM i LPT w taki sam sposób, jak są one skonfigurowane w systemie DOS:

```
Where is COM1" CF_Com1 '/dev/cua0'  
Where is COM2" CF_Com2 '/dev/cua1'  
Where is LPT1" CF_Lpt1 '/dev/lp0'
```

Następne pytanie dotyczy nazwy pliku, w którym rejestrowane będą komunikaty generowane podczas pracy Wine. Wybranie pliku o nazwie `CON` spowoduje przesyłanie ich do standardowego urządzenia wyjściowego, czyli najczęściej konsoli – takie rozwiązanie umożliwia przekierowanie ich do dowolnego pliku podczas uruchamiania Wine.

```
Log messages to which file (CON = stdout) 'CON'
```

Domyślnie Wine generuje mnóstwo komunikatów, które mogą nieco zwalniać jego działanie. Jeśli chcesz tego uniknąć, możesz skierować je do urządzenia `/dev/null`, podając jego nazwę zamiast nazwy pliku.

Następnie wyświetlana jest dłuża lista typów komunikatów i pytanie:

```
Exclude which messages from the log 'WM_SIZE;WM_TIMER'
```

Jeśli nie zależy Ci na informacjach o stanie systemu Wine, możesz pozostawić wartość domyślną. Poszczególne klasy komunikatów o błędach mogą być osobno włączane i wyłączone, mogą również być przekierowywane z wiersza poleceń.

Na koniec skrypt `Configure` wyświetla zawartość globalnego pliku konfiguracyjnego utworzonego na podstawie dostarczonych informacji i daje możliwość edytowania go za pomocą domyślnego edytora tekstu:

```
Do you want to edit it using vi (Y/N) [N]?
```

Można oczywiście później edytować utworzony plik dowolnym edytorem tekstów, dla tego w tym miejscu można zrezygnować z tej możliwości.

Automatyczna konfiguracja

Po utworzeniu pliku `wine.conf` skrypt `Configure` rozpoczyna modyfikowanie kodu źródłowego za pomocą programu `xmkmf`. Jest to program, który służy do generowania plików `makefile` dla X Window – tworzy on plik `Makefile` na podstawie informacji zapisanych w pliku `Imakefile`, biorąc pod uwagę szczegóły konfiguracji konkretnego systemu X Window.

Tworzenie pliku wykonywalnego

Aby utworzyć plik wykonywalny programu Wine, wystarczy wydać polecenie

```
make
```

Najtrudniejsza część konfiguracji Wine jest już za tobą. Mimo tego to właśnie komplikacja zabiera najwięcej czasu – w systemie opartym na procesorze Pentium z zegarem 90 MHz trwa ona około ośmiu minut. Aby program mógł zostać poprawnie skonsolidowany, potrzebne będą biblioteki `-lxext`, więc zainstaluj je najpierw z dysku CD-ROM.

Używanie programu Wine

Używanie Wine sprowadza się do wydania polecenia

```
wine nazwa_programu
```

Można również konfigurować ten program za pomocą rozlicznych opcji podawanych w wierszu poleceń, pozwalających na przykład na uruchomienie debugera umożliwiającego wyszukiwanie usterek tkwiących w tym programie.

Parametry konfiguracyjne

Globalny plik konfiguracyjny programu Wine o nazwie `wine.conf` można zwykle znaleźć w katalogu `/usr/local/etc`. Zapisane w nim parametry są w większości oparte na informacjach podanych podczas wcześniejszej konfiguracji i są zorganizowane w sposób przypominający format plików `.ini` używanych w systemie Windows. Poniżej przedstawiamy przykładową zawartość tego pliku, wraz z krótkimi komentarzami.

Poniższe wpisy określają, do których katalogów systemu Linux prowadzić będą DOS-owe litery dysków:

```
[drives]
A=/a
C=/c
```

Następne parametry określają położenie plików systemu Windows i Wine:

```
[wine]
Windows=:\\windows
System=c:\\windows\\system
Temp=c:\\temp
Path=c:\\windows;c:\\windows\\system
SystemResources=/users/wine/wine950122/sysres.dll
```

Kolejna sekcja dotyczy odwzorowania czcionek MS-Windows i czcionek systemu X (gwiazdka jest używana jako symbol wieloznaczny w nazwach czcionek systemu X).

```
[fonts]
system=-helvetica
mssansserif=-helvetica
msserif=-times
fixedsys=-fixed
arial=-helvetica
helv=-helvetica
roman=-times
default=-*
```

Fragmenty przedstawione poniżej definiują odwzorowania odpowiednich portów szeregowych i równoległych systemu Linux na ich DOS-owe odpowiedniki.

```
[serialports]
Com1=/dev/cua0
Com2=/dev/cua1

[parallelports]
Lpt1=/dev/lp0
```

Poniższe parametry określają klasy rejestrów komunikatów diagnostycznych i miejsce, w którym mają one być zapisywane.

```
[spy]
File=CON
Exclude=WM_SIZE;WM_TIMER
```

Opcje dostępne z wiersza poleceń

Wiersz poleceń przy uruchamianiu programu Wine ma następujący format:

```
wine opcje_wine program opcje_programu
```

na przykład:

```
bash# /usr/wine/wine -debugmsg +all /c/windows/winmine.exe
```

Dostępne opcje zostały zebrane w tabeli 59.1.

Tabela 59.1. Opcje programu Wine dostępne z wiersza poleceń

Opcja	Znaczenie
-depth n	Pozwala zmienić głębię koloru, dzięki czemu Wine może używać innej niż domyślnej liczby kolorów. Osiem planów bitowych daje 256 kolorów, co wystarcza do większości zastosowań.
-desktop geom	Uruchamia aplikację MS-Windows na pulpicie o podanym rozmiarze, na przykład podanie liczb 850x620 spowodowałoby otwarcie okna o rozmiarach 850 na 620 punktów. Uruchomienie pulpitu powoduje również wyeliminowanie modalnego zachowania się okienek programów systemu Windows.
-display nazwa	Pozwala wyświetlić okno na innym niż domyślny widoku, dzięki czemu można uruchamiać aplikacje MS-Windows na innym terminalu X połączonym do sieci.
-iconic	Uruchamia aplikację zminimalizowaną do postaci ikony. Opcja ta ma taką samą funkcję, jak opcja „Uruchom zminimalizowane” Menedżera Programów systemu MS-Windows.
-debug	Załącza debugger przed uruchomieniem właściwej aplikacji.
-name nazwa	Ustala nazwę aplikacji. Dzięki temu rozwiązaniu można określić inną niż domyślną (<code>wine</code>) nazwę aplikacji widzianą w systemie X Window.
-privatemap	Powoduje użycie prywatnej palety kolorów. Ta opcja ma zastosowanie szczególnie w przypadku aplikacji wykorzystujących dużą liczbę kolorów. Jej użycie może spowodować nieprawidłowe wyświetlanie kolorów w innych oknach w czasie, gdy okno programu Wine jest aktywne.
-synchronous	Załącza tryb wyświetlania synchronicznego. Ta opcja może poważnie zwolnić działanie aplikacji, ponieważ powoduje, że system X Window będzie przed wysłaniem każdego polecenia czekał na zakończenie wykonywania poprzedniego. Aplikacje systemu X mogą przesyłać polecenia do serwera, który może – ale nie musi – działać w tym samym systemie. W niektórych sytuacjach załączenie synchronizacji okazuje się niezbędne, aby zapobiec optymalizacji operacji graficznych przez serwer X.
-backingstore	Jest to opcja optymalizacyjna, umożliwiająca serwerowi X obsługę zdarzeń <code>expose</code> bez przerwania działania programu klienta.
-spy plik	Załącza rejestrowanie informacji diagnostycznych do pliku o podanej nazwie. Podobny efekt można uzyskać używając mechanizmów przekierowania danych wyjściowych.
-debugmsg nazwa	Załącza lub wyłącza informacje diagnostyczne poszczególnych typów. Aby otrzymać listę obsługiwanych klas informacji diagnostycznych, wydaj polecenie: <code>wine -debugmsg help help</code>

Debugger programu Wine

Program Wine jest wyposażony w wewnętrzny debugger, umożliwiający wyszukiwanie przyczyn problemów wynikających z tkwiących w nim usterek. Jeśli program systemu MS-Windows zakończy działanie ze względu na błąd, w oknie terminalu, z którego uruchomiony został program Wine, uruchamiany jest debugger. Jeśli nie jesteś zainteresowany wyszukiwaniem przyczyny problemu, możesz zakończyć jego działanie wydając polecenie `quit` i przejść do następnego podrozdziału.

Debugger programu Wine jest podobny do programu `gdb`. Umożliwia zakładanie pułapek, sprawdzanie i modyfikowanie wartości rejestrów oraz poszczególnych komórek pamięci. Mimo tego jego funkcjonalność jest dość ograniczona – tabela 59.2 zawiera wszystkie dostępne polecenia.

Tabela 59.2. Polecenia wewnętrznego debugera programu Wine

Polecenie	Funkcja
<code>break</code>	Ustawia pułapkę pod wskazanym adresem, który może również być reprezentowany przez wartość symboliczną. Po wykonaniu instrukcji znajdującej się pod tym adresem, wykonanie programu Wine zostanie wstrzymane. Przykładowo, polecenie <code>break * GDI_Ordinal_24</code> spowoduje zatrzymanie programu po wejściu do procedury systemu Windows rysującej elipsę, dostępnej pod wewnętrzna nazwą <code>GDI.24</code> .
<code>bt</code>	Wyświetlenie drogi, która prowadziła do aktualnego miejsca wykonania programu (ang. <i>backtrace</i>). Wyświetlane adresy są adresami powrotu, a nie wywołań podprogramów.
<code>cont</code>	Powoduje kontynuowanie wykonania programu po osiągnięciu pułapki lub wystąpieniu błędu.
<code>define</code>	Podstawia wartość pod nazwę symboliczną, na przykład <code>define mojaprocedure 0x000001c6</code>
<code>disable</code>	Powoduje wyłączenie określonej pułapki. Pułapki ustawiane za pomocą polecenia <code>break</code> są numerowane. Aby któryś z nich wyłączyć, należy określić jej numer za pomocą polecenia <code>info</code> , a następnie wykorzystać go podając jako parametr polecenia <code>disable</code> , na przykład aby wyłączyć pułapkę o numerze 1 należy wydać polecenie <code>disable 1</code>
<code>enable</code>	Umożliwia załączenie pułapki o określonym numerze – polecenie to ma działanie odwrotne do polecenia <code>disable</code> . Aby załączyć wyłączoną poprzednio pułapkę o numerze 1, należy wydać polecenie <code>enable 1</code> .
<code>help</code>	Wyświetla tekst pomocy dotyczącej dostępnych polecień.
<code>info</code>	Wyświetla informacje o następujących parametrach: <code>reg</code> – dane o wartościach przechowywanych w rejestrach; <code>stack</code> – zawartość stosu; <code>break</code> – dane o poszczególnych pułapkach; <code>segments</code> – informacje o rozmieszczeniu w pamięci używanych segmentów.
<code>mode</code>	Pozwala przełączyć się pomiędzy trybem 16 i 32-bitowym.

cd. na następnej stronie

Tabela 59.2. cd. Polecenia wewnętrzne debugera programu Wine

Polecenie	Funkcja
print	Wyświetla wartość podanego wyrażenia.
quit	Umożliwia zakończenie działania debugera i uruchomionego programu MS-Windows.
set	Pozwala zapisywać dowolne wartości do poszczególnych rejestrów i komórek pamięci.
symbolfile	Ładuje plik zawierający wartości symboliczne. Odpowiedni plik o nazwie <code>wine.sym</code> jest rozprowadzany wraz z programem Wine.
x	Wyświetla zawartość pamięci w kilku różnych formatach. Składnia tego polecenia jest następująca: <code>x / format adres</code> . Parametr <code>format</code> może przyjmować następujące wartości: <code>x</code> 32-bitowa liczba całkowita szesnastkowa <code>d</code> 32-bitowa liczba całkowita dziesiętna <code>w</code> 16-bitowa liczba szesnastkowa <code>b</code> bajt <code>c</code> pojedynczy znak <code>s</code> napis ASCII zakończony zerem <code>I</code> instrukcja procesora i386 Przed formatem można również podać liczbę, określającą ilość powtórzeń, na przykład aby wyświetlić 10 kolejnych instrukcji rozpoczynając od danego adresu, należy wydać polecenie: <code>x / 10 I 0x00001cd</code>

Aby móc wykorzystać możliwości tkwiące w tym debuggerze, trzeba rozumieć ogólne zasady działania debugerów programów asemblerowych dla platformy i386. Jeśli naprawdę chcesz szukać usterek w programie Wine, niezbędny będzie kod asemblerowy generowany przez kompilator GCC.

Zasady działania programu Wine

Program Wine składa się z części ładującej programy systemu MS-Windows do pamięci i z biblioteki funkcji systemu MS-Windows.

Ładowanie programów do pamięci

Program Wine zaczyna swoje działanie od załadowania obrazu pliku wykonywalnego systemu MS-Windows do pamięci. Dodatkowo ładowane są wszystkie wymagane biblioteki DLL i zasoby. Pliki wykonywalne systemu MS-Windows mają inną strukturę, niż uruchamiane w systemie DOS, nazywaną NE (ang. *New Executable*). Biblioteki DLL i pliki czcionek również używają struktury NE, co nieco upraszcza zadanie postawione przed programem Wine.

Do pamięci muszą zostać załadowane poszczególne segmenty obrazu pliku NE, a następnie należy rozwiązać wszystkie odniesienia do innych bibliotek DLL i wywołań funkcji systemu Windows. Odwołania do funkcji nie zdefiniowanych wewnątrz obrazu składają się z nazwy modułu i numeru porządkowego funkcji, na przykład wywołanie funkcji rysującej elipsę ma postać GDI.24. W module GDI zapisane są funkcje graficzne systemu Windows, natomiast numer 24 jest właśnie numerem porządkowym funkcji rysującej elipę.

Po załadowaniu do pamięci obrazu pliku wykonywalnego, Wine po prostu przechodzi do wykonania zdefiniowanej w nim funkcji `WinMain()`. Ponieważ zarówno Linux, jak i MS-Windows działają w oparciu o instrukcje procesorów rodziny i386, nie jest konieczna emulacja rozkazów. W momencie, gdy wystąpi wywołanie funkcji systemu Windows, jest ono przechwytywane przez program Wine i przekazywane do odpowiedniej biblioteki.

Biblioteka Wine

Program Wine zamienia wywołania standardowych funkcji MS-Windows na odpowiadające im wywołania funkcji systemu X lub UNIX. Dla przykładu, wywołanie funkcji rysującej elipsę w systemie MS-Windows ma następującą postać:

```
Ellipse (hdc, xLeft, yTop, xRight, yBottom);
```

Parametry `xLeft`, `yTop`, `xRight`, i `yBottom` określają rozmiar prostokąta otaczającego elipsę. Natomiast w systemie X odpowiadające polecenie ma postać:

```
XDrawArc (display, d, gc, x, y, width, height, angle1, angle2);
```

Jak widać, podstawienie odpowiednich współrzędnych wymaga przeprowadzenia pewnych obliczeń. Pozostałe parametry funkcji `XDrawArc` można wyznaczyć bezpośrednio. Parametr `d` określa obszar, na którym można rysować – zwykle jest to po prostu uchwyt okienka. W systemie Windows jest on jednym z elementów struktury przekazywanej przez parametr `hdc`. Parametr `gc` określa kontekst graficzny, który funkcjonalnie odpowiada parametrowi `hdc` w systemie Windows. Ponieważ w systemie X możliwe jest wyświetlanie okien na innym terminalu podłączonym do sieci, parametr `display` określa widok, którego należy użyć. Wartość tego parametru pozostaje stała w ciągu całej sesji programu Wine. Ostatnim elementem, na który program Wine musi zwrócić uwagę, jest fakt, że funkcja `Ellipse` może również zostać wywołana w celu narysowania wypełnionej elipsy – program Wine sprawdza to w strukturze `hdc` i ewentualnie zamiast `XDrawArc` wywołuje funkcję `XFillArc`.

W systemie MS-Windows dostępnych jest prawie 300 podstawowych funkcji graficznych, dla których muszą zostać wykonane podobne translacje. Choć może wydawać się to dość pracochłonne, konwersje funkcji graficznych okazują się być jednym z prostszych zadań, które trzeba podejmować podczas emulacji systemu MS-Windows.

Gdzie kończy się Wine, a zaczyna MS-Windows?

Ponieważ do poprawnego działania program Wine potrzebuje pewnych elementów systemu MS-Windows, nie jest łatwo od razu powiedzieć, w którym miejscu kończy się Wine, a zaczyna MS-Windows. Wine w obecnej wersji zapewnia poprawną obsługę wywołań funkcji wchodzących w skład następujących modułów systemu MS-Windows:

commdlg	Common Windows Dialog, najpopularniejsze okna dialogowe;
gdi	Graphics Device Interface, moduł z funkcjami graficznymi;
kernel	interfejs jądra systemu operacyjnego;
mmsystem	interfejs podsystemu obsługi multimedialnych;
mouse	funkcje obsługi myszy;
shell	biblioteka funkcji powłoki systemu Windows 3.1;
sound	podsystem obsługi dźwięku;
toolhelp	funkcje ułatwiające wyszukiwanie usterek;
user	funkcje biblioteki Microsoft Windows User Interface;
win87em	moduł obsługi i emulacji koprocesora;
winsock	moduł Windows Socket (obsługa TCP/IP).

Aby używać funkcji, które nie są zaimplementowane w samym programie Wine, niezbędny jest dostęp do poszczególnych części systemu MS-Windows. Jednym z przykładów jest biblioteka `OLECLL`, która obsługuje klientów OLE. Autorzy programu Wine znacznie ograniczyli liczbę wymaganych plików. Projekt zakłada całkowite uniezależnienie się od plików systemu MS-Windows, włączając w to programy użytkowe i struktury plików tworzone podczas instalacji aplikacji MS-Windows.

Niektóre z najprostszych aplikacji, na przykład `WINMINE.EXE` czy `SOL.EXE` już dziś działają nie wymagając żadnego dodatkowego kodu systemu MS-Windows czy dostępu do katalogów tego systemu. Choć nie jest wymagana żadna konkretna organizacja katalogów, możesz na przykład spróbować uruchomić aplikację `WINMINE.EXE` w następujący sposób:

1. skopiuj pliki `winmine.exe` i `win.ini` do katalogu linuxowego, na przykład `/users/windows`;
2. zmodyfikuj wartość zmiennej `Windows` przechowywaną w pliku `wine.conf` tak, by wskazywała na odpowiedni katalog – w naszym przypadku `/users/windows`;

3. odmontuj partycję DOS-ową;
4. uruchom program Wine.

Ograniczenia programu Wine

Tylko kilka programów systemu MS-Windows działa prawidłowo pod kontrolą programu Wine. Na szczęście można z dość dużą pewnością przewidzieć, czy dany program będzie działać poprawnie bez konieczności uruchamiania go. Niestety, istnieje dość spora grupa aplikacji, które najprawdopodobniej nigdy nie będą działać pod kontrolą emulatora Wine.

Działające oprogramowanie

Obecna wersja programu Wine obsługuje wiele programów użytkowych i gier rozprowadzanych wraz z systemem Windows 3.1. Pomiędzy poszczególnymi wersjami Wine występują jednak dość znaczne różnice – zmiany, które umożliwiają obsługę jednych aplikacji, mogą uniemożliwić poprawne działanie innych. Oto lista programów i gier, które działają zadowalająco dobrze pod kontrolą Wine:

```
✓ calc.exe  
✓ clock.exe  
✓ cruel.exe  
✓ golf.exe  
✓ notepad.exe  
✓ pipe.exe  
✓ pegged.exe  
✓ reversi.exe  
✓ winmine.exe
```

Używanie programu `winestat` do analizy programów systemu Windows

W skład pakietu Wine wchodzi również program użytkowy o nazwie `winestat`. Jest to w zasadzie ten sam program co Wine, tyle że zamiast uruchamiania programu MS-Windows poprzestaje on na załadowaniu go do pamięci i wyświetleniu informacji o powodzeniu tego przedsięwzięcia. Ładowane są również potrzebne biblioteki DLL i w razie ich braku

generowane są odpowiednie komunikaty. `winestat` analizuje też wszystkie odwołania do funkcji systemowych i zapisanych w bibliotekach DLL, weryfikując ich istnienie. Dla przykładu, uruchomienie programu `winestat` z aplikacją Paintbrush systemu MS-Windows daje następujące rezultaty:

```
KERNEL.1 not implemented
KERNEL.54 not implemented
KERNEL.113 not implemented
KERNEL.114 not implemented
KERNEL.121 not implemented
KERNEL.154 not implemented
KERNEL.178 not implemented
KERNEL.207 not implemented
KERNEL: 52 of 66 (86.7 %)
USER: 150 of 150 (100.0 %)
GDI.151 not implemented
GDI.307 not implemented
GDI.366 not implemented
GDI.439 not implemented
GDI: 80 of 84 (95.2 %)
SHELL: 9 of 9 (100.0 %)
KEYBOARD: 2 of 2 (100.0 %)
TOTAL: 293 of 305 winapi functions implemented (96.1 %)
```

Program `winestat` określa poszczególne niezaimplementowane funkcje za pomocą modułu, w którym są one zapisane, oraz numeru porządkowego. Jeśli wolałbyś znać nazwy tych funkcji, możesz zatrzeć do pliku o nazwie odpowiadającej nazwie modułu (z rozszerzeniem `.spec`) zapisanego w podkatalogu `if1632` katalogu z kodem źródłowym programu `wine`. Oto fragment pliku

```
kernel.spec:
#1 FATALEXIT
#2 EXITKERNEL
3 pascal GetVersion() GetVersion()
...
...
...
#54 pascal16 GETINSTANCEDATA
```

Wiersze rozpoczynające się od symbolu `#` są traktowane jako komentarze, co oznacza, że poszczególne funkcje nie są zaimplementowane. W powyższym przykładzie, są to między innymi funkcje `FATALEXIT` oraz `GETINSTANCEDATA`. Funkcja `FATALEXIT` używana jest do testowania programów w systemie MS-Windows w nieprawidłowych warunkach i rzadko jest przedmiotem zainteresowania większości użytkowników systemu MS-Windows. Funkcja `GETINSTANCEDATA` kopiuje dane konfiguracyjne z poprzedniego egzemplarza danej aplikacji. Jeśli uruchamiany jest tylko jeden egzemplarz, nie jest ona wykorzystywana.

Na koniec program `winestat` wyświetla procentową informację o liczbie obsługiwanych funkcji, która jest dość dobrą miarą tego, jak dobrze dana aplikacja może działać pod kontrolą programu `wine`. Niestety, jeśli nie jest obsługiwana choćby jedna funkcja niezbędna do zainicjalizowania danej aplikacji, jakkolwiek wynik poniżej 100% będzie niezadowalający.

Aplikacje, które w teście `winestat` uzyskają wynik ponad 95%, mają szansę na poprawne działanie. W przeciwnieństwie do programu DOSemu, Wine nie ma skłonności do pozostawiania Linuxa w stanie niestabilnym. Mimo tego zawsze należy zamykać kończące się błędem sesje programu Wine. Najłatwiej jest w tym celu uruchamiać Wine z osobnym pulpitem, na przykład za pomocą polecenia `wine -desktop 800x600 nazwa_programu`. Zwykłe metody wyłączania procesu systemu Windows za pomocą menedżera okienek powinny działać prawidłowo.



Kiedy zawiodą wszystkie inne metody zakończenia pracy błędnie działającego programu Wine, najlepiej przełączyć się na inną konsolę wirtualną i zamknąć proces Wine za pomocą polecenia `kill`. Dla przykładu, wcisnięcie klawiszy Alt+Control+F2 uaktywnia drugą konsolę wirtualną. Po zalogowaniu się można użyć polecenia `ps -ax | grep wine`, które pozwoli na znalezienie numeru procesu odpowiadającego sesji Wine. Następnie należy wydać polecenie `kill -15 pid`, gdzie `pid` jest wspomnianym wcześniej numerem.

Do sesji X Window można wrócić przełączając się na konsolę, na której jest ona uruchomiona. Jeśli nie wiesz, która to konsola, po prostu przełączaj się kolejno pomiędzy nimi, trzymając klawisze Alt i Control oraz wciskając kolejno klawisze funkcyjne.

Najważniejsze braki programu Wine

Prawdopodobnie najbardziej rzucającym się w oczy niedociągnięciem jest brak możliwości drukowania. Ponieważ jednak jest to proces skomplikowany, opracowanie interfejsu drukarki nie jest wcale proste. Oгромnym zadaniem byłoby umożliwienie obsługi wszystkich typów drukarek działających w systemie MS-Windows. Prawdopodobnie zaimplementowany zostanie tylko sterownik dla drukarek postscriptowych. Istniejące programy użytkowe dla systemu Linux, na przykład `Ghostscript`, potrafią następnie zamienić dokument postscriptowy na nadający się do wydrukowania za pomocą drukarek różnych typów, na przykład laserowych czy atramentowych drukarek HP.

Również 32-bitowy interfejs Windows API (`win32`) nie jest w większości obsługiwany. Obraz pliku wykonywalnego systemu Windows NT i Windows 95 ma format PE (ang. *Portable Executable*, przenośny plik wykonywalny) – `wine` potrafi co prawda ładować pliki zasobów (na przykład czcionki) w tym formacie, ale nie radzi sobie z plikami wykonywalnymi i bibliotekami DLL.

Oprogramowanie, które prawdopodobnie nigdy nie będzie działać

Projekt Wine nie zakłada obsługi wirtualnych sterowników urządzeń (Virtual Device Drivers, VDD). Wiąże się to z faktem, że pliki VDD wykorzystują inny format – LE (ang. *Linear Executable*), który nie jest obsługiwany przez moduł ładowający programu Wine. Ponieważ w plikach VDD często występują bezpośrednie odwołania do sprzętu, ich

współdziałanie z systemem Linux stwarzałyby poważne problemy. Jednym z zastosowań plików VDD w systemie Windows jest implementacja stosów TCP/IP. W programie Wine TCP/IP jest obsługiwane za pomocą biblioteki `winsock`, która z kolei wykorzystuje mechanizmy TCP/IP wbudowane w jądro systemu Linux.

Podsumowanie

W tym rozdziale przedstawiliśmy informacje dotyczące konfiguracji i używania programu Wine, który umożliwia emulowanie systemu MS-Windows w systemie X Window i uruchamianie niektórych aplikacji. Wine nie jest jedynym programem tego typu; istnieje jeszcze kilka innych emulatorów systemu Windows i interfejsów programów wykonywalnych, jak na przykład program WABI. Mimo tego program Wine jest najstarszym z nich (przynajmniej jeśli brać pod uwagę system Linux) i posiada dobre wsparcie ze strony użytkowników.

Używanie programu WAB (Windows Application Binary Interface) omówione jest w rozdziale 23. „Wabi”.

Rozdziały 26. „Programowanie w języku C” i 27. „Programowanie w C++” zawierają podstawowe informacje dotyczące programowania w językach C i C++ w systemie Linux.

W rozdziale 56. „Zarządzanie kodem źródłowym” omawiamy systemy kontroli wersji i narzędzia ułatwiające zarządzanie poszczególnymi wersjami dokumentów.

Rozdział 60.

HylaFAX

Tim Parker

W tym rozdziale:

- υ Instalacja programu HylaFAX
- υ Wysyłanie faksów
- υ Odbieranie faksów

Możliwość wysłania faksem dokumentu, który został napisany w ulubionym edytorze tekstów jest jednym z bardziej użytecznych aspektów pracy z komputerem. Pozwala zaoszczędzić czas, papier i wysiłek konieczny do wydrukowania kopii dokumentu, załatwiania jej do faksu i ręcznego wysłania. Programy faksujące pozwalają zarówno wysyłać faksy z poziomu dowolnego programu umożliwiającego drukowanie, jak i przyjmować nadchodzące faksy.

Choć najwydajniejsze programy obsługi faksu dla komputerów klasy PC przeznaczone są dla systemu Windows, istnieje również kilka programów tego typu przeznaczonych dla Linuxa i systemów UNIX-owych. Komercyjne programy do obsługi faksów można zakupić u wielu dostawców oprogramowania, ale na szczęście kilka z nich jest dostępnych za darmo. Jeśli planujesz regularne użytkowanie systemu linuxowego, możesz rozpatrzyć możliwość zainstalowania jednego z tych programów.

Najczęściej używanym programem faksującym dla systemu Linux jest darmowy program HylaFAX, napisany przez Sama Leffera. Nie jest on jedynie imitacją produktów komercyjnych – jest to w pełni funkcjonalny i konkurencyjny w stosunku do komercyjnych zintegrowany system przyjmowania i wysyłania faksów. Program HylaFAX do obsługi obrazów faksów wykorzystuje system GhostScript – jeśli chcesz dowiedzieć się o nim czegoś więcej, przeczytaj (o ile jeszcze tego nie zrobiłeś) rozdział 24. „Ghostscript i Ghostview”.

Instalacja programu HylaFAX

Program HylaFAX wchodzi w skład większości dystrybucji Linuxa rozpowszechnianych na płytach CD-ROM. Można go również załadować w wielu węzłach FTP i WWW. Możesz też natknąć się na starszy program FlexFAX, na którym został oparty program HylaFAX. Jeśli możesz wybierać, zainstaluj HylaFAX. Do instalacji większości wersji programu HylaFAX niezbędny będzie kompilator języka C++ (na przykład kompilator GCC dostarczany wraz z większością wersji Linuxa), ponieważ program ten jest rozprowadzany tylko w formie kodu źródłowego.



Program HylaFAX jest dostępny w większości węzłów FTP, ale węzłem macierzystym jest ftp.sgi.com. W którymś z katalogów powinny znajdować się informacje dotyczące tego programu. Dokumentacja programu HylaFAX jest dostępna także w węzłach FTP i na stronie WWW pod adresem <http://www.sgi.com>.

Jeśli chcesz obejrzeć dodatkowy opis programu HylaFAX i dokumentację dotyczącą jego rozwoju, zajrzyj na stronę <http://www.vix.com/hylafax>

Skompresowaną lub spakowaną programem tar wersję pliku zawierającego kod źródłowy programu HylaFAX należy rozpakować do jakiegoś utworzonego wcześniej katalogu. Jeśli na przykład posiadasz w katalogu /tmp plik o nazwie `HylaFax-v4.0-tar.gz`, powinieneś skopiować go do katalogu docelowego i rozpakować wyając polecenia:

```
mkdir /usr/fax
cd /usr/fax
cp /tmp/HylaFax-v4.0-tar.gz .
gunzip HylaFax-v4.0-tar.gz
tar xf HylaFax-v4.0-tar
```

Do wykonania tych poleceń niezbędne będą uprawnienia użytkownika `root`. Oczywiście zamiast katalogu /tmp należy podstawić faktyczną ścieżkę dostępu do pliku zawierającego kod źródłowy. Jeśli nie odpowiada Ci nazwa katalogu docelowego (/usr/fax), możesz zmienić również ją.

Kompilacja programu HylaFAX

Po rozpakowaniu wszystkich plików do katalogu docelowego należy uruchomić kompilator języka C++, który wygeneruje plik wykonywalny. Wcześniej warto jednak przejść plik z informacjami pomocniczymi, który zwykle ma nazwę `README` i przechowywany jest w podkatalogu `port/linux` – czasem zawarte są w nim istotne dane. Aby skompilować program HylaFAX, należy wydać następujące polecenia (będąc zalogowany jako użytkownik `root`):

```
configure
make clean
make install
```

Po wydaniu polecień `configure` i `make` trzeba będzie podać jeszcze kilka dodatkowych informacji – omówimy je za chwilę. Po wydaniu polecenia `configure` na ekranie wyświetlany jest szereg komunikatów dotyczących katalogów, których program HylaFAX będzie używał w czasie pracy.

HylaFAX configuration parameters are:

```
Directory for applications:      /usr/local/bin
Directory for lib data files:    /usr/local/lib/fax
Directory for lib executables:   /usr/local/lib/fax
Directory for servers:          /usr/local/etc
Directory for manual pages:     /usr/local/man
Directory for documentation:    /usr/local/doc/HylaFAX
Directory for spooling:         /usr/spool/fax
Type of uucp lock files:       /usr/spool/uucp
Mode for uucp lock files:      0444
Type of PostScript imager:      gs
PostScript imager program:     /usr/local/bin/gs
Default page size:             North American Letter
Default vertical res (lpi):    98
Directory for font metrics:    /usr/local/lib/afm
Location of sendmail program:  /usr/lib/sendmail
Are these ok [yes]?
```

Nie należy modyfikować domyślnych lokalizacji katalogów, ponieważ są one używane również przez wiele innych aplikacji linuxowych. Po wydaniu polecenia `make` należy poczekać kilka minut na zakończenie komplikacji, ponieważ składa się ona z kilku etapów i wymaga przetworzenia wielu plików rozsianych w całym systemie. Po zakończeniu procedury instalacyjnej powinieneś upewnić się, że katalog `/usr/local/bin/fax` istnieje i zawiera odpowiednie pliki – tylko wówczas będzie można używać programu HylaFAX.

Dodawanie modemów

Nastepnym krokiem jest podanie programowi HylaFAX informacji o modemach. Można to zrobić za pomocą skryptu o nazwie `faxaddmodem`, który poprowadzi Cię krok po kroku przez ten proces. HylaFAX potrafi poprawnie współpracować z większością faksmodemów zgodnych ze standardami Class 1 i Class 2, co w praktyce oznacza przeważającą większość dostępnych na rynku modemów umożliwiających obsługę faksu. Po uruchomieniu, skrypt `faxaddmodem` zadaje serię pytań; w większości przypadków odpowiedzi domyślne są prawidłowe (skrypt ten również należy uruchamiać posiadając uprawnienia użytkownika `root`):

```
# faxaddmodem
Verifying your system is set up properly for fax service...

There is no entry for the fax user in the password file.
The fax software needs a name to work properly; add it [yes]?

Added user "fax" to /etc/passwd
Added fax user to "/etc/passwd.sgi".

There does not appear to be an entry for the fax service in
either the yellow pages database or the /etc/services file;
should an entry be added to /etc/services [yes]?
```

```
There is no entry for the fax service in "usr/etc/inetd.conf";
should one be added [yes]?
```

```
Poking inetd so that it rereads the configuration file.
There does not appear too be an entry for the FaxMaster in
either the yellow pages database or the /usr/lib/aliases file;
should an entry be added to /usr/lib/aliases [yes]?
Users to receive fax-related mail [root]?
```

```
Rebuilding /usr/lib/aliases database.
46 aliases, longest 81 bytes, 923 bytes total
```

```
Done verifying system setup.
```

```
Serial port that modem is connected to []? cua1
```

```
Ok, time to set up a configuration file for the modem. The manual
page config(4F) may be useful during this process. Also be aware
that at any time you can safely interrupt this procedure.
```

```
No existing configuration. Let's do it from scratch.
```

```
Phone number of fax modem []? +1.613.838.1234
```

```
This is the phone number associated with the modem being configured.
It is passed as an "identity" to peer fax machines and it may
also appear on tag lines created by the fax server.
The phone number should be a complete international dialing specification
```

```
in the form +&lt;country code&gt;; &lt;area part&gt;.
Any others characters included for readability are automatically
removed if they might cause problems.
```

```
Area code []? 613
Country code [1]?
Long distance dialing prefix [1]?
International dialing prefix [011]?
Tracing during normal server operation [1]?
Tracing during send and receive sessions [11]?
Protection mode for received fax [0600]?
Rings to wait before answering [1]?
Modem speaker volume [off]?
```

```
The server configuration parameters are
```

```
FAXNumber:          +1.613.838.1234
AreaCode:           613
CountryCode:        1
LongDistancePrefix: 1
InternationalPrefix: 011
ServerTracing:     1
SessionTracing:    11
Recv FileMode:     0600
RingsBeforeAnswer: 1
SpeakerVolume:      off
Are these ok [yes]?
```

```
Now we are going to probe the tty port to figure out the type
of modem that is attached. This takes a few seconds, so be patient.
Note that if you do not have the modem cabled to the port, or the
modem is turned off, this may hang (just go and cable up the modem
or turn it on, or whatever).
```

```
Hmm, this looks like a class 1 modem.  
Product code is "1444".  
Modem manufacturer is "USRobotics".  
Modem model is "Courier".  
  
Using prototype configuration file config.usr-courier...  
  
The modem configuration parameters are:  
  
ModemRate: 19200  
ModemFlowControl: xonxoff  
ModemFlowControlCmd: &H2  
ModemSetupDTRCmd: S13=1&D2  
ModemSetupDCDCmd: &C1  
ModemDialCmd: DT%$@  
ModemResultCodesCmd: x4  
  
Are these ok [yes]?  
  
Startup a fax server for this modem [yes]?  
/usr/etc/faxd -m /dev/cua1
```

Jedynie punkty, na które należy zwrócić uwagę, to nazwa portu, do którego podłączony jest modem (użyty w przykładzie port `/dev/cua1` to drugi port szeregowy, ponieważ do pierwszego zwykle dołączona jest mysz), oraz numer telefonu i kod kraju określający linię, która ma być obsługiwana. Zauważ, że program HylaFAX tworzy nowe konto użytkownika o identyfikatorze `fax`, mającego ten sam numer identyfikacyjny (UID) co użytkownik `uucp`, ponieważ modemy są zwykle współużytkowane przez oba te programy. Program HylaFAX umożliwia konfigurację i używanie kilku różnych modemów, jeśli do systemu podłączonych jest kilka linii telefonicznych.

Program HylaFAX używa procesu `faxd.recv`, który umożliwia aplikacjom klienckim łączenie się z serwerem HylaFAX. Program `faxd.recv` jest zwykle uruchamiany przez proces `inetd` w momencie, gdy system jest uruchamiany w trybie wielodostępnym. Program konfiguracyjny prawdopodobnie sam dodał odpowiednie wpisy do plików startowych `inetd`. Jeśli nie, powinieneś dopisać do pliku `inetd` wiersz o następującej treści:

```
fax stream tcp nowait fax /usr/libexec/fax.d/faxd.recv faxd.recv
```

Aby uruchomić program rezydentny HylaFAX, należy wydać polecenie (podstawiając właściwą dla systemu nazwę portu, do którego podłączony jest modem):

```
/usr/etc/faxd -m /dev/cua1
```

Aby uruchamiać ten program automatycznie podczas startu systemu, należy dopisać to polecenie do pliku `/etc/rc.d`.

Do wysyłania i odbierania faksów przez system HylaFAX używany jest ten sam program rezydentny, `faxd`. Podczas odbierania faksu nie ma możliwości przerwania tego procesu – można co najwyżej wyłączyć modem.

Wysyłanie faksu

Aby wysłać faks za pomocą programu HylaFAX, należy uruchomić program `sendfax`. Zwykle za jego pomocą wysyła się dokumenty postscriptowe (lub ghostscriptowe) albo grafiki w formacie TIFF. Niektóre inne formaty plików (na przykład pliki ASCII czy pliki programu `troff`) są przed wysłaniem automatycznie konwertowane przez program `sendfax` do odpowiedniej postaci. W dokumentacji dostarczonej wraz z każdą wersją programu HylaFAX znajdują się dokładne dane co do obsługiwanych formatów plików.

Opcje programu `sendfax`

Jak widać w tabeli 60.1, w wierszu poleceń programu `sendfax` można określić wiele różnych opcji. Poszczególne opcje można również łączyć ze sobą.

Tabela 60.1. Opcje programu `sendfax` dostępne z wiersza poleceń

Opcja programu <code>sendfax</code>	Znaczenie
<code>-c</code>	Komentarz
<code>-d</code>	Określenie numeru telefonu
<code>-f</code>	Nadawca faksu
<code>-h</code>	Nazwa systemu
<code>-i</code>	Identyfikator, który zostanie umieszczony w faksie
<code>-k</code>	Czas, przez jaki należy ponawiać próby połączenia
<code>-l</code>	Tryb niskiej rozdzielczości
<code>-m</code>	Tryb średniej rozdzielczości
<code>-n</code>	Zapobiega wysyłaniu okładki
<code>-p</code>	Załącza numerowanie stron i okładki
<code>-r</code>	Nazwy odnośników
<code>-s</code>	Nazwa rozmiaru papieru (domyślnie 8.5x11)
<code>-t</code>	Liczba prób nawiązania połączenia
<code>-v</code>	Wyświetlanie informacji diagnostycznych na ekranie
<code>-x</code>	Nazwa firmy, do której wysyłany jest faks
<code>-y</code>	Adres odbiorcy
<code>-D</code>	Informowanie odbiorcy pocztą elektroniczną o nadaniu faksu
<code>-R</code>	Informowanie nadawcy pocztą elektroniczną o konieczności ponownego wysłania faksu

Przy wysyłaniu faksu program HylaFAX automatycznie generuje okładkę. Umieszczone są na niej informacje podane w wierszu poleceń przy uruchamianiu programu `sendfax`. Domyślnie używany jest tryb niskiej rozdzielczości (98 linii na cal), chyba że wartość ta zostanie zmieniona przez podanie opcji `-m` w wierszu poleceń (co daje 196 linii na cal – w programie HylaFAX rozdzielcość ta nazywana jest średnią). Program HylaFAX w razie niemożności nawiązania połączenia będzie próbował zadzwonić się do adresata przez 24 godziny – czas ten można zmienić podając inną wartość po opcji `-k`.

Okładki

Jeśli chcesz, by wysyłane okładki miały inną niż domyślna zawartość, powinieneś użyć polecenia `faxcover`. Pozwala ono utworzyć stronę w formacie PostScript, która będzie dołączana na początku każdego wychodzącego faksu.

Program `faxcover` również obsługuje kilka opcji podawanych z wiersza poleceń – zostały one zebrane w tabeli 60.2. Opcje te mogą być łączone ze sobą w dowolnym porządku.

Tabela 60.2. Opcje programu `faxcover` dostępne z wiersza poleceń

Opcja programu <code>faxcover</code>	Znaczenie
<code>-c</code>	Komentarz
<code>-C</code>	Nazwa pliku szablonu
<code>-f</code>	Identyfikator nadawcy
<code>-l</code>	Adres odbiorcy
<code>-n</code>	Numer faksu nadawcy
<code>-p</code>	Liczba stron
<code>-r</code>	Nazwy odnośników
<code>-s</code>	Nazwa rozmiaru papieru
<code>-t</code>	Odbiorca
<code>-v</code>	Numer docelowej linii głosowej
<code>-x</code>	Nazwa firmy odbierającej faks

Odbieranie faksu

Jeśli program HylaFAX będzie odbierał przychodzące faksy, będą one umieszczane w podkatalogu `recvq` katalogu, w którym zainstalowany jest program HylaFAX. Wszystkie nadchodzące faksy są zapisywane w postaci plików TIFF. Dostęp do katalogu z odebranymi faksami może być regulowany przez administratora systemu.

Po odebraniu faksu, HylaFAX przesyła go do użytkownika o identyfikatorze ustalanym na podstawie wartości zmiennej `FaxMaster`. Można również napisać skrypt, który sprawdzi, do którego z użytkowników skierowany jest dany faks i poinformuje go o fakcie jego nadania pocztą elektroniczną lub też wyśle faks prosto do kolejki drukarki.

Do odbierania faksów w systemie HylaFAX używany jest proces o nazwie `faxrcvd`. Program ten jest na tyle „sprytny”, że potrafi sam ocenić, czy nadchodzące połączenie jest połączeniem głosowym, czy też jest to połączenie mające na celu przesłanie faksu. Dzięki takiemu rozwiązaniu możliwe jest używanie tego samego modemu do odbierania faksów i połączeń modemowych pomiędzy systemami, które są normalnie obsługiwane przez `uucp` lub proces `getty`.

Podsumowanie

W tym rozdziale omówiliśmy zagadnienia związane z konfigurowaniem i używaniem systemu HylaFAX, który jest prawdopodobnie najbardziej wydajnym programem obsługi faksu przeznaczonym dla systemów linuxowych, pozostając przy tym aplikacją całkowicie darmową.

Jeśli chcesz dowiedzieć się czegoś więcej o interpreterze Ghostscript, który bywa użyteczny do przeglądania i tworzenia dokumentów postscriptowych, przejdź do rozdziału 24. „Ghostscript i Ghostview”.

Informacje dotyczące konfigurowania modemów przedstawione są w rozdziale 33. „Urządzenia”.

Rozdział 45. „Kopie zapasowe” omawia tworzenie kopii zapasowych, dzięki którym można uniknąć konieczności ponownej konfiguracji programu HylaFAX w razie awarii systemu.

Rozdział 61.

Gry

Ed Trejis i Tim Parker

W tym rozdziale:

- υ Które gry zainstalowałeś?
- υ Gry dla systemu X Window
- υ Gry działające w trybie tekstowym

Na większości płyt CD-ROM z systemem Linux zamieszczane są różnego rodzaju gry, a jeszcze większą ich różnorodność można znaleźć w węzłach FTP i na płytach CD-ROM z oprogramowaniem. Gry można z grubsza podzielić na te działające w systemie X oraz te, które działają w trybie tekstowym. W tym rozdziale przedstawimy gry należące do obu tych rodzajów. Ma on postać spisu wraz z krótkim omówieniem poszczególnych gier tekstowych i działających w systemie X.

Które gry zainstalowałeś?

Gry przedstawione w tym rozdziale pochodzą z kilku różnych pakietów instalacyjnych, dlatego może się zdarzyć, że niektóre z nich nie będą dostępne w Twoim systemie – dla przykładu, graficzne wersje gier `tetris`, `gnuchess` i `xfractint` są zwykle instalowane oddzielnie.

Jeśli któraś z przedstawionych tu gier zainteresuje Cię, możesz zainstalować ją z płyty CD-ROM, jeśli do tej pory tego nie zrobiłeś.

Gry dla systemu X Window

Do uruchomienia przedstawionych poniżej gier niezbędny jest system X Window. Większość gier przeznaczonych dla tego systemu znajduje się w kilku katalogach, w zależności od używanej wersji Linuxa. Zwykle gry można znaleźć w katalogach:

- υ /var/lib/games
- υ /usr/games
- υ /usr/lib/games
- υ /usr/local/games
- υ /usr/share/games

W wielu przypadkach zdarza się, że we wszystkich tych katalogach znajduje się po kilka gier.

Ponieważ system X jest okienkowym systemem graficznym, można się domyślić, że gry przeznaczone dla niego są oparte na grafice. Jest tak w rzeczywistości – prawie wszystkie z wymienionych niżej gier wykorzystują kolorowe grafiki bitmapowe. W wielu z nich można samemu określić paletę kolorów, która ma być wykorzystywana w grze.

Musisz jednak pamiętać o kilku rzeczach.

- υ Gry przygodowe i gry wideo używają specjalnie dla nich zaprojektowanego sprzętu. System X Window jest środowiskiem przeznaczonym nie tylko do gier i dlatego nie może zapewnić grom maksymalnej wydajności. Nawet dzisiejsze szybkie komputery osobiste nie są w stanie dorównać urządzeniom zaprojektowanym specjalnie do gier pod względem szybkości i płynności animacji.
- υ Gry wykorzystują zasoby sprzętowe i system operacyjny w stopniu o wiele większym, niż inne aplikacje. Dla osiągnięcia możliwie największej wydajności, gry tworzone są często przy użyciu różnych sztuczek programowych i sprzętowych. Z tego powodu zdarza się, że niektóre gry nie działają poprawnie w pewnych konfiguracjach lub też mają dziwne skutki uboczne swego działania.
- υ Gry przeznaczone dla systemu X to głównie aplikacje tworzone przez pojedynczych programistów. Zezwolili oni na ich darmowe rozpowszechnianie i wykorzystanie i oczekują na sugestie oraz pomoc w dalszym rozwijaniu. Nie należy jednak porównywać tych gier do standardów komercyjnych, ponieważ nie są one produktami komercyjnymi.
- υ Na płycie CD-ROM zawierającej wersję Slackware systemu Linux znajdują się dwa zestawy gier. Jeden z nich – zapisany w zestawie dysków oznaczonym literą Y – zawiera gry pochodzące z systemu BSD, natomiast w zestawie XAP znajdują się między innymi gry przeznaczone dla systemu X. Można zainstalować oba pakiety, a następnie usunąć te z gier, które nie przypadną użytkownikom do gustu.



Możliwe jest umieszczanie wszystkich gier w katalogu /usr/games, ale gry instalowane przez użytkowników powinny raczej trafić do katalogu /usr/local/games. Katalog /usr/games jest zarezerwowany dla gier rozprowadzanych wraz z systemem.

Poniżej zamieszczamy omówienie gier działających w systemie X, które powinieneś znaleźć w swoim systemie. Pamiętaj, że w związku z różnicami pomiędzy poszczególnymi instalacjami może się okazać, że posiadasz więcej lub też mniej gier, niż tu omawiamy.

Gry dostępne w menu głównym menedżera xdm

Jeśli używasz menedżera okien systemu X Window o nazwie `xdm`, jego menu główne (zwykle dostępne po naciśnięciu prawa klawisza myszy w czasie, gdy kursor jest nad obszarem okna głównego) zawiera podmenu o nazwie `Games`. Z tego menu można następnie wybrać kolejno podmenu `Demo` i `Gadgets`. Jeśli używasz innych menedżerów okienek, na przykład Motif, odpowiednie menu będą miały inną postać. Również same gry dostępne w tym menu zależą od wersji Linuxa. Poniżej przedstawiamy listę niektórych z nich, wraz z krótkimi opisami.

Spider

Jest to odmiana pasjansa. Dostępne są dwie wersje tej gry – Small i Large, różniące się tylko rozmiarem używanych kart i, co się z tym wiąże, wielkością zajmowanego okna.

Jeśli chcesz obejrzeć dokumentację dotyczącą tej gry, wydaj polecenie `man spider`.

Aby uruchomić tę grę, wpisz w oknie konsoli polecenie `spider`.

Celem gry `spider` jest ułożenie wszystkich kart jednokoloru w porządku od najstarszej do najmłodszej, co wymaga logicznego myślenia i planowania. Możliwe jest również układanie w porządku malejącym kart o różnych kolorach. Czasem takie postępowanie wydaje się poprawiać sytuację, w rzeczywistości powoduje znaczne wydłużenie rozgrywki. Dwie lub więcej kolejnych kart tego samego koloru przenoszone są grupowo. Spider jest wyzwaniem – nie próbuj grać w tę grę tylko dla zabicia czasu!

Puzzle

Jest to świetna wersja gry – najczęściej wykorzystywanej podczas dziecięcych przyjęć – której celem jest przedstawianie 15 kwadratowych elementów w siatce 4x4 tak, aby ułożyć zapisane na nich cyfry w odpowiednim porządku.

Jeśli chcesz obejrzeć dokumentację dotyczącą tej gry, wydaj polecenie `man puzzle`.

Aby uruchomić grę, wpisz w oknie konsoli polecenie `Puzzle`.

Wersja dla systemu X jest bardzo przyjemna w obsłudze, ponieważ elementy przesuwają się bezproblemowo, w przeciwieństwie do wersji plastikowej, w której miały one tendencję do zacinania się.

Po kliknięciu na prostokącie po lewej stronie elementy ustawiane są w przypadkowych pozycjach. Po kliknięciu po prawej stronie, gra sama się rozwiąże (spróbuj kliknąć na prostokącie po prawej stronie w sytuacji, gdy liczby są już w odpowiednim porządku).

GNU Chess

Jest to graficzna wersja programu GNU Chess, wykorzystująca program `xboard`.

Ostrzeżenie

Uruchomienie programu GNU Chess pod kontrolą `xboard` powoduje zużycie dużej ilości zasobów systemowych i może spowodować nawet załamanie systemu.

Utworzenie dodatkowego pliku lub partycji wymiany może zredukować czas oczekiwania na odpowiedź programu – nie przejmuj się jednak, nie jest to wina Twojego systemu, tylko programu GNU Chess.

Xtetris

Jeśli nigdy nie wciągnął Cię Tetris, masz teraz jedyną szansę. Jest to przyjemna w obsłudze implementacja tej gry dla systemu X, nie tracząc uroku (w przeciwieństwie do innych wersji) po przeniesieniu jej z gier wideo na komputery domowe.

Jeśli chcesz obejrzeć dokumentację dotyczącą tej gry, wydaj polecenie `man xtetris`.

Aby uruchomić grę, wpisz w oknie konsoli polecenie `xtetris`.

Gra oparta jest na przyjemnym dla oka zestawie kolorów, a animacja jest w miarę płynna. Jeśli jednak przywykłeś do innych wersji Tetrisa, powinieneś wziąć pod uwagę następujące wskazówki.

- v Strzałki w lewo i prawo powodują przesunięcie spadających klocków w odpowiednich kierunkach, natomiast w górę i w dół – obrót zgodnie z kierunkiem ruchu wskazówek zegara i w kierunku przeciwnym. Większość osób preferuje jeden z kierunków obrotu – powinieneś więc poeksperymentować i stwierdzić sam, który z nich bardziej Ci odpowiada.
- v Spacja, jak to zwykle bywa w implementacjach Tetrisa dla komputerów domowych, powoduje zrzucenie klocka na dno studni, zamiast przyspieszać jego ruch.
- v Kolory poszczególnych elementów, choć dość ładne, czasem bywają nieco mylące. Dla przykładu, jeden z klocków w kształcie litery L, który jest w innych wersjach Tetrisa fioletowy, w tej jest żółty, natomiast drugi z nich – dokładnie na odwrót. Jeśli jesteś przyzwyczajony do innych ustawień, może to być pewną niedogodnością.

Jaki jest cel tej gry? Należy tak ustawać poszczególne klocki, by nie pozostawiać między nimi żadnych luk. Po utworzeniu pełnej poziomej linii jest ona automatycznie usuwana. Gra kończy się po zapełnieniu całej studni klockami (niestety, gdy gra wymyka się spod kontroli, nie pojawia się Kozak niszczący klocki swą wielką stopą).

Xlander

Jest to nowa wersja starej gry znanej z automatów, Lunar Lander. Na ekranie przedstawiony jest widok z okna lądownika księżycowego. Należy miękko wylądować w wyznaczonym obszarze używając silnika głównego i kierunkowych. W przypadku niepowodzenia, po prostu się rozbijesz...

Jeśli chcesz obejrzeć dokumentację dotyczącą tej gry, wydaj polecenie `man xlander`.

Aby uruchomić grę, wpisz w oknie konsoli polecenie `xlander`.

Czasem zdarzają się problemy polegające na tym, że gra nieprawidłowo reaguje na wciskane klawisze – wówczas powierzchnia księżyca zbliża się bardzo szybko i nieuchronnie.

Xmahjongg

Jest to implementacja starej, chińskiej gry. Posiada bardzo atrakcyjną oprawę graficzną – ideogramy na poszczególnych klockach są wykonane bardzo estetycznie. Komputer oczywiście buduje zamek za Ciebie, co znacznie przyspiesza rozgrywkę.

Program `Xmahjongg` nie posiada własnej strony `man`.

Xvier

Xvier to gra zbliżona do popularnej gry w kółko i krzyżyk. Ruchy wykonuje się na zmianę na szachownicy o rozmiarach 5x5. Celem gry jest utworzenie rzędu złożonego z czterech elementów ułożonych poziomo, pionowo lub po przekątnej. Xvier różni się od gry w kółko i krzyżyk tym, że można wybrać tylko kolumnę, w której ma zostać umieszczony symbol – zostanie on automatycznie umieszczony na najniższej dostępnej pozycji.

Jeśli chcesz obejrzeć dokumentację dotyczącą tej gry, wydaj polecenie `man xvier`.

Aby uruchomić grę, wpisz w oknie konsoli polecenie `xvier`.

Możliwa jest również zmiana poziomu „inteligencji” komputera przez wciśnięcie w czasie gry klawiszy od 0 do 9. Należy jednak wziąć pod uwagę fakt, że po wybraniu jednego z wyższych poziomów komputer zastanawia się przez długi czas. Z tego powodu nie warto zwiększać poziomu trudności o więcej niż jeden. Domyślnie uruchomiony jest poziom zerowy i prawdopodobnie nie będziesz chciał wychodzić poza trzeci.

Ico

Po uruchomieniu gry `ico` na ekranie wyświetlany jest wielościan. W zależności od wybranej opcji, zajmuje on obszar własnego okna lub też jest wyświetlany w obszarze okna głównego.

Jeśli chcesz obejrzeć stronę `man` dotyczącą tej gry, wydaj polecenie `man ico`. Grę można uruchomić wpisując w wierszu poleceń systemu X Window polecenie `ico`. W zasadzie należy uruchamiać ją z wiersza poleceń ze względu na dostępne opcje. Jeśli zostanie uruchomiona z menu menedżera `xdm`, wielościan zostanie wyświetlony w małym, niepozornym okienku.

Jedną z interesujących opcji jest opcja `-color`, pozwalająca określić kolory ścianek wielościanu. Podając więcej niż jeden kolor, można otrzymać wielościan o ściankach różnych kolorów.

Po opcji `-color` należy podać dane dotyczące samych kolorów w następującym formacie: `rgb:<intens_czerwieni>/<intens_zieleni>/<intens_błękitu>`. Poszczególne intensywności określane są w notacji szesnastkowej, `000` to wartość najniższa, natomiast wartością najwyższą jest `fff`. Oto przykładowe polecenie:

```
ico -color rgb:000/888/fff rgb:e000/400/b80 rgb:123/789/def
```

Program `ico` dość intencyjnie korzysta z zasobów systemu i może zwolnić jego działanie.

Maze

Program `maze` rysuje labirynt, po czym go rozwiązuje. Nie ma sposobu na to, by rozwiązać go samodzielnie, dlatego program ten jest raczej programem demonstracyjnym niż grą. W szybkich systemach labirynt rozwiązywany jest zbyt szybko, by to zaobserwować!

Xeyes

Nie jest to gra w pełnym tego słowa znaczeniu, ale mimo to jest to bardzo miły programik. Po jego uruchomieniu na ekranie wyświetlana jest para oczu śledzących kurSOR myszy. Uruchomienie czterech czy pięciu kopii tego programu nadaje systemowi dość surrealistyczny wygląd.

Jeśli chcesz obejrzeć dokumentację dotyczącą tego programu, wydaj polecenie `man xeyes`.

Aby uruchomić grę, wpisz w oknie konsoli polecenie `xeyes`.

Xgas

Jest to program demonstrujący zachowanie się czystego gazu – aby można było popatrzeć na niego z przyjemnością, nie jest jednak konieczne posiadanie doktoratu z termodynamiki czy mechaniki statystycznej. Okno programu podzielone jest na dwie części, pomiędzy którymi jest jeden niewielki otwór. W obu częściach można ustalić różne temperatury.

Następnie należy umieścić cursor w którejś z części i wcisnąć lewy klawisz myszy – każde kliknięcie uwalnia nową molekulkę gazu poruszającą się w przypadkowym kierunku.

Jeśli chcesz obejrzeć dokumentację dotyczącą tej gry, wydaj polecenie `man xgas`.

Aby uruchomić grę, wpisz w oknie konsoli polecenie `xgas`.

Xlogo

Jest to niewielki program wyświetlający oficjalne logo systemu X Window.

Xroach

Jest to coś pośredniego pomiędzy grą a programem demonstracyjnym. Nie uruchamiaj tej gry, jeśli brzydzisz się insektami!

Jeśli mieszkales kiedykolwiek w budynku dotkniętym plagą karaluchów, ten program przywoła miłe (lub niezbyt miłe) wspomnienia. Za każdym razem, gdy uruchomisz egzemplarz programu `xroach`, nowe stado karaluchów zacznie biegać po ekranie, szukając okna, pod które można by się schować. W końcu chowają się wszystkie – przynajmniej do czasu, aż zamkniesz lub przesuniesz któryś z okien.

Jeśli chcesz obejrzeć dokumentację dotyczącą tej gry, wydaj polecenie `man xroach`.

Aby uruchomić grę, wpisz w oknie konsoli polecenie `xroach`.

Jeśli uruchamiasz program `xroach` z wiersza poleceń, możesz dodać opcję `-squish`, która umożliwia rozgniatanie insektów przez klikanie na nich. Niestety, są one dość szybkie, co nie ułatwia zadania. Można również określić kolor wnętrzności zabitych karaluchów.

Xhextris

Jest to wersja Tetrisa, w której poszczególne klocki składają się z sześciokątnych elementów. Aby uruchomić tę grę, wydaj w wierszu poleceń systemu X Window polecenie `xhextris`. Dla tego programu nie jest dostępna strona `man`.

Xbombs

Jest to wersja powszechnie znanej gry w sapera. Plansza podzielona jest na szereg pól, z których kilka zawiera miny. Twoim zadaniem jest oflagowanie wszystkich pól zawierających miny.

Grę tę uruchamia się wpisując w wierszu polecenie `xbombs`. Nie jest dostępna strona `man` dotycząca tej gry.

Po uruchomieniu gry wyświetlane jest pole, podzielone na niewielkie prostokąty, oraz okno punktacji.

Po kliknięciu na któryś z prostokątów jest on odkrywany. Gra kończy się w momencie, gdy trafisz na minę.

Bardziej prawdopodobne jest jednak, że trafisz na pole nie zawierające miny. Wówczas zostanie w nim wyświetlona liczba określająca, ile min znajduje się w polach sąsiadujących z odkrytym z którejś ze stron lub po przekątnej. Jeśli na polu nie ma żadnej cyfry, oznacza to, że żadne z pól sąsiednich nie zawiera miny. Takie pola zwykle odkrywają się w większych grupach. Przykładowo, jeśli na odkrytym polu pojawi się liczba 1, oznacza to, że na którymś z sąsiednich pól znajduje się mina. Jeśli odkryłeś już położenie miny sąsiadującej z polem zawierającym jedynkę, możesz bezpiecznie odkryć wszystkie pozostałe pola sąsiadujące z nim, ponieważ masz pewność, że żadne z nich nie zawiera już miny. W ten sposób można wydedukować położenie wszystkich min. Jeśli uda Ci się odkryć pole nie zawierające żadnej liczby (czyli nie sąsiadujące z żadną miną), gra automatycznie odkryje wszystkie takie pola sąsiadujące z odkrytym.

Kiedy wydaje Ci się, że odkryłeś, w którym miejscu umieszczona jest mina, możesz oznaczyć podejrzane pole flagą klikając na nim prawym przyciskiem myszy (jeśli przez przypadek klikniesz na tym polu lewym klawiszem myszy i faktycznie znajduje się na nim mina, gra jest niestety skończona). Prawy przycisk myszy włącza i wyłącza oznaczenie miny. Zauważ, że gra nie daje żadnych wskazówek co do tego, czy flaga została umieszczona prawidłowo.

Wkrótce zorientujesz się, że niektóre rozmieszczenia liczb dają pewną informację o położeniu miny; w innych przypadkach trzeba będzie nieco pogłówkować lub zdać się na lut szczęścia.

Oczywiście, czasem zdarza się, że przez pomyłkę wyleciszesz w powietrze. Aby zrestartować grę, kliknij dowolnym klawiszem myszy w obszarze okna punktacji. Jeśli uda Ci się odgadnąć położenie wszystkich min, czas, w jakim to zrobiłeś, zostanie zanotowany.

Xpaint

Jest to program służący do tworzenia prostych, kolorowych rysunków. Można go uruchomić wpisując w wierszu poleceń systemu X Window polecenie `xpaint`. Spowoduje to wyświetlenie menu zawierającego różnego typu narzędzia. Z menu File wybierz utworzenie nowego obrazka (ang. *new canvas*). W menu Tool zebrane są różne narzędzia, takie jak na przykład pędzle, ołówki, farba w aerosolu itp. Poniżej obrazka wyświetlana jest paleta dostępnych kolorów.

Jeśli chcesz obejrzeć stronę `man` dotyczącą tego programu, wydaj polecenie `man xpaint`.

Xfractint

Dzięki programowi Xfractint można w łatwy i bezbolesny sposób rozpocząć zabawę z fraktalami. Jeśli nie wiesz dokładnie, co to jest fraktal, powinieneś pobawić się tym programem. Prawie na pewno widziałeś już kiedyś fraktale.

Jeśli chcesz obejrzeć dokumentację dotyczącą tej gry, wydaj polecenie `man xfractint`.

Aby uruchomić grę, wpisz w oknie konsoli polecenie `xfractint`.

Program `xfractint` ma duże możliwości konfiguracyjne. Bez zagłębiania się w szczegóły matematyczne można z łatwością generować fraktale różnego typu.

Po uruchomieniu tego programu wyświetlane są dwa okna. Pierwsze z nich zawiera obraz fraktala (początkowo okno to jest puste), natomiast drugie pozwala wprowadzać polecenia. Można na przykład przejść do wybierania typu fraktala (ang. *Type*) i wybrać jeden z wielu rodzajów fraktali, który chciałbyś obejrzeć. Możesz również rozpoczęć generowanie obrazka wybierając pozycję *Select video mode*. Domyslnie rysowany jest fraktal Mandelbrota.

Po zakończeniu generacji obrazka (co może chwilę potrwać) można przejść do okna poleceń, wcisnąć klawisz `t`, a następnie wybrać z długiej listy inny typ fraktala. Na tym etapie nie warto zmieniać domyślnych wartości podpowiadanych przez program. Obejrzenie fraktali wszystkich typów i tak zajęłoby sporo czasu.

Aby zakończyć działanie programu `xfractint`, wciśnij dwukrotnie klawisz `Escape` w oknie poleceń.

Gry działające w trybie tekstowym

Gry działające w systemach UNIX-owych często mają za sobą długą historię. Wiele z nich powstało, zanim jeszcze rozpowszechniły się systemy potrafiące wyświetlać kolorową grafikę. Wszystkie te gry, za wyjątkiem gry Sasteroids, są oparte na trybie tekstowym. Oznacza to, że wszystkie elementy graficzne (o ile w ogóle jakieś występują) są wyświetlane za pomocą standardowych znaków, takich jak na przykład `A`, `*`, `|`, `x` itp. Wszystkie dane wprowadzane są z klawiatury (również tu gra Sasteroids jest wyjątkiem).

Zaletą gier pracujących w trybie tekstowym jest to, że do ich uruchomienia nie jest potrzebne środowisko graficzne czy okienkowe, wystarcza w zupełności monochromatyczny terminal. Tekstowa natura niektórych gier (jak na przykład gra w szubienicę) sprawia, że środowisko tekstowe jest wszystkim, czego im potrzeba – wymyślna, kolorowa grafika wcale nie jest niezbędna. Inne gry mogą w tej chwili być tylko ciekawostkami historycznymi: pokazują, jak wiele programista potrafił osiągnąć używając tylko i wyłącznie standardowego zestawu znaków, ale na pewno o wiele lepiej mogłyby być obsłużone w systemie graficznym.



Dwie z bardziej interesujących (i klasycznych) gier tekstowych, Rogue i Hack, nie wchodzą w skład dystrybucji Linuxa. Gry te używają ekranu terminalu do wyświetlania pomieszczeń i korytarzy lochów. Gracz (wraz z psem w grze Hack) porusza się po tychże lochach, odkrywając korytarze, wchodząc do kolejnych pomieszczeń (przy wchodzeniu do ciemnych, nieoświetlonych pokoi należy zachować szczególną ostrożność), zbierając ukryte skarby i magiczne przedmioty oraz walcząc z potworami (i uciekając przed nimi). Po odkryciu całego poziomu można jeździć na poziom niższy, który jest trudniejszy od poprzedniego.

Za każdym razem, gdy uruchamiana jest gra Hack lub Rogue, wygląd lochów jest inny. Każdy z potworów ma inne umiejętności walki, niektóre posiadają również specjalne możliwości. Magiczne przedmioty, takie jak pierścienie, pałeczki, zwoje i napoje mają najprzeróżniejsze właściwości. Niektóre ze znajdowanych przedmiotów, na przykład zbroje, mogą zostać zaczarowane czy usprawnione za pomocą mocy magicznych. Jeśli jednak znajdziesz przedmiot, który został przeklęty, lepiej pozostawić go na miejscu.

Zarówno Rogue jak i Hack mają swoich entuzjastów, ale gra Hack jest nowsza i bardziej rozbudowana, dlatego ma więcej zwolenników. Jeśli natknesz się na któryś z tych gier w Internecie, spróbuj w nią zagrać! Dostępne są również wersje tych gier przeznaczone dla systemu MS-DOS

Tekstowe gry przygodowe

Gry tego typu opierają się na prostej zasadzie: system informuje Cię o sytuacji, na przykład you are in a maze of small twisty passages, all alike (jestes w labiryncie wąskich, pokręconych i podobnych do siebie korytarzy), pozwalając wybrać drogę, którą chcesz się udać, przez wpisanie nazwy kierunku (np. forward – naprzód, east – wschód itp.) czy też podając jakieś proste działania typu take sword (weź miecz). Jeśli lubisz układanie puzzli, tego typu gry będą Ci się podobać. Akcja podąża dokładnie określona droga, a liczba możliwych czynności jest zwykle dość ograniczona.

Poniżej przedstawiamy przykładowy początek tekstowej gry przygodowej Battlestar, która zostanie omówiona w następnym podrozdziale. Polecenia są wpisywane po znaku zachęty, który ma postać >-:.

Version 4.2, fall 1984.

First Adventure game written by His Lordship, the honorable
Admiral D.W. Riggle

This is a luxurious stateroom.
The floor is carpeted with a soft animal fur and the great wooden
furniture is inlaid with strips of platinum and gold. Electronic
equipment built into the walls and ceiling is flashing wildly. The floor
shudders and the sounds of dull explosions rumble through the room. From

```
a window in the wall ahead comes a view of darkest space. There is a  
small adjoining room behind you, and a doorway right.  
>: right  
    These are the executive suites of the battlestar.  
Luxurious staterooms carpeted with crushed velvet and adorned with beaten  
gold open onto this parlor. A wide staircase with ivory banisters leads  
up or down. This parlor leads into a hallway left. The bridal suite is  
right.  
Other rooms are behind you.  
>: up  
    You are at the entrance to the dining hall.  
A wide staircase with ebony banisters leads down here.  
The dining hall is to the ahead.  
>: bye  
Your rating was novice.
```

Battlestar

Aby uruchomić tę grę, należy wydać polecenie `battlestar`. Przykładowa sesja tej gry przedstawiona została w poprzednim podrozdziale. Stronę `man` zawierającą informacje związane z tą grą można obejrzeć po wydaniu polecenia `man battlestar`.

Dungeon

Aby uruchomić tę grę, należy wydać polecenie `dungeon`. W trakcie gry wpisanie polecenia `help` pozwala uzyskać przydatne informacje. Zaczynasz na zewnątrz Lochów i musisz znaleźć do nich wejście. Dla gry Dungeon nie jest dostępna strona `man`.

Paranoia

Aby uruchomić tę grę, należy wydać polecenie `paranoia`. W tej zabawnej grze wcielasz się w tajnego agenta wykonującego bardzo niebezpieczną misję. W przeciwieństwie do większości innych gier tekstowych, Paranoia pozwala wybierać podejmowane akcje z menu. Dzięki temu nie trzeba szukać poleceń, które byłyby rozumiane przez grę. Dla gry Paranoia nie jest dostępna strona `man`.

Wump

Aby uruchomić tę grę, należy wydać polecenie `wump`. W tej grze wcielasz się w myśliwego polującego na potwora o imieniu Wumpus. Na początku jesteś zaopatrzony tylko w kilka wykonanych na zamówienie strzał, spryt i węch. Po rozpoczęciu gry możesz obejrzeć instrukcję zawierającą informacje o jej obsłudze.

Po wydaniu polecenia `man wumpus` można obejrzeć stronę `man` poświęconą tej grze.

Gry słowne

Dwie poniższe gry są wersjami popularnych zabaw polegających na odgadywaniu i tworzeniu wyrazów.

Boggle

Aby uruchomić tę grę, należy wydać polecenie `bog`. Gra ta jest wersją gry Boggle Deluxe firmy Parker Brothers. Do dyspozycji masz litery umieszczone w szachownicy o wymiarach 5x5. W czasie do trzech minut musisz wpisywać słowa składające się z zadanego liter. Domyślnie trzeba używać liter, które łączą się poziomo, pionowo lub po przekątnej, bez powtarzania żadnej z nich. Liczba mnoga i różne formy czasowników liczone są jako różne słowa, na przykład różnymi wyrazami są `use`, `uses`, `used` czy `user`. Takie zasady są zgodne z oficjalnymi stosowanymi w grze Boggle – można jednak je modyfikować.

Na koniec komputer wyświetla listę słów, które sam znalazł. Na pewno nie uda Ci się pobić komputera, ponieważ dopuszcza on wpisywanie tylko istniejących wyrazów. Jak się pewno przekonasz, słownik programu Boggle zawiera kilka niedopatrzeń – może to być nieco denerwujące, ale nie jest szczególnie poważną wadą.

Gra Boggle nie wymaga kolorowego terminalu, ale niewielki rozmiar liter powoduje, że oczy po chwili dość mocno się męczą.

Stronę `man` poświęconą programowi Boggle można obejrzeć po wydaniu polecenia `man boggle`.

Hangman

Aby uruchomić tę grę, należy wydać polecenie `hangman`. Kolorowa grafika nie jest w tym przypadku do niczego potrzebna. Gra jest angielską wersją popularnej również w Polsce gry w szubienicę (czy też wisielca), więc chyba nie trzeba tłumaczyć jej zasad. Na wszelki wypadek dostępna jest jednak strona `man`, którą można obejrzeć po wydaniu polecenia `man hangman`. Hangman dobiera wyrazy w sposób przypadkowy; czasem są one bardzo trudne do odgadnięcia.

Gry karciane

Ze względu na brak grafiki, gry te nie są tak atrakcyjne jak gry słowne.

Canfield

Aby uruchomić tę grę, należy wydać polecenie `canfield`. Gra ta jest odmianą pasjansa. Poświęconą jej stronę `man` można obejrzeć po wydaniu polecenia `man canfield`. Nie posiada ona jednak tak wielkiej zdolności pozerania czasu, jaka charakteryzuje wersje oparte na interfejsie graficznym.

Cribbage

Aby uruchomić tę grę, należy wydać polecenie `cribbage`. Jeśli jesteś fanem gry Cribbage, aplikacja ta przypadnie Ci do gustu. Poświęconą jej stronę `man` można obejrzeć po wydaniu polecenia `man cribbage`.

Go Fish

Aby uruchomić tę grę, należy wydać polecenie `fish`. Twoim przeciwnikiem jest komputer. Poświęconą jej stronę `man` można obejrzeć po wydaniu polecenia `man fish`. Dość uciążliwym aspektem tej gry jest fakt, że informacje o podejmowanych akcjach czasem wyświetlane są razem (przykładowo – jeśli Ty idziesz na ryby, komputer również musi to zrobić, a informacja o tym pojawia się w jednym bloku z poprzednią informacją).

Gry planszowe

Są to tekstowe wersje popularnych gier planszowych. Ich jakość jest dość różna; prawdopodobnie najlepiej dopracowana jest gra `backgammon`.

Backgammon

Aby uruchomić tę grę, należy wydać polecenie `backgammon`. Jeśli natomiast chcesz uruchomić łatwy do opanowania przewodnik, wydaj polecenie `teachgammon`. Choć w grze tej w niczym nie przeszkaź brak grafiki, brak możliwości wykorzystania urządzenia wskazującego, takiego jak na przykład mysz, jest dość irytujący, ponieważ wymusza wpisywanie danych o poszczególnych ruchach z klawiatury, na przykład w postaci `8-12` czy `4-5`. Wpisanie polecenia `?` w wierszu poleceń wyświetlonym podeczas gry umożliwia dostęp do pomocy zawierającej wskazówkę odnośnie wprowadzania danych o poszczególnych ruchach.

Wydanie polecenia `man backgammon` pozwala obejrzeć strony `man` poświęcone programom `backgammon` i `teachgammon`.

Szachy

W skład pakietu `gnuchess` wchodzi kilka gier powiązanych z szachami. Jeśli chcesz zagrać w szachy mając za przeciwnika komputer, wydaj polecenie `gnuchess`. Dostępny jest również program analizujący o nazwie `gnuan`. Gra potrafi drukować pozycje na szachownicy wykorzystując drukarkę postscriptową lub plik.

Dane o poszczególnych ruchach wprowadza się używając notacji standardowej, na przykład `e2-4`.

Pakiet `gnuchess` jest dość rozbudowany, dlatego zapoznawanie się z nim powinieneś rozpocząć od przejrzenia odpowiedniej strony `man`.

Mille Miglia

Aby uruchomić tę grę, należy wydać polecenie `mille`. Jest to linuxowa wersja gry symulującej wyścigi stworzonej przez firmę Parker Brothers. Ponieważ polecenia wymagane do jej obsługi nie są zbyt intuicyjne, warto najpierw zajrzeć na stronę `man`, którą można obejrzeć po wydaniu polecenia `man mille`.

Monopoly

Aby uruchomić tę grę, należy wydać polecenie `monop`. Jest to oparta o interfejs tekstowy wersja gry Monopoly firmy Parker Brothers. Komputer nie jest jednym z graczy – pełni on tylko rolę planszy, notując również informację o poszczególnych własnościach i finansach każdego z uczestników. Można oczywiście grać samemu, ale nieuchronnie prowadzi to do wygranej. Niestety, podczas gry nie jest wyświetlana żadna reprezentacja planszy, co utrudnia panowanie nad sytuacją i powoduje, że gra nie jest zbyt ciekawa. Dostępna jest również opisująca ją strona `man`.

Symulatory

Opisane poniżej gry pozwalają Ci sprawdzić swoje umiejętności pracy w różnych służbach. Są to gry otwarte, to znaczy nie opierają się na zdefiniowanym odgórnie scenariuszu. Wyświetlają zarówno komunikaty tekstowe, jak i elementy semigraficzne, takie jak na przykład ekran radaru.

Air Traffic Control

Aby uruchomić ten symulator stanowiska kontroli ruchu powietrznego, należy wydać polecenie `atc`. Najpierw należy jednak obejrzeć poświęconą mu stronę `man`, wyając polecenie `man atc` – w przeciwnym przypadku będziesz odpowiedzialny za jedną czy kilka katastrof lotniczych! Akcja gry rozgrywa się w czasie rzeczywistym. Większa dawka kofeiny prawdopodobnie pomoże w wypełnieniu misji.

Sail

Aby uruchomić tę grę, należy wydać polecenie `sail`. Wcielasz się w niej w kapitana statku, decydującego, jaki kurs należy przyjąć i jakiej broni użyć. Dostępnych jest ponad 20 scenariuszy, opartych głównie na historycznych bitwach morskich. Warto zajrzeć na stronę `man` dotyczącą tej gry (dostępną po wydaniu polecenia `man sail`), ponieważ niektóre z poleceń są niejasne i mylące.

Trek

Aby uruchomić tę grę, należy wydać polecenie `trek`. Grając w nią można polecieć tam, gdzie jeszcze nikt nie był, polować na Klingonów (lub stać się ich ofiarą) i tak dalej. Jeśli nie chcesz przynieść wstydu Federacji, powinieneś zajrzeć na stronę `man` dostępną po wydaniu polecenia `man trek`.

Gry „video”

Opisane niżej gry korzystają z całego ekranu terminalu, choć wyświetlana grafika składa się tylko ze standardowego zestawu znaków.

Robots

Aby uruchomić tę grę, należy wydać polecenie `robots`. Roboty wyświetlane na ekranie ścigają Cię; możesz bronić się przed nimi tylko powodując ich zderzanie się ze sobą – po takiej kolizji roboty wybuchają. Powstałe w wyniku zderzenia szczątki niszczą wszystkie roboty, które na nie wpadną. Po ekranie można poruszać się za pomocą klawiszy `hjkł`, czyli tak jak w edytorze `vi`; dozwolone są również ruchy po przekątnych, które można uzyskać wciskając klawisze `yubn`. Ty i roboty poruszacie się równocześnie: każdy Twój ruch powoduje ruch wszystkich robotów. Czasem może się jednak okazać, że jedynym wyjściem z sytuacji jest teleportowanie się w inne miejsce. Gra kończy się, gdy któremuś z robotów uda się Ciebie dogonić. Jeśli natomiast Tobie uda się pozbyć wszystkich robotów, przechodzisz do następnej planszy, na której czyha na Ciebie jeszcze liczniejsza ich zgraja. Dokumentacja tej gry jest dostępna na stronie `man` (wyświetlanej po wydaniu polecenia `man robots`).

W niektórych wersjach Linuxa gra ta została zmodyfikowana w ten sposób, że nie jest możliwe wykonanie ruchu powodującego kolizję z robotem (czyli powodującego przegrana), co odbiera całą przyjemność grania.

Snake

Aby uruchomić tę grę, należy wydać polecenie `snake`. Po ekranie możesz poruszać się za pomocą klawiszy `hjkł`, zbierając pieniądze (\$) i unikając węża (składającego się z liter s). Im więcej pieniędzy zbierzesz, tym głodniejszy jest wąż. Pomieszczenie można opuścić wchodząc na symbol # – jest to jedyna droga ucieczki przed wężem. W sytuacjach awaryjnych można również ratować się przeniesieniem na przypadkową pozycję – w tym celu należy wcisnąć klawisz w. Dokumentację tej gry można obejrzeć po wydaniu polecenia `man snake`.

Tetris

Aby uruchomić tę grę, należy wydać polecenie `tetris`. Jak wskazuje nazwa, jest to wersja gry w Tetrisa, wykorzystująca możliwości terminalu znakowego. Choć na pewno nie wygląda tak dobrze jak wersje dla systemu X Window czy innych systemów graficznych, jest bardzo wygodna w obsłudze, co powoduje, że gra jest bardzo przyjemna. Klocki przesuwa się klawiszami „,” oraz „/”, natomiast do ich obracania (w kierunku przeciwnym do ruchu wskazówek zegara!) służy klawisz „.”. Wciśnięcie spacji powoduje zrzucenie klocka na dno studni. Używając opcji programu `tetris` dostępnych z wiersza poleceń, można zmienić przyporządkowanie klawiszy. Szczegółowe informacje możesz znaleźć na stronie `man` wyświetlanej po wydaniu polecenia `man tetris`.

Worm

Aby uruchomić tę grę, należy wydać polecenie `worm`. Sterujesz robakiem, poruszającym się po ekranie i zjadającym pojawiające się tu i ówdzie cyfry. Połknięcie każdej cyfry powoduje odpowiednie wydłużenie robaka. Wpadnięcie na własny ogon lub na ścianę kończy się jego śmiercią. Jak długo potrafisz utrzymać robaka przy życiu? Weź pod uwagę fakt, że porusza się on do przodu nawet wtedy, gdy nie wydajesz żadnych poleceń.

Stronę `man` dotyczącą programu `worm` można obejrzeć po wydaniu polecenia `man worm`.

Gry matematyczne i programy użytkowe

Przedstawione poniżej programy są co prawda małe i interesujące, ale poza tym nie są szczególnie eksytujące.

Arithmetric

Aby uruchomić ten program, należy wydać polecenie `arithmetric`. Gra polega na podawaniu wyników prostych działań matematycznych. Działanie programu można przerwać wciskając klawisze `Control+C`. Strona `man` jest dostępna po wydaniu polecenia `man arithmetric`.

Programy do kodowania tloczonych kart BCD, alfabetu Morse'a i taśmy perforowanej.

Aby uruchomić program zamieniający wpisywany przez Ciebie tekst na postać odpowiadającą tej kodowanej na tloczonych kartach BCD, wydaj polecenie `bcd`. Jeśli chcesz dowiedzieć się, jaka będzie postać tekstu po zakodowaniu go alfabetem Morse'a, wydaj polecenie `morse`, natomiast na postać używaną w taśmach perforowanych możesz zmienić go wyając polecenie `ppt`. Jeśli programy te zostaną uruchomione bez żadnego tekstu w wierszu poleceń (który jest interpretowany jako tekst do przetłumaczenia), wchodzą w tryb interaktywny. Po wcisnięciu klawisza `Enter` wprowadzony wiersz tekstu jest poddawany konwersji (kod klawisza `Enter` również jest kodowany). Strona `man` dotycząca programu `bcd` zawiera także informacje o dwóch pozostałych programach.

Factor

Aby uruchomić ten program, należy wydać polecenie `factor`. Potrafi on rozkładać liczby na czynniki pierwsze. Wydanie polecenia `factor <liczba>` powoduje rozłożenie na czynniki podanej liczby, natomiast program wywołyany bez argumentów działa w trybie interaktywnym. Liczby muszą mieścić się w przedziale od -2147483647 do 2147483648. Oto przykładowy wynik działania programu `factor`:

```
darkstar:/usr/games$ factor
123
123: 3 41
36
36: 2 2 3 3
```

```
1234567  
1234567: 127 9721  
63473882377743928323  
factor: ouch  
darkstar:/usr/games$
```

Primes

Aby uruchomić ten program, należy wydać polecenie `primes`. Jeśli w wierszu poleceń podana zostanie liczba, program `primes` wyświetli wszystkie liczby pierwsze nie większe od podanej; jeśli nie, program najpierw czeka na wprowadzenie liczby określającej zakres. Program jest zadziwiająco szybki! Stronę `man` można obejrzeć wydając polecenie `man primes`.

Gra dla kilku graczy: Hunt

W tej grze musi brać udział kilku uczestników. Wymagane jest podłączenie co najmniej jednego dodatkowego terminala znakowego (na przykład poprzez port szeregowy).

Gra graficzna: Sasteroids

Ta gra do poprawnego działania wymaga karty graficznej zgodnej ze standardem VGA. Aby ją uruchomić, wydaj polecenie `sasteroids`. Gra przejmuje kontrolę nad ekranem, przełączając terminal w tryb graficzny. Jest to wersja znanej z automatów do gier gry Asteroids. Do poruszania statkiem służą klawisze:

- strzałka w lewo** obrót zgodnie z kierunkiem ruchu wskazówek zegara
- strzałka w prawo** obrót przeciwnie do kierunku ruchu wskazówek zegara
- strzałka w górę** przyspieszenie
- strzałka w dół** uruchomienie osłony (statek wyposażony jest tylko w jedną osłonę)
- lewy Control** strzał
- lewy Alt** hiperprzestrzeń

Przyzwyczajenie się do sposobu poruszania statkiem może zajść dłuższą chwilę. Wygląd gry jest zupełnie odmienny od jej pierwowzoru. Nie jest dostępna strona `man`.

Inne gry

Przedstawione poniżej gry mogą początkowo wydawać się trudne, ale mogą dostarczyć wielu godzin uzależniającej rozrywki.

Sokoban

Wyobraź sobie, że znajdujesz się w magazynie mającym postać labiryntu, w którym przechowywane jest mnóstwo beli bawełny. Każda z nich jest na tyle ciężka, że możesz ją poruszyć tylko pchając, nie da się natomiast jej pociągnąć. Nie należy więc wepchnąć beli do kąta, z którego nie uda się jej już wydostać. Kolejne etapy gry są coraz bardziej skomplikowane – aby przejść do następnego etapu, należy przepchnąć wszystkie bele do obszaru załadunku. Kod źródłowy tej gry jest dostępny w węźle sunsite.unc.edu, w pliku `sokoban-src.tar.gz`.

DOOM

Ta wciągająca, choć kontrowersyjna ze względu na okrucieństwo gry, doczekała się również wersji linuxowej. Jest to pełna wersja gry, z wyborną grafiką oraz obsługą dźwięku, nie odbiegającą od wersji dla systemu DOS. Jedynym problemem, na który należy zwrócić uwagę, jest fakt, że paleta kolorów systemu X Window może zostać zmieniona, jeśli przesuniesz kurSOR poza okno terminala X, w którym uruchomiony jest DOOM. Poza tym, aby móc posłuchać efektów dźwiękowych, należy przekompilować jądro dodając do niego obsługę karty dźwiękowej. Wersja 1.666 gry DOOM nie obsługuje zewnętrznych plików WAD (sugeruję zdobycie wersji zarejestrowanej).

Conquest

Jest to skomplikowana gra, polegająca na podbijaniu świata, wyposażona w równie złożoną instrukcję. Na szczęście przynajmniej pliki są rozprowadzane w postaci skompilowanej, więc nie trzeba tworzyć ich samodzielnie. Należy pamiętać o tym, aby używając pliku `xconq` wydać polecenie `xset fp rehash`, które pozwoli na przywrócenie właściwych czcionek. Podobna gra, o nazwie Empire, jest dostępna w węźle tsx-11.mit.edu również w formie kodu źródłowego – ta wersja wymaga jednak połączenia sieciowego.

Programy demonstracyjne i użytkowe

Opisane poniżej programy mogą Cię zainteresować – informacje o fazach Księżyca są przydatne, szczególnie jeśli jesteś wilkołakiem.

Caesar

Aby uruchomić ten program, należy wydać polecenie `caesar`. Program ten próbuje odszyfrować zakodowane słowa. Stronę `man` można obejrzeć po wydaniu polecenia `man caesar`.

Fortune

Wydanie polecenia `fortune` powoduje wyświetlenie aforyzmu, anegdoty czy powiedzonka.

Number

Program `number` zamienia postać numeryczną liczby na odpowiadającą jej postać słowną w języku angielskim, na przykład wydanie polecenia `number 41` spowoduje wyświetlenie tekstu `forty-one`.

Fazy księżyca

Aby uruchomić ten program, należy wydać polecenie `pom` (ang. *Phase Of the Moon*). Wyświetla on informacje o aktualnej fazie księżyca. Strona `man`, dostępna po wydaniu polecenia `man pom`, podaje, że informacje te mogą być przydatne do przewidywania własnych zachowań oraz zachowań innych osób.

Rain

Aby uruchomić ten program, należy wydać polecenie `rain`. Program ten wyświetla na ekranie terminala zmarszczki podobne do powstających w czasie burzy na powierzchni kałuży. Na większości terminali działa on jednak o wiele za szybko, nie przypominając oryginału nawet w przybliżeniu.

Worms

Aby uruchomić ten program, należy wydać polecenie `worms` (nie pomył go z opisany wcześniejszym programem `worm`). Ekran zostanie zapelniony przez wijące się robaki. Podobnie jak program `rain`, ten program również działa na konsolach linuxowych o wiele za szybko. Stronę `man` można obejrzeć po wydaniu polecenia `man worms`.

Podsumowanie

Teraz możesz już marnować czas, siedząc przed komputerem i grając w ulubione gry. Jeśli nie zainstalowałaś jeszcze systemu X, może skłonią Cię do tego dostępne gry?

Jeśli chcesz zainstalować system X Window, przejdź do rozdziału 22. „Instalacja i konfiguracja systemu X Window”.

Jeżeli chcesz tworzyć własne gry, powinieneś zapoznać się z rozdziałami dotyczącymi języków programowania, które znajdują się w części piątej, począwszy od rozdziału 25. „gawk”.

Tworzenie kopii zapasowych, dzięki którym dane o Twoich najlepszych wynikach osiągniętych w grach będą bezpieczne, omówione zostało w rozdziale 45. „Kopie zapasowe”.

Rozdział 62.

Adabas-D

i inne bazy danych

Tim Parker

W tym rozdziale:

- υ Bazy danych kompatybilne z dBASE
- υ dbMan
- υ Adabas-D
- υ LINCKS
- υ Inne bazy danych

W tym rozdziale omówimy niektóre z popularniejszych programów obsługi baz danych przeznaczonych dla systemu Linux. Skupimy się głównie na takich aplikacjach, jak Adabas-D, FlagShip i dbMan V. Omówimy również skrótnie program LYNCKS, który jest darmowym, obiektowym systemem zarządzania bazami danych (DBMS, DataBase Management System) działającym w systemie Linux.

Bazy danych kompatybilne z dBASE

Około dziesięć lat temu w powszechnym użyciu był tylko jeden system baz danych, dBASE firmy AshtonTate. Zanim pojawił się system Windows, praktycznie każda baza danych dla systemu DOS tworzona była w dBASE, a w miarę migracji systemów UNI-X-owych na mniejsze komputery zaczęły pojawiać się również wersje dBASE przeznaczone dla tych systemów. Choć właściciel pakietu dBASE zmieniał się kilkakrotnie, wersja dla Windows nie pojawiła się na rynku dość szybko (ani nie pozostała na nim zbyt długo), co spowodowało znaczny spadek jego popularności. Wkrótce zamiast dBASE zaczęto używać innych baz danych.

W związku z opracowaniem szybszych i lepszych wersji systemu dBASE, kilka firm wprowadziło kompatybilne z nim produkty, rozszerzające możliwości tego języka. Produkty były ogólnie nazywane xBase, aby podkreślić ich powiązania z systemem dBASE. Kilka z nich zyskało dość dużą popularność wśród programistów – w szczególności Clipper, kompatybilny z dBASE kompilator, który w ogromnym stopniu przyspieszał działanie tworzonych programów.

Choć wielu programistów uważa system dBASE za przestarzały, napisano tysiące (jeśli nie miliony) aplikacji działających w tym relacyjnym systemie baz danych. Wiele z nich działa po dziś dzień, bądź w niezmienionej formie, bądź też po przeniesieniu do systemów xBase i nowszych systemów operacyjnych. Ponieważ mało prawdopodobne jest, że języki rodziny xBase kiedykolwiek odejdą w zapomnienie, nie powinno być żadną niespodzianką to, że opracowano ich wersje dla systemu Linux.

Firma oferująca system FlagShip, który jest systemem baz danych kompatybilnym zarówno z dBASE, jak i Clipper, proponuje jego wersje dla różnych systemów operacyjnych, w większości UNIX-owych. Wersja dla Linuxa jest produktem komercyjnym, kosztującym w Stanach Zjednoczonych około 200\$. Dostępnych jest również kilka wersji demonstracyjnych, których używać można tylko przez 30 dni, dzięki czemu można przekonać się, czy aplikacje stworzone w systemie dBASE czy Clipper będą działać pod kontrolą Linuxa. Jeśli działają prawidłowo i chcesz przenieść je na tę platformę, możesz zakupić pełną wersję systemu FlagShip.

Co to jest xBase?

xBase to dość ogólny termin, obejmujący języki programowania oparte na języku dBASE. Dla systemu DOS są to głównie programy FoxPro (którego właścicielem jest obecnie Microsoft), dBASE V (własność firmy Borland) i Clipper (własność firmy Computer Associates).

W języku xBase można znaleźć instrukcje występujące w większości innych języków programowania, takie jak IF, ELSE, ENDIF czy WHILE. Struktura tego języka nie jest jednak zaprojektowana do zastosowań ogólnych, ale tak, by uprościć dostęp do informacji przechowywanych w bazach danych. Dla przykładu, instrukcja GOTO w języku xBase nie służy do przejścia do określonego punktu w programie, ale do rekordu w bazie danych. xBase zawiera również różne inne polecenia o bardzo dużych możliwościach, służące do przetwarzania plików oraz pobierania danych z formularzy.

Również ustalanie zależności pomiędzy plikami jest w systemie xBase bardzo proste. Nazwy wszystkich pól zapisanych w pliku są przechowywane w jego nagłówku. Nowe pola mogą być dodawane bez modyfikowania operujących na nich programów. Takie rozwiązanie pozwala programom na korzystanie z tej samej bazy danych na różne sposoby, przy wykorzystaniu informacji zapisanych w nagłówku.

Trzej główni producenci baz danych xBase zignorowali system Linux jako platformę dla swoich produktów. Dostępny jest jednak system FlagShip oraz dbMan (firmy Verisoft Corporation). Oba te produkty działają pod kontrolą różnych wersji systemu UNIX.

Nie ma większego sensu porównywanie tych dwóch systemów. FlagShip jest oparty na języku Clipper w wersji 5, natomiast pakiet dbMan przypomina dBASE III+ czy FoxPlus. FlagShip, podobnie jak Clipper, jest kompilatorem, natomiast dbMan działa głównie jako interpreter, choć możliwe jest „skompilowanie” programu dbMan. FlagShip jest poza tym językiem obiektowym, co sprawia, że jego filozofia jest zupełnie inna, niż filozofia języka dbMan, FoxPro czy dBASE. Języki Clipper i FlagShip posiadają kilka cech upodabniających je do języka C. Podobieństwo to jest zaletą dla użytkowników systemu Linux.

Również rynki, dla jakich przeznaczone są te dwa produkty, są różne. dbMan jest przeznaczony głównie dla użytkowników indywidualnych. Jeśli chcesz napisać program, którego będziesz mógł używać do śledzenia czasu poświęcanego klientom czy do przechowywania listy telefonów lub danych o sprzedaży, dbMan jest aplikacją dla Ciebie.

Program FlagShip nie nadaje się do tworzenia aplikacji wykonujących tylko proste operacje na bazach danych, takie jak utrzymywanie listy adresów itp., czyli programów tworzonych przez zwykłych użytkowników. Nie chodzi bynajmniej o to, że nie radzi sobie on z takimi zadaniami, ale aby wykorzystać jego możliwości, trzeba najpierw zapoznać się z samym językiem. FlagShip jest kierowany raczej do osób, które chcą tworzyć własne pakiety oprogramowania. Zgodnie z tradycją, bazy danych systemu dBASE składają się z pliku danych o rozszerzeniu .DBF i plików z indeksami. Format plików danych jest w zasadzie taki sam we wszystkich wersjach języków xBase. Trudno jednak znaleźć dwa produkty używające tego samego formatu plików indeksowych – mimo to zarówno w systemie FlagShip, jak i dbMan można używać tych samych plików z rozszerzeniem .DBF.

Co to jest FlagShip?

FlagShip jest kompilatorem, co oznacza, że tworzy on kod nadający się do bezpośredniego uruchomienia, bez udziału pośredniego pseudo kodu. Nie istnieje żadna interpretowana wersja języka FlagShip, więc do opracowywania złożonych aplikacji możesz potrzebować takiego interpretera, jak FoxPro czy dBASE. FlagShip został zaprojektowany tak, by umożliwić istniejącym programom xBase działanie bez żadnych modyfikacji (albo przy minimalnych modyfikacjach, dotyczących tylko problemów z nazwami plików) w systemie Linux i innych wersjach UNIX-a. Nie są pobierane żadne opłaty za stworzone aplikacje, więc po opracowaniu i skompilowaniu programu można bez przeszkoła rozprowadzać go za darmo.



Wersje demonstracyjne programu FlagShip można załadować z wiele węzłów sieci Internet; można je znaleźć na przykład używając jednej z wyszukiwarek, takich jak AltaVista (<http://www.altavista.digital.com>) czy Yahoo! (<http://www.yahoo.com>), lub też bezpośrednio u dystrybutora programu FlagShip, firmy Multisoft Datentechnik GmbH, którego strona WWW znajduje się pod adresem <http://www.fship.com>. Pod adresem <ftp://fship.com/pub/multisoft> dostępny jest również serwis FTP. Wersja demonstracyjna jest dość spora (ok. 4,5 MB). Dokumentacja różnego typu jest również osią-

galna w węzłach WWW i FTP.

Upewnij się, że ładujesz odpowiednią wersję językową, ponieważ w większości węzłów dostępna jest zarówno wersja angielska, jak i niemiecka; podobnie sprawa wygląda z dokumentacją (ponieważ siedziba firmy Multisoft Datentechnik GmbH znajduje się w Niemczech).

FlagShip jest całkowicie kompatybilny z dBASE i Clipperem, podobnie jak z większością innych systemów xBase, takich jak Fox, FoxPlus, FoxPro, dbMan, QuickSilver i innymi. Dostępna wszystkie przydatne funkcje spotykane w systemach xBase, na przykład:

- υ obsługa makropoleczeń,
- υ tabele, obiekty i bloki kodu,
- υ funkcje i polecenia definiowane przez użytkownika
- υ indeksy i sortowanie tablic
- υ kompatybilność z większością formatów xBase, takich jak .dbf, .dbt, .mem, .lbl, .frm i .fmt
- υ interfejs pozwalający na łączenie programów w języku C i FlagShip w obrębie jednej aplikacji.

FlagShip nie posiada odpowiednika wiersza poleceń systemu dBASE, nie umożliwia również pracy interaktywnej, jak czyni to większość systemów xBase. Dostępny jest jednak program `dbo`, rozprowadzany na licencji public domain, który pozwala w sposób interaktywny definiować funkcje, indeksy, dodawać, usuwać i wyszukiwać rekordy, oraz przeglądać pliki.

Interfejs użytkownika systemu FlagShip oparty jest na bibliotece `curses`. Podczas jego instalacji tworzony jest zestaw plików `terminfo` specjalnie dla systemu FlagShip. Uručhamiąc program FlagShip w oknie terminalu systemu X, możesz zamiast ramek otrzymać dziwne symbole. W takim przypadku może nie pomóc nawet zmiana wartości parametru `acsc` wpisu `fslinuxterm` w pliku `terminfo`. Spróbuj wówczas zastosować czcionki `vga`, które są rozprowadzane z programem DOSemu.

FlagShip nie zawiera funkcji przeznaczonych specjalnie do obsługi rozwijanych menu. Programiści radzą sobie z tym używając do tworzenia poziomych menu polecenia `@PROMPT/MENU`, natomiast do tworzenia menu pionowych – funkcji `ACHOICE()`. Klawisze skrótu można definiować za pomocą polecenia `SET KEY id klawisz TO`. Zwykle polecenia są wywołaniami funkcji. Wewnątrz tych funkcji można wywołać funkcję `READVAR()`, która poinformuje, w którym z pól znajdował się kursor w czasie, gdy wcisnięty został klawisz. Pole wprowadzania danych może zostać uaktywnione przez dodanie polecenia `VALID` do instrukcji `@SAY/GET`. Również w tym przypadku polecenie jest wywołaniem funkcji, wewnątrz której można na przykład poszukać wprowadzonego przez użytkownika ciągu znaków w bazie danych.

W języku FlagShip dostępne są funkcje zarządzające okienkami, które działają w bardzo wygodny sposób, ale nie wchodzą one w skład pakietu podstawowego – aby uzyskać do nich dostęp, trzeba zakupić bibliotekę `FStools`. Jak sugeruje sama nazwa, jest to klon biblioteki Clipper Tools. Funkcje obsługi okien wchodzą również w skład biblioteki `NanForum` (zawierającej poza tym funkcje matematyczne i statystyczne), która jest rozpowszechniana zgodnie z warunkami licencji public domain.

Programy w języku FlagShip mogą być łączone ze stronami WWW, co daje użytkownikom WWW dostęp do baz danych, wraz z możliwością ich aktualniania. Ta cecha, w połączeniu z możliwością wiązania kodu z programami napisanymi w języku C i C++, powoduje że program FlagShip jest systemem zarządzania bazami danych o bardzo dużych możliwościach.

Do programu FlagShip dołączany jest program `fsman`, dzięki któremu można w każdej chwili uzyskać dostęp do potrzebnej dokumentacji (w sumie jest jej ponad 1000 stron), co oznacza, że nie trzeba posiadać ogromnego podręcznika rozłożonego na stałe na biurku. Przykładowe fragmenty kodu pochodzące z dokumentacji mogą być zapisywane jako pliki tekstowe, co pozwala w łatwy sposób włączać je do swych programów. Można oczywiście również używać myszy, kopując fragmenty tekstu pomiędzy oknem programu FlagShip i `fsman`.

FlagShip nie jest tylko kompilatorem przeniesionym z platformy DOS-owej. Został on zaprojektowany tak, by można było skorzystać ze wszystkich możliwości udostępnianych przez systemy UNIX-owe. Kod źródłowy działa w systemie Linux szybciej, niż analogiczny w systemie DOS (skompilowany na przykład kompilatorem Clipper), ponieważ system Linux jest po prostu lepiej zaprojektowany. System FlagShip nie posiada również niektórych ograniczeń systemów xBase dla DOS i Windows.

Jeśli chcesz przenieść aplikacje w języku Clipper na platformę linuxową albo szukasz prostego systemu obsługi relacyjnych baz danych, program FlagShip spełni Twoje oczekiwania.

Instalowanie programu FlagShip

Program FlagShip jest dostępny głównie na płytach CD-ROM, w węzłach WWW i archiwach FTP. W większości przypadków rozpowszechniane są dwie wersje tego programu, a wybór jednej z nich zależy od wykorzystywanej wersji Linuxa. Zwykle pliki są spakowane do jednego archiwum o nazwie `fsdemo.tar` (dla wersji demonstracyjnej). Jedna z wersji przeznaczona jest dla nowszych wersji Linuxa, ponieważ wykorzystuje format plików wykonywalnych o nazwie ELF. Druga wersja, oznaczana zwykle nazwą `aout` (pochodzącą od formatu plików wykonywalnych `a.out`), działa w każdym systemie linuxowym. Choć wersja ELF jest o wiele bardziej elastyczna i ma większe możliwości, w skład pakietu demonstracyjnego wchodzi często tylko wersja `aout`.

Po umieszczeniu pliku `fsdemo.tar` w odpowiednim katalogu (najlepiej pustym i nowo utworzonym) należy rozpakować go, wydając polecenie

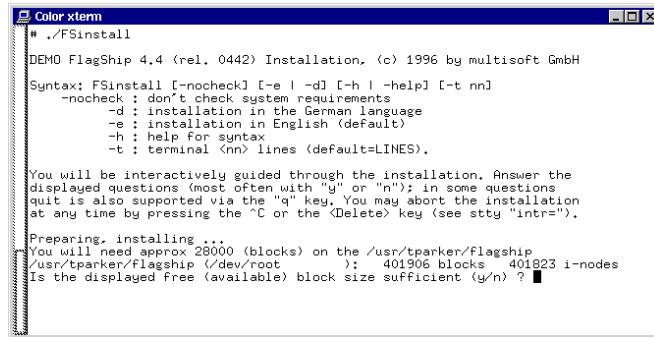
```
tar xvf fsdemo.tar
```

Spowoduje ono utworzenie pewnej liczby plików, a pomiędzy nimi plików `FSinstall.set` i `FSinstall`. W pliku `FSinstall.set` znajdują się definicje zmiennych środowiskowych wykorzystywanych podczas instalacji – należy go uruchomić, wpisując po prostu jego nazwę. Na ekranie nie zostaną wyświetcone żadne komunikaty. Po uruchomieniu pliku `.set` należy wykonać polecenie `FSinstall`, które spowoduje rozpoczęcie właściwego procesu instalacji.

Jeśli program instalacyjny zostanie poprawnie załadowany, wyświetlane zostanie pytanie dotyczące ilości wolnego miejsca w systemie plików, jak pokazuje rysunek 62.1.

Rysunek 62.1.

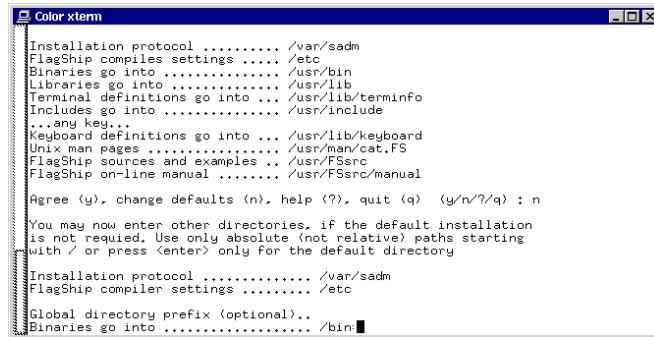
Program instalacyjny systemu FlagShip sprawdza, czy na dysku jest wystarczająca ilość wolnego miejsca



Następnie będzie można wybrać docelowy katalog instalacji – w większości przypadków powinieneś zaakceptować wartość podpowiadaną przez program instalacyjny. Jeśli chcesz zmienić któryś z wartości domyślnych, możesz to zrobić po prostu wpisując nową wartość w odpowiedzi na monit programu, jak pokazano na rysunku 62.2. Po zakończeniu programu instalacyjnego można już zacząć używać systemu FlagShip.

Rysunek 62.2.

W razie potrzeby można zmienić wartości podpowiadane przez program instalacyjny



Używanie programu FlagShip

Jeśli używałeś wcześniej programu Clipper lub innego kompilatora xBase, znasz już większość poleceń wykorzystywanych w języku FlagShip. Wprowadzone modyfikacje wynikają głównie z dostosowania tego języka do wymagań i możliwości systemów UNIX-owych, ale poza tym używanie języka FlagShip jest bardzo proste. Trzeba pamiętać o tym, że FlagShip nie jest programem do interaktywnego projektowania aplikacji; nie pomaga on w żaden sposób w tworzeniu kodu programu. Jest to po prostu kompilator. Możesz używać go do tworzenia aplikacji tylko pod warunkiem, że znasz język dBASE – nie jest on bowiem przeznaczony do nauki tego języka.

Jeśli przygotowałeś już pliki źródłowe .prg, możesz uruchomić kompilator FlagShip. Składnia odpowiedniego polecenia jest następująca:

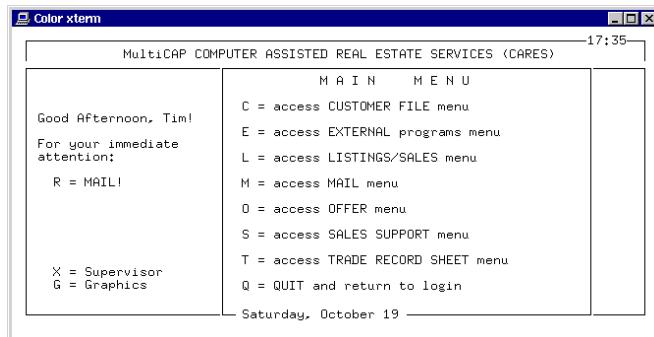
```
FlagShip nazwa_prog.prg -onazwa_prog_wynikowego -Mstart
```

`nazwa_prog` to nazwa pliku, zawierającego główną część kodu źródłowego programu (wywołującą pozostałe części), natomiast `nazwa_prog_wynikowego` to nazwa, jaka zostanie nadana skompilowanej, gotowej do uruchomienia wersji programu (kompilator języka C domyślnie przyjmuje nazwę `a.out`). Jeśli program główny nie wywołuje wszystkich plików programów niezbędnych do prawidłowego działania całej aplikacji, trzeba skompilować odpowiednie pliki oddzielnie, a następnie powiązać je ze sobą.

Po skompilowaniu aplikacja może zostać uruchomiona podobnie jak w systemie DOS czy innym systemie operacyjnym. Rysunek 62.3 przedstawia aplikację systemu DOS przeniesioną na platformę linuxową, skompilowaną za pomocą programu FlagShip i uruchomioną. Jedyną modyfikacją kodu źródłowego, którą trzeba było wprowadzić, było dostosowanie ścieżek dostępu do plików do wymagań systemu Linux. Jak widać, nawet grafika tworzona za pomocą znaków ASCII została zachowana prawidłowo, dzięki czemu program może być bez problemu uruchamiany z dowolnego terminalu obsługiwanej przez system Linux.

Rysunek 62.3.

Aplikacja skompilowana kompilatorem FlagShip działa w systemie Linux tak samo, jak działała w systemie DOS



Przenoszenie istniejących aplikacji

O co więc trzeba zadbać przy przenoszeniu na platformę linuxową istniejących aplikacji w języku dBASE czy Clipper? Na początek potrzebny będzie oczywiście kod progra-

mu, zapisany w plikach z rozszerzeniem .prg. Przenieś te pliki do systemu linuxowego w dowolny sposób – czy to za pomocą dyskietki, połączenia sieciowego, czy też inną drogą. Program FlagShip jest na tyle sprytny, że potrafi ignorować wielkość liter – może to brzmieć banalnie, ale w rzeczywistości jest to poważny problem, ponieważ większość programistów pracujących w systemie DOS tworzy programy używając zarówno wielkich, jak i małych liter – nie ma to dla nich większego znaczenia, natomiast w systemie UNIX wielkość liter jest bardzo istotna, co może powodować wiele kłopotów.

Do programu FlagShip dołączona jest dokumentacja dotycząca modyfikowania kodu źródłowego tak, by działał prawidłowo (jest ona również dostępna na stronie WWW), ale w zasadzie prawie wszystkie aplikacje powinny działać bez żadnych problemów.

FlagShip zamienia kod źródłowy w języku dBASE na odpowiadający mu kod w języku C, który następnie jest komplikowany. Z tego względu potrzebny jest również kompilator języka C, który na szczęście jest częścią prawie każdego systemu UNIX-owego i linuxowego. Jeśli chcesz używać programu FlagShip, a nie zainstalowałeś jeszcze pakietu służącego do programowania w języku C, musisz to najpierw zrobić, w przeciwnym razie program FlagShip nie będzie działał poprawnie, generując komunikaty o błędach. Nie jest potrzebny kompilator języka C++ – w zupełności wystarczy kompilator języka C, dostępny na każdej płycie CD-ROM zawierającej dystrybucję Linuxa (nie wyłączając płyty rozprowadzanej z tą książką). Proces uruchamiania aplikacji przez program FlagShip jest dość prosty:

1. wstępne przetworzenie kodu źródłowego, dzięki czemu wykrywane są błędy składniowe i inne często popełniane błędy; jeśli jakieś błędy zostaną wykryte, są one zgłasiane i program kończy działanie;
2. tłumaczenie kodu na język C;
3. komplikacja za pomocą kompilatora języka C, co prowadzi do powstania pliku pośredniego .obj;
4. konsolidacja skompilowanego pliku wraz z bibliotekami programu FlagShip do postaci wykonywalnej.

Utworzony plik wykonywalny może zostać uruchomiony z wiersza poleceń systemu Linux.

Jeszcze krótka uwaga dla doświadczonych użytkowników systemów dBASE i Clipper: nie trzeba martwić się o nakładki, ponieważ w systemach UNIX-owych nie są one potrzebne. Dzięki temu, że system używa pamięci wirtualnej, możliwe jest ładowanie do pamięci aplikacji o prawie dowolnym rozmiarze (choć istnieją pewne ograniczenia, mogą one zostać zmienione). Z tego powodu nie trzeba martwić się nakładkowaniem, tak jak to miało miejsce w systemach dBASE i Clipper – zamiast tego można śmiało połączyć cały kod w jeden duży plik wykonywalny, pozwalając systemowi Linux zająć się jego ładowaniem.

dbMan

Program dbMan jest interpreterem. Po jego uruchomieniu na ekranie wyświetlany jest wiersz poleceń ze znakiem zachęty, który ma formę `CMD:`. W tym wierszu należy wpisywać wszystkie polecenia. Rozwiążanie to jest zbliżone do używanego w systemie dBASE – tam znakiem zachęty była kropka. Początkujący użytkownicy mogą skorzystać z polecenia `ASSIST`, które przywoła na ekran interfejs obsługiwany za pomocą menu, podobny do tych dostępnych w systemach FoxPro czy dBASE.

Interfejs oparty na systemie menu nie jest szczególnie rozbudowany. Pozwala na pracę tylko z jednym plikiem w tym samym czasie, co oznacza, że nie można ustalać żadnych powiązań pomiędzy plikami. Z poziomu tego menu można również uruchomić prosty generator programów, ale tu również nałożone jest ograniczenie umożliwiające pracę tylko z jednym plikiem.

Programy w systemie dbMan można kompilować, ale proces ten nie powoduje powstania pliku wykonywalnego. Zamiast tego, tworzony jest plik z rozszerzeniem `.run`, do uruchomienia którego niezbędny jest program dbMan. W wierszu poleceń programu dbMan można także wydać polecenie `CREATE REPORT` lub `MODIFY REPORT`, które powoduje uruchomienie modułu do tworzenia raportów. Pozwala on na wyświetlanie danych z użyciem operatorów relacyjnych. Funkcja `PMENU()` pozwala tworzyć rozwijane menu. Niestety, nie jest ona wyposażona w żadne mechanizmy pozwalające na tymczasowe wyłączenie którejś z jego menu.

Obsługa okien w systemie dbMan odbiega od tej znanej z innych systemów xBase. Przed zdefiniowaniem nowego okna trzeba wywołać procedurę `PUSHWIND()`, która odłoży na stos dane o oknie bieżącym. Kiedy program jest w stanie początkowym, cała zawartość ekranu traktowana jest jako okno. Procedura `WINDOW()` służy do tworzenia nowego okna. Po zakończeniu pracy z oknem należy wywołać procedurę `POPWIND()`, która spowoduje zdjęcie ze stosu i uaktywnienie poprzedniego okna.

W systemie dbMan można zdefiniować tylko jeden klawisz skrótu, wywołując funkcję `ONKEY()`. Nie będzie ona miała żadnego efektu, dopóki nie zostanie wywołane następujące po niej polecenie – zwykle jest to instrukcja `DO`, obsługująca naciśnięcie danego klawisza.

Polecenie `BROWSE` może zostać wywołane z wieloma różnymi opcjami. Używając go, można przeglądać jedynie wyszczególnione pola, można również określić szerokość poszczególnych pól, oraz to, czy mogą one być edytowane. Lista pól może zawierać pola zapisane w innych plikach, co pozwala w łatwy sposób wiązać ze sobą dane pochodzące z różnych baz.

dbMan nie używa informacji zapisanych w pliku `termcap` czy `terminfo`. Zamiast tego zawiera on plik o nazwie `dbmterm.dbm`. Zawartość tego pliku jest zbliżona do zawartości pliku `termcap`. Nie ma w nim wpisów dotyczących terminalu X czy konsoli, więc trzeba stworzyć własne wpisy w oparciu o już istniejące.

dbMan nie zawiera interfejsu pozwalającego na wykonywanie funkcji napisanych w języku C czy asemblerze, można więc używać tylko funkcji oferowanych przez ten system. W wersji, którą testowałem (o numerze 5.32), było kilka dość poważnych błędów. Naj-

ważniejszym z nich był chyba fakt, że pliki procedur po prostu nie działały, jeśli były to pliki z rozszerzeniem .prg. Po skompilowaniu ich do postaci plików z rozszerzeniem .run wszystko działało poprawnie.



Program dbMan jest rozpowszechniany przez firmę

VERASOFT Corporation
4340 Alamaden Expressway, #110
San Jose, CA95118
(408) 723-9044

Adabas-D

Adabas-D to baza danych możliwościami dorównującą aplikacjom komercyjnym dla systemów UNIX-owych, kosztującym tysiące dolarów. Obsługuje język SQL, pozwala również na pracę z wieloma popularnymi formatami baz danych – warto więc rozważyć jej zainstalowanie, jeśli potrzebujesz większych możliwości, niż oferowane przez wspomniane wcześniej programy xBase.



Na rynku działa kilku dystrybutorów programu Adabas-D. Prawdopodobnie najpewniejszym z nich jest firma Caldera, która rozpowszechnia również inne produkty dla systemu Linux oraz własną wersję dystrybucji Slackware, o nazwie Caldera OpenLinux. Z firmą tą można skontaktować się pod adresem:

Caldera Incorporated
633 South 550 East
Provo, Utah 84606
801-377-7687
knfo@caldera.com
<http://www.caldera.com>

Instalowanie programu Adabas-D

System Adabas-D jest zwykle rozpowszechniany na płytach CD-ROM, ale udostępniają go również niektóre węzły FTP. Choć wersja rozpowszechniana przez firmę Caldera reklamowana jest jako przeznaczona do pracy z systemem Caldera OpenLinux, działa ona znanym zamiast wszystkimi implementacjami Linuxa. Instalacja pakietu jest prosta. Podana poniżej procedura działa we wszystkich wersjach systemu RedHat, Slackware i OpenLinux, które udało nam się przetestować. Założymy tu, że oprogramowanie Adabas-D znajduje się na płycie CD-ROM. Jeśli załadowałeś odpowiedni plik z sieci, powinieneś umieścić go w osobnym katalogu i pominiąć kroki dotyczące montowania systemu plików.

Zaloguj się jako `root`, proces instalacji wymaga bowiem uprawnień administratora. Pierwszym krokiem jest zamontowanie systemu plików zapisanego na płycie CD-ROM. Można to zrobić wydając polecenie

```
mount /dev/cdrom
```

W tej wersji polecenia `mount` pominęliśmy argumenty dotyczące urządzenia oraz punktu zamontowania, które omówiliśmy w poprzednich rozdziałach. System plików zostanie zamontowany w katalogu `/mnt/cdrom` – jeśli chcesz, możesz oczywiście wymusić zamontowanie go w innym katalogu. W dalszej części tego rozdziału przyjmiemy, że punktem zamontowania jest właśnie katalog `/mnt/cdrom`.

Po zamontowaniu płyty CD-ROM przejdź do katalogu zawierającego pliki instalacyjne, na przykład wydając polecenie

```
cd /mnt/cdrom
```

Wyświetlenie informacji o zawartości katalogu powinno pokazać pliki systemu Adabas-D. Większość z nich znajduje się w podkatalogach katalogu głównego.

Dla uproszczenia życia użytkownika, ta wersja systemu Adabas-D zawiera skrypt instalacyjny, który bierze na siebie wszystkie czynności, jakie normalnie należałoby wykonać ręcznie. Aby uruchomić ten skrypt, wystarczy wydać polecenie

```
./install
```

Kropka i ukośnik poprzedzające nazwę skryptu oznaczają, że jest on umieszczony w katalogu bieżącym. Jeśli zostałyby one pominięte, przed sprawdzeniem zawartości bieżącego katalogu Linux szukałby pliku o nazwie `install` w innych lokalizacjach, co zwykle prowadzi do uruchomienia innego programu instalacyjnego. Jeśli w Twojej wersji pakietu Adabas-D nie ma skryptu instalacyjnego, w którymś z plików powinien znajdować się dokładny opis czynności, które należy wykonać, aby go zainstalować.

Jeśli używasz skryptu instalacyjnego, będziesz miał możliwość zainstalowania kilku pakietów oprogramowania oraz zdecydowania, w którym katalogu mają się one znaleźć. Dla uzyskania pełnego i kompletnego systemu Adabas-D należy zainstalować wszystkie dostępne pakiety; jedynym wyjątkiem są pakiety obsługujące języki, których nie potrzebujesz (Adabas-D zwykle rozprowadzany jest wraz z angielską i niemiecką dokumentacją oraz zestawem poleceń). W skład pakietu w wersji Personal Edition wchodzą cztery składniki.

- ⦿ `adabas-6.1PE`. Jądro programu oraz wszystkie pliki wykonywalne i konfiguracyjne (ok. 46 MB).
- ⦿ `adabas-mydb`. Przykładowa baza danych zawierająca dane o niemieckich liniach kolejowych (ok. 74 MB).
- ⦿ `adabas-docEN`. Dokumentacja w języku angielskim (105 MB).
- ⦿ `adabas-dokDE`. Dokumentacja w języku niemieckim (79 MB).

Przed uruchomieniem programu Adabas-D należy ustawić odpowiednie wartości dwóch zmiennych środowiskowych dla każdego użytkownika, który będzie korzystał z baz danych. Zmienne te mają następujące znaczenie:

- υ **DBROOT** – katalog programu (na przykład `/opt/adabas-d`),
- υ **PATH** – do istniejącej wartości ścieżki przeszukiwania należy dodać podkatalog bin katalogu **DBROOT** (co zwykle uzyskuje się przez wpisanie `$DBROOT/bin:$PATH`).

Pakiet `adabas-mydb` pozwala na łatwe opanowanie obsługi systemu Adabas-D na podstawie przykładowej bazy danych dotyczącej kolei w Niemczech (która jest również sama w sobie warta obejrzenia, jeśli interesujesz się koleją). Można uruchomić ten przykład w oknie terminalu X Window, aby przekonać się, jakie możliwości ma system Adabas-D. Aby uruchomić program demonstracyjny, uruchom system X Window (po ustawieniu odpowiednich wartości zmiennych środowiskowych) i wydaj polecenie

`panel`

Nazwa użytkownika, którą należy podać, to `control`, również hasło ma postać `control`. Nazwa bazy danych to `MYDB`. Kliknij na przycisku `Connect` i poczekaj, aż kontrolka przypominająca światła na skrzyżowaniu zmieni kolor na zielony. Baza danych zawiera również zdjęcia niektórych niemieckich pociągów. Jeśli chcesz obejrzeć dołączone obrazki, w oknie terminalu wydaj polecenie `fotos`, następnie podaj nazwę użytkownika `demo` i hasło `demo` oraz tę samą co poprzednio nazwę bazy danych (`MYDB`).

System Adabas-D powinien już działać poprawnie. Jeśli występują jakieś problemy, upewnij się, że wartości wspomnianych wcześniej zmiennych środowiskowych są prawidłowe oraz że podczas instalacji nie wystąpił żaden błąd.

Dokumentacja systemu Adabas-D znajduje się w podkatalogu `doc` katalogu głównego systemu Adabas-D (na przykład `/opt/adabas-d/doc`).

LINCKS

LINCKS to obiektowy system zarządzania bazami danych. Znakomicie nadaje się on do użytku w sieci oraz w sytuacjach, gdy pliki mają być współużytkowane za pomocą zdalnych wywołań procedur. Aby móc w pełni wykorzystać możliwości oferowane przez ten program, trzeba mieć nieco doświadczenia w pracy z siecią i posiadać działającą sieć. Pakiet nie jest raczej przeznaczony dla systemu jednostanowiskowego, ponieważ po prostu mnóstwo jego możliwości pozostałoby niewykorzystanych.



Pakiet LINCKS można załadować z katalogu `/pub/linux/apps/database/ lincks` na serwerze `sunsite.unc.edu`.

LINCKS oparty jest na obiektowej strukturze pozwalającej jedynie na dołączanie nowych składników. Nowe obiekty definiowane są na podstawie innych obiektów. Pomiędzy obiektami można tworzyć powiązania, obrazujące związki pomiędzy danymi. Dla obiektu można również zdefiniować widok (lub nawet kilka widoków), który pozwala określić, w jaki sposób dane będą prezentowane użytkownikowi. Widoki mogą być dziedziczone.

Głównym interfejsem tego pakietu jest program `xlincks`. Z jego pomocą, używając poleceń zbliżonych do tych wykorzystywanych w edytorze `emacs`, można w sposób interaktywny przeglądać bazy danych. Interfejs nieco przypomina w działaniu funkcje hipertekstowe dostępne na stronach WWW. Po kliknięciu na wyróżnionym elemencie program wyświetla dodatkowe informacje na zadany temat.

Pomoc dostępna jest w dwóch postaciach: jako pomoc kontekstowa oraz jako baza danych, którą można swobodnie przeglądać. Dostęp do pliku pomocy można uzyskać w każdej chwili, wciskając klawisz `Help`. Zawartość tego pliku jest zorganizowana w bardzo przezroczysty sposób, dzięki czemu może on zostać wykorzystany również do nauki systemu LINCKS od podstaw. W archiwach systemu `sunsite` dostępny jest również podręcznik w wersji postscriptowej.

W pakiecie LINCKS znajduje się kilka programów. Program `dbroot` służy do tworzenia nowych baz danych. Aby uwolnić bazę danych od obiektów, które nie są już używane, należy wydać polecenie `cutoff`. Serwerem całej aplikacji jest program `netserv`, który uruchamia procesy `db`s obsługujące każdego łączącego się z nim klienta.

Inne bazy danych

Oczywiście narzędzia xBase, które omówiliśmy w tym rozdziale, nie są jedynymi systemami obsługi baz danych dla platformy linuxowej. Istnieje również mnóstwo programów niekompatybilnych z systemem dBASE. Przyjrzymy się jeszcze króciutko niektórym z nich (większość jest –dostępna za darmo i może zostać załadowana z Internetu).

- υ **mbase v5.0** to system obsługi relacyjnych baz danych stworzony pierwotnie dla komputerów Amiga, a następnie przeniesiony na inne platformy. Do obsługi baz danych używany jest język przypominający język C. Kompilacja programu `mbase` wymaga biblioteki `ncurses` oraz sporej ilości czasu. Zdarzają się również problemy z plikiem `Makefile`, który może wymagać ręcznej modyfikacji. Jeśli potrzebujesz taniego systemu dostępu do baz danych, zbliżonego do języka C, możesz rozpatrzyć używanie tego systemu. Jednak programy FlagShip lub dbMan są bardziej stabilne i mają większe możliwości.
- υ **onyx** to program do tworzenia baz danych za pomocą języka zbliżonego do języka C. Polecenie `make config` powoduje uruchomienie procesu instalacyjnego, podczas którego należy odpowiedzieć na szereg pytań, dzięki którym możliwe będzie prawidłowe skonfigurowanie bazy danych dla systemu Linux.
- υ **DBF** to zestaw programów służących do manipulowania bazami zgodnymi z xBase (z rozszerzeniem `.dbf`). Niektóre z tych programów potrafią na przykład dodać do pliku nowe rekordy lub warstwy informacji (`dbfadd`), inne potrafią wyświetlać zapisane w bazie danych informacje (`dbflist`). Program `dbft` pozwala na wyświetlenie informacji o strukturze bazy danych i poszczególnych jej elementów.
- υ **typhoon** to jeszcze jeden system obsługi relacyjnych baz danych. Wyróżnia go fakt, że składnia języka używanego do obsługi baz danych praktycznie niczym nie różni się od języka C. Zanim jednak będzie można używać go do poważniejszych zastosowań, program musi jeszcze zostać dopracowany.

Podsumowanie

Programy FlagShip oraz dbMan są doskonałą metodą na przenoszenie istniejących aplikacji dBASE i kompatybilnych na platformę linuxową. W większości przypadków mogą one działać o wiele szybciej i zwykle nie wymagają żadnych modyfikacji. W czasie tworzenia tej książki opracowywana była wersja programu FlagShip dla systemu X Window.

Wersja demonstracyjna programu pozwala programistom pracować nad systemami baz danych w systemie Linux. Wersja pełna, będąca produktem komercyjnym, zawiera 1500 stron dokumentacji i kosztuje około 200\$, wydaje się więc być niezłą propozycją dla dostawców oprogramowania. dbMan jest programem alternatywnym, przeznaczonym dla użytkowników, którym nie są potrzebne rozszerzenia oferowane przez FlagShip i możliwość komplikacji programów.

Rozdział 63.

StarOffice

Tim Parker

W tym rozdziale:

- υ Instalacja pakietu StarOffice
- υ Uruchamianie StarOffice
- υ Importowanie i eksportowanie plików

StarOffice 5.0 to przeznaczony dla środowiska X Window pakiet programów biurowych działających w systemie Linux. Jest on rozprowadzany wraz z niektórymi wersjami Linuksa (na przykład Caldera OpenLinux), można go również załadować w niektórych węzłach FTP i na stronach WWW. Wersja StarOffice 5.0, która jest wersją najnowszą, zawiera trzy aplikacje: StarWriter, który jest edytorem tekstów, arkusz kalkulacyjny Star-Calc oraz program do tworzenia prezentacji o nazwie StarImpress, potrafiący również generować dokumenty w języku HTML.

Choć możliwości tego pakietu nie są aż tak ogromne, jak możliwości pakietu Office firmy Microsoft czy WordPerfect Suite firmy Corel, to i tak najprawdopodobniej będziesz zaskoczony jego możliwościami i przydatnością. Narzędzia wchodzące w skład pakietu są najczęściej wykorzystywany w codziennej pracy programami, potrafiącymi w dodatku współpracować z bazami danych (więcej informacji o bazach danych możesz znaleźć w rozdziale 62.), co powoduje, że stanowią one pełną platformę biurową dla systemów linuxowych. Znakomicie nadają się również do wykorzystania w domu.

Program StarOffice działa ze wszystkimi dostępnymi obecnie wersjami Linuksa, pod warunkiem, że pracuje w nich również serwer X Window.



Program StarOffice jest dostępny również dla innych platform, na przykład SCO UNIX, jednak w wersji dla nich kosztuje około 550\$. Użytkownicy systemu Linux mogą zakupić go w zestawach oferowanych za znacznie niższą cenę, od firm takich jak na przykład Caldera. Dokładniejsze informacje znajdziesz pod adresem

<http://www.caldera.com>.

Instalacja pakietu StarOffice

Pakiet StarOffice nie jest instalowany za pomocą standardowych programów instalujących, ale za pomocą specjalnego skryptu. Aby zainstalować ten pakiet, musisz być zalogowany jako `root`. StarOffice jest aplikacją systemu X, więc przed rozpoczęciem instalacji należy uruchomić ten system, a poszczególne polecenia wpisywać w oknie terminalu.

Aby zainstalować pakiet StarOffice, który zwykle dostarczany jest na płytach CD-ROM, należy najpierw zamontować odpowiednią płytę w punkcie, którego zwykle używasz, na przykład wydając polecenie:

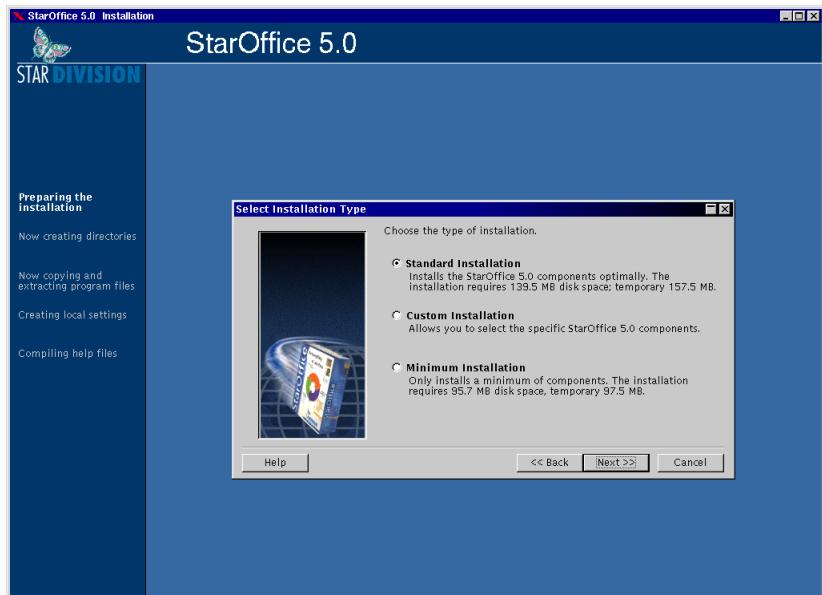
```
mount /dev/cdrom /usr/cdrom
```

Następnie należy przejść do katalogu, w którym dostępne są pliki zapisane na płycie CD-ROM. Dalej należy przejść do katalogu o nazwie StarOffice 5.0 (pozostałe katalogi zawierają pliki pomocnicze i informacje o prawach autorskich), podkatalogu `bin`, i w końcu uruchomić skrypt instalacyjny, wydając polecenie

```
./setup
```

Kropka i ukośnik przed właściwą nazwą pliku oznaczają, że jest to plik zapisany w katalogu bieżącym, dzięki czemu na pewno wykonany zostanie ten program, o który chodzi. Po uruchomieniu i wyświetleniu informacji o prawach autorskich zostanie wyświetlone okno przedstawione na rysunku 63.1. W większości przypadków najlepiej w tym momencie zdecydować się na instalację standardową, chyba że chcesz zainstalować tylko niektóre ze składników pakietu. Jeśli masz mało miejsca na dysku, również powinieneś pomyśleć o zrezygnowaniu z instalacji plików, które nie są niezbędne.

Rysunek 63.1.
Instalator pakietu
StarOffice daje
możliwość wyboru
rodzaju instalacji



Skrypt instalacyjny automatycznie kopiuje wszystkie potrzebne pliki, wyświetlając percentowy wskaźnik zaawansowania tej operacji. Po zakończeniu instalacji skrypt kończy działanie.

Uruchamianie StarOffice

Domyślnie program StarOffice instalowany jest w katalogu `/root/Office50`. Jeśli wybrałeś instalację ręczną, możesz zmienić tę ścieżkę, ale nie jest to konieczne w większości wersji Linuxa. Katalog ten oraz jego podkatalog `bin` należy dodać do ścieżki przeszukiwania, dzięki czemu będzie można łatwo uruchamiać pakiet z dowolnego miejsca w systemie plików.

Aby uruchomić pakiet StarOffice, otwórz okno terminalu (jeśli żadne nie jest jeszcze otwarte), przejdź za pomocą polecenia `cd` do katalogu `/root/Office50/bin` (jeśli nie dołączyleś go do ścieżki przeszukiwania) i wydaj polecenie:

```
soffice &
```

Jeśli w skład ścieżki przeszukiwania nie wchodzi ani katalog programu StarOffice, ani katalog bieżący (.), należy poinformować interpreter poleceń, że pliku wykonywalnego ma szukać w katalogu bieżącym, wydając polecenie

```
./soffice &
```

Ampersand na końcu polecenia spowoduje, że zostanie ono wykonane w tle, dzięki czemu nie będzie blokowało dostępu do terminalu X. Polecenie `soffice` jest najprostszym sposobem uruchamiania tego pakietu, choć możliwe jest również uruchamianie poszczególnych aplikacji.



StarOffice jest jednym z najbardziej stabilnych pakietów biurowych; podczas ponad miesiąca nieprzerwanych testów na platformie Slackware nie załamał się ani razu. Niewiele aplikacji (a nawet niewiele systemów operacyjnych) może pochwalić się takimi wynikami.

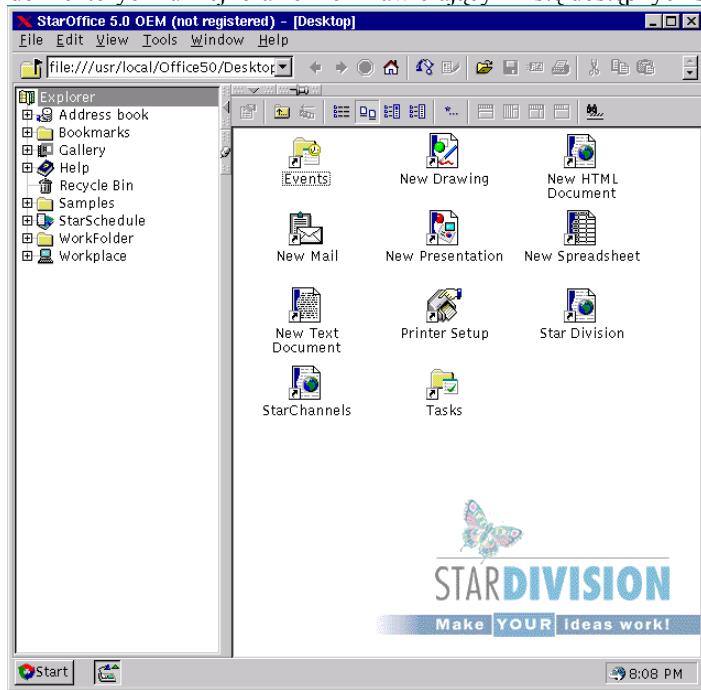
Po uruchomieniu programu StarOffice wyświetlane jest okno główne, przedstawione na rysunku 63.2. Jest ono podzielone na część zawierającą przeglądarkę plików oraz drugą, w której wyświetlany jest zestaw ikon pozwalających na uruchomienie poszczególnych aplikacji.

StarWriter

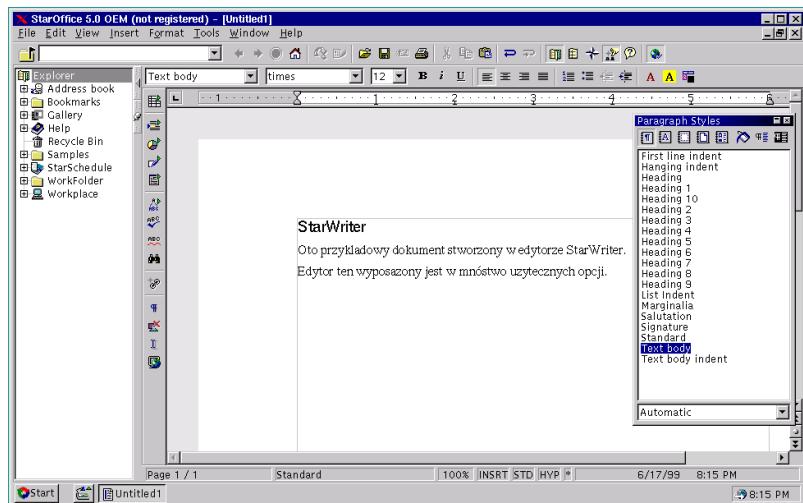
StarWriter to edytor tekstów wchodzący w skład pakietu StarOffice. Aby uruchomić ten program z nowym dokumentem, kliknij dwukrotnie na ikonę New Document w oknie głównym programu StarOffice. Spowoduje to uruchomienie programu StarWriter, jak widać na rysunku 63.3. Lewa część okna nadal zawiera przeglądarkę plików, natomiast w części prawej wyświetlone jest okno edytora tekstów, wraz z ikonami pozwalającymi na szybki dostęp do niektórych funkcji oraz oknem zawierającym listę dostępnych stylów.

Rysunek 63.2.

Okno główne programu StarOffice pozwala poruszać się w systemie plików i uruchamiać poszczególne aplikacje



Rysunek 63.3.
StarWriter
to edytor tekstów,
przypominający
edytor Microsoft
Word



Edytor StarWriter zachowuje się podobnie jak system Windows, zarówno pod względem dostępnych polecień, jak i skrótów klawiaturowych oraz ikonek dostępnych w górnej części ekranu. Dzięki temu jego obsługa jest bardzo łatwa (szczególnie dla użytkowników, którzy pracowali wcześniej z takimi edytorem, jak Word czy WordPad).

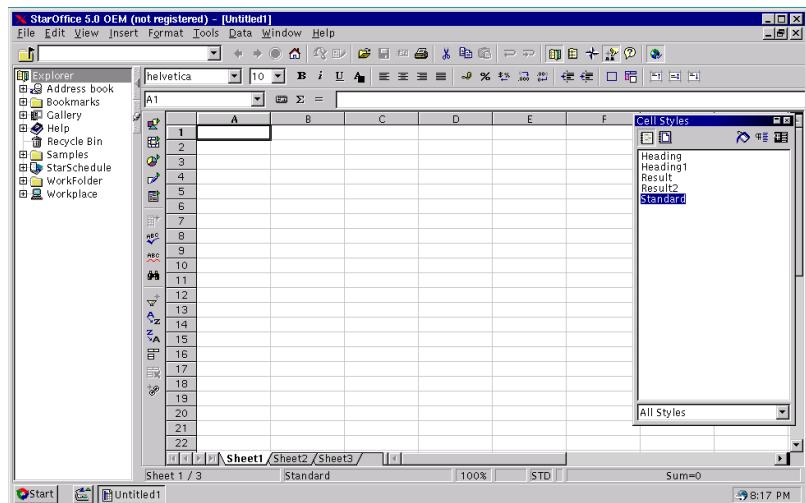
StarWriter zawiera również mnóstwo drobnych udogodnień, które na pewno spodobają się użytkownikom Linuxa. Można o nich poczytać używając systemu pomocy pakietu StarOffice. Dla przykładu, jeśli wysydasz dużą liczbę wiadomości pocztą elektroniczną czy też intensywnie korzystasz z grup dyskusyjnych, jesteś na pewno przyzwyczajony do akcentowania fragmentów tekstu w sposób zgodny ze standardem ASCII. Na przykład, aby pogrubić dany fragment tekstu, zwykle otacza się go gwiazdkami, na przykład pisząc *A teraz coś bardzo ważnego*. Podkreślenie uzyskuje się przez umieszczenie symbolu podkreślenia przed i po wyróżnionym fragmencie, czyli _w taki sposób_. StarWriter potrafi zorientować się, o co Ci chodzi, i zastosować odpowiedni kroj czcionki. Dzięki temu używając tego edytora nie musisz zmieniać swoich przyzwyczajeń.

StarCalc

StarCalc, jak widać na rysunku 63.4, jest arkuszem kalkulacyjnym. Podobnie jak StarWriter, StarCalc zachowuje się prawie tak samo jak program Excel wchodzący w skład pakietu Microsoft Office. Jeśli więc potrafisz obsługiwać program Excel, nie będziesz miał żadnych problemów z programem StarCalc.

Rysunek 63.4.

StarCalc to arkusz kalkulacyjny, mający możliwości podobne do oferowanych przez program Microsoft Excel



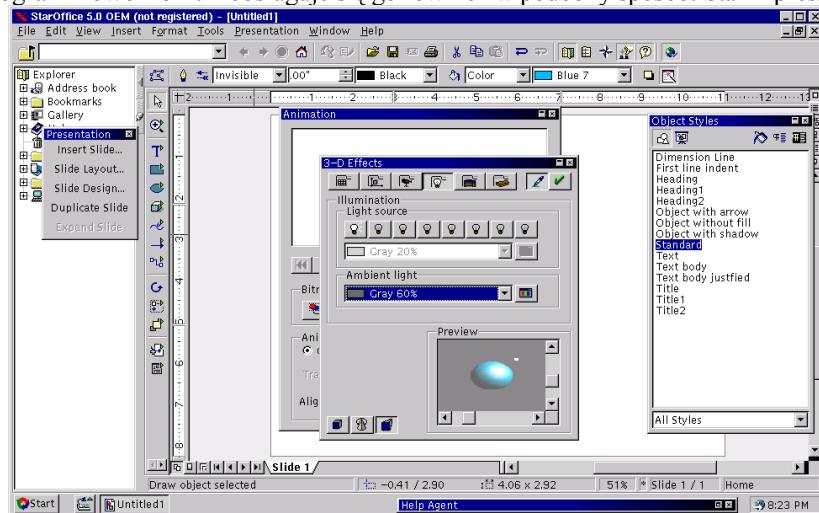
Jedną z cech programu StarCalc, której brakuje w wielu innych programach tego typu (nie wyłączając programów dla Windows 95), jest to, że do opisu równań można używać języka naturalnego. Można również na przykład pojedyńczym poleceniem utworzyć tabelę, w której pierwszy rzad i pierwsza kolumna będą pełniły funkcję nagłówka.

StarImpress

Program StarImpress może znaleźć wiele zastosowań – można go na przykład wykorzystać do tworzenia prezentacji graficznych. Jak widać na rysunku 63.5, StarImpress bardzo przypomina program PowerPoint – obsługuje się go również w podobny sposób. StarImpress

Rysunek 63.5.

StarImpress to program do tworzenia prezentacji wchodzący w skład pakietu StarOffice



zawiera kilka bardziej zaawansowanych możliwości, na przykład potrafi bardzo szybko tworzyć efekty trójwymiarowe. Dzięki temu może konkuruwać z innymi programami dla platform UNIX-owych przeznaczonymi do tego celu, na przykład pakietem CorelDraw! dla UNIX-a, które są o wiele droższe.

Bez żadnych problemów można również używać programu StarImpress wraz z innymi aplikacjami pakietu, osadzając w nich utworzone grafiki. Do importowania dokumentów używa się techniki zbliżonej do OLE, dzięki czemu w dokumencie programu StarWriter czy StarCalc można umieścić dowolną liczbę grafik pochodzących z programu StarImpress.

Program StarImage jest programem pomocniczym, pozwalającym na szybką konwersję plików graficznych do różnych formatów oraz na manipulację ich zawartością. StarImage potrafi obsługiwać pliki w formatach .BMP, .JPEG, .GIF, .TIFF i .XBM. Większość jego funkcji pokrywa się z udostępnianymi przez inne programy graficzne, na przykład xv, ale scalenie go z pakietem StarOffice ułatwia pracę z różnymi narzędziami wchodzącymi w jego skład.

Importowanie i eksportowanie plików

Jak mogłeś się przekonać, pakiet StarOffice bardzo przypomina analogiczny pakiet firmy Microsoft. Rozciąga się to również na możliwości importowania i eksportowania plików innych formatów. Choć własny format, w którym StarOffice zapisuje dokumenty, arkusze kalkulacyjne i grafiki, nie jest kompatybilny z programami Microsoft Word, Excel i PowerPoint, dostępna jest opcja pozwalająca na zapisanie dokumentów w innych formatach. Jednym z tych formatów jest właśnie format używany przez pakiet Microsoft Office, co znakomicie ułatwia przenoszenie dokumentów pomiędzy tymi pakietami.

Oprócz formatów zgodnych z pakietem Office, StarOffice umożliwia również użycie formatów tekstowych, takich jak ASCII, RTF i HTML. Ostatni z nich jest oczywiście bardzo przydatny do tworzenia stron WWW.

Po przeniesieniu plików pochodzących z aplikacji firmy Microsoft do systemu linuxowego możliwe jest importowanie ich do pakietu StarOffice. Można następnie zapisać je w tym samym formacie bądź też w formacie używanym przez pakiet StarOffice. Możliwość wymiany plików pomiędzy pakietem StarOffice i programami dla systemu Windows bardzo ułatwia pracę.

Podsumowanie

Teraz już wiesz, w jaki sposób można zainstalować pakiet StarOffice, wiesz również, do czego służą poszczególne jego składniki. Jest to jeden z najbardziej użytecznych pakietów dla systemu Linux. Z jego pomocą możesz używać Linuxa nie tracąc kompatybilności z systemem Windows.

Konfigurowanie systemu XFree86, który jest niezbędny do tego, by uruchomić pakiet StarOffice, opisane jest w rozdziale 22. „Instalacja i konfiguracja XFree86”.

Jeśli chcesz utworzyć własną stronę WWW w oparciu o dokumenty wygenerowane przez pakiet StarOffice, przejdź do rozdziału 51. „Konfiguracja węzła WWW”.

Narzędzia przeznaczone do obsługi baz danych omówione są w rozdziale 62. „Adabas-D i inne bazy danych”.

Rozdział 64.

Program Lone-Tar firmy Lone Star Software

Tim Parker

W tym rozdziale:

- υ Co to jest Lone-Tar?
- υ Interfejs programu Lone-Tar
- υ Instalacja programu Lone-Tar
- υ Tworzenie kopii zapasowych za pomocą programu Lone-Tar
- υ Weryfikacja plików
- υ Przywracanie wcześniejszych wersji plików
- υ Programy użytkowe i środowisko: dostosowywanie programu Lone-Tar do własnych potrzeb

Jak mogłeś się przekonać czytając rozdział 32. „Podstawy administracji systemem”, tworzenie kopii zapasowych systemu plików może być zajęciem złożonym i irytującym, szczególnie jeśli nie dysponujesz napędem taśmowym albo innym nośnikiem o odpowiedniej pojemności. Jeśli musisz korzystać z dyskietek, Twoja sytuacja jest dość nieciekawa, ponieważ wykonanie pełnej kopii systemu plików zajmie kilkadziesiąt, jeśli nie kilkaset dyskietek. Dlatego większość użytkowników nie posiadających nośników o większej pojemności po prostu rezygnuje z wykonywania pełnych kopii zapasowych.

Wielu użytkowników uważa, że program `tar`, przeznaczony do wykonywania kopii zapasowych jest trudny i nieprzyjemny w obsłudze. Na domiar złego zdarza się, że zgłasza on błędy powodowane przez najróżniejsze przyczyny, zmuszając Cię do wykonania kopii zapasowej od początku. W wielu większych systemach UNIX-owych program `tar` został zastąpiony specjalnie zaprojektowanym programem, opartym na interfejsie graficznym; niestety, w systemie Linux program taki nie jest jeszcze dostępny. Mimo tego istnieje kilka programów mogących zastąpić program `tar`; najlepszym z nich jest chyba program Lone-Tar firmy Lone Star Software.

Co to jest Lone-Tar?

O programie Lone-Tar można myśleć jak o poprawionej wersji programu `tar`. Posiada on wszystkie możliwości, które oferuje `tar`, rozbudowano go ponadto o kilka nowych, których w tym programie zdecydowanie brakowało. Program Lone-Tar nie używa standardowego programu `tar`, choć jego zachowanie jest dość podobne. Lone-Tar jest dostępny w wersjach dla wielu platform UNIX-owych i innych i jest kompatybilny pomiędzy wszystkimi tymi platformami. Można na przykład wykonać kopię zapasową plików w systemie DOS i przywrócić zapisane w niej pliki pod kontrolą systemu Linux.

Podobnie jak program `tar`, Lone-Tar może tworzyć kopie zapasowe plików i przywracać pliki z kopii zapisanych na dyskietkach, taśmach, dyskach twardych i innych nośnikach. Lone-Tar potrafi również zapisywać informacje o wszystkich plikach specjalnych, dowiązaniach (symbolicznych i stałych), plikach wirtualnych i partycjach – możliwości takich nie posiada program `tar`.

Program Lone-Tar umożliwia wykonywanie kopii zapasowych na dwóch nośnikach o różnych pojemnościach, co nie jest łatwe do osiągnięcia za pomocą programu `tar`. Poza tym program Lone-Tar zawiera doskonale działające procedury wykrywania i korekcji błędów, co pozwala na przywrócenie zapisanego systemu plików nawet w sytuacji, gdy zapisane dane zostały w pewnym stopniu uszkodzone. Program `tar` w takiej sytuacji po prostu kończy działanie, pozbawiając kopię zapasową jakiejkolwiek wartości.



Program Lone-Tar wydaje się być bardzo użyteczny, nieprawdaż? Niestety, jest on produktem komercyjnym i należy za niego zapłacić firmie Lone Star Software. W niektórych węzłach FTP dostępna jest wersja demonstracyjna, działająca przez określony czas, która pozwala zapoznać się z możliwościami tego programu. Jeśli uznasz, że program jest tego wart, po upłynięciu okresu próbnego można zakupić jego pełną wersję. Wraz z nią otrzymasz dobrze opracowany podręcznik. Więcej informacji o programie Lone-Tar można uzyskać kontaktując się z firmą Lone Star Software:

13987 W Annapolis Ct.
Mt. Airy, MD 21765
(800) LONE-TAR
cowboy@cactus.com
<http://www.cactus.com>
[ftp.cactus.com](ftp://ftp.cactus.com)

Interfejs programu Lone-Tar

Program Lone-Tar posiada dwa interfejsy. Może być obsługiwany za pomocą systemu menu lub też z wiersza poleceń. Dla zapewnienia kompatybilności, interfejs oparty na

wierszu poleceń jest bardzo podobny do interfejsu programu `tar`. Dzięki temu użytkownik programu `tar`, który zdecydował się używać programu Lone-Tar ze względu na jego większe możliwości, nie musi od nowa przyswajać sobie całego zestawu poleceń. Jak jednak miałeś już okazję się przekonać, zestaw poleceń programu `tar` jest dość dziwny, niewygodny i trudny do opanowania.

Aby przyzwyczaić się do składni programu `tar`, potrzeba dość sporo czasu, więc program Lone-Tar oferuje również o wiele wygodniejszy interfejs obsługiwany za pomocą menu. Wszystkie funkcje programu Lone-Tar dostępne są zarówno z wiersza poleceń, jak i z menu – działają one dokładnie tak samo. Jeśli jednak nie masz doświadczenia w używaniu programu `tar`, o wiele łatwiej będzie Ci posługiwać się systemem menu.

Różnicę pomiędzy obydwoma interfejsami niech zobrazuje składnia polecenia `lone-tar` (bardzo zbliżona zresztą do składni polecenia `tar`):

```
lone-tar [MICrTUXPZ] [bdefhklmnpvFEADVR] [tapefile] [block size]
[compression limit] [0-9] [floppy/tape size] files ...
```

Jeśli pomyliś się przy podawaniu któregoś z parametrów, zarówno `tar` jak i Lone-Tar wygenerują komunikaty o dostępnych opcjach i kodach błędów, przedstawione na rysunku 64.1. Można również wyświetlić te informacje wydając polecenie

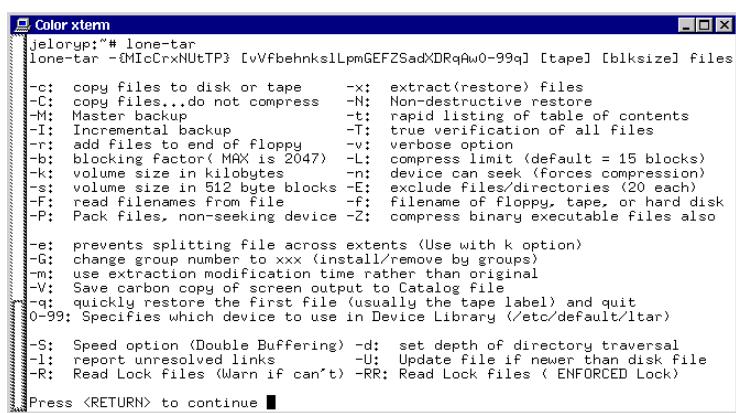
```
lone-tar
```

Natomiast interfejs obsługiwany za pomocą menu, przedstawiony na rysunku 64.2, jest o wiele bardziej przyjazny i łatwiejszy w obsłudze. Każda z pozycji menu głównego, dostępna po wcisnięciu odpowiedniego klawisz, prowadzi do menu podrzędnego.

Wybór interfejsu zależy oczywiście tylko od Ciebie, ale jeśli nie jesteś przyzwyczajony do składni programu `tar`, powinieneś raczej zdecydować się na użycie systemu menu. Weterani systemów UNIX-owych mogą chcieć pozostać przy używaniu wiersza poleceń, ale system menu jest o wiele łatwiejszy w użyciu i pozwala uniknąć błędów typograficznych, dlatego w dalszej części tego rozdziału skupimy się głównie na interfejsie opartym na systemie menu.

Rysunek 64.1.

Pierwsza strona komunikatów generowanych przez program Lone-Tar



```
jeloryp:"># lone-tar
lone-tar -fMlcCrXNUTP3 [vVfbhnksLpmGEFZSadXDRqAw0-99q] [tape] [blksize] files
-c: copy files to disk or tape      -x: extract	restore) files
-C: copy files...do not compress    -N: Non-destructive restore
-M: Master backup                   -t: rapid listing of table of contents
-I: Incremental backup              -T: true verification of all files
-r: add files to end of floppy     -v: verbose option
-b: blocking factor( MAX is 2047)   -l: compress limit (default = 15 blocks)
-k: volume size in kilobytes       -n: device can seek (forces compression)
-s: volume size in 512 byte blocks -E: exclude files/directories (20 each)
-F: read filenames from file      -f: filename of floppy, tape, or hard disk
-P: Pack files, non-seeking device -Z: compress binary executable files also

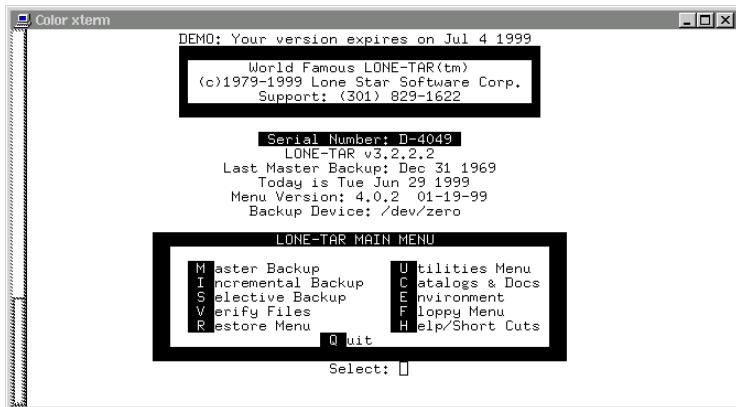
-e: prevents splitting file across extents (Use with k option)
-G: change group number to xxx (install/remove by groups)
-m: use extraction modification time rather than original
-V: Save carbon copy of screen output to Catalog file
-q: quickly restore the first file (usually the tape label) and quit
0-99: Specifies which device to use in Device Library (/etc/default/ltar)

-S: Speed option (Double Buffering) -d: set depth of directory traversal
-l: report unresolved links        -U: Update file if newer than disk file
-R: Read Lock files (Warn if can't) -RR: Read Lock files ( ENFORCED Lock)

Press <RETURN> to continue ■
```

Rysunek 64.2.

Interfejs programu Lone-Tar obsługiwanego za pomocą menu jest o wiele bardziej przyjazny i łatwiejszy w obsłudze



Instalacja programu Lone-Tar

Proces instalowania programu Lone-Tar jest bardzo prosty. Należy zalogować się jako użytkownik `root` i przejść do katalogu `/tmp`. Następnie należy rozpakować wszystkie pliki z płyty CD-ROM lub dyskietki (jeśli posiadasz wersję programu Lone-Tar na dyskietce) polecienniem `tar`. Jeśli na przykład pliki programu Lone-Tar znajdują się na dyskietce w pierwszej stacji dysków, należy wydać polecenia:

```
cd /tmp
tar xvf /dev/rfd0
```

Polecenie to spowoduje odtworzenie wszystkich plików zapisanych na dyskietce w pierwszej stacji dysków (`/dev/rfd0`) i zapisanie ich w katalogu bieżącym.



W niektórych systemach pierwsza stacja dysków nie nazywa się `/dev/rfd0`, ale `/dev/fd0`. Jeśli przy wydawaniu powyższego polecenia system wygeneruje komunikat `Device unknown (nieznane urządzenie)`, należy zamiast wartości `/dev/rfd0` podać `/dev/fd0` i spróbować jeszcze raz.

Jeśli instalujesz pliki z płyty CD-ROM, możesz skopiować odpowiednie pliki używając po prostu polecenia `cp`. Jeśli na przykład pliki programu Lone-Tar są zapisane w katalogu `/lone-tar`, a płyta CD-ROM jest zamontowana w katalogu `/cdrom`, powinieneś wydać polecenia:

```
cd /tmp
cp /cdrom/lone-tar/* .
```

Dokładna postać tego polecenia zależy od położenia plików programu Lone-Tar i punktu zamontowania płyty CD-ROM.

Po skopiowaniu plików do katalogu `/tmp`, można rozpocząć właściwą instalację, wydając polecenie

```
./init.ltar
```

Program `init.ltar`, stworzony przez Lone Star Software, potrafi automatycznie zainstalować wszystkie składniki programu Lone-Tar. Podczas instalacji należy odpowiedzieć na zestaw pytań dotyczących nośnika, na którym zapisywane będą dane i jego pojemności; będzie również możliwość wydrukowania instrukcji obsługi. Instrukcję można wydrukować w każdej chwili wybierając odpowiednie polecenie z menu programu Lone-Tar.

Aby uruchomić menu programu Lone-Tar, należy wydać polecenie

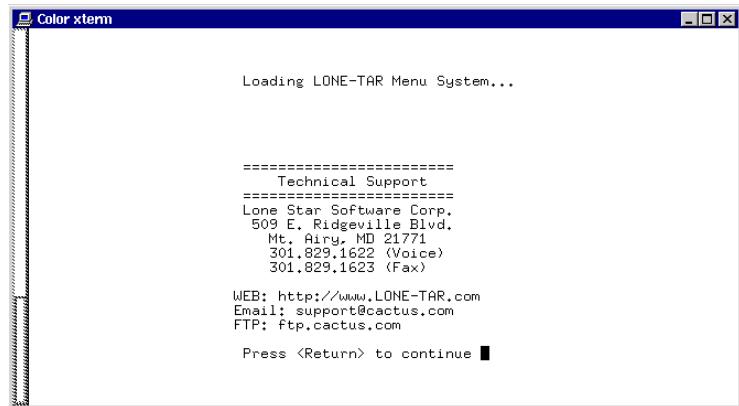
```
ltmenu
```

Na ekranie wyświetlony zostanie komunikat, przedstawiony na rysunku 64.3. Wciśnięcie klawisza `Enter` prowadzi do wyświetlenia menu przedstawionego na rysunku 64.2. Aby uruchomić program Lone-Tar w trybie obsługi z wiersza poleceń, należy wydać polecenie

```
lone-tar
```

które spowoduje wyświetlenie informacji pomocniczych lub też podać po nim wszystkie wymagane opcje.

Rysunek 64.3.
Jeśli program
Lone-Tar został
zainstalowany
prawidłowo,
po uruchomieniu go
w trybie
interaktywnym
powinien zostać
wyświetlony
ekran powitalny



Tworzenie kopii zapasowych za pomocą programu Lone-Tar

Tworzenie kopii zapasowych jest integralną częścią pracy z systemem linuxowym, zarówno w przypadku, gdy jest on używany do poważnej pracy, jak i do zabawy. Powód jest prosty: ponowna instalacja i konfiguracja systemu i wszystkich programów tak, aby przywrócić pierwotny stan rzeczy, jest procesem bardzo długotrwałym i może w niektórych przypadkach prowadzić do błędów. Przywrócenie systemu plików z taśmy lub innego nośnika trwa natomiast tylko kilka minut – choć wymaga systematycznego wyko-

nywania kopii. Jeśli w systemie zapisane są jakieś ważne informacje, regularne kopie zapasowe należy wykonywać również dlatego, że odtworzenie utraconych danych może często okazać się niemożliwe.



Jeśli jesteś zmuszony do używania dyskietek jako nośnika kopii zapasowych, powinieneś z menu programu Lone-Tar wybrać opcję Floppy. Prowadzi ona do menu umożliwiającego tworzenie kopii na dyskietkach, zamiast na taśmach czy dyskach twardych.

Program Lone-Tar pozwala na wykonywanie dwóch rodzajów kopii zapasowych: pełnych oraz przyrostowych. Kopia pełna zawiera dane o całym systemie plików – zapisany jest w niej każdy plik przechowywany w systemie. Kopie przyrostowe tworzy się pomiędzy kolejnymi wykonaniami kopii pełnych i zawierają one dane tylko o tych plikach, które zostały zmienione od czasu wykonania ostatniej kopii. Tworzenie kopii przyrostowej jest szybsze i nie wymaga tak dużo miejsca jak kopia pełna. System Linux orientuje się, które z plików zostały zmodyfikowane, na podstawie atrybutów pliku – dzięki temu tylko te pliki, które zostały utworzone lub zmodyfikowane od czasu utworzenia ostatniej kopii zapasowej, są włączane do kopii przyrostowej. W przypadku wystąpienia awarii należy najpierw przywrócić dane z kopii pełnej, a następnie ze wszystkich kolejnych kopii zapasowych wykonanych od czasu jej utworzenia. Jeśli natomiast potrzebne jest przywrócenie tylko kilku plików, często wystarczy wykorzystanie samej kopii przyrostowej.

Częstość wykonywania kopii pełnych i przyrostowych zależy od wykorzystania systemu, ilości zmienianych codziennie danych i ich ważności. Dla regularnie wykorzystywanego systemu można co tydzień wykonywać kopię pełną na taśmie o dużej pojemności, natomiast kopie przyrostowe mogą być wykonywane każdej nocy.

Jeśli system nie jest wykorzystywany zbyt intensywnie, można na przykład tworzyć kopie pełne co miesiąc, a przyrostowe co tydzień, choć takie rozwiązań nie jest raczej zalecane. W bardziej obciążonych systemach można zrezygnować z kopii przyrostowych i co noc wykonywać pełną kopię systemu plików. Jak się za chwilę przekonasz, jedną z zalet programu Lone-Tar jest możliwość automatyzacji tworzenia kopii zapasowych.

Aby uruchomić tworzenie pełnej kopii zapasowej (nazywanej w programie Lone-Tar *master backup*), w menu głównym wciśnij klawisz m. Zostaniesz zapytany o to, czy nie chcesz pominąć któregoś z systemów plików (jak na rysunku 64.4). Domyslnie program Lone-Tar stworzy kopię zapasową wszystkich plików w systemie, ale możesz zdecydować, że któryś z katalogów nie powinny być kopowane, na przykład katalog, w którym zamontowana jest płyta CD-ROM. Jeśli jest ona zamontowana w katalogu /cdrom, powinieneś poinformować program Lone-Tar o tym, że nie powinien wykonywać kopii tego katalogu (zresztą tworzenie kopii zapasowej płyty CD-ROM nie ma większego sensu). Można również nie wykonywać kopii zamontowanych dysków sieciowych.

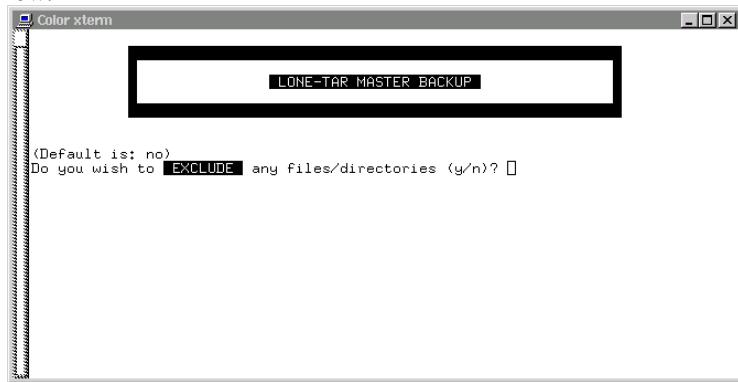
Następnie pojawi się monit o włożenie odpowiedniego nośnika, na przykład taśmy, po czym rozpoczyna się tworzenie kopii zapasowej. Mogą również zostać wyświetlane komunikaty o konieczności wymiany nośnika. Program Lone-Tar sprawdza, czy napęd taśmowy (lub inny) jest gotowy do pracy.

Program Lone-Tar spodziewa się, że każdy nośnik, na którym wykonywana jest kopii zapasowa, będzie oznaczony za pomocą pliku zawierającego informacje, że jest to nośnik przeznaczony do pracy z programem Lone-Tar. Nie trzeba oznaczać kopii w ten

sposób, ale dzięki temu program Lone-Tar ma ułatwione zadanie podczas odzyskiwania zapisanych plików.

Rysunek 64.4.

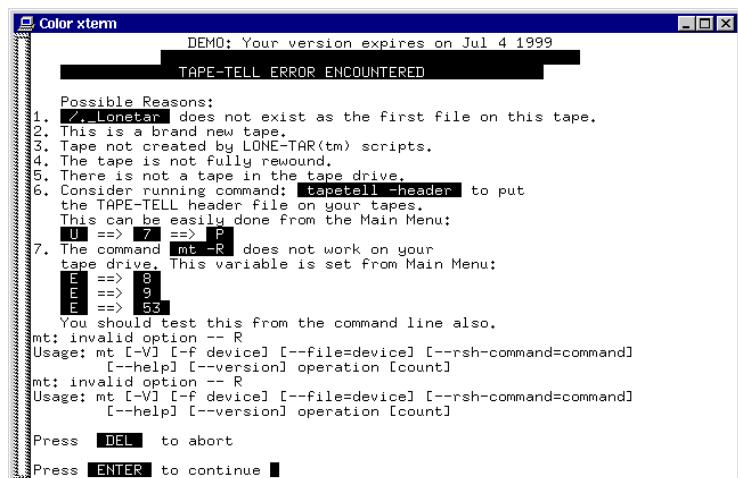
Podczas tworzenia pełnej kopii można zrezygnować z zapisywania informacji o niektórych katalogach



Komunikat o błędzie pokazany na rysunku 64.5 mówi, że program Lone-Tar stwierdził, że w napędzie nie ma taśmy lub też włożona została taśma, która nie została wcześniej oznaczona. Oznaczenie taśmy nie jest konieczne – wciśnięcie klawisza Enter spowoduje rozpoczęcie procesu kopiowania mimo jego braku.

Rysunek 64.5.

Program Lone-Tar wyświetla ostrzeżenie, jeśli stwierdzi, że w napędzie nie ma taśmy lub też taśma nie została odpowiednio oznaczona



Na rysunku 64.6 przedstawione są komunikaty wyświetlane podczas wykonywania kopii zapasowej. Program Lone-Tar wyświetla na ekranie nazwę i kilka informacji o każdym pliku, który jest kopowany.

Jeśli jeden egzemplarz nośnika nie wystarcza dla wykonania kopii, program Lone-Tar poprosi o wymianę nośnika. Przed ponownym przystąpieniem do tworzenia kopii program czeka na wciśnięcie klawisza Enter. Po zakończeniu całego procesu ponownie wyświetlane jest menu główne.

Podczas tworzenia kopii można również kompresować zapisywane pliki, dzięki czemu na jednym egzemplarzu nośnika można zapisać większą ilość danych. Musisz sam zdecydować, czy chcesz używać kompresji – odpowiednie dane można podać podczas

Rysunek 64.6.

Podczas wykonywania kopii zapasowej Lone-Tar wyświetla nazwy plików i dodatkowe informacje o nich, dzięki czemu można ocenić, czy program działa prawidłowo

```
a ./usr/bin/printop, 145 blocks...compressing==== 53 blocks!! (63.26%)
VOL=24403Mb
a ./usr/bin/mh/promter, 121 blocks...compressing==! 55 blocks!! (55.26%)
VOL=24403Mb
a ./usr/bin/mh/pgped, 2 blocks
a ./usr/bin/mh/pgshow, 2 blocks
a ./usr/bin/mh/anno, 125 blocks...compressing==== 57 blocks!! (54.25%)
VOL=24403Mb
a ./usr/bin/mh/burst, 124 blocks...compressing==! 58 blocks!! (54.46%)
VOL=24403Mb
a ./usr/bin/mh/comp, 167 blocks...compressing==== 99 blocks!! (40.55%)
VOL=24403Mb
a ./usr/bin/mh/dist, 167 blocks...compressing==== 99 blocks!! (40.54%)
VOL=24403Mb
a ./usr/bin/mh/folder, 128 blocks...compressing==! 59 blocks!! (54.12%)
VOL=24403Mb
a ./usr/bin/mh/folders, 128 blocks linked to ./usr/bin/mh/folder
a ./usr/bin/mh/forw, 175 blocks...compressing==! 92 blocks!! (47.38%)
VOL=24403Mb
a ./usr/bin/mh/inc, 171 blocks...compressing==== 103 blocks!! (39.87%)
VOL=24403Mb
a ./usr/bin/mh/mark, 120 blocks...compressing==! 0
```

instalacji. Zalet tego rozwiązana nie trzeba chyba nikomu tłumaczyć: większa ilość danych zajmuje mniej miejsca. Ma ono niestety również wady: wykonanie kopii zapasowej trwa dłużej, a skompresowane pliki mogą być odtworzone tylko za pomocą programu Lone-Tar. Nieskompresowana kopia zapasowa może natomiast być odczytana zarówno przez program Lone-Tar, jak i program tar, co jest bardzo ważne przy przenoszeniu kopii zapasowych pomiędzy różnymi systemami.

Czas potrzebny do utworzenia kopii zapasowej zależy od wielu czynników, głównie od wielkości systemu plików, prędkości urządzenia, do którego wysyłane są dane, oraz obciążenia systemu. Jeśli posiadasz szybki komputer oraz urządzenie pozwalające na zapis dużej ilości danych, program Lone-Tar potrafi wykonać kopię w o wiele krótszym czasie, niż program tar. Prędkość jest ograniczana głównie przez prędkość samych urządzeń, na których wykonywane są kopie. Dla przykładu napęd DAT oparty na interfejsie SCSI jest o wiele szybszy niż napęd QIC, wykorzystujący kasety z taśmą. Po wykonaniu kilku kopii przyzwyczaisz się do czasu, jakiego wymaga ten proces. Jeśli potrzeba na to naprawdę długiego czasu, warto uruchamiać tworzenie kopii zapasowych w nocy, w czasie gdy śpisz, czy też wtedy, gdy nie używasz systemu. Weź pod uwagę fakt, że wykonanie pełnej kopii systemu trwa zwykle od godziny do kilku godzin, zależnie od prędkości napędu.



Upewnij się, że odpowiednio opisłeś nośnik, umieszczając na nim przynajmniej informacje o dacie wykonania kopii i jej rodzaju. Informacje te powinny być jasne i czytelne, ponieważ nigdy nie wiadomo, kiedy będą musiały zostać odczytane.

Proces wykonywania kopii przyrostowych przebiega prawie tak samo, jak tworzenie kopii pełnych. Ponieważ jednak kopie przyrostowe zwykle są o wiele mniejsze niż kopie pełne, ich wykonywanie trwa również o wiele krócej. Powinieneś przyzwyczaić się do tego, że kopie przyrostowe tworzy się codziennie i dodatkowo wtedy, gdy zapisujesz dane, na których utratę nie możesz sobie pozwolić. Lepiej spędzić 10 minut na wykony-

waniu kopii zapasowej niż potem stracić kilka godzin na odtwarzanie ostatniego rozdziału swojej najnowszej książki.

Wykonanie częściowej kopii systemu (ang. *selective backup*) jest możliwe po wcisnięciu w menu głównym klawisza **s**. Taka kopia zawiera tylko wybrane fragmenty systemu plików. Po wybraniu plików, które mają wchodzić w jej skład, proces tworzenia kopii przebiega w zwykły sposób.

Weryfikowanie plików

Opcja weryfikacji utworzonej kopii pozwala na dodatkowe podniesienie poziomu bezpieczeństwa danych. Pozwala ona na odczytanie każdego zapisanego pliku i porównanie jego zawartości z plikiem oryginalnym. Dzięki temu można wychwycić wszystkie problemy z nośnikiem zanim jeszcze będzie za późno.

Dobrym zwyczajem jest weryfikacja plików po wykonaniu każdej kopii zapasowej. Jest to szczególnie ważne po utworzeniu pełnej kopii systemu plików.

Pamiętaj o tym, że niektóre pliki mogą zostać zmodyfikowane w czasie pomiędzy utworzeniem ich kopii a rozpoczęciem weryfikacji, w zależności od tego, czy system jest w tym czasie używany, czy nie. Również niektóre programy działające przez cały czas, na przykład obsługujące pocztę czy rejestrowanie informacji mogą zmieniać zawartość niektórych plików w czasie wykonywania kopii i po zakończeniu tego procesu. Program Lone-Tar podczas weryfikacji stwierdza, że pliki te nie są takie same, jak zapisane i zgłasza błędy. Powinieneś uważnie przejrzeć wygenerowany raport, dzięki czemu będziesz mógł zorientować się, które komunikaty są istotne, a które pojawiły się ze względu na nieprzerwane działanie systemu.

Przywracanie wcześniejszych wersji plików

Kiedy musisz przywrócić wcześniejszą wersję pliku, katalogu czy całego systemu plików, musisz najpierw odszukać właściwą kopię zapasową. Jeśli stosowałeś zarówno kopie pełne, jak i przyrostowe, musisz odnaleźć ostatnią kopię pełną i wszystkie kopie przyrostowe wykonane od czasu jej utworzenia. Jeśli szukasz tylko wersji kilku usuniętych przez przypadek plików, musisz wiedzieć, na której z taśm możesz je znaleźć.

Aby uruchomić proces przywracania wcześniejszych wersji plików, z menu głównego programu Lone-Tar wybierz opcję **R** (ang. *restore*). Spowoduje to wyświetlenie menu, które zostało przedstawione na rysunku 64.7. Nazwy poszczególnych opcji nie pozostawiają wątpliwości co do ich znaczenia. Aby na przykład przywrócić wszystkie pliki zapisane na taśmie, należy wybrać opcję

`Restore entire tape to hard disk.`

Pozostałe opcje menu Restore pozwalają na przywrócenie wybranych katalogów i plików, w oparciu o podane nazwy (możliwe jest również używanie symboli wieloznacznych). Można również utworzyć listę katalogów i przywrócić je wszystkie za jednym zamachem.

Inną możliwością jest wybranie katalogów, których zawartości nie należy odtwarzać; wówczas przywrócona zostanie poprzednia wersja wszystkich pozostałych katalogów.

Rysunek 64.7.

*Menu Restore
programu Lone-Tar*



Po określeniu, które pliki i katalogi mają zostać odtworzone, program Lone-Tar prosi o włożenie odpowiedniego nośnika do napędu, po czym rozpoczyna odtwarzanie. Podobnie jak podczas wykonywania kopii, również teraz na ekranie wyświetlane są informacje o poszczególnych odtwarzanych plikach.



Podczas odtwarzania nawet pojedynczych plików czy katalogów, program Lone-Tar musi przeszukać całą taśmę, by je znaleźć, co może zabrać dłuższą chwilę, zależnie od prędkości urządzenia. Nie przejmuj się więc tym, że przez kilka minut nic się nie dzieje. Diody kontrolne urządzenia, w którym znajduje się nośnik, powinny sygnalizować, że jest ono używane.

Jeśli do odtworzenia plików potrzeba więcej niż jednego egzemplarza nośnika, program Lone-Tar poprosi o jego wymianę – cykl taki będzie powtarzany aż do zakończenia odtwarzania.

Jeśli chcesz przywrócić pliki zarówno z kopii pełnej, jak i późniejszych kopii przyrostowych, powinieneś powtórzyć cały proces dla każdej z kopii. Jeśli na przykład usunąłeś przez przypadek cały katalog, możesz przywrócić jego zawartość z ostatniej kopii pełnej, a następnie odtworzyć wszystkie wprowadzone później modyfikacje powtarzając cały proces dla każdej z kopii przyrostowych. Poszczególne uaktualnienia trzeba przeprowadzać ręcznie, za każdym razem korzystając z opcji dostępnych w menu Restore.

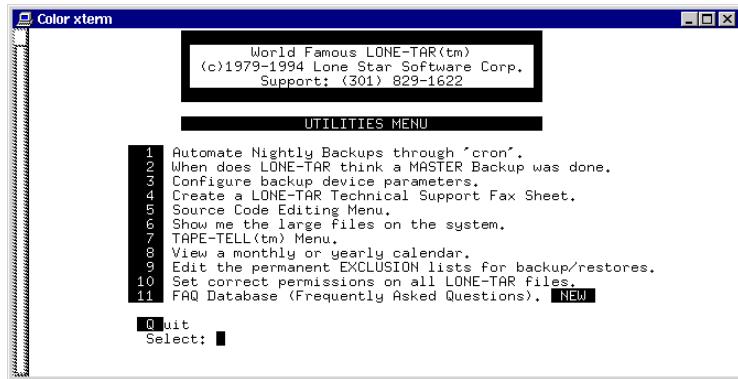
Po zakończeniu odtwarzania plików, ponownie wyświetlana jest zawartość menu głównego. Należy sprawdzić, czy pliki trafiły na swoje miejsce i czy wszystko wygląda prawidłowo.

Moduł Tape-Tell programu Lone-Tar potrafi dostarczyć kilku informacji o czasie ostatniego użycia taśmy. Działa on w oparciu o plik programu Lone-Tar, który może zostać zapisany na początku nośnika. Problem ten przedstawiliśmy w podrozdziale dotyczącym wykonywania kopii zapasowych.

Programy użytkowe i środowisko: dostosowywanie programu Lone-Tar do własnych potrzeb

Menu Utilities (programy użytkowe) programu Lone-Tar, pokazane na rysunku 64.8, zawiera pewną liczbę przydatnych poleceń i funkcji. Większość opcji tego menu posiada nazwy nie pozostawiające wątpliwości co do ich przeznaczenia, ale dla zwykłych użytkowników przydatnych jest tylko kilka z nich. Warto na przykład co jakiś czas sprawdzać datę wykonania ostatniej pełnej kopii zapasowej, co może przypomnieć o tym, że najwyższa pora wykonać nową kopię.

Rysunek 64.8.
*Menu Utilities
programu Lone-Tar*



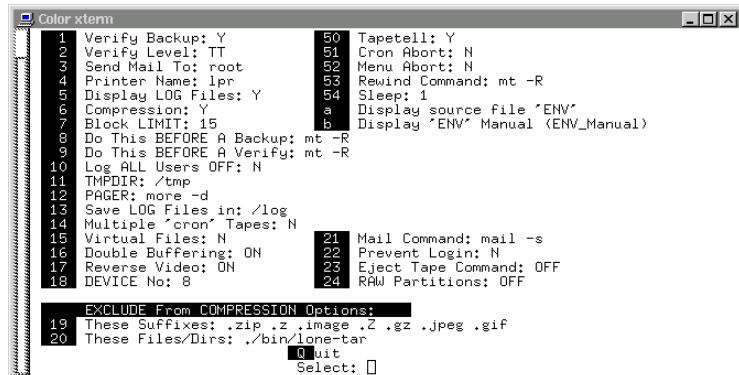
Jeśli kopie zapasowe mają być wykonywane automatycznie, warto skorzystać z możliwości współpracy z programem `cron`. Program ten został przedstawiony w jednym z wcześniejszych rozdziałów – pozwala on na wykonywanie poleceń o określonych porach, bez konieczności fizycznego przebywania przy komputerze. Dzięki takiemu rozwiązaniu można z łatwością zlecić wykonywanie kopii zapasowych na przykład każdej nocy, dwa razy w tygodniu czy w dowolnych innych porach. Nie trzeba przy tym znać szczegółów obsługi programu `cron` – program Lone-Tar sam zajmie się wprowadzeniem odpowiednich informacji do plików konfiguracyjnych.

Menu Utilities pozwala również zmienić urządzenie, na którym zapisywane będą kopie zapasowe, oraz szczegóły konfiguracji, więc jeśli posiadasz więcej niż jedno urządzenie przeznaczone do tworzenia kopii zapasowych, możesz z łatwością przełączać się pomiędzy nimi. Takie rozwiązanie jest szczególnie przydatne, jeśli do wykonywania pełnych kopii systemu chcesz wykorzystywać inne urządzenie, niż do tworzenia kopii przyrostowych.

Menu Environment (środowisko), przedstawione na rysunku 64.9, zawiera listę wszystkich ustawień konfigurujących działanie programu Lone-Tar. Wiele z tych wartości zostało ustalonych podczas instalacji tego programu, ale również tu można je w razie potrzeby zmodyfikować.

Rysunek 64.9.

*Menu Environment
programu Lone-Tar
pozwala
na dostosowanie
działania programu
do własnych potrzeb*



Podsumowanie

Program Lone-Tar ma również wiele możliwości, o których w tym rozdziale nawet nie wspomnialiśmy – odsyłamy więc do lektury dokumentacji dostępnej w sieci Internet. Używanie tego programu do tworzenia kopii zapasowych i odtwarzania wcześniejszych wersji plików jest łatwe i szybkie, ponieważ nie trzeba przejmować się skomplikowaną składnią poleceń i uważać na pomyłki typograficzne. Powinieneś wypróbować wersję demonstracyjną tego programu, a przekonasz się, jak łatwo jest ją obsługiwać.

Jeśli chcesz dowiedzieć się czegoś więcej o montowaniu systemów plików zapisanych na dyskietkach czy płytach CD-ROM, przejdź do rozdziału 33. „Urządzenia”.

Tworzenie kopii zapasowych tak, by zabezpieczyć się przed wszystkimi sytuacjami awaryjnymi, jest omówione bardziej szczegółowo w rozdziale 45. „Kopie zapasowe”.

Jeżeli chcesz uzyskać dodatkowe informacje o tworzeniu kopii zapasowych w ciągu nocy, przejdź do rozdziału 46. „cron i at”.

Część dziewiąta

Dodatki

W tej części:

- υ Węzły FTP i listy dyskusyjne poświęcone Linuxowi
- υ Komercyjni dystrybutorzy Linuxa
- υ Projekt dokumentacji Linuxa
- υ Publiczna licencja GNU
- υ Informacje o prawach autorskich
- υ Zawartość płyt CD-ROM

Dodatek A

Węzły FTP i listy dyskusyjne poświęcone Linuxowi

W tym dodatku:

- υ Węzły FTP
- υ Systemy BBS (Bulletin Board Systems)
- υ Listy dyskusyjne

Jeśli posiadasz dostęp do Internetu, czy to bezpośredni, czy też za pośrednictwem usługi internetowego, możesz skorzystać z oferty dodatkowych źródeł oprogramowania i informacji dotyczących systemu Linux. Dwa najpopularniejsze z nich to węzły FTP oraz listy dyskusyjne o tematyce linuxowej.

Jeśli nie posiadasz dostępu do Internetu, możesz zdobyć informacje i oprogramowanie również z innych źródeł, na przykład z węzłów BBS czy płyt CD-ROM publikowanych przez firmy zajmujące się dystrybucją darmowego oprogramowania.

Węzły FTP

FTP to protokół pozwalający na dostęp do zdalnych systemów plików i ładowanie zapisanych w nich danych. Jest on łatwy w użyciu i umożliwia szybkie uaktualnianie posiadanych programów osobom posiadającym dostęp do sieci Internet.

Użytkownicy nie mający możliwości używania protokołu FTP, ale mający dostęp do poczty elektronicznej, mogą skorzystać z programu `ftpmail`, mającego podobne możliwości jak program FTP.

Co to jest FTP?

FTP (ang. *File Transfer Protocol*, protokół transmisji plików) to jeden z protokołów rodziny TCP/IP. Protokoły TCP/IP są powszechnie używane w wielu sieciach lokalnych i Internecie. Prawie wszystkie systemy UNIX-owe używają protokołów TCP/IP.

Protokół FTP służy do przesyłania plików pomiędzy komputerami obsługującymi system TCP/IP. Dostępne są również podobne do FTP programy przeznaczone do współpracy z innymi protokołami.

Aby można było używać protokołu FTP, na obu końcach połączenia musi być uruchomiony odpowiedni program. Aby załadować pliki z systemu zdalnego, należy uruchomić oprogramowanie klienta FTP i zlecić mu połączenie się z odpowiednim oprogramowaniem po stronie serwera.

W Internecie działa wiele węzłów FTP oferujących tak zwane archiwa FTP. Są to komputery skonfigurowane w ten sposób, aby umożliwić wszystkim chętnym ładowanie oprogramowania. Niektóre węzły FTP są węzłami zwierciadlanymi, co oznacza, że ich zawartość jest kopią innego węzła, nazywanego węzłem pierwotnym. W takim przypadku należy łączyć się z tym serwerem, który oferuje większą prędkość transmisji, ponieważ dostępne oprogramowanie jest dokładnie takie samo jak w węźle pierwotnym.

Zwykle po uzyskaniu połączenia należy się zalogować, co oznacza, że trzeba być użytkownikiem systemu zdalnego (czyli posiadać własny identyfikator oraz hasło). Ponieważ niemożliwe jest przyznanie osobnego identyfikatora i hasła każdemu, kto chce skorzystać z ogólnodostępnych zasobów, w wielu systemach używa się tzw. anonimowego FTP. Rozwiążanie to pozwala każdemu zalogować się do systemu po podaniu identyfikatora `ftp` lub `anonymous` czy `guest`, bądź to bez podania hasła, bądź też po podaniu swojego adresu e-mailowego.

Łączenie się z systemem zdalnym i ładowanie plików

Połączenie się z systemem zdalnym za pomocą FTP jest łatwe. Założymy, że masz dostęp do Internetu (bezpośredni lub za pośrednictwem usługodawcy internetowego); w takim przypadku należy po prostu uruchomić program obsługujący połączenia FTP, podając jako argument wywołania nazwę węzła, z którym chcesz się połączyć. Jeśli więc jesteś połączony z Internetem, wystarczy, że wydasz następujące polecenie:

```
ftp sunsite.unc.edu
```

Spowoduje ono uruchomienie klienta FTP i próbę nawiązania połączenia z systemem `sunsite.unc.edu`. Po nawiązaniu połączenia (zakładając, że system zdalny pozwala na połączenia FTP) system zdalny oczekuje na podanie identyfikatora użytkownika. Jeśli

dozwolone są anonimowe połączenia FTP, zwykle pojawia się odpowiednia informacja. Oto przykład pochodzący z węzła

sunsite.unc.edu:

```
ftp sunsite.unc.edu
331 Guest login ok, send your complete e-mail address as password.
Enter username (default: anonymous): anonymous
Enter password [tparker@tpci.com]:
|FTP| Open
230-           WELCOME to UNC and SUN's anonymous ftp server
230-           University of North Carolina
230-           Office FOR Information Technology
230-           sunsite.unc.edu
230 Guest login ok., access restrictions apply.
FTP>
```

Po zakończeniu procesu logowania wyświetlana jest zachęta `FTP>`. Oznacza to, że system jest gotów do przyjmowania poleceń FTP. W niektórych systemach dodatkowo wyświetlane jest krótki komunikat zawierający instrukcje dotyczące ładowania plików, ograniczeń nałożonych na Ciebie jako użytkownika anonimowego oraz położenia bardziej przydatnych plików. Przykładowe komunikaty z węzła `sunsite.unc.edu` mają postać:

```
To get a binary file, type: BINARY and then: GET "File.Name" newfilename
To get a text file, type:      ASCII      and then: GET "File.Name"
newfilename
Names MUST match upper, lower case exactly. Use the "quotes" as shown.
To get a directory, type: DIR. To change directory, type: CD "Dir.Name"
To read a short text file, type GET "File.Name" TT
For more, type HELP or see FAQ in gopher.
To quit, type EXIT or Control-Z.

230- If you e-mail to info@sunsite.unc.edu you will be sent help
□ information
230- about how to use the different services sunsite provides.
230- We use the Wuarchive experimental ftpd. If you "get"
□ <directory>.tar.Z
230- or <file>.Z it will compress and/or tar it on the fly. Using ".gz"
□ instead
230- of ".Z" will use the GNU zip (/pub/gnu/gzip*) instead, a superior
230- compression method.
```

Po zalogowaniu się do systemu zdalnego można przeglądać zawartość katalogów i poruszać się po nich używając poleceń znanych z systemu Linux. Na przykład, aby wyświetlić zawartość bieżącego katalogu, należy wydać polecenie `ls` (niektóre wersje FTP obsługują również jego DOS-owy odpowiednik, `dir`). Do zmiany katalogu bieżącego służy polecenie `cd`. Jeśli chcesz przejść do katalogu nadrzędnego (czyli tego, w którym zawarty jest katalog bieżący), wydaj polecenie `cd ..` lub `cdup`. Podczas używania FTP nie są dostępne żadne skróty klawiaturowe, co oznacza, że trzeba wpisywać nazwy plików i katalogów w pełnej i poprawnej postaci.

Kiedy znajdziesz w systemie zdalnym jakiś plik, który chciałbyś załadować do swojego komputera, użyj polecenia `get`, którego argumentem jest nazwa tegoż pliku:

```
get "plik1.txt"
```

Polecenia `get` (załaduj, pobierz) i `put` (wyślij) mają punkt odniesienia w komputerze lokalnym, co oznacza, że wydając polecenie `get`, każesz oprogramowaniu pobrać plik z serwera i umieścić go w swoim komputerze, natomiast polecienniem `put` wysyłasz pliki ze swojego komputera do serwera (mamy tu sytuację dokładnie odwrotną, niż w przypadku programu `telnet`, w którym polecenia są odniesione do komputera zdalnego). Ważne jest, by dobrze zapamiętać, które polecenie działa w którą stronę, w przeciwnym przypadku może zdarzyć się, że niechcący pozbędziesz się jakichś ważnych danych.

Cudzysłów otaczający nazwy plików w większości wersji FTP nie jest wymagany, ale zabezpiecza przed złym interpretowaniem polecenia przez serwer (konkretnie przed interpretacją znaków zastrzeżonych powłoki), więc stosowanie go jest dobrą praktyką.

FTP pozwala na przesyłanie plików w dwóch trybach: `binary` (binarnym) oraz `ASCII` (pliki tekstowe). Niektóre systemy przełączają się pomiędzy nimi automatycznie po rozpoznaniu pliku binarnego, ale nie należy zakładać, że jest tak zawsze. Dla pewności zawsze warto ustawić odpowiedni tryb samodzielnie. Aby załączyć tryb binarny (wymagany do ładowania plików wykonywalnych), wydaj polecenie

```
binary
```

Do trybu tekstowego możesz wrócić wydając polecenie `ASCII`. Ponieważ jednak węzły FTP zwykle odwiedza się po to, by załadować jakieś archiwum, nowe wersje programów `ftp`, warto zawsze używać trybu binarnego. Pliki binarne przesłane w trybie ASCII nie będą się nadawały do użycia. Tryb ASCII pozwala tylko na przesyłanie znaków wchodzących w skład standardu ASCII (kodowanych za pomocą siedmiu bitów), w plikach binarnych natomiast mogą występować również kody nie należące do tego standardu. Przesłanie pliku ASCII w trybie `binary` nie narusza jego zawartości, choć w niektórych sytuacjach transmisja może być bardziej narażona na zakłócenia.

Po wydaniu polecenia `get` system zdalny przesyła dane do Twojego komputera, wyświetlając krótkie podsumowanie po zakończeniu tej operacji. W trakcie przesyłania danych nie jest wyświetlany żaden wskaźnik postępu, więc w przypadku większych plików powinieneś uzbroić się w cierpliwość.

```
FTP> get "file1.txt"
200 PORT command successful.
150 BINARY data connection for FILE1.TXT (27534 bytes)
226 BINARY Transfer complete.
27534 byte received in 2.35 seconds (12Kbytes/s).
```

Aby zakończyć pracę programu `ftp`, wydaj polecenie `quit` lub `exit`. Oba te polecenia powodują zamknięcie sesji w systemie zdalnym, a następnie zakończenie działania programu klienta.

Używanie programu `ftpmail`

Jeśli nie masz dostępu do usług FTP, natomiast masz możliwość korzystania z poczty elektronicznej (e-mail), nadal masz szansę na ładowanie oprogramowania poprzez sieć. Rozwiązanie to przydatne jest w systemach nie pozwalających na bezpośrednią połą-

czenia FTP, ale dopuszczających obsługę poczty, na przykład udostępniających konta UUCP. Aby załadować pliki z węzła FTP za pośrednictwem poczty elektronicznej, użyj programu `ftpmail`.

Wspomniany wcześniej węzeł `sunsite.unc.edu` jest jednym z głównych archiwów FTP poświęconych Linuxowi obsługujących `ftpmail` (wszystkie wymienione w tym dodatku węzły FTP również umożliwiają połączenia za pomocą `ftpmail`). Jeśli chcesz otrzymać instrukcję obsługi tego programu, wyślij wiadomość do użytkownika `ftpmail` w dowolnym z tych systemów (na przykład do `ftpmail@sunsite.unc.edu`). Treść tego listu powinna zawierać jedynie słowo `help`.

Po otrzymaniu Twojego zgłoszenia, `ftpmail` prześle Ci instrukcje dotyczące korzystania z tej usługi. W większości przypadków musisz po prostu wpisać polecenia, jakie wydałbyś w programie `ftp`, jako treść swojego listu. Na przykład, by otrzymać informację o zawartości katalogu `/pub/Linux`, wyślij wiadomość o następującej treści:

```
open sunsite.unc.edu
cd /pub/Linux
ls
quit
```

Jeśli chcesz załadować plik `README` za pośrednictwem poczty elektronicznej, wyślij wiadomość o treści:

```
open sunsite.unc.edu
cd /pub/Linux
binary
get README
quit
```

`ftpmail` jest wolniejszy od `ftp`, ponieważ trzeba czekać, aż e-mail pokona drogę do serwera docelowego, zostanie zinterpretowany przez program `ftpmail`, odpowiedź zostanie odpowiednio sformatowana i w końcu wysłana do Ciebie. Mimo wszystko jest to dobry sposób przesyłania plików dla osób nie mających dostępu do FTP. Ma on również tę zaletę, że nie wymaga logowania się do serwera, jeśli chcesz tylko obejrzeć zawartość katalogów. Może to być użyteczne, gdy chcesz okresowo sprawdzać, czy nie pojawiło się tam nowe oprogramowanie.

Węzły FTP poświęcone Linuxowi

Lista węzłów FTP udostępniających Linuxa powoli ulega zmianom, niemniej jednak adresy zebrane poniżej były dostępne w chwili, gdy książka ta szła do druku. Wiele z nich to kopie innych serwerów (ang. *mirrors*), mających dokładnie taką samą zawartość jak serwery główne (ang. *home sites*).

Węzeł, który znajduje się najbliżej Twojego miejsca zamieszkania, może być zidentyfikowany przez identyfikator kraju na końcu adresu (`uk` – Wielka Brytania, `fr` – Francja, `pl` – Polska itp.). Większość wersji FTP pozwala używać zarówno nazwy domenowej, jak i adresu IP serwera, choć nazwa nie może być poprawnie zinterpretowana przez lokalną bramkę internetową, użycie adresu IP jest najlepszą metodą.

Serwerami pierwotnymi są: `tsx-11.mit.edu`, `sunsite.unc.edu`, oraz `nic.funet.fi`. To właśnie w nich najpierw dostępne są nowe wersje programów. Większość serwerów przedstawionych w powyższej tabeli to kopie serwerów pierwotnych.

Tabela A.1. Węzły FTP zawierające oprogramowanie dla Linuksa i ich adresy IP

Nazwa węzła	Adres IP	Katalog
<code>tsx-11.mit.edu</code>	18.172.1.2	/pub/linux
<code>sunsite.unc.edu</code>	152.2.22.81	/pub/Linux
<code>nic.funet.fi</code>	128.214.6.100	/pub/OS/Linux
<code>ftp.mcc.ac.uk</code>	130.88.200.7	/pub/linux
<code>ftp.informatik.rwth-aachen.de</code>	131.226.255.3	/pub/Linux
<code>ftp.dfv.rwth-aachen.de</code>	137.226.4.111	/pub/linux
<code>ftp.ibp.fr</code>	132.227.60.2	/pub/linux
<code>ftp.uu.net</code>	137.39.1.9	/systems/unix/linux
<code>Wuarchive.wustl.edu</code>	128.252.135.4	/systems/linux
<code>ftp.win.tue.nl</code>	131.155.70.19	/pub/linux
<code>ftp.stack.urc.tue.nl</code>	131.155.140.128	/pub/linux
<code>ftp.ibr.cs.tu-bs.de</code>	134.169.34.15	/pub/os/linux
<code>ftp.denet.dk</code>	129.142.6.74	/pub/OS/linux

Grupy dyskusyjne w sieci Usenet

Usenet to zbiór list (grup) dyskusyjnych (ang. *newsgroups*), dostępnych dla użytkowników Internetu. Istnieje ponad 10000 różnych grup dyskusyjnych, generujących codziennie setki megabajtów wiadomości. Kilka spośród tych grup (obejmujących każdy chyba temat) poświęconych jest Linuxowi.

Dostęp do list dyskusyjnych można uzyskać za pomocą odpowiedniego oprogramowania, podłączając się bezpośrednio do Internetu lub też do lokalnego serwera grup dyskusyjnych. Inne sieci, takie jak CompuServe, America Online i Delphi, również umożliwiają korzystanie z grup dyskusyjnych. Pozwalają na to także w pewnym zakresie systemy BBS.

Grupy dyskusyjne podzielone są na trzy kategorie: główne, dostępne dla wszystkich użytkowników, lokalne, zazwyczaj obejmujące tylko dany region geograficzny, oraz inne, które nie mogą być obsługiwane przez wszystkie serwery ze względu na panujące w nich

reguły. Głównymi grupami dyskusyjnymi poświęconymi Linuxowi, działającymi w chwili pisania tej książki, są:

- ⦿ `comp.os.linux.admin` – poświęcona administrowaniu systemami linuxowymi,
- ⦿ `comp.os.linux.advocacy` – tu można wygłosić swoją opinię na tematy związane z Linuxem,
- ⦿ `comp.os.linux.announce` – ogłoszenia istotne dla społeczności linuxowej,
- ⦿ `comp.os.linux.answers` – pytania i rozwiązania problemów dotyczących Linuxa,
- ⦿ `comp.os.linux.development` – ogólnie o pracach nad Linuxem,
- ⦿ `comp.os.linux.development.apps` – o pracach nad aplikacjami linuxowymi,
- ⦿ `comp.os.linux.development.system` – o pracach nad systemem operacyjnym,
- ⦿ `comp.os.linux.hardware` – skupia się na obsłudze urządzeń zewnętrznych w systemie Linux,
- ⦿ `comp.os.linux.help` – pytania i porady na temat Linuxa,
- ⦿ `comp.os.linux.misc` – tematy dotyczące Linuxa nie zawierające się w tematyce innych grup,
- ⦿ `comp.os.linux.networking` – poświęcona w całości problemom sieci,
- ⦿ `comp.os.linux.setup` – o problemach związanych z instalacją i konfiguracją Linuxa.

Te grupy dyskusyjne powinny być osiągalne we wszystkich węzłach sieci Usenet, chyba że z jakichś powodów dostępu do nich zabroni administrator systemu.

Inne grupy poświęcone Linuxowi często się zmieniają, głównie dlatego, że obejmują zasięgiem niewielki region albo są zdominowane przez osoby o bardzo jednostronnych poglądach; szczególnie grupy `.alt` (ang. *alternate* – inne) mają tendencję do pojawiania się i znikania. Jedną z takich grup aktywnych podczas pisania tej książki jest `alt.uu.comp.os.linux.questions`.

Istnieją również grupy regionalne, które nie podlegają powszechnej dystrybucji, oraz grupy w których używa się języka innego niż angielski. Oto przykłady takich grup:

```
dc.org.linux-users  
de.comp.os.linux  
fr.comp.os.linux  
tn.linux
```

Jeżeli masz dostęp do sieci Usenet, przeglądaj regularnie informacje o powstawaniu nowych i zakończeniu działalności istniejących grup o tematyce związanej z Linuxem.

Serwery pozwalające na dostęp do sieci Usenet zwykle zapewniają też aktualną listę działających grup dyskusyjnych, którą można łatwo przeglądać.

Większość informacji w grupach dyskusyjnych poświęconych Linuxowi dotyczy problemów i zagadnień związanych z instalacją, konfiguracją, administracją i ogólnie użytkowaniem tego systemu operacyjnego. Przewija się przez nie wiele wartościowych informacji, warto więc przeglądać je regularnie. Najciekawsze z wiadomości dotyczących danego wątku są często zbierane razem i udostępniane w formie archiwów w węzłach FTP.

Dodatek B

Komercyjni dystrybutorzy Linuxa

W tym dodatku:

- υ Dystrybucja Debian
- υ Płyty CD-ROM Yggdrasil Plug-and-Play oraz Biblia Linuxa
- υ Linux na płytach CD-ROM firmy Nascent
- υ CD-ROM Unifix 1.02
- υ InfoMagic Developer's Resource CD-ROM kit
- υ Linux Quaterly CD-ROM
- υ Linux Systems Labs
- υ Sequoia International Motif Development Package
- υ Takelap Systems Ltd.
- υ Trans-Ameritech Linux Plus BSD CD-ROM
- υ Caldera OpenLinux

W tym dodatku przedstawiamy listę firm, które zajmują się komercyjną dystrybucją systemu Linux. W dodatku A „Węzły FTP i listy dyskusyjne poświęcone Linuxowi” można znaleźć listę węzłów FTP, w których system Linux jest dostępny za darmo. Nie-wątpliwą zaletą wynikającą z zakupu Linuxa od dostawcy komercyjnego jest to, że otrzymujemy od razu całe mnóstwo oprogramowania, dzięki czemu nie trzeba zwracać sobie głowy jego zdobywaniem. Poniższa lista jest również prowadzona przez redakcję miesięcznika Linux Journal:

Linux Journal
P.O. Box 85867
Seattle, WA 98145-1867
Telefon: (206) 527-3385
Fax: (206) 527-2806

Plik *Linux Distribution-HOWTO* także zawiera aktualne informacje o firmach, które kompletują pakiety oprogramowania dla systemu Linux. Lista ta jest utrzymywana przez Matta Welsha, maw@sunsite.unc.edu. Plik ten można załadować z węzła tsx-11.mit.edu, gdzie jest dostępny pod nazwą </pub/linux/docs/HOWTO/Distribution-howto>.

Dystrybucja Debian

The Debian Linux Association
Station 11
P.O. Box 3121
West Lafayette, IN 47906

Wersje testowe programów dostępne są w węźle sunsite.unc.edu, w katalogu </pub/Linux/distributions/debian>.

Płyty CD-ROM Yggdrasil Plug-and-Play oraz Biblia Linuxa

Yggdrasil Computing, Incorporated
4880 Stevens Creek Blvd., Suite 205
San Jose, CA 95129-1034

E-mail: info@yggdrasil.com
WWW: <http://www.yggdrasil.com>
FTP: <http://ftp.yggdrasil.com>
Telefon: (800) 261-6630, (408) 261-6630
Fax: (408) 261-6631

Linux na płytach CD-ROM firmy Nascent

Nascent Technology
Linux from Nascent CD-ROM
P.O. Box 60669
Sunnyvale, CA 49088-0669
Telefon: (408) 737-9500
Fax: (408) 241-9390
E-mail: nascent@netcom.com

CD-ROM Unifix 1.02

Unifix Software GmbH
Postfach 4918
D-38039 Braunschweig, Germany
Telefon: +49 (0)531 515161
Fax: +49 (0)531 515162

Fintronic Linux Systems

Fintronic USA, Inc.
1360 Willow Rd., Suite 205
Menlo Park, CA 94025
Telefon: (415) 325-4474
Fax: (415) 325-4908
E-mail: linux@fintronic.com
<http://www.fintronic.com/linux/catalog.html>

InfoMagic Developer's Resource CD-ROM kit

InfoMagic, Inc.
PO Box 30370
Flagstaff, AZ 86003-0370

E-mail: info@infomagic.com
WWW: <http://www.infomagic.com>
Telefon: (800)-800-6613, (520)-526-9852
Fax: (520)-526-9573

Linux Quaterly CD-ROM

Morse Telecommunication, Inc.
26 East Park Avenue, Suite 240
Long Beach, NY 11561
Zamówienia: (800) 60-MORSE
Wsparcie techniczne: (516) 889-8610
Fax: (516) 889-8665
E-mail: Linux@morse.net

Linux Systems Labs

Linux System Labs
18300 Tara Drive
Clinton Twp., MI 48036
Telefon: (800) 432-0556
E-mail: drivin@vela.acs.oakland.edu

Sequoia International Motif Development Package

Sequoia International, Inc.
600 West Hillsboro Blvd., Suite 300
Deerfield Beach, FL 33441
Telefon: (305) 480-6118

Takelap Systems Ltd.

The Reddings
Court Robin Lane, Llangwm
Usk, Gwent, United Kingdom NP5 1ET
Telefon: +44 (0)291 650357
E-mail: info@ddrive.demon.co.uk

Trans-Ameritech Linux Plus BDS CD-ROM

Trans-Ameritech Linux Plus BSD CD-ROM

Trans-Ameritech Enterprises, Inc.

2342A Walsh Avenue

Santa Clara, CA 95051

Telefon: (408) 727-3883

E-mail: roman@trans-ameritech.com

Caldera OpenLinux

Caldera, Inc.

633 South 550 East

Provo, Utah 84606

E-mail: info@caldera.com

WWW: <http://www.caldera.com>

FTP: <ftp://ftp.caldera.com>

Telefon: (801)-377-7678

Fax: (801)-377-8752

Dodatek C

Projekt dokumentacji Linuxa

Nad projektem dokumentacji Linuxa (Linux Documentation Project), którego celem jest opracowanie zestawu dokumentów pełniących funkcję kompletnego podręcznika systemu Linux, pracuje grupa ochotników. Koordynatorem prac jest Matt Welsh, wspomagany przez Larsa Wireniusa i Michaela K. Johnsona.

Strona główna dotycząca Linuxa prowadzona przez Matta Welsha dostępna jest pod adresem

<http://sunsite.unc.edu/mdw/linux.html>

Organizatorzy projektu zachęcają wszystkich ochotników do włączenia się w prace nad dokumentowaniem systemu Linux. Jeśli posiadasz dostęp do Internetu, możesz przyłączyć się do kanału DOC listy dyskusyjnej Linux-Activists, wysyłając do użytkownika linux-activists-request@niksula.hut.fi wiadomość o treści

X-Mn-Admin: join DOC

Jeśli masz jakieś pytania, chcesz podzielić się pomysłami czy pieniędzmi, nie krępuj się i skontaktuj się z koordynatorem tego projektu. Adres e-mailowy Matta Welsha to mdw@sunsite.unc.edu, natomiast adres, pod którym można się z nim skontaktować za pośrednictwem zwykłej poczty, jest następujący:

205 Gray Street
Wilson, NC 27893

Dodatek D

Publiczna licencja GNU

W tym dodatku:

- υ Publiczna licencja GNU, wersja 2, czerwiec 1991
- υ E.1 Preambuła
- υ E.2 Warunki licencji GNU: kopiowanie, dystrybucja i modyfikowanie plików
- υ Jak zastosować zasady licencji GNU do własnych programów

System Linux jest rozpowszechniany zgodnie z licencją GNU General Public License (nazywaną również GPL *copyleft*, w odróżnieniu od *copyright*), którą zamieszczamy tu po to, aby wyjaśnić wszelkie niedomówienia związane z prawami do tego systemu.

Linux nie zalicza się do oprogramowania typu shareware czy public domain. Prawa do większej części jądra systemu Linux od 1993 roku posiada Linus Torvalds, a do innych części systemu i jądra systemu prawa autorskie posiadają ich twórcy. Jak widać, system Linux jest objęty prawem autorskim jego twórców.

Mimo tego można rozprowadzać go zgodnie z zasadami przedstawionej poniżej licencji GPL.

Publiczna licencja GNU, wersja 2, czerwiec 1991

*Copyright 1989, 1991
Free Software Foundation, Inc.
675 Mass Ave,
Cambridge, MA 02139
USA*

Każdy ma prawo kopiować i rozpowszechniać ten dokument w wersji niezmienionej, nie jest natomiast dozwolone wprowadzanie w nim jakichkolwiek modyfikacji.

E.1 Preambuła

Licencje obejmujące większość oprogramowania mają na celu ograniczenie możliwości jego kopiowania i modyfikowania. Publiczna licencja GNU ma umożliwić użytkownikom oprogramowania jego kopiowanie i modyfikowanie oraz zapewnić darmowość dla wszystkich użytkowników. Licencja ta stosuje się do większości programów tworzonych w ramach projektu Free Software Foundation oraz do wszelkich aplikacji, przy których tworzeniu używano tych programów (niektóre inne programy tworzone pod patronatem Free Software Foundation są również objęte licencją GNU).

Licencję GNU można również stosować do tworzonych przez siebie programów.

Programy objęte licencją GNU mogą, ale nie muszą być rozpowszechniane bezpłatnie. Została ona stworzona tak, aby umożliwić wszystkim dystrybucję darmowych programów (jak również pobieranie za nie opłaty), zapewnić dostęp do ich kodu źródłowego, umożliwić modyfikowanie i wykorzystanie fragmentów kodu w nowych programach i poinformować wszystkich zainteresowanych o tych możliwościach.

Aby chronić prawa użytkowników, wprowadziliśmy restrykcje zabraniające komukolwiek pozbawiania innych możliwości korzystania z tych praw czy też ograniczania ich.

Ograniczenia te przekładają się bezpośrednio na pewne obowiązki nałożone na dystrybutorów oprogramowania i osoby, które je modyfikują.

Jeśli na przykład rozpowszechniasz kopie programu objętego licencją GNU, czy to za darmo, czy też pobierając jakieś opłaty, musisz przyznać wszystkim klientom takie same prawa, jakie sam posiadasz. Musisz również udostępnić kod źródłowy programu oraz opublikować informacje o prawach przysługujących użytkownikowi oprogramowania.

Na ochronę praw użytkownika składają się dwa czynniki:

- 1. prawa autorskie**
- 2. licencja GNU, pozwalająca zgodnie z prawem kopiować, rozpowszechnać i modyfikować oprogramowanie.**

Aby chronić autorów oprogramowania, licencja nie daje żadnej gwarancji poprawnej pracy programów (które między innymi dzięki temu mogą pozostać darmowe). Jeśli oprogramowanie jest modyfikowane, autor zmian musi poinformować o fakcie ich wprowadzenia użytkowników, którym go oferuje, przez co ewentualne problemy ze zmodyfikowaną wersją nie odbiąją się niekorzystnie na dobrym imieniu autora wersji pierwotnej.

Zagrożeniem dla darmowego oprogramowania jest również patentowanie rozwiązań programistycznych. Aby uniknąć sytuacji, w których dystrybutorzy darmowego oprogramowania samodzielnie patentują zawarte w nich rozwiązania, pozbawiając użytkowników dostępu do nich, wyjaśniamy, że każde opatentowane rozwiązanie musi być dostępne za darmo lub też nie może w ogóle zostać objęte patentem.

Poniżej przedstawiono szczegółowe warunki kopiowania, dystrybucji i modyfikacji programów.

E.2 Warunki licencji GNU: kopiowanie, dystrybucja i modyfikowanie plików

Ta licencja dotyczy wszystkich programów i innych dzieł, które opatrzone są informacją o prawach autorskich stwierdzającą, że mogą być rozpowszechniane na warunkach określonych przez licencję GNU General Public License. Określenie „Program” tyczy się będzie dowolnego programu lub innego dzieła, natomiast „dzieło oparte na Programie” oznacza zarówno sam Program, jak i dowolne dzieło pochodne, chronione prawem autorskim, czyli dzieło zawierające Program lub jego fragment, zarówno w wersji pierwotnej, jak i zmodyfikowanej czy przetłumaczonej na inny język (tłumaczenie bez ograniczeń jest jedną z postaci modyfikacji).

Działania inne niż kopiowanie, dystrybucja i modyfikacja wychodzą poza zakres tej licencji. Uruchamianie Programu również nie jest przez nią w żaden sposób ograniczone, natomiast wyniki jego działania są objęte licencją tylko w przypadku, gdy w ich skład wchodzą elementy oparte ściśle na działaniu Programu (niezależnie natomiast od tego, czy są one wynikiem działania Programu).

Poniższe punkty stosują się w zależności od przeznaczenia Programu.

3. Można kopiować i rozpowszechniać niezmienione kopie kodu źródłowego Programu, w postaci takiej, w jakiej go otrzymano, na dowolnym nośniku, pod warunkiem, że do każdej kopii dołączona zostanie widoczna i wyraźna informacja o prawach autorskich i o braku gwarancji na Program. Wszystkie te informacje i inne informacje związane z licencją powinny być rozpowszechniane wraz z Programem w postaci niezmienionej.

Za fizyczne skopiowanie Programu można pobierać opłaty, można również gwarantować Program za darmo lub za opłatą.

4. Można modyfikować kopię Programu i kopiować dowolne jego fragmenty, co upoważnia do tworzenia nowych dzieł opartych o Program, oraz do kopiowania i rozpowszechniania ich zgodnie z warunkami opisanymi w punkcie 1., pod warunkiem, że spełnione zostaną również poniższe wymagania.
 - a. Zmodyfikowane pliki muszą zawierać informację o tym, że zostały zmodyfikowane, wraz z danymi o dacie modyfikacji i jej autorze.
 - b. Każde rozpowszechniane lub publikowane dzieło, które w całości lub części jest oparte na Programie lub jego fragmencie, musi być dostępne za darmo dla wszystkich jego odbiorców, zgodnie z warunkami tej licencji.
 - c. Jeśli zmodyfikowany Program pracuje w sposób interaktywny, przyjmując polecenia z wiersza poleceń, musi on na początku pracy (po uruchomieniu w najprostszym sposobie) wyświetlać lub drukować komunikat zawierający informacje o prawach autorskich oraz o tym, że Program nie jest objęty żadną gwarancją (chyba że gwarancja taka zostanie udzielona przez osobę modyfikującą

kod lub dystrybutora) i że można go rozprowadzać zgodnie z tymi warunkami. Komunikat powinien również zawierać wskazówki co do sposobu, w jaki można obejrzeć pełną kopię tej licencji (jeśli jednak Program działa w sposób interaktywny i nie wyświetla takiego komunikatu, również dzieło oparte na nim nie musi go wyświetlać).

Wymagania powyższe dotyczą dzieła opartego na Programie, traktowanego jako całość. Jeśli pewne fragmenty tego dzieła, mogące być traktowane jako oddzielne, działające niezależnie od pozostałych moduły, nie są oparte na Programie, licencja nie musi się do nich stosować, o ile są one rozprowadzane jako oddzielne dzieła. Jeśli natomiast są one rozprowadzane wraz z częściami opartymi na Programie jako jedno dzieło, całość dystrybucji musi podlegać warunkom licencji, pozwalającej na rozszerzanie dzieła o kolejne moduły, bez względu na ich autorstwo.

Z tego powodu licencja ta nie jest odpowiednia, jeśli autor chce zastrzec sobie wyłączne prawo do modyfikowania Programu. Jej intencją jest wymuszenie prawa do kontrolowania dystrybucji pochodnych lub tworzonych zespołowo dzieł opartych na Programie.

Rozprowadzanie na tym samym nośniku Programu (czy też dzieła opartego na Programie) oraz innych dzieł nie wymusza stosowania do nich warunków licencji.

5. Można kopiować i rozprowadzać Program (oraz dzieło oparte na Programie, zgodnie z warunkami z punktu 2) w postaci kodu pośredniego i w postaci wykonywalnej, zgodnie z warunkami z punktu 1 i 2, pod warunkiem zapewnienia realizacji przynajmniej jednego z poniższych punktów.

- a.** Do wersji wykonywalnej (na nośniku używanym zwykle do wymiany oprogramowania) należy dołączyć odpowiedni kod źródłowy, w wersji czytelnej dla komputera, objęty warunkami określonymi w punkcie 1 i 2.
- b.** Do wersji wykonywalnej należy dołączyć pisemną ofertę, ważną co najmniej przez trzy lata, pozwalającą wszystkim zainteresowanym na zakup kompletnej kopii kodu źródłowego, w wersji odpowiedniej dla danego komputera, za cenę nie wyższą niż koszt sporządzenia tej kopii. Kod źródłowy musi być objęty warunkami określonymi w punkcie 1 i 2 i rozprowadzany na nośniku używanym zwykle do wymiany oprogramowania.
- c.** Do wersji wykonywalnej należy dołączyć informację o tym, w jaki sposób można uzyskać kod źródłowy, otrzymaną od wcześniejszego dystrybutora. To rozwiązanie może być zastosowane wyłącznie w przypadku dystrybucji niekomercyjnej i wtedy, gdy otrzymana kopia miała postać plików pośrednich lub wykonywalnych, zgodnie z warunkami z punktu b.

Kod źródłowy dzieła to taka jego postać, która najlepiej nadaje się do wprowadzania modyfikacji. Dla pliku wykonywalnego pełny kod źródłowy to kod źródłowy wszystkich modułów, pliki zawierające definicje interfejsu, skrypty pozwalające na komplikację i instalację plików wykonywalnych. Kod źródłowy nie musi jednak zawierać składników, które są rozprowadzane normalnie (w formie kodu źródłowego lub plików wykonywalnych) wraz z głównymi częściami systemu (jądrem, kompilatorem itp.), pod kontrolą którego plik wykonywalny ma pracować, chyba że sam ten składnik towarzyszy plikowi wykonywalnemu.

Jeśli dystrybucja plików wykonywalnych czy pośrednich opiera się na udostępnieniu ich w jakimś miejscu, udostępnienie w tym samym miejscu kodu źródłowego zgadza się z warunkami tej licencji, choć użytkownicy nie muszą otrzymywać kodu źródłowego wraz z plikami wykonywalnymi.

6. Nie można kopiować, modyfikować i rozprowadzać Programu na warunkach innych, niż wymienione w tej licencji. Każda próba kopiowania, modyfikacji i udzielania licencji na Program jest nieważna i odbiera wszelkie prawa przyznawane przez tę licencję. Nie są jednak pozbawiane tych praw osoby, które nie łamią zasad licencji.
7. Akceptacja licencji nie jest wymagana, ponieważ nie jest ona potwierdzana żadnym podpisem. Mimo tego nie można modyfikować i kopować Programu w żaden inny sposób – takie działanie jest sprzeczne z prawem. Modyfikowanie i rozprowadzanie programu (lub dowolnego dzieła opartego na Programie) oznacza zaakceptowanie warunków licencji, wraz ze wszystkimi jej warunkami dotyczącymi kopiowania, modyfikacji i rozprowadzania Programu.
8. Po przekazaniu kopii Programu (albo dzieła opartego na Programie) innemu użytkownikowi nabywa on wszelkie prawa, które posiadała osoba będąca pierwotnie w posiadaniu Programu, dotyczące kopiowania, dystrybucji i modyfikacji. Nakładanie jakichkolwiek ograniczeń na odbiorcę Programu jest niedozwolone. Osoba rozprowadzająca Program nie jest odpowiedzialna za wymuszanie zgodności z licencją przez osoby trzecie.
9. Jeśli w wyniku procesu sądowego, udowodnienia naruszenia praw patentowych czy z jakichkolwiek innych przyczyn osoba rozprowadzająca oprogramowanie zostanie ograniczona jakimś warunkami (określonymi przez wyrok sądu, postanowienie ugody itp.), nie zwalnia jej to od przestrzegania warunków licencji. Jeśli nie jest możliwa dystrybucja Programu zgodnie z licencją, nie kolidująca z innymi zobowiązaniemi stałymi, nie można rozprowadzać Programu w ogóle. Jeśli na przykład patent nie zezwala na darmową dystrybucję Programu dla wszystkich otrzymujących jego kopię bezpośrednio czy pośrednio, jedyną drogą na pogodzenie warunków patentu i licencji jest zaprzestanie rozprowadzania Programu.

Jeśli którykolwiek z fragmentów tego punktu okaże się w jakichś szczególnych okolicznościach nieprawidłowy, należy wziąć pod uwagę fakt, że pozostała jego część nadal stosuje się do Programu, a całość powinna być stosowana w pozostałych okolicznościach.

Celem tego punktu nie jest namawianie kogokolwiek do łamania praw patentowych czy innych praw własności czy też do kwestionowania ważności tego typu praw. Jedynym jego celem jest ochrona spójności systemu rozprowadzania darmowego oprogramowania, którego zasady są określone przez licencję publiczną. Wiele osób zdecydowało się poświęcić swoje programy na rzecz tego systemu, ufając w jego spójność i ścisłe stosowanie. Tylko od autora zależy, czy chce on rozprowadzać oprogramowanie poprzez dowolny inny system dystrybucji – ta decyzja nie może zostać wymuszona przez osobę udzielającą licencji.

Ten punkt ma na celu wyjaśnienie spodziewanych konsekwencji wynikających z dalszej części licencji.

- 10.** Jeśli dystrybucja lub użycie Programu jest w pewnych krajach ograniczone prawem patentowym lub autorskim, komunikat informujący o zasadach rozprowadzania programu może zawierać dane o geograficznych ograniczeniach dotyczących jego rozprowadzania. W takim przypadku licencja zawiera ograniczenia określone w jej treści.
- 11.** Free Software Foundation zastrzega sobie prawo do publikowania zmodyfikowanych wersji licencji General Public License. Nie będą się one różnić ogólnymi zasadami, natomiast mogą zawierać fragmenty pozwalające na rozwiązywanie nowych kwestii.
- 12.** Każda wersja licencji jest opatrzona numerem. Jeśli informacja dołączona do programu mówi, że stosuje się do niego wersja licencji o danym numerze i wszystkie wersje późniejsze, dystrybutor może sam zdecydować, czy chce stosować wersję określoną explicite czy też któryś z wersji późniejszych. Jeśli w programie nie jest określona wersja licencji, można wybrać dowolną wersję opublikowaną kiedykolwiek przez Free Software Foundation.
- 13.** Jeśli części Programu mają zostać włączone do innego dzieła rozprowadzanego za darmo, jednak na innych warunkach, należy uzyskać na to zgodę autora. W przypadku oprogramowania, które jest własnością Free Software Foundation, należy skontaktować się z tą organizacją – czasem taka zgoda jest udzielana. Decyzja będzie zależeć od tego, czy dzieła oparte na darmowym oprogramowaniu pozostaną dostępne dla wszystkich, oraz od tego, czy zachowana zostanie zgodność z ideą dzielenia i wielokrotnego wykorzystania oprogramowania.
BRAK GWARANCJI
- 14.** PONIEWAŻ PROGRAM OBJĘTY JEST LICENCJĄ NIE PRZEWIDUJĄCĄ POBIERANIA ŻADNYCH OPŁAT, NIE JEST ON OBJĘTY ŻADNĄ GWARANCJĄ W ZAKRESIE DOZWOLONYM PRZEZ OBOWIĄZUJĄCE PRAWO. POZA SYTUACJAMI, GDY INNE ZASADY SĄ OKREŚLONE NA PIŚMIE, PROGRAMY UDOSTĘPNIANE SĄ TAK, JAK ZOSTAŁY STWORZONE, BEZ GWARANCJI ŻADNEGO TYPU, OKREŚLONEJ BEZPOŚREDNIO CZY POŚREDNIO, WŁĄCZAJĄC W TO GWARANCJE OSÓB SPRZEDAJĄCYCH OPROGRAMOWANIE I GWARANCJE PRZYDATNOŚCI PROGRAMÓW DO OKREŚLONYCH CELÓW. RYZYKO ZWIĄZANE Z DZIAŁANIEM I JAKOŚCIĄ PROGRAMU SPOCZYWA W CAŁOŚCI NA BARKACH UŻYTKOWNIKA. JEŚLI PROGRAM ZAWIERA BŁĘDY, UŻYTKOWNIK PONOSI WSZELKIE KOSZTY ZWIĄZANE Z EWENTUALnymi NAPRAWAMI CZY POPRAWKAMI.
- 15.** W ŻADNYM PRZYPADKU, CHYBA ŻE JEST TO WYMAGANE PRZEZ ODPOWIEDNIE PRAWO LUB ZOSTAŁA WYRAŻONA PISEMNA ZGODA, OSOBA POSIADAJĄCA PRAWA AUTORSKIE CZY INNE OSOBY WSPÓŁUCZESTNICZĄCE W TWORZENIU PROGRAMU NIE MOGĄ BYĆ OBCIĄŻANE ODPOWIEDZIALNOŚCIĄ ZA EWENTUALNE SZKODY

POWSTAŁE W WYNIKU DZIAŁANIA PROGRAMU, WŁĄCZAJĄC W TO SZKODY NATURY OGÓLNEJ, PRZYPADKOWE CZY POWSTAŁE ZGODNIE Z DZIAŁANIEM PROGRAMU, POWSTAŁE W WYNIKU NIEWŁAŚCIWEGO LUB WŁAŚCIWEGO WYKORZYSTANIA PROGRAMU (WŁĄCZAJĄC W TO RÓWNIEŻ UTRATĘ DANYCH, OTRZYMANIE NIEPRAWIDŁOWYCH DANYCH, STRaty PONOSZONE PRZEZ UŻYTKOWNIKA LUB OSOBY TRZECIE, CZY TEŻ NIEPRAWIDŁOWE DZIAŁANIE PROGRAMU W OBECNOŚCI INNYCH PROGRAMÓW), NAWET JEŚLI POSIADACZ PRAW AUTORSKICH CZY WSPÓŁTWÓRCA PROGRAMU ZOSTAŁ POINFORMOWANY O MOŻLIWOŚCI ZAISTNIENIA TAKICH STRAT.

KONIEC OKREŚLENIA WARUNKÓW

Jak zastosować zasady licencji GNU do własnych programów

Jeśli opracowujesz nowy program i chcesz, by był on dostępny dla możliwie szerokiej rzeszy użytkowników, najlepiej udostępnić go za darmo zgodnie z powyższymi warunkami, dzięki czemu każdy będzie mógł go rozprowadzać i modyfikować.

Aby to zrobić, należy do programu dołączyć przedstawione poniżej informacje. Najbezpieczniej umieścić je na początku każdego pliku z kodem źródłowym, dzięki czemu nie będzie wątpliwości co do tego, których plików dotyczą te warunki. Każdy plik powinien zawierać informację o prawach autorskich i wskazówkę, gdzie należy szukać pełnej treści warunków licencji; w wersji angielskiej:

<w pierwszym wierszu należy umieścić nazwę programu i krótki opis jego zastosowania>

Copyright (C) 19xx <imię i nazwisko autora>
This program is free software; you can redistribute it and/or modify it
under the terms of the GNU General Public License as published by the
Free Software Foundation; either version 2 of the License, or (at your
option) any later version.
This program is distributed in the hope that it will be useful, but
WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
Public License for more details. You should have received a copy of the
GNU General Public License along with this program; if not, write to the
Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
lub w wersji polskiej:
<w pierwszym wierszu należy umieścić nazwę programu i krótki opis jego zastosowania>

Copyright (C) 19xx <imię i nazwisko autora>
Ten program jest darmowy; można go rozprowadzać i modyfikować zgodnie
z warunkami określonymi w wersji 2 lub wersjach późniejszych licencji GNU
General Public License opublikowanej przez Free Software Foundation.
Program ten jest rozprowadzany po to, aby jak najlepiej służył
użytkownikom, ale nie jest on objęty żadną gwarancją, nawet gwarancja
wynikającą z obowiązków sprzedawcy czy też odnoszącą się do przydatności

programu do określonych zastosowań. Szczegółowe informacje znajdują się w licencji GNU General Public License. Jej kopię powinieneś otrzymać wraz z tym programem; jeśli tak nie jest, napisz do Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Należy również podać informację o tym, jak można skontaktować się z autorem, zarówno drogą elektroniczną, jak i pocztową.

Jeśli program działa w trybie interaktywnym, powinien na początku wyświetlać informację podobną do następującej:

```
Gnomovision version 69, Copyright (C) 19xx imię i nazwisko autora
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
This is free software, and you are welcome to redistribute it under
certain conditions; type 'show c' for details.
```

Hipotetyczne polecenia `show w` i `show c` powinny powodować wyświetlenie odpowiednich fragmentów licencji General Public License. Oczywiście postać tych poleceń może być inna, mogą to nawet być kliknięcia myszą na odpowiednich pozycjach menu – wszystko zależy od samego programu.

Jeśli to konieczne, powinieneś również zdobyć podpis swojego pracodawcy (jeśli pracujesz jako programista), dyrektora szkoły itp., na dokumencie stwierdzającym, że zrzeka się on praw autorskich do tworzonego przez Ciebie programu. Oto treść przykładowego dokumentu tego typu:

*Yoyodyne, Inc., wyrzeka się wszelkich praw do programu „Gnomovision”, autorstwa
Jamesa Hackera
<aktualny podpis>, 1 IV 1998
Ty Coon, prezes Yoyodyne, Inc.*

Licencja General Public License nie zezwala na dołączanie programów do programów komercyjnych. Jeśli Twój program jest biblioteką podprogramów, możesz rozważyć zwolenie na dołączanie tej biblioteki do tworzonych aplikacji. Jeśli chcesz to zrobić, powinieneś zamiast tej licencji zastosować warunki licencji GNU Library General Public License.

Dodatek E

Informacje

o prawach autorskich

Prawa autorskie do jądra systemu Linux (lata 1991, 1992, 1993 i 1994) posiada Linus Torvalds (inne osoby posiadają prawa autorskie do niektórych sterowników, systemów plików i innych części jądra systemu). Jest ono objęte warunkami licencji GNU General Public License (dokładniejsze informacje znajdziesz w pliku `COPYING` w katalogu `/usr/src/linux`).

Również wiele innych pakietów oprogramowania wchodzących w skład dystrybucji podlega licencji GNU General Public License, której treść można znaleźć w pliku `COPYING` w katalogu `/usr/src/linux`.

Programy te zostały między innymi opracowane przez Uniwersytet Kalifornijski w Berkeley i instytucje z nim związane:

Copyright (c) 1980, 1983, 1986, 1988, 1990, 1991 The Regents of the University of California. Wszelkie prawa zastrzeżone.

Dozwolona jest dystrybucja i używanie programów w formie kodu źródłowego i w postaci wykonywalnej, zarówno w postaci identycznej z oryginalną, jak i zmodyfikowanej, po spełnieniu następujących warunków.

1. Przy rozpowszechnianiu kodu źródłowego należy dołączyć powyższą notkę o prawach autorskich, listę obowiązujących warunków i podane poniżej zastrzeżenie.
2. Przy rozpowszechnianiu plików binarnych należy dołączyć do dokumentacji lub innych materiałów rozpowszechnianych wraz z programem powyższą notkę o prawach autorskich, listę obowiązujących warunków i podane poniżej zastrzeżenie.
3. Wszystkie materiały promujące programy muszą zawierać następującą informację:

Produkt zawiera oprogramowanie opracowane na Uniwersytecie Kalifornijskim w Berkeley, oraz przez jego współpracowników.

-
4. Nazwa uniwersytetu ani nazwy instytucji z nim współpracujących nie mogą być wykorzystane do promowania czy potwierdzania jakości produktów opartych na tym programie bez uzyskania pisemnej zgody.

PROGRAM TEN JEST ROZPROWADZANY W POSTACI, W JAKIEJ ZOSTAŁ STWORZONY, I NIE JEST ON OBJĘTY GWARANCJĄ ŻADNEGO TYPU, OKREŚLONĄ BEZPOŚREDNIO CZY POŚREDNIO, WŁĄCZAJĄC W TO GWARANCJE OSÓB SPRZEDAJĄCYCH OPROGRAMOWANIE I GWARANCJE PRZYDATNOŚCI PROGRAMÓW DO OKREŚLONYCH CELÓW. W ŻADNYM PRZYPADKU OSOBA POSIADAJĄCA PRAWA AUTORSKIE CZY INNE OSOBY WSPÓŁUCZESTNICZĄCE W TWORZENIU PROGRAMU NIE MOGĄ BYĆ OBCIĄŻANE ODPOWIEDZIALNOŚCIĄ NA ŻADNEJ PŁASZCZYŹNIE (ROZUMIANĄ JAKO ODPOWIEDZIALNOŚĆ BEZPOŚREDNIA, POŚREDNIA, CZY WYNIKAJĄCA Z ZANIEDBANIA) ZA EWENTUALNE SZKODY POŚREDNIO CZY BEZPOŚREDNIO POWSTAŁE W WYNIKU DZIAŁANIA PROGRAMU, WŁĄCZAJĄC W TO SZKODY PRZYPADKOWE CZY POWSTAŁE ZGODNIE Z DZIAŁANIEM PROGRAMU (RÓWNIEŻ DOSTARCZENIE NIEWŁAŚCIWYCH DÓBR CZY USŁUG, UTRATĘ DANYCH, UŻYTECZNOŚCI, ZYSKÓW LUB PRZERWANIE DZIAŁANIA ZA ROBKOWEGO) W DOWOLNY SPOSÓB, NAWET JEŚLI POSIADACZ PRAW AUTORSKICH CZY WSPÓŁTWÓRCA PROGRAMU ZOSTAŁ POINFORMOWANY O MOŻLIWOŚCI ZAISTNIENIA TAKICH STRAT.

Dystrybucja Slackware zawiera programy kompresujące firmy Info-ZIP. Programy te (Zip, UnZip i pochodne) są darmowe i można załadować je zarówno w postaci kodu źródłowego, jak i w wersji wykonywalnej w wielu węzłach FTP, na przykład w węźle ftp://ftp.uu.net/pub/archiving/zip/*. Oprogramowanie to jest całkowicie darmowe – za jego używanie nie ponosi się żadnych dodatkowych czy ukrytych kosztów. Dziękujemy firmie Info-ZIP! :^)

Pliki źródłowe programów Zip/Unzip można również znaleźć w katalogu `slackware_source/a/base`.

Skrypty instalacyjne systemu Slackware są objęte prawem autorskim Patricka Volkerdinga, 1993, 1994, Moorhead, Minnesota, USA. Wszelkie prawa zastrzeżone.

Dystrybucja i używanie tego oprogramowania, w postaci pierwotnej lub zmodyfikowanej, jest dozwolone po spełnieniu następujących warunków.

5. Rozprowadzane oprogramowanie musi zawierać poniższą notkę o prawach autorskich, tę listę warunków oraz poniższe zastrzeżenie.

PROGRAM TEN JEST ROZPROWADZANY W POSTACI, W JAKIEJ ZOSTAŁ STWORZONY, I NIE JEST ON OBJĘTY GWARANCJĄ ŻADNEGO TYPU, OKREŚLONĄ BEZPOŚREDNIO CZY POŚREDNIO, WŁĄCZAJĄC W TO GWARANCJE OSÓB SPRZEDAJĄCYCH OPROGRAMOWANIE I GWARANCJE PRZYDATNOŚCI PROGRAMÓW DO OKREŚLONYCH CELÓW. W ŻADNYM PRZYPADKU AUTOR NIE MOŻE BYĆ OBCIĄŻANY ODPOWIEDZIALNOŚCIĄ NA ŻADNEJ PŁASZCZYŹNIE (ROZUMIANĄ JAKO ODPOWIEDZIALNOŚĆ BEZPOŚREDNIA, POŚREDNIA, CZY WYNIKAJĄCA Z ZANIEDBANIA) ZA EWENTUALNE SZKODY

POŚREDNIO CZY BEZPOŚREDNIO POWSTAŁE W WYNIKU DZIAŁANIA PROGRAMU, WŁĄCZAJĄC W TO SZKODY PRZYPADKOWE CZY POWSTAŁE ZGODNIE Z DZIAŁANIEM PROGRAMU (RÓWNIEŻ DOSTARCZENIE NIEWŁAŚCIWYCH DÓBR CZY USŁUG, UTRATĘ DANYCH, UŻYTECZNOŚCI, ZYSKÓW LUB PRZERWANIE DZIAŁANIA ZAROBKOWEGO) W DOWOLNY SPOSÓB, NAWET JEŚLI POSIADACZ PRAW AUTORSKICH CZY WSPÓŁTWÓRCA PROGRAMU ZOSTAŁ POINFORMOWANY O MOŻLIWOŚCI ZAISTNIENIA TAKICH STRAT.

Slackware to marka zastrzeżona przez Patricka Volkerdinga. Używać jej w odniesieniu do dystrybucji Linuxa można po spełnieniu następujących warunków:

6. Aby dystrybucja mogła być określana jako Slackware, nie może być różnić się od wersji dostępnej w centralnym węźle FTP (<ftp.cdrom.com>). Ten warunek ma na celu zapewnienie spójności, niezawodności i dobrego imienia dystrybucji Slackware. Każdy, kto chce rozprowadzać wersję zmodyfikowaną, musi uzyskać zgodę Patricka Volkerdinga, volkerdi@ftp.cdrom.com (czyli certyfikat stwierdzający, że wersja jest w rozsądny sposób pozbawiona usterek). Jeśli dystrybucja będzie spełniać odpowiednie kryteria, zostanie wydana pisemna zgoda na używanie marki Slackware.
7. Do dystrybucji musi być dołączony cały odpowiedni kod źródłowy (jest to również wymaganie licencji GNU General Public License).
8. Marka Slackware może być używana jako znak firmowy czy nazwa produktu (lub jej części) tylko po uzyskaniu pisemnej zgody. Należy podkreślić, że nadal można rozprowadzać dystrybucje zmodyfikowane – zabronione jest tylko określanie ich jako Slackware. Osobiście nie lubię nakładać na nikogo żadnych ograniczeń, ale ograniczenia te nie mają na celu utrudnienia komukolwiek życia. Chcę się tylko upewnić, że do komercyjnych wersji dystrybucji Slackware nie są wprowadzane żadne usterek. Zdarzało się to w przeszłości, a rezultaty były takie, że moja skrzynka pocztowa była zalewana prośbami o pomoc. Próbuje zabezpieczyć się przed takimi sytuacjami.

Pytania dotyczące tych warunków należy kierować bezpośrednio do Patricka Volkerdinga, volkerdi@ftp.cdrom.com.

Prawa autorskie do programu XView3.2-X11R6:

© Copyright 1989, 1990, 1991 Sun Microsystems, Inc. Patenty firmy Sun oczekują na zatwierdzenie w USA i innych krajach. OPEN LOOK jest znakiem zastrzeżonym firmy USL i może być używany za zgodą właścicieli. ©

© Copyright Bigelow & Holmes, 1986, 1985. Lucida jest znakiem zastrzeżonym firmy Bigelow & Holmes. Używanie tego znaku jest dozwolone tylko w połączeniu z obrazami i czcionkami opisanymi w tym pliku.

SUN MICROSYSTEMS, INC. I BIGELOW & HOLMES NIE BIORĄ ODPOWIEDZIALNOŚCI ZA PRZYDATNOŚĆ TEGO KODU ŹRÓDŁOWEGO DO JAKICHKOLWIEK CELÓW. JEST ON ROZPROWADZANY TAK, JAK ZOSTAŁ STWORZONY I NIE JEST OBJĘTY ŻADNĄ GWARANCJĄ POŚREDNIĄ CZY BEZPOŚREDNIĄ. SUN MICROSYSTEMS, INC., USL I BIGELOW & HOLMES WYPIERAJĄ SIĘ WSZELKICH GWARANCJI DOTYCZĄCYCH TEGO KODU ŹRÓDŁOWEGO, WŁACZAJĄC W TO GWARANCJE OBOWIĄZUJĄCE SPRZEDAWCĘ I ODPOWIEDZIALNOŚĆ ZA PRZYDATNOŚĆ DO JAKICHKOLWIEK CELÓW. W ŻADNYM PRZYPADKU FIRMY SUN MICROSYSTEMS, INC., USL I BIGELOW & HOLMES NIE MOGĄ BYĆ OBCIĄŻANE ODPOWIEDZIALNOŚCIĄ ROZUMIANĄ JAKO ODPOWIEDZIALNOŚĆ BEZPOŚREDNIA, POŚREDNIA, CZY WYNIKAJĄCA Z ZANIEDBANIA ZA EWENTUALNE SZKODY POŚREDNIO CZY BEZPOŚREDNIO POWSTAŁE W WYNIKU UŻYCIA TEGO KODU ŹRÓDŁOWEGO, WŁACZAJĄC W TO SZKODY PRZYPADKOWE CZY POWSTAŁE ZGODNIE Z DZIAŁANIEM PROGRAMU, RÓWNIEŻ DOSTARCZENIE NIEWŁAŚCIWYCH DÓBR CZY USŁUG, UTRATĘ DANYCH, UŻYTECZNOŚCI LUB ZYSKÓW.

Stosowane są też różne inne warunki rozprowadzania programów. Szczegółowe informacje znaleźć można w dokumentacji dołączanej do oprogramowania.

Choć staraliśmy się zamieścić tu pełny kod źródłowy programów wchodzących w skład tego projektu, być może o czymś zapomnieliśmy. Jeśli odkryjesz, że czegoś brakuje, możemy udostępnić kopie odpowiednich plików – wystarczy zapytać!



Jesteśmy zobowiązani zapewnić dostęp do brakującego kodu źródłowego programów objętych licencją GPL przez trzy lata, zgodnie z punktem E2.3.b licencji Gnu General Public License:

b. Do wersji wykonywalnej należy dołączyć pisemną ofertę, ważną co najmniej przez trzy lata, pozwalającą wszystkim zainteresowanym na zakup kompletnej kopii kodu źródłowego, w wersji odpowiedniej dla danego komputera, za cenę nie wyższą niż koszt sporządzenia tej kopii. Kod źródłowy musi być objęty warunkami określonymi w punkcie 1 i 2 i rozprowadzany na nośniku używanym zwykle do wymiany oprogramowania.

Poza tym, jeśli stwierdzisz brak jakichś plików (nawet jeżeli nie potrzebujesz ich kopii), wyślij pocztą elektroniczną wiadomość pod adres volkerdi@ftp.cdrom.com, co pozwoli na uzupełnienie braków w przyszłości.

Dodatek F

Zawartość płyt CD-ROM

W tym dodatku:

- υ Nagradzany system operacyjny
- υ Oprogramowanie

Na płytach CD-ROM dołączonych do tej książki znajdziesz dystrybucję systemu Linux w wersji RedHat 5.0.

Nagradzany system operacyjny

System RedHat Linux, wybierany przez dwa lata z rzędu systemem operacyjnym roku przez Infoworld, jest coraz szerzej stosowany zamiast innych systemów operacyjnych przez użytkowników potrzebujących stabilnego, bezpiecznego i wydajnego systemu pracującego na komputerach klasy PC.

Pulpit

System RedHat pozwala na dowolne skonfigurowanie pulpitu. Można wybierać spośród kilku dostępnych programów zarządzających okienkami. Można przeglądać Sieć, kompilować programy, formatować dyskietki (wszystko w jednym czasie) i czerpać przyjemność z pracy z tym niesamowicie stabilnym systemem operacyjnym, radzącym sobie z wielozadaniowością o wiele lepiej niż inne systemy. Komputery działające pod kontrolą RedHat Linux mogą działać całe miesiące bez żadnego załamania.

Serwer internetowy

Zaraz po zainstalowaniu system RedHat Linux może działać jako serwer usług internetowych, takich jak strony WWW, poczta elektroniczna, DNS i inne, dla wielu węzłów i z obsługą domen wirtualnych. Za pomocą programu RPM (RedHat Package Manager) można

z łatwością aktualniać oprogramowanie systemowe, ładując najnowsze, najbezpieczniejsze wersje programów za pomocą pojedynczego polecenia. RPM pozwala również podnieść poziom bezpieczeństwa systemu informując, które z plików zostały zmodyfikowane.

Platforma edukacyjna

W jaki sposób można lepiej poznać system operacyjny, niż zapoznając się z jego pełnym kodem źródłowym? System Linux pozwala na poznanie tajników swego działania „od podszewki” (nie jest dostępny kod aplikacji komercyjnych). RedHat Linux rozprowadzany jest wraz z kompilatorami języków C, C++, F77, językami interpretowanymi (python, perl, Tcl/Tk, scheme0) oraz narzędziami przeznaczonymi do obliczeń matematycznych i inżynierskich (spice, GNUploat, xfing). Serwer Apache pozwala na naukę programowania skryptów CGI i tworzenie dokumentów w języku HTML.

Oprogramowanie

Odsyłamy do lektury dokumentacji dołączonej do oprogramowania rozprowadzanego przez innych dystrybutorów (zwykle zawartej w pliku `readme.txt` czy `licence.txt`) i zalecamy trzymanie się zawartych w nich wytycznych.

Skorowidz

