

IDŹ DO

PRZYKŁADOWY ROZDZIAŁ



SPIS TREŚCI

KATALOG KSIĄŻEK

KATALOG ONLINE

ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK

DODAJ DO KOSZYKA

CENNIK I INFORMACJE

ZAMÓW INFORMACJE
O NOWOŚCIACH

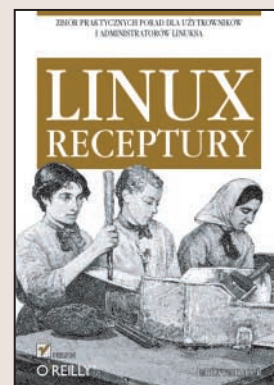
ZAMÓW CENNIK

CZYTELNIA

FRAGMENTY KSIĄŻEK ONLINE

Linux. Receptury

Autor: Carla Schroder
Tłumaczenie: Adam Jarczyk
ISBN: 83-7361-879-1
Tytuł oryginału: [Linux Cookbook](#)
Format: B5, stron: 560



Zbiór praktycznych porad dla użytkowników i administratorów Linuksa

O popularności Linuksa i jej powodach napisano już bardzo dużo. Dostępny nieodpłatnie system operacyjny, stabilny, bezpieczny i możliwy do zastosowania zarówno w serwerach, jak i stacjach roboczych – brzmi to niemal jak utopia, a jednak jest prawdą. Wielu użytkowników komputerów, zachęconych opiniami na temat Linuksa, zdecydowało się na jego instalację. Pierwszy kontakt z systemem w większości sytuacji przebiega bez problemów dzięki dopracowanemu modułowi instalacyjnemu i graficznemu interfejsowi użytkownika. Problemy rozpoczynają się w momencie przejścia do bardziej zaawansowanych zagadnień. Tryb tekstowy, polecenia z dziesiątkami opcji i parametrów, pliki konfiguracyjne liczące dziesiątki linijek i trudna do zlokalizowania dokumentacja dość skutecznie odstraszą od prób poznawania tajników Linuksa.

„Linux. Receptury” to książka, dzięki której Linux wyda się mniej przerażający. Zawiera rozwiązania większości problemów mogących pojawić się podczas pracy z Linuksem. Opisuje sposoby konfigurowania systemu, administrowania nim, znajdowania informacji w dokumentacji oraz korzystania z poleceń i narzędzi systemowych. Wszystko, co może sprawiać kłopoty użytkownikowi lub administratorowi Linuksa, zostało tu wyjaśnione w jasny i czytelny sposób. Każda receptura, poza omówieniem problemu i przedstawieniem gotowego rozwiązania, zawiera także analizę, która jest bardzo pomocna przy dostosowywaniu sposobu postępowania do własnych potrzeb.

- Przeszukiwanie dokumentacji systemowej
- Instalowanie oprogramowania z pakietów RPM i z kodu źródłowego
- Wykrywanie nowego sprzętu
- Uruchamianie i zatrzymywanie systemu
- Administracja kontami użytkowników i systemem plików
- Aktualizowanie jądra systemu
- Nagrywanie płyt CD i DVD
- Konfiguracja wielosystemowa
- Tworzenie kopii zapasowych i odtwarzanie systemu
- Konfiguracja usług sieciowych
- Serwer WWW i poczty elektronicznej

Stosowanie gotowych rozwiązań najczęściej występujących problemów to oszczędność nie tylko pracy, ale i czasu.



Spis treści

Przedmowa	17
1. Znajdowanie dokumentacji	21
1.1. Wprowadzenie	21
1.2. Strony man — wprowadzenie	22
1.3. Znajdowanie odpowiednich stron man	24
1.4. Znajdowanie zgubionych stron man	25
1.5. Czytanie stron man bez odpowiedniej przeglądarki	26
1.6. Konfiguracja manpath	27
1.7. Korzystanie ze stron info	28
1.8. Drukowanie stron man	28
1.9. Drukowanie stron info	29
1.10. Drukowanie wybranych stron man lub info	30
1.11. Znajdowanie całej dokumentacji programu	31
2. Instalacja i zarządzanie oprogramowaniem w systemach opartych na RPM.....	35
2.1. Wprowadzenie	35
2.2. Instalowanie RPM-ów	36
2.3. Aktualizacja RPM-ów	37
2.4. Usuwanie RPM-ów	38
2.5. Gromadzenie informacji o zainstalowanych RPM-ach	39
2.6. Zdobywanie informacji z niezainstalowanych RPM-ów	41
2.7. Znajdowanie niedawno zainstalowanych RPM-ów	42
2.8. Przebudowa bazy danych RPM	42
2.9. Śledzenie bibliotek skompilowanych ze źródeł w systemie RPM	43
2.10. Rozwiązywanie problemów z instalacją RPM	45
2.11. Instalowanie RPM-ów źródłowych	46
2.12. Zmiany opcji kompilacji RPM-a źródłowego	47
2.13. Instalacja Yum	49
2.14. Konfiguracja Yum	50
2.15. Instalacja i aktualizacja pakietów za pomocą Yum	51
2.16. Usuwanie pakietów za pomocą Yum	52
2.17. Gromadzenie za pomocą Yum informacji o zainstalowanych pakietach	52
2.18. Konserwacja Yum	53

3. Instalacja i zarządzanie oprogramowaniem w systemach opartych na Debianie ...	55
3.1. Wprowadzenie	55
3.2. Zdobywanie oprogramowania dla systemu Debian	56
3.3. Instalacja pakietów Debiana z płyt CD-ROM	58
3.4. Instalacja pakietów w systemach Debian	59
3.5. Usuwanie pakietów z systemu Debian	60
3.6. Instalacja ze źródeł w systemie Debian	60
3.7. Aktualizacja pakietów w systemie Debian	61
3.8. Aktualizacja systemu Debian	62
3.9. Przejście do nowszej wersji Debiana	63
3.10. Prowadzenie mieszanego systemu Debian	64
3.11. Wyszukiwanie pakietów zainstalowanych w systemie Debian	65
3.12. Utrzymanie pamięci podręcznej pakietów w Debianie	67
3.13. Rozwiązywanie konfliktów zależności	68
3.14. Tworzenie lokalnego repozytorium Debiana	69
3.15. Wybór serwerów zwierciadlanych pakietów dla apt-proxy.conf	70
3.16. Dodajemy własną pamięć podręczną pakietów do apt-proxy.conf	71
4. Instalacja programów z kodu źródłowego	73
4.1. Wprowadzenie	73
4.2. Przygotowanie systemu do kompilacji programów ze źródeł	73
4.3. Generowanie listy plików przy instalacji ze źródeł w celu ułatwienia deinstalacji	75
4.4. Instalacja programów z kodu źródłowego	75
4.5. Tworzenie pakietów z kodu źródłowego za pomocą CheckInstall	77
5. Wykrywanie sprzętu bez otwierania obudowy	79
5.1. Wprowadzenie	79
5.2. Wykrywanie sprzętu za pomocą lspci	80
5.3. Gromadzenie informacji o sprzęcie za pomocą dmesg	82
5.4. Obraz uruchomionego sprzętu w /proc	83
5.5. Przeglądanie partycji dysku narzędziem fdisk	86
5.6. Obliczenie pojemności dysku twardego	87
6. Edycja plików tekstowych za pomocą JOE i Vim	89
6.1. Wprowadzenie	89
6.2. Znajdowanie poleceń JOE	91
6.3. Dostosowanie JOE do własnych potrzeb	92
6.4. Zapis preferencji JOE w osobnym pliku	93
6.5. Kopiowanie pomiędzy plikami w JOE	94
6.6. Wyszukiwanie i zastępowanie w JOE	95
6.7. Zaznaczanie tekstu pionowo w JOE	96

6.8. Wyszukiwanie i otwieranie plików w JOE	97
6.9. Szybka nauka edytora Vim	98
6.10. Tworzenie autotekstu za pomocą skrótów Vim	100
6.11. Przypisywanie poleceń do kombinacji klawiszy	101
6.12. Dostosowanie ustawień edytora Vim do własnych potrzeb	102
6.13. Szybka nawigacja w edytorze Vim za pomocą znaczników	104
6.14. Powrót do pracy w miejscu jej ukończenia — sesje edytora Vim	105
6.15. Ustawienie domyślnego edytora	107
6.16. Wykrywanie opcji kompilacji edytora Vim	108
7. Uruchamianie i zamykanie systemu Linux.....	109
7.1. Wprowadzenie	109
7.2. Zmiana poziomu działania po starcie	111
7.3. Zmiana domyślnego poziomu działania	112
7.4. Uruchamianie i zatrzymywanie X	113
7.5. Zarządzanie poziomami działania w Debianie	114
7.6. Tworzenie tekstowych i graficznych poziomów działania w Debianie	115
7.7. Zarządzanie poziomami działania w systemie Red Hat	116
7.8. Ręczna konfiguracja uruchamiania usług	118
7.9. Ręczne uruchamianie i zatrzymywanie usług	119
7.10. Wyłączanie i przeładowywanie Linuksa	119
7.11. Wyłączanie i ograniczanie dostępu do Ctrl+Alt+Del	121
7.12. Automatyczne wyłączanie systemu	121
8. Zarządzanie użytkownikami i grupami	123
8.1. Wprowadzenie	123
8.2. Rozróżnianie kont użytkowników i kont systemowych	124
8.3. Znajdowanie UID i GID użytkownika	126
8.4. Dodawanie użytkowników poleceniem useradd	126
8.5. Dodawanie użytkowników poleceniem adduser	128
8.6. Modyfikacja kont użytkowników	129
8.7. Usuwanie użytkownika	130
8.8. Łatwe i przyjemne zabijanie procesów użytkownika	131
8.9. Blokowanie kont	132
8.10. Zarządzanie hasłami	133
8.11. Dodawanie grup poleceniem groupadd	134
8.12. Usuwanie grup poleceniem groupdel	134
8.13. Tworzenie użytkownika systemowego	135
8.14. Tworzenie grup systemowych poleceniem addgroup	136
8.15. Dodawanie i usuwanie członków grupy	136
8.16. Kontrola poprawności plików haseł	137

8.17. Dodawanie wsadowe użytkowników	138
8.18. Masowa zmiana haseł	143
8.19. Dodawanie wsadowe użytkowników do grup	144
8.20. Chwilowy dostęp do konta root za pomocą polecenia su	145
8.21. Przyznawanie ograniczonych uprawnień użytkownika root za pomocą sudo	146
8.22. Limity dyskowe	148
9. Zarządzanie plikami i partycjami	151
9.1. Wprowadzenie	151
9.2. Ustawianie uprawnień do plików i katalogów z użyciem numerycznej notacji chmod	157
9.3. Operacje wsadowe z chmod	158
9.4. Ustawianie uprawnień do plików i katalogów z użyciem symbolicznej notacji chmod	159
9.5. Zmiana właściciela pliku poleceniem chown	161
9.6. Operacje wsadowe z chown	161
9.7. Tworzenie katalogu współużytkowanego za pomocą setgid i bitu lepkości	162
9.8. Ustawianie domyślnych uprawnień poleceniem umask	164
9.9. Montowanie i odmontowywanie dysków wymiennych	165
9.10. Konfiguracja montowania systemów plików w /etc/fstab	167
9.11. Montowanie i odmontowywanie systemów plików na dyskach twardych	169
9.12. Znajdowanie nazw urządzeń na potrzeby mount i fstab	170
9.13. Tworzenie plików i katalogów	173
9.14. Usuwanie plików i katalogów	174
9.15. Kopiowanie, przenoszenie i zmiana nazw plików i katalogów	175
9.16. Tworzenie linuksowych partycji dyskowych poleceniem fdisk	176
9.17. Tworzenie systemu plików na nowej partycji	177
10. Łatanie, modyfikacje i aktualizacje jądra	179
10.1. Wprowadzenie	179
10.2. Dodawanie nowych funkcji do jądra 2.4	181
10.3. Odchudzanie standardowego jądra 2.4	185
10.4. Aktualizacja do najnowszej stabilnej wersji jądra 2.4	186
10.5. Kompilacja jądra 2.6	187
10.6. Dodawanie nowych funkcji do jądra 2.6	188
10.7. Dodawanie nowego modułu ładowalnego jądra	189
10.8. Instalacja poprawek	190
10.9. Usuwanie poprawki	193
10.10. Tworzenie obrazu initrd	193
10.11. Tworzenie dyskietki startowej w Debianie	194
10.12. Tworzenie dyskietki startowej w systemie Red Hat	195

11. Nagrywanie CD i DVD	197
11.1. Wprowadzenie	197
11.2. Znajdowanie adresu SCSI nagrywarki CD lub DVD	200
11.3. Włączenie emulacji SCSI dla nagrywarek CD i DVD IDE/Atapi	201
11.4. Nagrywanie CD z danymi do powszechnego użytku	203
11.5. Tworzenie drzew katalogów na CD	205
11.6. Kopiowanie CD lub DVD	206
11.7. Kasowanie CD-RW	207
11.8. Nagrywanie wielosesyjnej płyty CD z danymi	208
11.9. Tworzenie startowego dysku CD	209
11.10. Dzielenie dużego pliku na kilka CD	210
11.11. Nagrywanie DVD z danymi	211
11.12. Nagrywanie płyt audio CD dla standardowych odtwarzaczy CD	213
12. Zarządzanie programem rozruchowym i start wielosystemowy	217
12.1. Wprowadzenie	217
12.2. Migracja z LILO do GRUB	218
12.3. Instalacja GRUB bez dyskietki	220
12.4. Instalacja GRUB za pomocą grub-install	222
12.5. Przygotowanie komputera do uruchamiania różnych systemów Linux	223
12.6. Dodawanie kolejnych instalacji Linuksa do systemu	224
12.7. Znajdowanie parametrów startowych z powłoki poleceń GRUB	226
12.8. Konfiguracja partycji startowej	228
12.9. Tworzenie menu startowego GRUB	229
12.10. Modyfikacje menu.lst	230
12.11. Dodawanie Windows 95, 98 lub Me do systemu Linux	231
12.12. Dodawanie Windows NT, 2000 lub XP do startu wielosystemowego	233
12.13. Przywracanie GRUB w MBR za pomocą płyty CD Knoppix	235
12.14. Ochrona plików systemowych hasłem GRUB	236
12.15. Blokowanie dostępu użytkowników do konkretnych pozycji menu GRUB	237
12.16. Tworzenie obrazka startowego GRUB	238
12.17. Uruchamianie Linuksa z LILO	239
12.18. Wybór systemu Linux do uruchomienia w LILO	241
12.19. Start wielosystemowy Linuksa i Windows za pomocą LILO	242
12.20. Tworzenie dyskietki startowej LILO	243
12.21. Ochrona LILO hasłem	244
12.22. Tworzenie kopii zapasowej MBR	245
13. Ratowanie i przywracanie systemu za pomocą dystrybucji Knoppix	247
13.1. Wprowadzenie	247
13.2. Uruchomienie systemu Knoppix	247
13.3. Tworzenie dyskietki startowej Knoppiksa	249

13.4. Zapis konfiguracji systemu Knoppix w pamięci USB PenDrive	250
13.5. Tworzenie trwałego, zaszyfrowanego katalogu /home dla systemu Knoppix	250
13.6. Kopiowanie plików z innego PC z systemem Linux	251
13.7. Kopiowanie plików na udział Samba	253
13.8. Kopiowanie plików na CD-R/RW	254
13.9. Edycja plików konfiguracyjnych z poziomu systemu Knoppix	254
13.10. Instalacja oprogramowania w systemie Knoppix	255
13.11. Przywracanie hasła konta root	256
13.12. Instalacja systemu Knoppix na dysku twardym	257
13.13. Skanowanie Windows w poszukiwaniu wirusów za pomocą systemu Knoppix	257
14. Drukowanie przy użyciu CUPS	259
14.1. Wprowadzenie	259
14.2. Instalacja drukarki w autonomicznym PC z systemem Linux	261
14.3. Udostępnianie drukarek klientom linuksowym	264
14.4. Udostępnianie drukarki bez rozwiązywania nazw	265
14.5. Obsługa klientów Windows bez Samby	266
14.6. Udostępnianie drukarek za pomocą Samby w mieszanej sieci lokalnej	267
14.7. Tworzenie dedykowanego serwera drukarek CUPS	268
14.8. Drukowanie rozproszone z użyciem klas	269
14.9. Ograniczanie dostępu do drukarek i klas	270
14.10. Rozwiązywanie problemów	271
15. Konfiguracja grafiki i zarządzanie serwerem X Window	273
15.1. Wprowadzenie	273
15.2. Jednoczesne korzystanie z X Window i konsoli	276
15.3. Instalacja nowej karty graficznej	278
15.4. Edycja XF86Config	279
15.5. Włączenie akceleracji sprzętowej 3D w XFree86/DRI	280
15.6. Rozwiązywanie problemów z akceleracją sprzętową 3D	281
15.7. Konfiguracja dwumonitorowa	283
15.8. Wybór ServerLayout przy starcie	285
15.9. Ustawienie domyślnego ServerLayout	288
15.10. Konfiguracja startx	288
15.11. Zmiana graficznego menedżera logowania	289
15.12. Jednoczesne uruchamianie różnych menedżerów okien za pomocą Xnest	290
16. Tworzenie i przywracanie danych z kopii zapasowych.....	293
16.1. Wprowadzenie	293
16.2. Lokalne transfery i synchronizacja plików za pomocą rsync	295
16.3. Bezpieczne przesyłanie plików z użyciem rsync i ssh	296

16.4. Tworzenie serwera kopii zapasowych rsync	298
16.5. Zabezpieczanie modułów rsync	300
16.6. Instalacja anonimowego publicznego serwera rsync	301
16.7. Uruchamianie demona rsync przy starcie systemu	302
16.8. Precyzyjne wybieranie plików	303
16.9. Automatyzacja tworzenia kopii zapasowych za pomocą rsync przez ssh	304
16.10. Ograniczenie pasma sieci używanego przez rsync	305
16.11. Wskazywanie ścieżek w rsync	306
16.12. Instalacja rsync w klientach Windows	306
16.13. Tworzenie wiadomości dnia dla rsync	307
16.14. Tworzenie startowej płyty CD do przywracania systemu z pomocą Mondo Rescue	308
16.15. Weryfikacja kopii zapasowej Mondo	311
16.16. Tworzenie startowej płyty DVD do przywracania systemu z pomocą Mondo Rescue	312
16.17. Klonowanie systemów linuksowych z pomocą Mondo Rescue	313
16.18. Zastosowanie mindi-kernel	314
16.19. Przywracanie systemu z płyty ratunkowej Mondo	314
16.20. Przywracanie wybranych plików z płyty Mondo	315
17. Dostęp zdalny.....	317
17.1. Wprowadzenie	317
17.2. Konfiguracja OpenSSH po raz pierwszy	318
17.3. Generowanie nowych kluczy hosta	320
17.4. Uwierzytelnianie za pomocą kluczy publicznych	321
17.5. Korzystanie z dodatkowych par kluczy	323
17.6. Logowanie bez hasła z użyciem narzędzia ssh-agent	324
17.7. Lepsze logowanie bez hasła z użyciem keychain	325
17.8. Logowanie bez hasła dla zadań cron	326
17.9. Automatyczne wyłączenie ssh-agent przy wylogowaniu	327
17.10. Dostosowanie wiersza zachęty Bash dla ssh	327
17.11. Tunelowanie X przez SSH	329
17.12. Łączenie z komputera Windows	330
17.13. Uprawnienia do plików ssh	331
18. Kontrola wersji.....	333
18.1. Wprowadzenie	333
18.2. Tworzenie prostego lokalnego repozytorium RCS	335
18.3. Pobieranie starszych wersji plików z RCS	336
18.4. Porównywanie wersji pliku w RCS	338
18.5. Zarządzanie systemowymi plikami konfiguracyjnymi za pomocą RCS	339
18.6. CVS w roli lokalnego repozytorium dla pojedynczego użytkownika	341

18.7. Dodawanie nowych plików do repozytorium CVS	343
18.8. Usuwanie plików z repozytorium CVS	344
18.9. Tworzenie wspólnego repozytorium CVS	345
18.10. Udostępnianie jednego repozytorium kilku grupom	346
18.11. Dostęp do zdalnego repozytorium CVS	347
18.12. Aktualizacja plików roboczych w CVS	348
18.13. Pobieranie określonych starszych wersji z CVS	349
18.14. Anonimowe repozytorium CVS tylko do odczytu, wykorzystujące Pserver	351
18.15. Dublowanie repozytorium CVS	353
18.16. Zapisywanie plików binarnych w CVS	354
18.17. Tworzenie obrazów wersji za pomocą etykiet	355
18.18. Tworzenie gałęzi stabilnych i rozwojowych projektu	356
18.19. Dostosowanie środowiska CVS do własnych potrzeb	358
18.20. Planowanie miejsca na repozytorium CVS	359
19. Synchronizacja czasu przez NTP	361
19.1. Wprowadzenie	361
19.2. Instalacja lokalnego serwera czasu	362
19.3. Łączenie się z lokalnym serwerem czasu	363
19.4. Kontrola dostępu	365
19.5. Wybór puli NTP	366
19.6. Łączenie z serwerem czasu za pomocą łącza tymczasowego	368
19.7. Konfiguracja kilku serwerów czasu	370
19.8. Uwierzytelnianie kluczami w protokole NTP	371
20. Instalacja serwera pocztowego Postfix	373
20.1. Wprowadzenie	373
20.2. Instalacja serwera pocztowego POP3	374
20.3. Instalacja serwera pocztowego POP3 w systemie Debian	378
20.4. Testowanie serwera pocztowego SMTP/POP3	379
20.5. Wysyłanie poczty internetowej	382
20.6. Odbieranie poczty internetowej	382
20.7. Instalacja Cyrus-SASL do autoryzacji SMTP	383
20.8. Instalacja Cyrus-SASL w Debianie	385
20.9. Konfiguracja smtp-auth do uwierzytelniania użytkowników	386
20.10. Uwierzytelnianie Postfiksa na innym serwerze za pomocą smtp-auth	388
20.11. Konfiguracja pełnej nazwy domeny	389
20.12. Instalacja serwera poczty IMAP	390
20.13. Podłączanie użytkowników	392
20.14. Udostępnianie folderów IMAP	393

20.15. Domeny wirtualnych skrzynek pocztowych w Postfixie	394
20.16. Tworzenie listy dystrybucyjnej z pomocą couriermlm	397
20.17. Zarządzanie listą couriermlm	399
20.18. Dostęp do poczty przez WWW	400
20.19. Tabele kodów odpowiedzi i poleceń SMTP	401
21. Walka ze spamem i złośliwymi programami.....	403
21.1. Wprowadzenie	403
21.2. Podstawowa lista kontrolna: przygotowania do tworzenia zabezpieczeń przed spamem i malware	405
21.3. Bezpieczne testowanie nowych reguł antyspamowych w Postfixie	406
21.4. Podstawowe konfiguracje reguł antyspamowych w Postfixie	407
21.5. Tworzenie białych list	410
21.6. Czarne listy oparte na DNS-ie	411
21.7. Odrzucanie wiadomości z załącznikami	413
21.8. Instalacja Clam Anti-Virus na serwerze Postfiksa	414
21.9. Instalacja SpamAssassin na serwerze Postfiksa z Amavisd-new	417
21.10. Instalacja programu SpamAssassin bez Amavisd-new	418
22. Serwer WWW Apache	421
22.1. Wprowadzenie	421
22.2. Instalacja Apache 2.0 z kodu źródłowego	425
22.3. Dodawanie nowych modułów po instalacji	428
22.4. Ustawienia uprawnień i własności plików Apache	429
22.5. Lokalny dostęp do podręcznika użytkownika Apache	430
22.6. Konfiguracja prostego publicznego serwera WWW	431
22.7. Przekierowanie URL do nowego katalogu	432
22.8. Przydzielenie użytkownikom indywidualnych katalogów WWW	432
22.9. Uruchamianie Apache przy starcie systemu	433
22.10. Prowadzenie wielu domen w Apache	434
22.11. Używanie osobnych plików dziennika dla hostów wirtualnych	436
22.12. Blokowanie dostępu do witryn lokalnych z internetu	437
22.13. Zabezpieczanie wybranych katalogów hasłami	438
22.14. Kontrola nad robotami przeszukującymi WWW za pomocą robots.txt	440
22.15. Blokowanie niepożądanych gości	442
22.16. Tworzenie własnych stron obsługujących błędy	442
22.17. Modyfikacje domyślnych stron błędów w Apache	443
22.18. Tworzenie czytelnych indeksów katalogów	444
22.19. Dostarczanie stron w różnych językach z użyciem Content Negotiation	445
22.20. Używanie ikon favicon	447
22.21. Przeglądanie dzienników dostępu Apache za pomocą narzędzia Webalizer	448

23. Udostępnianie plików i drukarek oraz uwierzytelnianie w domenach za pomocą Samby.....	451
23.1. Wprowadzenie	451
23.2. Prosty anonimowy serwer plików Samba dla systemów Windows	453
23.3. Tworzenie sieci komputerów równorzędnych Windows i Linux	456
23.4. Włączanie udostępniania plików w komputerach Windows	457
23.5. Uwierzytelnianie na serwerze Samby	460
23.6. Wsadowa konwersja użytkowników systemowych na użytkowników Samby	462
23.7. Logowanie do Samby z Windows 95/98/Me	463
23.8. Rozwiązywanie problemów z szyfrowaniem haseł w systemach Windows	463
23.9. Kontrola dostępu do udziału za pomocą ACL	464
23.10. Tworzenie udziałów publicznych dla użytkowników	465
23.11. Dostęp do katalogów macierzystych użytkowników za pomocą Samby	466
23.12. Tworzenie podstawowego kontrolera domeny za pomocą Samby	467
23.13. Dołączanie Windows 95/98/Me do domeny obsługiwanej przez Sambę	470
23.14. Dołączanie Windows NT/2000 do domeny obsługiwanej przez Sambę	470
23.15. Dołączanie Windows XP do domeny obsługiwanej przez Sambę	471
23.16. Obsługa profili mobilnych	472
23.17. Podłączanie klientów linuksowych do serwera plików lub sieci równorzędnej w Sambie	473
23.18. Podłączanie klientów linuksowych do grup roboczych Samby za pomocą narzędzi wiersza poleceń	476
23.19. Łączenie klientów linuksowych z domeną Samby za pomocą przeglądarek graficznych	478
23.20. Łączenie klientów linuksowych z domeną Samby za pomocą narzędzi wiersza poleceń	478
23.21. Synchronizacja haseł Samby i Linuksa	479
23.22. Udostępnianie drukarek linuksowych w Windows	480
23.23. Udostępnianie drukarek Windows dla systemów Linux	481
23.24. Uruchamianie aplikacji Windows pod Linuksem za pomocą CrossOver Office	482
24. Rozwiązywanie nazw	487
24.1. Wprowadzenie	487
24.2. Lokalne rozwiązywanie nazw za pomocą plików hosts	489
24.3. Instalacja serwera DHCP	491
24.4. Konfiguracja klientów DHCP	492
24.5. Dodawanie statycznych hostów do DHCP	493
24.6. Prowadzenie publicznego serwera DNS	494
24.7. Instalacja djbdns	495
24.8. Przenoszenie plików dzienników tinydns i dnscache	496

24.9. djbdns jako lokalny buforujący serwer nazw	497
24.10. Konfiguracja klientów linuksowych i Windows do współpracy z buforującym serwerem DNS	499
24.11. Publiczny serwer DNS wykorzystujący tinydns	501
24.12. Instalacja prywatnego serwera tinydns	503
24.13. Proste równoważenie obciążenia za pomocą tinydns	504
24.14. Synchronizacja z drugim serwerem tinydns	505
24.15. Lokalny buforujący serwer nazw BIND	506
24.16. Prywatny serwer DNS BIND	508
24.17. Kontrola składni	512
24.18. Konfiguracja publicznego serwera DNS BIND	512
24.19. Konfiguracja wtórnego serwera BIND	515
24.20. Proste równoważenie obciążenia za pomocą serwera BIND	517
24.21. Testowanie serwera tinydns	518
24.22. Testowanie i odpytywanie serwerów DNS za pomocą narzędzi dig i dnstrace	518
A Źródła dokumentacji Linuksa	521
B Źródła online	525
C Typy plików Microsoftu	527
D Skrypt startowy CVSD.....	531
Skorowidz.....	533

Instalacja programów z kodu źródłowego

4.1. Wprowadzenie

Mimo wszystkich dostępnych menedżerów pakietów i narzędzi do rozwiązywania zależności bywa, że preferowana jest kompilacja z kodu źródłowego. Na przykład potrzebny nam program może być niedostępny w postaci pakietu, możemy potrzebować ścisłej kontroli nad tym, które opcje i funkcje mają być wbudowane, lub chcemy zoptymalizować program dla konkretnej architektury. Wielu doświadczonych administratorów zaleca kompilację z kodu źródłowego programów krytycznych dla bezpieczeństwa (np. ssh i wszystkie serwery).

Przy kompilacji z kodu źródłowego zaleca się przeczytanie wszystkich instrukcji. Wprawdzie procedura złożona z poleceń `configure — make — make install` jest w miarę standardowa, lecz zdarza się wiele wyjątków spowodowanych dziwactwami autorów. Często dostępnych jest wiele opcji konfiguracji, o których mówi tylko dokumentacja programu.

Jeśli ktoś woli korzystać z pakietów, to wszystkie narzędzia do budowy własnych pakietów RPM i `.deb` są dostępne za darmo. Jednakże krzywa uczenia się używania RPM lub budowy własnych `.deb` jest dość stroma. Jest jeszcze trzecia opcja — `CheckInstall`. Program ten jest doskonałym narzędziem pozwalającym z łatwością budować z kodu źródłowego własne pakiety RPM, Debiana lub Slackware.

4.2. Przygotowanie systemu do kompilacji programów ze źródeł

Problem

Wiemy, że do budowania programów ze źródeł potrzebny jest kompilator i, być może, jeszcze jakieś inne narzędzia, lecz nie wiemy dokładnie jakie.

Rozwiązanie

Potrzebne będą programy z dwóch kategorii:

- Niezbędne narzędzia programistyczne wspólne dla wszystkich systemów linuksowych.
- Konkretnie biblioteki lub narzędzia zależne od kompilowanego programu.

Oto lista typowych narzędzi programistycznych dla Linuksa:

GNU coreutils

Duży zbiór niezbędnych narzędzi systemowych: *shellutils* (narzędzia powłoki), *fileutils* (narzędzia plikowe) i *textutils* (narzędzia tekstowe). Pełna lista znajduje się pod adresem <http://www.gnu.org/software/coreutils/>; patrz też *info coreutils*.

GNU binutils

Narzędzia do pracy na plikach binarnych (<http://www.gnu.org/software/binutils/>).

gcc

GNU Compiler Collection — zbiór kompilatorów obejmujący języki C, C++, Objective-C, Fortran, Java i Ada oraz biblioteki dla nich.

GNU tar

Narzędzie archiwizacyjne do obsługi archiwów tar kodu źródłowego (z rozszerzeniem *.tar*).

gunzip

Narzędzie do kompresji często używane w połączeniu z tar (rozszerzenie *.tar.gz*).

bunzip2

Format superkompresji do pakowania i rozpakowywania archiwów tar (rozszerzenie *.bz2*).

make

To polecenie odczytuje opcje konfiguracji i inicjuje kompilację plików programu.

Dokumentacja dla kompilowanej aplikacji powinna powiedzieć o wszystkim, co jest niezbędne do pomyślnej kompilacji.

Analiza

W większości dystrybucji Linuksa dostępna jest opcja instalacji „Core Development Tools” lub podobna, więc nie musimy polować na narzędzia i instalować ich indywidualnie.

Trzeba będzie przeczytać dokumentację kompilowanej aplikacji, aby wyszukać wszelkie wymagania dla danego programu. Archiwum tar kodu źródłowego powinno zawierać pliki *README*, *INSTALL* lub inną dokumentację. Radzę przeczytać wszystko. Skrypt konfiguracyjny po uruchomieniu sprawdza w systemie, czy wszystkie potrzebne elementy są obecne. Jeśli cokolwiek brakuje, skrypt kończy pracę i zwraca komunikat o błędzie, określając, co jeszcze jest potrzebne.

Zobacz również

- Rozdział 14. książki *LPI Linux Certification in a Nutshell* Jeffa Deana (O'Reilly)

4.3. Generowanie listy plików przy instalacji ze źródeł w celu ułatwienia deinstalacji

Problem

Chcemy wiedzieć, które pliki są instalowane w systemie przy instalacji programu z kodu źródłowego, co pozwoli znaleźć i usunąć je wszystkie, gdy zdecydujemy, że program jest już niepotrzebny. Niektórzy autorzy zapobiegliwie udostępniają opcję „make uninstall”, co pozwala całkowicie odinstalować program, lecz wielu tak nie robi.

Rozwiązanie

Możemy za pomocą standardowych narzędzi linuksowych wygenerować listę wszystkich plików w systemie przed instalacją, a następnie listę plików po instalacji. Porównanie tych dwóch list poleceniem `diff` pozwoli zbudować listę świeżo zainstalowanych plików. Poniższy przykład opiera się na JOE (ang. *Joe's Own Editor*):

```
# find / | grep -v -e ^/proc/ -e ^/tmp/ -e ^/dev/ > joe-preinstall.list
```

Po skompilowaniu i zainstalowaniu nowego programu wygenerujemy drugą listę:

```
# find / | grep -v -e ^/proc/ -e ^/tmp/ -e ^/dev/ > joe-postinstall.list
```

a następnie za pomocą `diff` listę plików zainstalowanych przez Joe:

```
# diff joe-preinstall.list joe-postinstall.list > joe-installed.list
```

Analiza

Używając `find` w połączeniu z `grep`, możemy z łatwością wykluczyć katalogi, które nie mają znaczenia dla ostatecznej listy. Opcja `-v` polecenia `grep` odwraca wzorzec wyszukiwania, tak że są wyszukiwane wiersze niezgodne z podanym wzorcem. `-e` definiuje wzorzec wyszukiwania, natomiast `^` oznacza początek wiersza. Razem mówi to: „Ignoruj wiersze zaczynające się od...”

Nie musimy zawracać sobie głowy katalogami `/proc` i `/tmp`, ponieważ ich zawartość nieustannie się zmienia. Plikami `/dev` zarządza system, więc również możemy je zignorować. Jest to też ważny środek bezpieczeństwa — gdy usuwamy program ręcznie, używając wygenerowanej listy różnic, `/proc`, `/tmp` i `/dev` są katalogami, których nie powinniśmy w żadnym przypadku ruszać.

Zobacz również

- `grep(1)`, `find(1)`, `diff(1)`

4.4. Instalacja programów z kodu źródłowego

Problem

Chcemy zainstalować program z kodu źródłowego, lecz trudno jest poruszać się po gęstwinie archiwów `tar`, plików `makefile` itp.

Rozwiązanie

Najpierw należy rozpakować archiwum tar, a następnie skonfigurować (`configure`), skompilować (`make`) i zainstalować (`install`) program.

Zacniemy pracę w katalogu mieszczącym archiwum tar i drzewo katalogów źródeł. Poniższy przykład opiera się na JOE (Joe's Own Editor):

```
# cd /usr/src/downloads
# tar zxvf joe-2.9.8.tar.gz
# cd joe-2.9.8
# ls
# less README
# less INFO
# ./configure --help
# ./configure <opcje, jeśli są potrzebne>
# make
# make install | tee joe-makeinstall
```

Ostatnie polecenie zapisuje wyjście procedury instalacji do pliku tekstowego *joe-makeinstall*.

Niektóre programy są archiwizowane za pomocą narzędzia *bzip2* zamiast bardziej tradycyjnego *gzip*. Archiwa *.bz2* rozpakowuje się następująco:

```
# tar jxvf joe-2.9.8.tar.bz2
```

Do odinstalowania programu skompilowanego z kodu źródłowego posłużysz poleceniu:

```
# make uninstall
```

Zadziała to tylko wtedy, jeśli autor programu przewidział opcję `make uninstall`. Przekierowanie wyjścia polecenia `make install` do pliku tekstowego pozwala zapamiętać informacje, które mogą się przydać, gdy trzeba będzie usunąć wszystkie pliki ręcznie. Listę możemy też wygenerować tak jak w recepturze 4.3.

Analiza

Kroki opisane w niniejszej recepturze są standardowym procesem instalacji programów z kodu źródłowego. Jednakże nie wszyscy autorzy programów posługują się tą samą procedurą. Przed rozpoczęciem pracy powinniśmy przejrzeć dokumentację programu.

Najważniejsze jest zapoznanie się z opcjami *configure*. Niektóre programy, np. Apache, mają dziesiątki opcji kompilacji. Z punktu widzenia bezpieczeństwa rozsądnie jest wkompiłować obsługę tylko tego, co naprawdę będzie potrzebne. Jest to szczególnie ważne w przypadku serwerów mających kontakt z niezaufanymi sieciami, na przykład serwerów WWW i poczty.

Kompilacja programów z kodu źródłowego ma następujące zalety:

- Możemy skonfigurować dokładnie takie opcje, jakich nam potrzeba.
- Możemy zoptymalizować program dla konkretnej architektury systemu.
- Mamy absolutną kontrolę nad tym, co ma być zainstalowane.

Wady też się znajdują:

- Uaktualnienia i usuwanie programów może być nieprzyjemne.
- Od piekła zależności dzieli tylko jeden krok.
- Kompilacja dużego programu może trwać godzinami.

Pewne serwery powinny być kompilowane z kodu źródłowego. Na przykład serwer WWW Apache naprawdę wymaga kompilacji z kodu źródłowego, aby pozwolić na pełne dostosowanie do konkretnych potrzeb i optymalizację.

W przypadku systemów biurkowych możemy sobie to darować. Są zbyt duże i złożone. Dla nich optymalne będą wygodne dystrybucje Linuksa oparte na pakietach.

Zobacz również

- *info tar, make(1), bzip2(1)*

4.5. Tworzenie pakietów z kodu źródłowego za pomocą CheckInstall

Problem

Chcemy stworzyć z kodu źródłowego pakiet dla systemu Slackware, Red Hat lub Debian, ponieważ dana aplikacja nie jest dostępna w postaci potrzebnego nam pakietu. Z lektury dotyczącej budowania pakietów wynika, że jest to bardzo skomplikowane zadanie. Czy jest jakiś łatwiejszy sposób?

Rozwiązanie

Posłużymy się CheckInstall. Ponownie za przykład weźmiemy Joe's Own Editor. Dla systemu Debian procedura wygląda tak:

```
# mkdir /doc-pak
# tar zxvf joe-2.9.8.tar.gz
# cd joe-2.9.8
# ./configure
# make
# checkinstall -D
```

Narzędzie CheckInstall zastępuje polecenie *make install*, musi więc zostać uruchomione w katalogu głównym drzewa katalogów kodu źródłowego. Dalej wystarczy postępować zgodnie ze wskazówkami. CheckInstall zbuduje i zainstaluje pakiet *.deb*, co możemy sprawdzić:

```
$ dpkg -l | grep joe
ii joe          2.9.8-1      joe's own editor
```

I to wszystko. Program został zainstalowany i jest gotowy do pracy. Kopia pakietu pozostała w katalogu źródłowym.

Do zbudowania pakietu Slackware posłuży polecenie:

```
# checkinstall -S
```

Do zbudowania pakietu RPM:

```
# checkinstall -R
```

Analiza

W katalogu *doc-pak* CheckInstall umieszcza plik *README* i pozostałą dokumentację programu. Jeśli nie utworzymy tego katalogu, CheckInstall zapyta, czy chcemy utworzyć domyślny katalog na dokumentację. Jeżeli odpowiemy, że nie, pakiet nie będzie zawierał dokumentacji.

CheckInstall wykorzystuje macierzyste menedżery pakietów dla instalowanego programu: RPM w systemie Red Hat, *installpkg* w Slackware i *.apt* w Debianie. Do usunięcia pakietu CheckInstall możemy po prostu użyć menedżera pakietów z naszego systemu.

CheckInstall obsługuje wszelkie skrypty instalacyjne, na przykład:

```
# checkinstall -D make install_packages
# checkinstall -R make modules_install
# checkinstall -S install.sh
# checkinstall -D setup
```

Przypominam o przeczytaniu pliku *README* programu i innej dołączonej dokumentacji. Nie wszystkie pakiety źródłowe używają tradycyjnej procedury *configure-make-make install*. Niektóre wykorzystują skrypty instalacyjne, jak powyższe przykłady.

CheckInstall jak na razie nie pozwala na tworzenie pakietu bez automatycznego zainstalowania, lecz to może się zmienić w przyszłych wersjach.

Zobacz również

- Strona macierzysta CheckInstall (<http://asic-linux.com.mx/~izto/checkinstall/news.php>)