

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

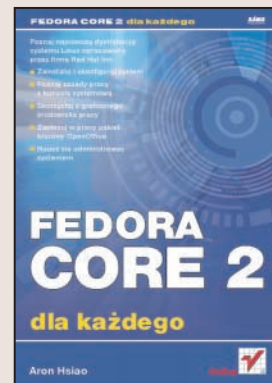
Fedora Core 2 dla każdego

Autor: Aron Hsiao

ISBN: 83-7361-552-0

Tytuł oryginału: [Teach Yourself
Red Hat Linux Fedora in 24 Hours](#)

Format: B5, stron: 528



Fedora Core 2 to otwarta platforma linuksowa przeznaczona do użytku osobistego albo dla małych firm. Jej twórca – firma Red Hat Inc. wykorzystuje ją w roli podstawy dla systemów klasy wyższej, takich jak Red Hat Enterprise Linux. Systemy operacyjne firmy Red Hat pozostają najlepiej znaną i najbardziej lubianą spośród wszystkich odmian Linuksa. Uważa się, że to właśnie dzięki firmie Red Hat Linux trafił ze świata idei na komercyjny rynek. Systemy operacyjne Red Hat cechują się także doskonałym zrównoważeniem mocy, wygody instalacji i łatwości zarządzania. Jeśli chcesz wypróbować ten system, otwarta platforma firmy Red Hat – Fedora Core 2 – nadaje się do tego wręcz idealnie.

Książka „Fedora Core 2 dla każdego” to wprowadzenie do systemu Fedora Core 2, przeznaczone dla osób, które nigdy wcześniej nie miały do czynienia z systemami linuksowymi. Zawiera informacje związane z instalacją i konfiguracją systemu, wykorzystywaniem go w domu i pracy oraz podstawami administracji Fedorą.

- Podział dysku na partycje i instalacja systemu
- Pierwsze uruchomienie Linuksa
- Praca z systemem plików
- Korzystanie z konsoli systemowej
- System składu dokumentów LaTeX
- Korzystanie z internetu z poziomu powłoki systemowej
- Tworzenie skryptów powłoki
- Graficzne środowiska pracy GNOME i KDE
- Pakiet biurowy OpenOffice
- System X Window
- Administracja systemem w środowisku tekstowym i graficznym
- Mechanizmy zabezpieczeń
- Instalowanie oprogramowania
- Konfigurowanie usług sieciowych
- Tworzenie kopii zapasowych



Spis treści

O Autorze	17
Wstęp	19
Część I Instalowanie systemu Fedora.....	23
Rozdział 1. Przygotowania do instalacji systemu Fedora	25
Spis sprzętu	25
Pojemność pamięci.....	26
Szybkość procesora.....	27
Pojemność dysku twardego	28
Sprzęt komunikacyjny.....	28
Ocena sprzętu	29
Jak zrobić miejsce na system Linux?.....	30
Co to są partycje?.....	30
Repartycjonowanie niedestrukcyjne.....	31
Defragmentacja dysku przed użyciem FIPS.....	32
Tworzenie dyskietki FIPS i uruchamianie programu FIPS	33
Korzystanie z programu FIPS	35
Repartycjonowanie destrukcyjne	38
Unikanie repartycjonowania przez dodanie dysku twardego	40
Uruchamianie instalatora Fedory.....	41
Podsumowanie	41
Pytania i odpowiedzi	41
Warsztat.....	42
Quiz.....	42
Odpowiedzi.....	42
Ćwiczenia.....	42
Rozdział 2. Instalowanie systemu Fedora	43
Uruchamianie instalatora Fedory.....	43
Rozpoczęcie instalacji	43
Wybór języka	44
Konfiguracja klawiatury.....	44
Konfiguracja myszy	45
Konfiguracja monitora	46
Typ instalacji.....	47
Konfiguracja partycjonowania dysku.....	48
Konfigurowanie dysku.....	49

Konfiguracja programu rozruchowego.....	56
Konfiguracja sieci	58
Konfiguracja zapory sieciowej.....	59
Obsługa dodatkowych języków.....	60
Wybór strefy czasowej.....	60
Konfiguracja konta.....	61
Wybieranie grup pakietów	62
Przed instalacją	64
Instalacja pakietów.....	64
Gratulacje!.....	65
Podsumowanie	65
Pytania i odpowiedzi	66
Warsztat.....	66
Quiz.....	66
Odpowiedzi.....	66
Rozdział 3. Uruchamianie systemu, logowanie i konfiguracja	67
Uruchamianie Fedory	67
Witaj w systemie Fedora Core 2!	69
Logowanie się do systemu	69
Części ekranu logowania.....	69
Konfigurowanie drukarki i połączenia internetowego.....	71
Konfigurowanie drukarki	71
Konfigurowanie telefonicznego połączenia internetowego.....	73
Uruchamianie narzędzia Network Configuration	76
Wylogowywanie się.....	76
Podsumowanie	77
Pytania i odpowiedzi	77
Warsztat.....	78
Quiz.....	78
Odpowiedzi.....	78
Ćwiczenia.....	78
Część II Korzystanie z konsoli Linuksa	79
Rozdział 4. Nawigowanie po systemie Linuks za pomocą konsoli	81
Po co uczyć się obsługi Linuksa za pomocą konsoli?	81
Konsole wirtualne	82
Przełączanie konsoli.....	82
Logowanie się do konsoli wirtualnej.....	83
Podstawowe informacje o powłoce	83
Powłoka jako interpreter poleceń.....	83
Zasady korzystania z powłoki	84
Praca z systemem plików	85
Wiele drzew, wiele korzeni: system plików Windows.....	85
Jedno drzewo, jeden korzeń: system plików Linuksa.....	86
Katalogi i pliki	86
Katalog domowy	88
Katalog roboczy	89
Manipulowanie plikami i katalogami.....	91
Ścieżki względne.....	91
Usuwanie plików i katalogów	95
Pliki wykonywalne.....	95
Dowiązania symboliczne.....	96

Prawa dostępu	97
Długie listingi plików	98
Tożsamość i przynależność plików	99
Przynależność i prawa dostępu do plików	100
Więcej przykładów praw dostępu	101
Zmiana praw dostępu	102
Podsumowanie	104
Pytania i odpowiedzi	104
Warsztat	105
Quiz	105
Odpowiedzi	105
Ćwiczenia	106
Rozdział 5. Zaprzęganie konsoli do pracy	107
Tworzenie, edytowanie i zapisywanie plików tekstowych	107
Tworzenie plików tekstowych za pomocą edytora vim	108
Wstawianie tekstu w edytorze vi	109
Edycja tekstu w edytorze vi	110
Zapisywanie pliku, zamykanie edytora i dalsza lektura	111
Tworzenie plików tekstowych za pomocą edytora emacs	112
System menu w edytorze emacs	113
Najważniejsze polecenia klawiaturowe edytora emacs	114
Efektywne zarządzanie grupami plików	116
Grupowanie plików w wierszu poleceń	116
Zapobieganie rozwijaniu nazw plików	118
Szybkie wyszukiwanie plików i katalogów	119
Szukanie plików za pomocą polecenia find	120
Przeszukiwanie całego systemu plików za pomocą polecenia locate	120
Zapisywanie listy znalezionych plików	122
Wyszukiwanie wzorców w plikach tekstowych	122
Wyszukiwanie plików, które zawierają określone słowa	123
Używanie wyników poleceń do złożonych zadań	124
Łączenie poleceń za pomocą potoków	124
Używanie wyników jednego polecenia jako argumentów drugiego	126
Sterowanie programami powłoki	127
Przełączanie otwartych aplikacji	127
Wznawianie zadania za pomocą polecenia fg	128
Uruchamianie zadania w tle za pomocą polecenia bg	129
Ostatnie uwagi o sterowaniu zadaniami	130
Podsumowanie	130
Pytania i odpowiedzi	131
Warsztat	132
Quiz	132
Odpowiedzi	132
Ćwiczenia	132
Rozdział 6. Uzyskiwanie pomocy podczas korzystania z konsoli	133
Strony podręcznika systemowego	133
Korzystanie ze stron podręcznika: podstawy	133
Rozdziały podręcznika	135
Lokalizowanie stron podręcznika za pomocą wyszukiwania tematycznego	136
Korzystanie z systemu GNU info	137
Nawigacja po systemie info	138
Efektywne korzystanie z systemu info	139

Drzewo katalogów /usr/share/doc	139
Czytanie dodatkowych dokumentów dołączonych do aplikacji.....	139
Przeszukiwanie drzewa /usr/share/doc	140
Pomoc zawarta w poleceniach.....	143
Podsumowanie	144
Pytania i odpowiedzi	144
Warsztat.....	145
Quiz.....	145
Odpowiedzi.....	145
Ćwiczenia.....	145
Rozdział 7. Praca bez myszy	147
Drukowanie z poziomu wiersza poleceń	147
Tworzenie zadań wydruku	148
Wyświetlanie zadań wydruku	148
Usuwanie zadań z kolejki.....	148
Tworzenie dokumentów o wysokiej jakości.....	149
Program formatujący LaTeX-2e	150
Podstawy LaTeX-a.....	150
Tworzenie pustego dokumentu LaTeX-a	151
Formatowanie i drukowanie pierwszego dokumentu	153
Wybór stylu strony.....	154
Tytuł dokumentu i nazwisko autora	155
Kontrolowanie akapitów, podziałów wiersza i stron.....	155
Organizowanie dłuższych tekstów	156
Formatowanie treści dokumentu	158
Wstawianie znaków specjalnych.....	160
Wszystko razem	160
Więcej informacji o LaTeX-u	162
Wykonywanie obliczeń matematycznych za pomocą kalkulatora binarnego.....	163
Uruchamianie programu bc i wykonywanie podstawowych obliczeń.....	163
Używanie zmiennych.....	164
Automatyzowanie obliczeń w programie bc	164
Pisanie zaawansowanych skryptów programu bc.....	165
Tworzenie i sortowanie list danych.....	165
Tworzenie list, które można przeszukiwać lub sortować	166
Wyświetlanie żądanych pozycji	166
Sortowanie danych na liście.....	167
Podsumowanie	168
Pytania i odpowiedzi	169
Warsztat.....	169
Quiz.....	169
Odpowiedzi.....	170
Ćwiczenia.....	170
Rozdział 8. Używanie sieci bez grafiki	171
Przeglądanie sieci WWW na konsoli	171
Uruchamianie przeglądarki Lynx	172
Korzystanie z przeglądarki Lynx.....	172
Obsługa plików cookie w programie Lynx	173
Zarządzanie zakładkami w programie Lynx.....	175
Zarządzanie pocztą e-mail na konsoli.....	176
Pobieranie wiadomości za pomocą protokołów Post Office Protocol 3	
lub Internet Message Access Protocol	176
Tworzenie pliku .fetchmailrc	177
Zarządzanie pocztą elektroniczną za pomocą programu mutt	178

Logowanie się w zdalnym systemie linuksowym lub uniksowym	181
Zdalne logowanie za pomocą programu telnet.....	182
Zdalne logowanie za pomocą programu ssh.....	183
Wymiana plików z komputerami linuksowymi i uniksowymi za pomocą programu ftp... 184	
Uruchamianie programu ftp i logowanie się	184
Nawigacja podczas sesji ftp	186
Przykładowa sesja ftp.....	187
Wymiana plików z systemami Windows za pomocą programu smbclient	189
Wyświetlanie udziałów w komputerze Windows.....	189
Łączenie się z udziałem Windows	189
Nawigacja i kopiowanie plików w programie smbclient.....	190
Podsumowanie	191
Pytania i odpowiedzi	191
Warsztat.....	192
Quiz.....	192
Odpowiedzi.....	192
Ćwiczenia.....	192
Rozdział 9. Ujarzmianie potęgi powłoki.....	193
Rozszerzamy repertuar poleceń.....	193
Wysyłanie tekstu na standardowe wyjście za pomocą polecenia echo.....	194
Wykonywanie prostych obliczeń za pomocą polecenia expr	194
Wyświetlanie początku lub końca pliku tekstowego za pomocą poleceń head i tail ...	195
Edytowanie strumieni danych za pomocą polecenia sed	195
Zmienne powłoki i przytaczanie.....	197
Tworzenie i podstawianie zmiennych	197
Przytaczanie	198
Zmienne środowiskowe.....	199
Tworzenie własnych poleceń za pomocą skryptów powłoki	200
Początek skryptu powłoki	201
Przetwarzanie argumentów wiersza polecenia	201
Zmiana skryptu w plik wykonywalny	202
Używanie poleceń warunkowych.....	203
Wielokrotne testowanie.....	206
Wielokrotne wykonywanie operacji na wstępnie zdefiniowanym zbiorze.....	207
Oprócz skryptów powłoki	209
Podsumowanie	211
Pytania i odpowiedzi	211
Warsztat.....	212
Quiz.....	212
Odpowiedzi.....	212
Ćwiczenia.....	213
Część III Korzystanie z pulpitu Linuksa.....	215
Rozdział 10. Wprowadzenie do pulpitu Fedory	217
Uwagi dotyczące środowisk GNOME i KDE w Fedorze.....	217
Logowanie się na pulpicie.....	218
Nawigacja po pulpicie.....	220
Uruchamianie aplikacji	221
Używanie przycisków sterujących oknem	223
Przesuwanie, zmiana wymiarów, minimalizowanie i maksymalizowanie okien	223
Menu aplikacji.....	224

Praca z wieloma oknami.....	224
Zmiana aktywnej aplikacji	225
Minimalizowanie i przywracanie okien za pomocą paska zadań	226
Wirtualne obszary robocze.....	226
Który obszar roboczy jest aktywny?	227
Wybieranie nowego obszaru roboczego.....	227
Przenoszenie działającej aplikacji do innego obszaru roboczego.....	228
Korzystanie z menu zarządzania oknem w KDE.....	228
Wylogowywanie się z pulpitu GNOME.....	229
Podsumowanie	229
Pytania i odpowiedzi	230
Warsztat.....	231
Quiz.....	231
Odpowiedzi.....	231
Ćwiczenia.....	231
Rozdział 11. Praca z plikami na pulpicie.....	233
Tworzenie nowego pliku tekstowego za pomocą edytora tekstu.....	233
Korzystanie z menedżera plików.....	236
Otwieranie okna menedżera plików	236
Nawigacja po drzewie katalogów.....	238
Praca z plikami i katalogami	239
Otwieranie, edycja i zamykanie istniejącego pliku	240
Wycinanie, kopiowanie i wklejanie plików	241
Powielanie pliku w bieżącym katalogu	242
Zaznaczanie wielu plików	242
Tworzenie dowiązania symbolicznego.....	243
Zmiana nazwy pliku.....	244
Usuwanie plików.....	244
Zmiana praw dostępu do plików	244
Tworzenie nowego katalogu	246
Porządkowanie lub sortowanie ikon.....	246
Manipulowanie plikami za pomocą przeciągania i upuszczania	248
Przenoszenie pliku do katalogu albo na pulpit	248
Przesuwanie pliku między dwoma oknami katalogów.....	249
Kontekstowe przeciąganie i upuszczanie	250
Praca z koszem	250
Przywracanie usuniętych plików.....	251
Opróżnianie kosza.....	251
Podsumowanie	252
Pytania i odpowiedzi	252
Warsztat.....	252
Quiz.....	252
Odpowiedzi.....	252
Ćwiczenia.....	253
Rozdział 12. Wprowadzenie do pakietu OpenOffice	255
Aplikacje OpenOffice.....	255
Uruchamianie aplikacji OpenOffice.....	256
Tworzenie i formatowanie dokumentu w programie OpenOffice Writer	256
Uruchamianie programu OpenOffice Writer.....	257
Wprowadzanie tekstu	258
Edycja tekstu	259
Zmiana wyglądu tekstu	260
Zmiana formatowania akapitów	262

Zmiana rozmiaru marginesów i układu strony	263
Zapisywanie pliku	264
Otwieranie pliku	265
Drukowanie pliku	266
Praca z programem OpenOffice Calc	267
Wprowadzenie do programu Calc	267
Wpisywanie etykiet tekstowych	269
Wprowadzanie i formatowanie danych liczbowych	270
Wprowadzanie formuł	272
Używanie funkcji w formułach	274
Kopiowanie formuł	276
Drukowanie, zapisywanie i otwieranie arkuszy kalkulacyjnych	277
Podsumowanie	278
Pytania i odpowiedzi	279
Warsztat	279
Quiz	279
Odpowiedzi	280
Ćwiczenia	280
Rozdział 13. Sieć na pulpicie	281
Wprowadzenie do Mozilli	281
Uruchamianie przeglądarki Mozilla	281
Przeglądanie sieci WWW za pomocą Mozilli	282
Odwiedzanie witryny WWW	282
Nawigacja po witrynach WWW	284
Zapisywanie adresów	285
Przeglądanie stron na kartach	286
Wyłączanie okien wyskakujących	286
Zamykanie Mozilli	287
Czytanie i pisanie listów	287
Uruchamianie programu Evolution	288
Konfigurowanie programu Evolution	288
Pisanie nowej wiadomości e-mail	292
Pisanie wiadomości w formacie HTML	294
Dołączanie pliku do wiadomości	295
Wysyłanie wiadomości	296
Pobieranie nowej poczty	297
Odpowiadanie na wiadomość lub przesyłanie jej do innego użytkownika	297
Dostęp do załącznika	298
Drukowanie i usuwanie	298
Zamykanie programu Evolution	299
Dostęp do sieci Windows	299
Dostęp do plików Windows na pulpicie GNOME	300
Korzystanie z protokołu FTP	300
Łączenie się ze zdalnym systemem	301
Kopiowanie plików do zdalnego systemu	302
Kopiowanie plików ze zdalnego systemu	303
Zamykanie połączenia FTP	303
Podsumowanie	303
Pytania i odpowiedzi	304
Warsztat	304
Quiz	304
Odpowiedzi	304
Ćwiczenia	305

Rozdział 14. Uzyskiwanie pomocy podczas pracy na pulpicie	307
Korzystanie z pomocy w aplikacjach	307
Znajdowanie i uruchamianie pomocy w aplikacjach.....	308
Opcja Informacje o (About) w menu Pomoc (Help)	309
Korzystanie z opcji Co to jest? w aplikacjach KDE.....	310
Przeglądanie zawartości pomocy w aplikacjach GNOME	310
Przeglądanie podręczników aplikacji KDE	312
Używanie systemowej pomocy w środowiskach GNOME i KDE	313
Uruchamianie i używanie przeglądarki pomocy GNOME.....	313
Uruchamianie i używanie Centrum pomocy KDE	314
Czytanie stron man i info za pomocą programu KDE Konqueror.....	316
Podsumowanie	316
Pytania i odpowiedzi	317
Warsztat.....	318
Quiz.....	318
Odpowiedzi.....	318
Ćwiczenia.....	318
Rozdział 15. Dostosowywanie pulpitu do własnych potrzeb	319
Korzystanie z centrum sterowania GNOME	319
Modyfikacja działania myszy.....	319
Zmiana wyglądu okien.....	322
Zmiana tapety pulpitu	324
Zmiana wygaszacza ekranu.....	325
Zmiana rozdzielczości i częstotliwości odświeżania.....	327
Zmiana innych ustawień pulpitu	328
Korzystanie z Centrum sterowania KDE.....	329
Zmiana właściwości myszy.....	329
Zmiana stylu okien.....	331
Zmiana kolorów aplikacji.....	333
Zmiana krawędzi i pasków tytułu okien.....	334
Zmiana tapety pulpitu	335
Zmiana wygaszacza ekranu.....	336
Konfigurowanie paska zadań.....	337
Dodawanie ikony do paska zadań w środowisku GNOME.....	338
Dodawanie ikony do paska zadań w środowisku KDE	338
Przesuwanie ikony na pasku zadań	339
Usuwanie ikony z paska zadań.....	339
Dodatkowa konfiguracja paska zadań	339
Podsumowanie	340
Pytania i odpowiedzi	340
Warsztat.....	341
Quiz.....	341
Odpowiedzi.....	341
Ćwiczenia.....	341
Rozdział 16. Zaawansowane korzystanie z pulpitu Linuksa	343
Aplikacja terminala	343
Uruchamianie aplikacji terminala.....	344
Uruchamianie aplikacji pulpitowych za pomocą terminala.....	345
Sterowanie zadaniami w wierszu poleceń terminala.....	345
Zachowywanie działających zadań za pomocą polecenia nohup	347
Dostęp z wiersza polecenia do plików na pulpicie.....	347

Korzystanie z aplikacji systemu X Window.....	348
Używanie terminala bez GNOME lub KDE	349
Używanie edytora emacs na pulpicie	349
Używanie aplikacji pomocniczych systemu X Window	350
Interakcja z użytkownikiem w skryptach powłoki	351
Polecenie xmessage w skryptach powłoki.....	352
Przykładowy skrypt xmessage	352
Drugi przykładowy skrypt xmessage	353
Pisanie skryptów menedżera plików Nautilus	355
Tworzenie i używanie skryptów powłoki w Nautilusie.....	355
Przykładowy skrypt Nautilusa.....	355
Podsumowanie	357
Pytania i odpowiedzi	357
Warsztat.....	358
Quiz.....	358
Odpowiedzi.....	358
Rozdział 17. Zdalne używanie aplikacji	359
Protokół systemu X Window.....	359
Łączenie się z ekranem X za pomocą ssh.....	360
Wyświetlanie pojedynczych zdalnych aplikacji za pomocą ssh.....	361
Wyświetlanie zdalnych aplikacji na lokalnym ekranie za pomocą ssh.....	362
Zdalne wyświetlanie lokalnych aplikacji za pomocą ssh	363
Konfigurowanie zapory sieciowej do obsługi zdalnego ekranu	363
Ręczne łączenie się z ekranem X	364
Przygotowywanie ręcznego połączenia z ekranem X.....	364
Zezwalanie na przychodzące połączenia X za pomocą programu xhost	365
Zdalne wyświetlanie lokalnych aplikacji	367
Lokalne wyświetlanie zdalnych aplikacji.....	368
Zezwalanie na zdalne sesje X i uruchamianie sesji	369
Konfigurowanie menedżera logowania do obsługi XDMCP	369
Uruchamianie zdalnej sesji X.....	371
Podsumowanie	371
Pytania i odpowiedzi	372
Warsztat.....	373
Quiz.....	373
Odpowiedzi.....	373
Ćwiczenia.....	374
Część IV Zagadnienia zaawansowane	375
Rozdział 18. Administrowanie systemem z wiersza poleceń	377
Polecenie su.....	377
Zarządzanie procesami	378
Wyświetlanie listy działających procesów	379
Regulowanie priorytetu procesu.....	380
Usuwanie działających procesów.....	381
Zarządzanie usługami.....	382
Poziomy pracy.....	383
Wybieranie usług uruchamianych automatycznie.....	384
Zatrzymywanie, uruchamianie i restartowanie usług	387
Zarządzanie systemami plików	387
Tworzenie systemów plików	388
Montowanie i odmontowanie systemów plików	390
Modyfikowanie pliku /etc/fstab.....	393

Zarządzanie kontami	394
Dodawanie i usuwanie kont użytkowników	394
Dodawanie i usuwanie grup	395
Zarządzanie grupami	395
Zmiana tożsamości grupowej	396
Zarządzanie okresowymi zadaniami za pomocą polecenia cron	397
Dodawanie systemowych procesów cron	397
Edytowanie osobistych procesów cron	398
Zamykanie i ponowne uruchamianie systemu	399
Podsumowanie	400
Pytania i odpowiedzi	401
Warsztat	401
Quiz	401
Odpowiedzi	402
Ćwiczenia	402
Rozdział 19. Administrowanie systemem w środowisku pulpitowym	403
Zarządzanie procesami	403
Zmiana priorytetu procesu	404
Usuwanie procesów	405
Zarządzanie usługami	406
Narzędzie Konfiguracja usług	406
Włączanie i wyłączanie usług	407
Zatrzymywanie, uruchamianie i restartowanie usług	408
Zarządzanie interfejsami sieciowymi	409
Edytowanie statycznych informacji IP albo adresowania DHCP	409
Ręcznie konfigurowanie informacji DNS	410
Włączanie i wyłączanie interfejsów sieciowych	411
Zarządzanie kontami	411
Dodawanie i usuwanie kont użytkowników	412
Dodawanie i usuwanie grup	413
Edytowanie przynależności do grup	413
Czytanie dzienników systemowych	414
Montowanie i odmontowanie systemów plików	415
Formatowanie urządzenia lub partycji	416
Podsumowanie	417
Pytania i odpowiedzi	417
Warsztat	417
Quiz	418
Odpowiedzi	418
Rozdział 20. Podstawowe informacje o bezpieczeństwie	419
Zarządzanie zaporą sieciową Fedory	419
Program Security Level Configuration	420
Zezwalanie na dodatkowe rodzaje ruchu	421
Zaawansowane prawa dostępu	422
Zmiana przynależności pliku	422
Tryb liczbowy polecenia chmod	423
Specjalne prawa dostępu	424
Ochrona konta użytkownika root	425
Korzystanie z grupy wheel	426
Dodawanie użytkowników do grupy wheel	427
Zmiana przynależności i praw dostępu do polecenia su	427
Automatyczne wylogowywanie użytkowników	428
Ustawianie limitu czasu logowania	428
Usuwanie mniej popularnych powłok	429

Podsumowanie	430
Pytania i odpowiedzi	431
Warsztat	431
Quiz	431
Odpowiedzi	432
Ćwiczenia	432
Rozdział 21. Instalowanie oprogramowania	433
Instalowanie i usuwanie komponentów Fedory	433
Uruchamianie narzędzia Package Management	434
Instalowanie oprogramowania	435
Usuwanie oprogramowania	436
Używanie oprogramowania pochodzącego z innych źródeł	437
Niespełnione zależności	438
Korzystanie z polecenia rpm	438
Instalowanie pakietów RPM za pomocą polecenia rpm	439
Aktualizowanie pakietów RPM za pomocą polecenia rpm	439
Rozwiązywanie problemów z zależnościami	439
Uzyskiwanie informacji za pomocą polecenia rpm	440
Odinstalowywanie oprogramowania za pomocą polecenia rpm	441
Rozwiązywanie problemów z zależnościami cyklicznymi	443
Aktywatory aplikacji	443
Tworzenie aktywatora aplikacji w środowisku GNOME	443
Tworzenie aktywatora aplikacji w środowisku KDE	445
Podsumowanie	446
Pytania i odpowiedzi	446
Warsztat	447
Quiz	447
Odpowiedzi	447
Ćwiczenia	447
Rozdział 22. Udostępnianie plików w sieci	449
Zanim zaczniesz	449
Oferowanie usług NFS	450
Uruchamianie narzędzia NFS Server Configuration	450
Dodawanie i konfigurowanie udziałów NFS	451
Automatyczne uruchamianie usługi NFS — środowisko pulpitu	453
Konfigurowanie NFS za pomocą wiersza poleceń	454
Automatyczne uruchamianie usługi NFS — wiersz poleceń	455
Przepuszczanie ruchu NFS przez zaporę sieciową	455
Współdzielenie plików z systemami Windows	456
Instalowanie serwera plików Windows	457
Instalowanie programu SWAT	457
Uruchamianie programu SWAT	
i konfigurowanie podstawowych parametrów Samby	458
Konfigurowanie udziałów Samby	460
Uruchamianie i automatyczne uruchamianie usług współdzielenia plików Windows	462
Tworzenie kont usługi współdzielenia plików Windows	463
Przepuszczanie usługi współdzielenia plików Windows przez zaporę sieciową	464
Podsumowanie	464
Pytania i odpowiedzi	465
Warsztat	466
Quiz	466
Odpowiedzi	466

Rozdział 23. Udostępnianie usług WWW i FTP	467
Zanim zaczniesz	467
Uruchamianie serwera WWW	468
Instalowanie serwera Apache	468
Automatyczne uruchamianie serwera Apache	469
Przepuszczanie żądań WWW przez zaporę sieciową	470
Korzystanie z serwera WWW Apache	470
Włączanie witryn WWW w katalogach domowych	471
Narzędzie konfiguracyjne serwera Apache	473
Podstawowe zabezpieczenia serwera Apache	474
Dodatkowe informacje o konfigurowaniu serwera Apache	476
Uruchamianie serwera FTP	477
Włączanie i wyłączanie serwera FTP	477
Konfigurowanie usługi vsftpd	477
Przepuszczanie ruchu FTP przez zaporę sieciową	478
Kontrola dostępu do serwera FTP	478
Włączanie i wyłączanie anonimowego FTP	479
Podsumowanie	479
Pytania i odpowiedzi	480
Warsztat	480
Quiz	480
Odpowiedzi	480
Rozdział 24. Kopie zapasowe, rozwiązywanie problemów i odzyskiwanie systemu ..	481
Tworzenie kopii zapasowych i przywracanie danych	481
Kopia zapasowa na dysku	482
Tworzenie kopii zapasowych za pomocą polecenia tar	484
Przykłady tworzenia kopii zapasowej za pomocą polecenia tar	485
Przywracanie kopii zapasowych za pomocą polecenia tar	485
Przykłady przywracania kopii zapasowych za pomocą polecenia tar	487
Testowanie kopii zapasowych i wyświetlanie ich zawartości	487
Automatyczne tworzenie kopii zapasowych	488
Postępowanie w razie katastrofy	490
Uruchamianie narzędzia rescue	490
Naprawa systemu plików	491
Ratowanie plików	492
Inne krytyczne problemy	493
Problemy z systemem plików	493
Problemy z sieciowymi napastnikami	494
Podsumowanie	495
Pytania i odpowiedzi	496
Warsztat	497
Quiz	497
Odpowiedzi	497
Ćwiczenia	497
Dodatki	499
Dodatek A Rozwiązywanie problemów z programem instalacyjnym	501
Wczytywanie modułów podczas instalacji	501
Ostatnia deska ratunku: parametry modułu	502
Korzystanie z instalatora tekstowego	503

Rozdział 5.

Zaprzęganie konsoli do pracy

W tym rozdziale poznasz kilka pojęć, które będą tym bardziej użyteczne, im dłużej będziesz używał Linuksa. Zaczniemy od przedstawienia dwóch najpopularniejszych edytorów, z którymi będziesz stale spotykał się w świecie Linuksa. Następnie pokażemy, jak ułatwić sobie pracę na konsoli.

Podobnie jak w rozdziale 4. („Nawigowanie po Linuksie za pomocą konsoli”) także tutaj chodzi przede wszystkim o nabranie wprawy w posługiwaniu się powłoką oraz jej poleceniami i technikami, a nie o to, aby natychmiast zostać profesjonalistą. Staraj się zrozumieć, o co chodzi w każdym przykładzie, i nie przejmuj się, jeśli wszystkiego nie zapamiętasz.

Przed przystąpieniem do lektury niniejszego rozdziału zaloguj się do konsoli wirtualnej i przygotuj do wpisywania poleceń.

Tworzenie, edytowanie i zapisywanie plików tekstowych

W poprzednim rozdziale utworzyłeś kilka plików tekstowych za pomocą polecenia `touch`, ale były to *puste* pliki tekstowe — niezbyt przydatne do zwykłych zastosowań. Kiedy tworzysz plik tekstowy, zwykle chcesz, żeby zawierał wprowadzony przez Ciebie tekst.

W wierszu poleceń Linuksa do tworzenia plików tekstowych zwykle używa się jednego z dwóch skrajnie różnych edytorów. Pierwszy to klasyczne narzędzie uniksowe o nazwie `vi` (mówiąc dokładniej, w systemie Fedora Core 2 `vi` jest aliasem do programu `vim`); drugi to duży, rozszerzalny procesor tekstu o nazwie `emacs`. Oba programy

dobrze spełniają wymagania stawiane edytorom tekstu. Edytor `vi` podoba się osobom, które cenią minimalizm i szybkość, a `emacs` jest zwykle preferowany przez zaawansowanych użytkowników. Wybór edytora używanego w codziennej pracy jest w dużej mierze kwestią gustu. Jednakże edytor `vi` jest wykorzystywany przez niektóre narzędzia administracyjne, więc powinieneś nauczyć się przynajmniej podstawowej obsługi tego programu.



Różnice między edytorem i procesorem tekstu

Procesor tekstu zawiera funkcje związane ze składem komputerowym, takie jak wybór kroju czcionki, sterowanie układem strony oraz opcje formatowania wydruku. Edytor tekstu służy wyłącznie do edytowania liter i liczb; nie pozwala zmieniać wyglądu ani układu drukowanego tekstu.

Tworzenie plików tekstowych za pomocą edytora vim

Edytor `vi` (w systemie Fedora Core 2 `vim`) jest popularny, bo — w przeciwieństwie do edytora `emacs` — jest dostępny w niemal każdym systemie uniksowym, nawet najbardziej ograniczonym. Ponadto jest znacznie mniejszy i szybszy.

Zacznij teraz pracę nad pustym dokumentem, wpisując polecenie `vi` oraz argument w postaci nazwy pliku, który chcesz utworzyć. Nadaj plikowi nazwę `moj_plik_vi.txt`:

```
[ty@fedora2 ty]$ vi moj_plik_vi.txt
```



Wczytywanie istniejących plików

Uruchomienie edytora `vi` nie zawsze powoduje utworzenie pliku. Jeśli podasz programowi `vi` nazwę istniejącego pliku, zostanie on wczytany do edytora i będziesz mógł go zmodyfikować.

Kiedy program `vi` się uruchomi, zobaczysz kolumnę znaków tyldy (~) po lewej stronie konsoli. Znaki te wskazują, że w poszczególnych wierszach nie ma jeszcze żadnego tekstu. Nie próbuj niczego pisać; na razie jest to niemożliwe. Program `vi` to edytor wierszowy, który ma dwa tryby pracy. Kiedy uruchamiasz `vi`, program zaczyna działać w trybie nieprzeznaczonym do wprowadzania tekstu.

Jest to tak zwany **tryb poleceń**. Wszystko, co wpisujesz w tym trybie, łącznie ze zwykłymi znakami alfabetu, nie pojawia się w dokumencie, lecz jest interpretowane przez `vi` jako żądanie wykonania określonej operacji. W trybie poleceń możesz zapisywać plik, przesuwać kursor, usuwać frazy albo wiersze tekstu itd.

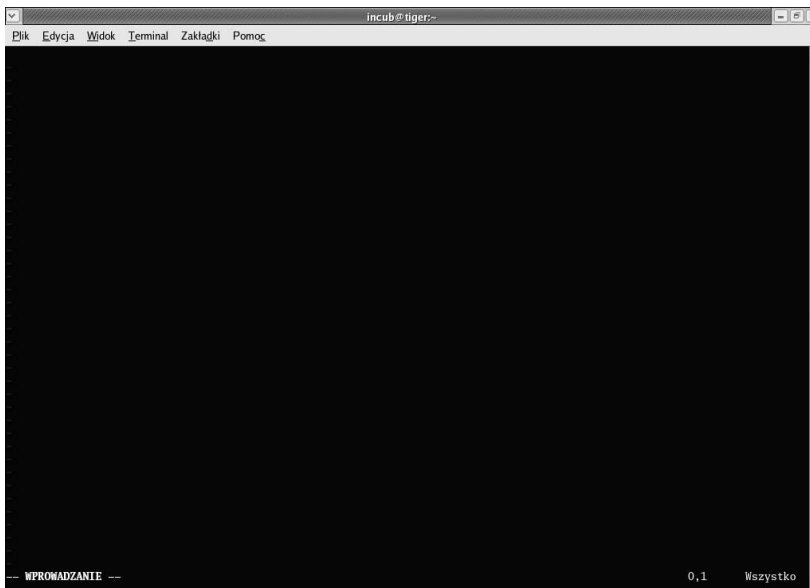
Bezpośrednie wprowadzanie tekstu jest możliwe tylko w **trybie wstawiania**; kiedy `vi` działa w tym trybie. Ta cecha edytora nieodmiennie sprawia problemy początkującym użytkownikom, ale zwykle odrobina praktyki wystarczy, żeby sprawnie posługiwać się systemem z dwoma trybami.

Wstawianie tekstu w edytorze vi

Aby zacząć wstawiać tekst do pliku, nad którym pracujesz (*moj_plik_vi.txt*), naciśnij klawisz *i*. Zwróć uwagę, że na dole ekranu pojawi się słowo WPROWADZANIE czyli *Insert* (zobacz rysunek 5.1).

Rysunek 5.1.

Edytor vi działa w trybie wstawiania i oczekuje na wprowadzenie tekstu. Wszystkie znaki wpisane z klawiatury są teraz interpretowane jako dane wejściowe, dopóki nie naciśniesz klawisza Esc



Wpisz kilka wierszy tekstu. Możesz przepisać przykładowe zdania albo wprowadzić własny tekst. Zauważ, że edytor *vi* nie zawija wierszy; kiedy dotrzesz do końca wiersza, musisz nacisnąć klawisz *Enter*, aby rozpocząć nowy wiersz:

Edytor *vi* jest bardzo rozpowszechniony w systemie Unix. Z tego powodu wszyscy przyszli użytkownicy Uniksa powinni nauczyć się jego obsługi.



Brak zawijania znaków w edytorze vi

Jeśli nie naciśniesz klawisza *Enter*, kiedy dotrzesz do prawego marginesu, tekst na ekranie zawinie się w środku słowa do następnego wiersza. Pamiętaj jednak, że tekst w buforze pamięciowym edytora *vi* pozostanie niezawinięty. Jeśli wypełnisz tekstem cały ekran, nie naciskając klawisza *Enter*, zapisany plik będzie zawierał *jeden długi wiersz* tekstu.

Może to mieć nieoczekiwane konsekwencje podczas drukowania, wysyłania pocztą, a nawet edytowania plików tekstowych.

Kiedy skończysz wpisywać tekst, naciśnij klawisz *Esc*, aby opuścić tryb wstawiania i wrócić do trybu poleceń. Zwróć uwagę, że słowo WPROWADZANIE zniknie z wiersza na dole ekranu.

Wstawiłeś kilka słów. A jeśli chciałbyś zmienić to, co napisałeś do tej pory? Sposoby dokonywania zmian w edytorze *vi* nie zawsze są oczywiste.

Edycja tekstu w edytorze vi

W edytorze vi do nawigacji oraz do edycji tekstu używa się poleceń klawiaturowych. Najczęściej używane polecenia zebrano w tabeli 5.1.

Tabela 5.1. Polecenia klawiaturowe programu vi do nawigacji i edycji tekstu

Klawisz	Operacja
<i>l</i>	Przesuwa kursor o jeden znak w prawo.
<i>h</i>	Przesuwa kursor o jeden znak w lewo.
<i>j</i>	Przesuwa kursor w dół o jeden wiersz <i>tekstu</i> (nie wiersz ekranu).
<i>k</i>	Przesuwa kursor w górę o jeden wiersz <i>tekstu</i> (nie wiersz ekranu).
<i>x</i>	Usuwa znak pod kursorem.
<i>d#<spacja></i>	Usuwa # znaków pod kursorem i na prawo od kursora.
<i>dd</i>	Usuwa bieżący wiersz.
<i>i</i>	Wstawianie: włącza tryb wstawiania; wstawianie zaczyna się od bieżącej pozycji kursora.
<i>a</i>	Uzupełnianie: włącza tryb wstawiania; wstawianie zaczyna się na prawo od bieżącej pozycji kursora.
<i>A</i>	Uzupełnianie: włącza tryb wstawiania; wstawianie zaczyna się od końca bieżącego wiersza <i>tekstu</i> (nie wiersza ekranu).
<i>Esc</i>	Włącza tryb poleceń.

Spróbuj użyć klawiszy nawigacyjnych i edycyjnych w trybie poleceń do zmodyfikowania wpisanego przed chwilą tekstu, do usunięcia wiersza albo kilku znaków oraz do wstawienia lub uzupełnienia tekstu w pliku. Jeśli poczujesz się zdezorientowany albo klawiatura przestanie reagować podczas edytowania tekstu, naciśnij kilkakrotnie klawisz *Esc*, aby upewnić się, że wróciłeś do trybu poleceń.



Klawisze strzałek w edytorze vi

Jeśli wykonywałeś ćwiczenia i eksperymentowałeś z edytorem vi, prawdopodobnie odkryłeś, że jego obsługa jest łatwiejsza, niż sugerowałby powyższy opis. Fedora zawiera ulepszoną wersję edytora vi o nazwie vim. W tej wersji vi można poruszać się po tekście za pomocą klawiszy strzałek, nawet w trybie wstawiania, pod warunkiem, że terminal jest poprawnie skonfigurowany (konsola Linuksa spełnia ten warunek).

Ulepszona wersja vi nie jest jednak dostępna w większości systemów uniksowych, a nawet nie we wszystkich systemach linuxowych. Co więcej, nawet w Fedorze niektóre operacje administracyjne ograniczają Cię do tradycyjnych klawiszy vi. Powinieneś zapoznać się więc z ich działaniem na wypadek, gdybyś w przyszłości musiał pracować z tradycyjnym edytorem vi.

Poeksperymentuj przez chwilę z edytorem vi i spróbuj przyswoić sobie zasady jego obsługi. W Linuksie będziesz miał z nim do czynienia na tyle często, że nie będzie to stracony czas.

Zapisywanie pliku, zamykanie edytora i dalsza lektura

Aby zapisać właśnie utworzony plik, trzeba wprowadzić dość tajemnicze polecenie. W trybie poleceń wpisz dwukropek (:). Zauważ, że dwukropek pojawia się w lewym dolnym rogu ekranu, a kursor przenosi się tuż obok dwukropka; edytor `vi` czeka teraz na wpisanie bardziej złożonego polecenia. Wpisz małą literę `w` i naciśnij klawisz `Enter`. Edytor `vi` wyświetli w dolnym ekranu następujący komunikat:

```
"moj_plik_vi.txt" [Nowy] 2L, 141C written
```

Edytor podaje najpierw nazwę zapisanego pliku; w tym przypadku jest to `moj_plik_vi.txt`. Następnie wyświetla liczbę wierszy w pliku (2) oraz liczbę znaków (141). Zapisalesz swój pierwszy plik `vi`.

W edytorze `vi` istnieje też funkcja „zapisz jako”. Aby ponownie zapisać tekst, tym razem w nowym pliku, wpisz dwukropek, literę `w`, spację i nazwę `moj_nowy_plik_vi.txt`. Edytor wyświetli komunikat:

```
"moj_nowy_plik_vi.txt" [Nowy] 2L, 141C written
```

Teraz w swoim katalogu macierzystym masz dwie kopie pliku — `moj_plik_vi.txt` i `moj_nowy_plik_vi.txt`. Ponieważ jest to książka dla początkujących, nie będziemy się więcej zajmować edytorem `vi`. Aby zamknąć program, wpisz dwukropek i literę `q`. W ten sposób opuścisz edytor i wrócisz do wiersza poleceń.



Przypomnienie o zapisie danych w edytorze `vi`

Jeśli spróbujesz zamknąć edytor bez zapisania pliku, nad którym pracowałeś, edytor `vi` wyświetli odpowiedni komunikat o błędzie, a następnie wróci do trybu poleceń.

Jeśli rzeczywiście chcesz zamknąć program bez zapisywania pliku, wpisz polecenie `:q!` (ze znakiem wykrzyknika) zamiast `:q`; edytor zastosuje się do Twojego życzenia.

Edytor `vi` jest bardzo zaawansowany i wart dokładniejszego przestudiowania, zarówno ze względu na liczne funkcje zwiększające wydajność edytowania tekstu, jak i dlatego, że często będziesz miał z nim do czynienia podczas pracy w wierszu poleceń Linuksa lub Uniksa. Jeśli masz odwagę, możesz przejrzeć obszerny system pomocy edytora `vi`, ponownie uruchamiając edytor i wpisując `:help` w trybie poleceń. Stronę tytułową systemu pomocy pokazano na rysunku 5.2.

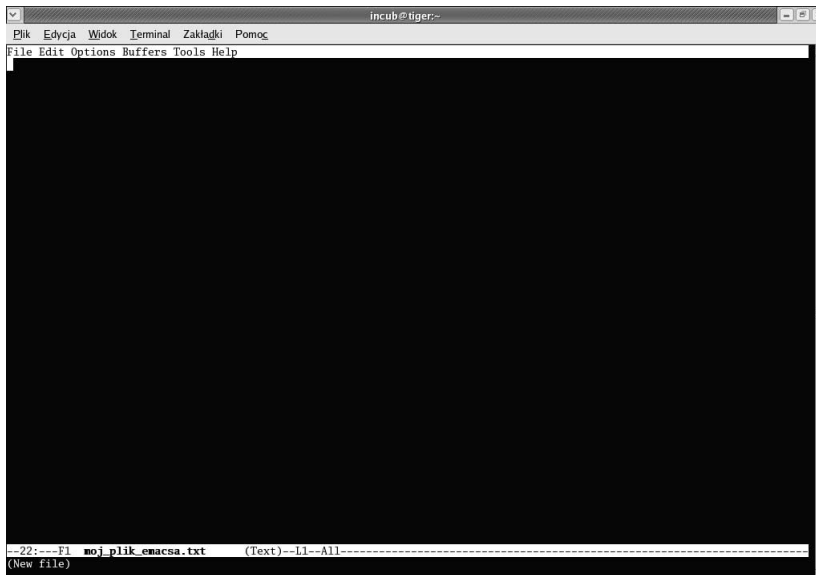


Więcej informacji o edytorze `vi`

Bardziej szczegółowy przewodnik po edytorze `vi` znajdziesz w książce *Sams Teach Yourself Unix in 24 hours*, której autorem jest Dave Taylor.

Rysunek 5.3.

*Edytor emacs
wygląda
zdecydowanie
bardziej
przystępnie
niż vi
i funkcjonuje
w sposób
zbliżony
do tego,
do którego
przyzwyczajeni
są użytkownicy
Windowsa*

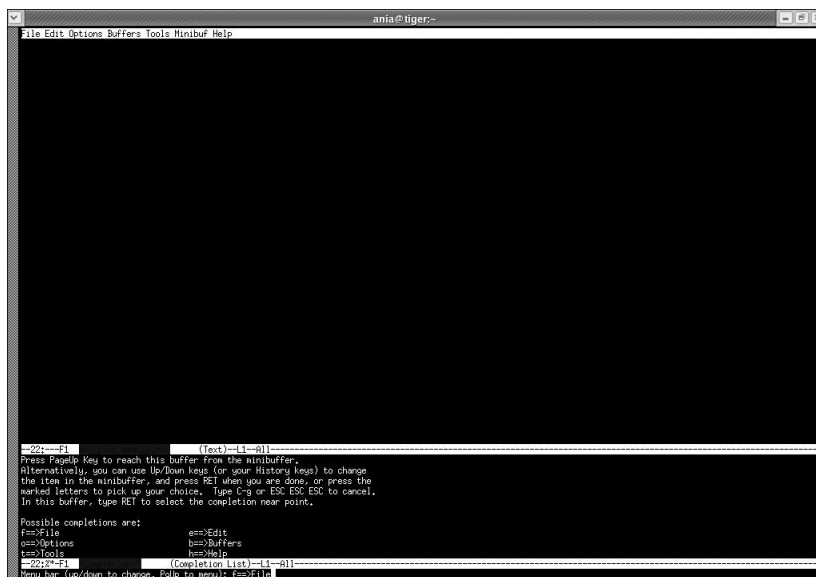


System menu w edytorze emacs

Choć na górze ekranu edytora emacs znajduje się coś, co wygląda jak menu, podczas pracy na konsoli nie da się go uaktywnić. Można jednak w niemal dowolnym momencie nacisnąć klawisz *F10*, aby uzyskać dostęp do względnie intuicyjnego menu. Po naciśnięciu klawisza *F10* ekran dzieli się na dwie części. W dolnej części wyświetlane są opcje, które można uaktywnić jednym naciśnięciem klawisza (zobacz rysunek 5.4).

Rysunek 5.4.

*Po wyświetleniu
menu edytora
emacs podczas
pracy na konsoli
możesz
wybierać
poszczególne
pozycje,
naciskając
odpowiedni
klawisz*



Aby zapisać plik, nad którym pracowałeś, wpisz `f`, aby otworzyć menu *File*, a następnie wpisz `s`. W niewielkim obszarze na dole ekranu (nazywanym **minibuforem**) pojawi się komunikat o zapisaniu pliku:

```
wrote /home/ty/moj_plik_emacsa.txt
```

Za pomocą menu przywoływanego klawiszem *F10* możesz wykonywać większość podstawowych operacji edycyjnych oraz zapisywać i czytać pliki. Większość użytkowników woli posługiwać się edytorem `emacs` niż `vi` właśnie ze względu na menu *F10* oraz prostszy sposób edytowania tekstu.

Najważniejsze polecenia klawiaturowe edytora emacs

Jeśli masz zręczne palce, będziesz mógł używać edytora `emacs` i menu *F10* przez długie miesiące, nie wpadając w kłopoty. Jednakże przypadkowo naciśnięty klawisz może sprawić, że `emacs` zacznie piszczeć i wyświetlać zagadkowe komunikaty o błędzie albo niezrozumiałe żądania w minibuforze.

W takich okolicznościach z pomocą przychodzi kombinacja klawiszy zapisana w dokumentacji edytora `emacs` jako `C-g` (przytrzymaj wciśnięty klawisz *Ctrl* i naciśnij klawisz `g`). Jest to polecenie przerwania bieżącej operacji. Edytor `emacs` jest tak rozbudowany, że daje wiele okazji do przełączenia się w nieznaną stan; w większości przypadków wystarczy kilkakrotnie nacisnąć `C-g`, aż wszystko wróci do normy.

W tabeli 5.2 zebrano kilka innych interesujących poleceń klawiaturowych edytora `emacs` — niektóre z nich są dostępne w systemie menu, inne nie. Aby zrozumieć tę tabelę, musisz najpierw zapoznać się z formatem poleceń edytora:

- ♦ `C-x`, gdzie `x` jest literą, oznacza: przytrzymaj naciśnięty klawisz *Ctrl* i naciśnij klawisz litery.
- ♦ `C-xy`, gdzie `x` i `y` są literami, oznacza: przytrzymaj wciśnięty klawisz *Ctrl*, naciśnij pierwszą literę, zwolnij klawisz *Ctrl* i pierwszą literę, a następnie naciśnij drugą literę.
- ♦ `M-x`, gdzie `x` jest literą, oznacza: przytrzymaj wciśnięty klawisz *Alt* i naciśnij literę.
- ♦ `C-x C-y`, gdzie `x` i `y` są literami, oznacza: przytrzymaj wciśnięty klawisz *Ctrl*, naciśnij pierwszą literę, a następnie bez zwalniania klawisza *Ctrl* naciśnij drugą literę.

Wielkość liter w tych kombinacjach ma znaczenie, więc wpisuj wskazane litery, a nie ich małe lub wielkie odpowiedniki.

Zanim skończymy opis edytora `emacs`, podamy jeszcze jedno polecenie, od którego zaczyna się droga do zostania prawdziwym znawcą tego programu. Aby poznać dostępne polecenia, naciśnij `M-x`, a następnie klawisz *Tab*. Pojawi się bufor z listą dostępnych poleceń edytora `emacs`. Naciśnij `C-xo`, aby przenieść kursor do tego bufora; możesz teraz przeglądać dostępne polecenia za pomocą klawiszy *Page Up* i *Page Down*. Jeśli znajdziesz jakieś interesujące polecenie, wpisz je w minibuforze i naciśnij klawisz *Enter*.

Tabela 5.2. Przydatne polecenia klawiaturowe programu emacs

Polecenie	Opis
C-ht	Uruchamia samouczek edytora emacs, w którym udokumentowanych jest znacznie więcej kombinacji klawiszy.
C-x C-f	Otwiera lub tworzy plik w bieżącym okienku edycyjnym; nazwę pliku albo ścieżkę do pliku wpisuje się w minibuforze.
C-x C-s	Zapisuje bieżący plik.
C-x2	Dzieli pionowo bieżące okienko edycyjne.
C-x3	Dzieli poziomo bieżące okienko edycyjne.
C-x1	Rozciąga bieżące okienko edycyjne na cały ekran (ukrywa lub usuwa inne okienka).
C-x0	Wybiera następne okienko edycyjne.
C-xk	Zamyka (bez zapisywania) bieżący bufor. Możesz użyć tego polecenia na przykład do zamknięcia samouczka.
C-xb	Przełącza bieżący bufor; nazwę bufora wpisuje się w minibuforze. Jeśli nie pamiętasz nazw otwartych buforów, naciśnij klawisz <i>Tab</i> , aby wyświetlić pełną listę.
C-xi <i>plik</i>	Wstawia plik w bieżącym położeniu kursora. Nazwę pliku wpisuje się w minibuforze.
C-x C-c	Zamyka edytor emacs.
C-g	Przerywa bieżącą operację edytora.

Wśród wielu poleceń edytora emacs znajdziesz poniższe:

- ♦ M-x *dunnet* uruchamia tekstową grę przygodową.
- ♦ M-x *auto-fill-mode* włącza zawijanie wierszy.
- ♦ M-x *calendar* otwiera nowe okienko z kalendarzem na trzy miesiące.
- ♦ M-x *ansi-term* otwiera powłokę w bieżącym okienku. Zanim wydasz to polecenie, wpisz C-x2 albo C-x3, aby móc jednocześnie edytować plik i pracować z powłoką, przełączając się między okienkami za pomocą kombinacji C-x0 (zobacz rysunek 5.5).

Rysunek 5.5.

Ekran edytora emacs podzielony poziomo za pomocą kombinacji C-x2. W dolnym okienku uruchomiono powłokę poprzez wpisanie C-x0, a następnie M-x *ansi-term*. Można przełączać się między okienkami za pomocą kombinacji C-x0

The screenshot shows the Emacs editor interface with a terminal window open at the bottom. The terminal displays system information including CPU usage, memory usage, and a list of running processes. The Emacs interface shows the terminal output in a separate window, with the main editor area above it.

```

File Edit Options Buffers Tools Help
-----
[niepamiętam jak to działa, jest bardzo proste, znaczenie trudniej jest opisać
pozostałe funkcje tego edytora]

[22:47:41] (C-x) [2-x] [1]
-----
PID USER PR NI VIRT RES SHR S CPU% ZMEM TIME+ COMMAND
1 root 16 0 1954 464 1316 S 0.0 0.1 0:04.32 init
2 root 34 19 0 0 0 0 S 0.0 0.0 0:00.00 ksoftirqd/0
3 root 15 -10 0 0 0 0 S 0.0 0.0 0:00.45 eventsd/0
4 root 15 -10 0 0 0 0 S 0.0 0.0 0:00.00 klockd/0
5 root 15 -10 0 0 0 0 S 0.0 0.0 0:00.00 khajpar
6 root 15 0 0 0 0 0 S 0.0 0.0 0:00.00 khaid
7 root 15 0 0 0 0 0 S 0.0 0.0 0:00.02 pdflush
10 root 15 -10 0 0 0 0 S 0.0 0.0 0:00.00 aisc/0
3 root 15 0 0 0 0 0 S 0.0 0.0 0:00.31 kswapd0
115 root 21 0 0 0 0 0 S 0.0 0.0 0:00.00 kseriad
156 root 24 0 0 0 0 0 S 0.0 0.0 0:00.00 scsi_eh_0
158 root 25 0 0 0 0 0 S 0.0 0.0 0:00.00 scsi_eh_1
164 root 15 0 0 0 0 0 S 0.0 0.0 0:01.33 kjsuaid
1173 root 15 0 0 0 0 0 S 0.0 0.0 0:00.48 kjsuaid
1581 root 16 0 2318 588 1296 S 0.0 0.1 0:00.20 eglogd
1595 root 15 0 3421 460 2244 S 0.0 0.1 0:00.05 klogd
1616 rpc 15 0 1998 580 1372 S 0.0 0.1 0:00.00 portmap

[22:47:41]
[Target: char min]--L146--Bus
Rtfosaving...done

```



Tworzenie kopii zapasowych plików przez program emacs

Jeśli zaczniesz korzystać z edytora `emacs`, wkrótce zauważysz, że od czasu do czasu w Twoim katalogu domowym pojawiają się pliki o nazwach podobnych do tych, z którymi pracowałeś, ale z niewielkimi uzupełnieniami. Jeśli na przykład pracowałeś z plikiem `moj_plik.txt`, w tym samym katalogu możesz znaleźć plik `#moj_plik.txt#` albo `moj_plik.txt~`.

Są to „pliki bezpieczeństwa”, utworzone przez edytor `emacs` w celu ochrony Twoich danych.

Kiedy zamykasz program bez zapisywania pliku, `emacs` zapisuje go i tak, pod tą samą nazwą, ale ze znakami krzyżyka (`#`) z przodu i z tyłu, na wypadek, gdybyś później postanowił przywrócić dokonane zmiany.

Kiedy dokonujesz zmian w istniejącym pliku i zapisujesz je, `emacs` zachowuje pierwotny (niezmodyfikowany) plik pod tą samą nazwą, uzupełniając ją o znak tyldy (`~`), na wypadek, gdybyś chciał później wrócić do pierwotnej wersji pliku.

Jeśli nie potrzebujesz automatycznie zapisanych plików ani ich kopii zapasowych, możesz je usunąć poleceniem `rm`, choć przedtem warto wczytać je do edytora `emacs` i sprawdzić, czy nie zawierają ważnych danych.

Efektywne zarządzanie grupami plików

Kiedy zaczniesz regularnie pracować z większą liczbą plików oraz z edytorami, takimi jak `vi` i `emacs`, zbudujesz bibliotekę danych, raportów i niedokończonych plików.

Polecenia takie jak `mv`, `cp`, `rm` i `ls` przydają się do organizowania plików. Kiedy jednak Twoja kolekcja plików się powiększy, będziesz potrzebował dodatkowych narzędzi, dostarczanych przez powłokę oraz kilku bardziej zaawansowanych poleceń. Możesz grupować pliki, aby szybciej i wydajniej je wyszukiwać. W kolejnych podrozdziałach wyjaśnimy, jak grupować pliki w Linuksie, aby sprawniej nimi zarządzać.

Grupowanie plików w wierszu poleceń

W rozdziale 4. nauczyłeś się kilku podstawowych metod zarządzania plikami, polegających na wpisywaniu poleceń i nazw plików — po jednej lub po dwie — za monitem powłoki. Czasem warto jednak mieć możliwość odwołania się do wielu plików jednocześnie, bez wpisywania wszystkich nazw. Powłoka dostarcza narzędzi, które umożliwiają zgrupowanie podobnych nazw plików, dzięki czemu nie trzeba wpisywać ich pojedynczo.

Zilustrujmy to na przykładzie. Przypuśćmy, że chcesz zapisać pliki `moj_plik_vi.txt`, `moj_nowy_plik_vi.txt` oraz `moj_plik_emacsa.txt` w katalogu o nazwie *pierwsze pliki*. Korzystając z umiejętności nabytych w podrozdziale „Manipulowanie plikami i katalogami” w rozdziale 4. najpierw tworzysz katalog za pomocą polecenia `mkdir`, a następnie przenosisz poszczególne pliki za pomocą polecenia `mv`:

```
[ty@fedora2 ty]$ mkdir pierwsze_pliki
[ty@fedora2 ty]$ mv moj_plik_vi.txt moj_nowy_plik_vi.txt moj_plik_emacsa.txt
pierwsze_pliki
[ty@fedora2 ty]$
```

To bardzo dużo pisania jak na zwykłe przenoszenie trzech plików tekstowych w dogodniejsze miejsce. Wydaje się, że powinien istnieć jakiś łatwiejszy sposób — i on istnieje. W wierszu poleceń gwiazdka (*) jest jednym ze znaków specjalnych, które służą do **rozwijania nazw plików** — logicznego grupowania plików w taki sposób, żeby nie trzeba było wpisywać poszczególnych nazw i żeby dało się manipulować wszystkimi plikami jednocześnie. Rozwijanie nazw plików to po części nauka, a po części sztuka; korzystasz z narzędzi do dopasowywania wzorców, aby zebrać pliki w grupę, a następnie przekazujesz tę grupę powłóce. Brzmi to skomplikowanie, ale rozwijanie nazw plików jest w rzeczywistości bardzo proste. Oto przykład zastosowania tej techniki w opisanym przed chwilą zadaniu. Spróbuj użyć poniższego polecenia `mv`:

```
[ty@fedora2 ty]$ mv *.txt pierwsze_pliki
```

Sprawdźmy, czy polecenie odniosło pożądany skutek:

```
[ty@fedora2 ty]$ ls
pierwsze_pliki
[ty@fedora2 ty]$ ls pierwsze_pliki
moj_nowy_plik_vi.txt moj_plik_emacsa.txt moj_plik_vi.txt
[ty@fedora2 ty]$
```

Najwyraźniej nowe, uproszczone polecenie zadziałało poprawnie. Fraza `*.txt` zgrupowała wszystkie pliki, których nazwy są zakończone czterema znakami `.txt`.

Zanim przejdziemy do innych przykładów, przyjrzyj się tabeli 5.3, w której wymienione są znaki najczęściej używane do rozwijania nazw plików.

Tabela 5.3. *Znaki używane do dopasowywania wzorców i ich działanie*

Wzorec	Opis
*	Dopasowuje wszystkie znaki, niezależnie od ich liczby.
?	Dopasowuje dowolny pojedynczy znak.
[a-b] (zakres)	Dopasowuje pojedynczy znak we wskazanym zakresie; na przykład [A-Z] dopasowuje litery X lub P, ale nie cyfrę 9 albo literę a.
[AaBbCc] (lista)	Dopasowuje dowolny znak spośród wymienionych na liście.



Nieoczekiwane działanie zakresów

Litery w komputerze są reprezentowane wewnątrz za pomocą specjalnego kodu o nazwie ASCII, którego układ odbiega od naszych przyzwyczajęń, więc niektóre wzorce mogą odnosić skutek inny od zamierzonego.

Na przykład wzorec `[a-Z]` nie dopasuje niczego, natomiast `[A-z]` dopasuje kilka znaków, które nie są wielkimi ani małymi literami. Z tego powodu powinieneś używać tylko takich zakresów, w których oba znaki są wielkimi literami, małymi literami albo cyframi.

Kilka przykładów pomoże Ci lepiej zrozumieć, co oznaczają te wzorce i jak za ich pomocą można grupować pliki w Linuksie. Utwórzmy tabelę przykładów. Przypuśćmy, że chcesz usunąć grupę plików ze swojego katalogu domowego. Tabela 5.4 przedstawia przykładowe argumenty polecenia `rm` oraz pliki, które zostałyby dopasowane (albo nie) przez każdy z nich.

Tabela 5.4. Przykładowe polecenia korzystające z rozwijania nazw plików i ich skutki

Polecenie	Skutek
<code>rm *.txt</code>	Usunie każdy plik z rozszerzeniem <code>.txt</code> — na przykład <code>a.txt</code> , <code>b.txt</code> , <code>halo.txt</code> albo <code>wszyscy_sa_super.txt</code> , ale nie <code>rower.gif</code> , <code>rachunki.xls</code> albo <code>stary_txt</code> .
<code>rm a*.jpg</code>	Usunie wszystkie pliki, których nazwa zaczyna się od litery <code>a</code> , a kończy trzema literami <code>jpg</code> — na przykład <code>a.jpg</code> , <code>ananas.jpg</code> , <code>andromeda.jpg</code> i <code>anastazja_jpg</code> , ale nie <code>kajak.jpg</code> , <code>plik.txt</code> , <code>malpy.gif</code> albo <code>makarony_lista</code> .
<code>rm k*a.*if</code>	Usunie pliki <code>kuchnia.gif</code> , <code>kura.tif</code> , <code>ka.zif</code> i <code>kawiarnia..if</code> , ale nie <code>kariera.if</code> , <code>kasyno.gif</code> albo <code>fontanna.tif</code> .
<code>rm k[aoi]t.*</code>	Usunie pliki <code>kot.gif</code> , <code>kit.txt</code> , <code>kat.jpg</code> , <code>kit.do.okien</code> , <code>kot.i.pies.txt</code> oraz <code>kat.astrofa</code> , ale nie <code>ket.gif</code> , <code>kotek.txt</code> albo <code>kot</code> .
<code>rm [a-f]*</code>	Usunie pliki <code>ananas.txt</code> , <code>banan.jpg</code> , <code>czarny.kot</code> , <code>dynia.i.melon</code> , <code>energia.mp3</code> i <code>faworyci_fanatyka</code> , ale nie <code>gildia.txt</code> , <code>xawery.jpg</code> albo <code>Barbie.gif</code> .
<code>rm *</code>	Usunie każdy plik w bieżącym katalogu roboczym. W przypadku dodania opcji <code>-rf</code> usunie każdy plik i każdy katalog w bieżącym katalogu roboczym (czy pamiętasz opcje <code>-r</code> i <code>-f</code> z rozdziału 4.?).



Postępowanie z wzorcami

Zachowaj ostrożność, kiedy grupujesz pliki za pomocą rozwijania nazw; nieprzemysłane wzorce mogą przynieść niepożądane skutki. Kiedy na przykład usuwasz pliki za pomocą poleceń pokazanych w tabeli 5.4, Linux przed usunięciem wszystkich plików pasujących do podanego wzorca nie ostrzega Cię ani nie pyta, czy jesteś tego pewien.

Rozwijania plików możesz używać w wielu okolicznościach, aby ułatwić sobie pracę na konsoli poprzez zmniejszanie liczby wpisywanych znaków przy zarządzaniu dużymi grupami plików. Na koniec spróbujemy przenieść pliki utworzone w edytorze `vi` z powrotem do katalogu domowego, a plik utworzony w edytorze `emacs` pozostawimy w katalogu *pierwsze_pliki*:

```
[ty@fedora2 ty]$ mv pierwsze_pliki/*vi* .
[ty@fedora2 ty]$ ls
moj_nowy_plik_vi.txt  moj_plik_vi.txt  pierwsze_pliki
[ty@fedora2 ty]$ ls pierwsze_pliki
moj_plik_emacsa.txt
[ty@fedora2 ty]$
```

Zapobieganie rozwijaniu nazw plików

Czasem rozwijanie nazw plików jest niepożądane — na przykład wtedy, kiedy chcesz użyć znaków specjalnych (takich jak gwiazdka albo znak zapytania) w nazwie pliku albo jako argumentu polecenia.

Przypuśćmy, że chcesz utworzyć w swoim katalogu domowym katalog o nazwie **nowy** i zapisać w nim kilka nowych plików, nad którymi ostatnio pracowałeś. W Linuksie nazwy plików i katalogów mogą zawierać gwiazdki. Kiedy jednak spróbujesz utworzyć taki katalog, otrzymasz następujący komunikat o błędzie:

```
[ty@fedora2 ty]$ mkdir *nowy*
mkdir: `moj_nowy_plik_vi.txt' istnieje, ale nie jest katalogiem
[ty@fedora2 ty]$
```

Powłoka zinterpretowała argument **nowy** jako wzorzec i próbowała dopasować go do plików w bieżącym katalogu roboczym. Pech chciał, że powłoka znalazła pasujący plik — wzorzec **nowy** pasuje do nazwy *moj_nowy_plik_vi.txt*, więc powłoka zadziałała tak, jakbyś wprowadził polecenie:

```
[ty@fedora2 ty]$ mkdir moj_nowy_plik_vi.txt
mkdir: `moj_nowy_plik_vi.txt' istnieje, ale nie jest katalogiem
[ty@fedora2 ty]$
```

W takich przypadkach chciałbyś, żeby powłoka traktowała wpisaną frazę jak zwykły tekst, a nie jak wzorzec. W tym celu musisz **przytoczyć** tekst (czyli ująć go w cudzysłów). Możesz to zrobić za pomocą pojedynczego lub podwójnego cudzysłowu (jest między nimi pewna różnica, o której będzie mowa w dalszych rozdziałach). Póki co, użyj pojedynczego cudzysłowu, aby utworzyć katalog **nowy** w swoim katalogu macierzystym:

```
[ty@fedora2 ty]$ mkdir '*nowy*'
[ty@fedora2 ty]$ ls
moj_nowy_plik_vi.txt  moj_plik_vi.txt  pierwsze_pliki  *nowy*
```

Udało ci się utworzyć katalog **nowy** dzięki ujęciu w cudzysłów frazy, która inaczej zostałaby potraktowana jako wzorzec do rozwijania nazw plików.

Szybkie wyszukiwanie plików i katalogów

Dowiedziałeś się, jak wyświetlić listę plików za pomocą polecenia `ls` i jak zmienić katalog roboczy za pomocą polecenia `cd`. Jeśli jednak chciałbyś wyszukać konkretny plik tylko za pomocą poleceń `ls` i `cd`, musiałbyś wędrować po całym drzewie katalogu domowego, wyświetlać zawartość każdego katalogu, katalogów w tych katalogach i tak dalej, dopóki nie znalazłbyśżądanego pliku. Polecenia `ls` i `cd` sprawdzają się w przypadku zbioru dwóch czy trzech plików, ale jeśli zbiór liczy sto plików? Albo tysiąc? Prędzej czy później wszystkie nazwy plików zaczynają wyglądać jednakowo.

Zamiast tracić cały dzień na ręczne wyszukiwanie plików, użyj poleceń `find` i `locate`, aby odnaleźć zagubione pliki szybko i bez wysiłku.

Szukanie plików za pomocą polecenia find

Polecenie `find` przeszukuje całe drzewa katalogów albo listy drzew katalogów pod kątem określonych nazw plików albo nazw pasujących do podanego wzorca. Oto składnia polecenia `find`:

```
find drzewo1 [drzewo2 ...] -name nazwapliku -print
```

Polecenie `find` wywołane w taki sposób szuka pliku o określonej nazwie w podanych drzewach katalogów. Możesz na przykład wyszukać plik `mój_plik_emacsa.txt` w swoim katalogu domowym:

```
[ty@fedora2 ty]$ find ~ -name moj_plik_emacsa.txt -print
/home/ty/pierwsze_pliki/moj_plik_emacsa.txt
[ty@fedora2 ty]$
```



Działanie znaku tyldy

Pamiętaj, że znak tyldy (`~`) jest dla powłoki odpowiednikiem katalogu `/home/ty`.

Polecenie `find` szybko znalazło plik `mój_plik_emacsa.txt`. Znajduje się on w katalogu `/home/ty/pierwsze_pliki`. W podobny sposób możesz wyszukać w swoim katalogu domowym wszystkie pliki, których nazwy kończą się rozszerzeniem `.txt`, podając wzorzec podobny do używanego podczas rozwijania nazw plików. Musisz jednak ująć wzorzec w cudzysłów, bo w przeciwnym razie powłoka dopasuje wzorzec do plików tekstowych w Twoim katalogu domowym, zanim polecenie `find` będzie miało szansę go zinterpretować:

```
[ty@fedora2 ty]$ find /home/ty -name '*.txt' -print
/home/ty/pierwsze_pliki/moj_plik_emacsa.txt
/home/ty/moj_plik_vi.txt
/home/ty/moj_nowy_plik_vi.txt
[ty@fedora2 ty]$
```

Polecenie `find` przeszukało Twój katalog domowy i znalazło trzy pliki o nazwie zakończonej na `.txt`. Wyświetlone zostały pełne ścieżki do tych plików.

Przeszukiwanie całego systemu plików za pomocą polecenia locate

Czasem zdarza się, że chcesz wyszukać konkretny plik w całym systemie plików Linuksa. W takich przypadkach polecenie `locate` często działa lepiej niż `find`. Wiesz już, jak przeszukać cały system plików za pomocą polecenia `find`; przypuśćmy teraz, że chcesz znaleźć wszystkie obrazy JPEG w systemie plików Linuksa. Możesz to zrobić za pomocą polecenia:

```
[ty@fedora2 ty]$ find / -name '*.jpg' -print
[...]
```

Przekonasz się jednak, że tego rodzaju wyszukiwanie bywa czasochłonne, ponieważ polecenie `find` zagląda do każdego zakątka dysku twardego, katalog po katalogu, szukając plików o nazwach zakończonych na `.jpg`. Na ekranie pojawi się też mnóstwo komunikatów o błędach, ponieważ polecenie `find` napotka wiele miejsc, do których — jako zwykły użytkownik — nie masz dostępu. Polecenie `locate` działa w takich przypadkach wydajniej, ponieważ odwołuje się do bazy danych, która indeksuje wszystkie pliki w systemie. Używanie polecenia `locate` jest proste — podaj mu po prostu argument w postaci tekstu, który chcesz wyszukać:

```
[ty@fedora2 ty]$ locate '*.jpg'
/usr/share/nautilus/patterns/stucco.jpg
/usr/share/nautilus/patterns/dark-gnome.jpg
/usr/share/nautilus/patterns/burlap.jpg
/usr/share/nautilus/patterns/chalk.jpg
/usr/share/gimp/2.0/scripts/images/texture1.jpg
/usr/share/gimp/2.0/scripts/images/texture3.jpg
/usr/share/gimp/2.0/scripts/images/texture.jpg
/usr/share/gimp/2.0/scripts/images/beavis.jpg
/usr/share/gimp/2.0/scripts/images/texture2.jpg
/usr/share/gtk-2.0/demo/background.jpg
[...]
/usr/games/chromium/data/doc/images/useItem00.jpg
/usr/local/games/ut2003_demo/Web/images/h_logo.jpg
/usr/local/games/ut2003_demo/Web/images/h_rgfx.jpg
/usr/local/games/enemy-territory/Docs/Help/Tech Help/vendor/gameLogo.jpg
/usr/local/games/enemy-territory/Docs/Help/Tech Help/vendor/redstripe.jpg
/usr/local/games/enemy-territory/Docs/Help/Tech Help/vendor/Activision.jpg
[ty@fedora2 ty]$
```



Wstrzymywanie wyświetlania wyników

Wyniki polecenia `locate` prawdopodobnie przewiną się poza ekran. Możesz wstrzymać wyświetlanie wyników polecenia `locate` (i innych poleceń), dołączając na końcu polecenia tekst `|more`:

```
locate '*.jpg' |more
```

Polecenie to będzie wyświetlać wyniki strona po stronie — naciśnięcie klawisza spacji spowoduje pokazanie następnego ekranu informacji.

Technika ta nosi nazwę **łączenia potokowego** i zostanie opisana w dalszym podrozdziale, zatytułowanym „Łączenie poleceń za pomocą potoków”.

Choć rzeczywista lista rysunków zależy od tego, jakie środowiska pulpitu i inne pakiety oprogramowania zainstalowałeś w rozdziale 2. („Instalowanie systemu Fedora”), niemal natychmiast otrzymasz długą listę pasujących plików.

Baza danych używana przez polecenie `locate` jest aktualizowana raz dziennie, kiedy system wykonuje polecenie `updatedb`. Dane wyświetlane przez polecenie `locate` czasem nie uwzględniają pracy, którą wykonałeś w ciągu ostatnich kilku godzin. Jednakże wykorzystanie bazy danych umożliwi bardzo szybkie wyszukiwanie wielu plików na całym dysku.

Zapisywanie listy znalezionych plików

Czasem warto zapisać wyniki wyszukiwania. Powłoka umożliwi zapisanie wyników poleceń, takich jak `find` i `locate`, przez *przekierowanie* ich wyjścia do pliku. Możesz następnie wczytać ten plik do edytora, takiego jak `emacs` lub `vi`, aby je zmodyfikować, wydrukować lub wykorzystać w inny sposób. Aby przekierować do pliku **standardowe wyjście** polecenia (informacje, które polecenie wyświetla na konsoli, kiedy realizuje Twoje żądania), użyj znaku większości (`>`) i wpisz po nim nazwę docelowego pliku. Spróbuj teraz zapisać listę plików JPEG w swoim systemie w pliku o nazwie *moje_pliki_jpeg.txt*:

```
[ty@fedora2 ty]$ locate '*.jpg' > moje_pliki_jpeg.txt
[ty@fedora2 ty]$ ls -l
-rw-rw-r-- 1 ty ty 50990 lip 27 14:55 moje_pliki_jpeg.txt
-rw-rw-r-- 1 ty ty 6 lip 27 14:30 moj_nowy_plik_vi.txt
-rw-rw-r-- 1 ty ty 6 lip 27 14:29 moj_plik_vi.txt
drwxrwxr-x 2 ty ty 4096 lip 27 14:39 *nowy*
drwxrwxr-x 2 ty ty 4096 lip 27 14:36 pierwsze_pliki
[ty@fedora2 ty]$
```

Jak widzisz, w Twoim katalogu domowym pojawił się nowy, całkiem spory plik o nazwie *moje_pliki_jpeg.txt*. Plik ten zawiera listę plików JPEG w systemie plików Linuksa, znalezionych przez polecenie `locate`. Jeśli chcesz obejrzeć plik, wczytaj go do edytora `emacs` lub `vi`.

Możesz przekierować wyjście niemal każdego polecenia dostępnego w powłoce. Dołączenie standardowego wyjścia polecenia do istniejącego pliku można zrealizować także za pomocą podwójnego znaku większości (`>>`). Aby na przykład dodać listę wszystkich plików GIF w systemie do pliku *moje_pliki_jpeg.txt*, wydaj polecenie:

```
[ty@fedora2 ty]$ locate '*.gif' >> moje_pliki_jpeg.txt
[ty@fedora2 ty]$
```



Ostrożność podczas przekierowywania wyjścia poleceń

Aby uniknąć przypadkowego nadpisania ważnych plików, zachowaj ostrożność podczas przekierowywania wyjścia poleceń do istniejącego pliku. Kiedy używasz znaku większości (`>`) w celu przekierowania wyjścia do pliku, który już istnieje, zawartość pierwotnego pliku zostaje nadpisana nowymi danymi. Pamiętaj więc, żeby używać podwójnego znaku większości (`>>`), aby nie nadpisać ważnych plików.

Wyszukiwanie wzorców w plikach tekstowych

Utworzyłeś w swoim katalogu domowym długi plik tekstowy o nazwie *moje_pliki_jpeg.txt*, który zawiera listę wszystkich obrazów JPEG i GIF w całym systemie plików Linuksa. Przypuśćmy jednak, że chciałbyś zawęzić tę listę i znaleźć tylko obrazy piłek.

Mógłbyś wczytać plik *moje_pliki_jpeg.txt* do edytora `emacs` lub `vi` i przejrzeć listę w poszukiwaniu plików spełniających to kryterium, ale szybciej znajdziesz żądane obrazy dzięki przeszukaniu pliku *moje_pliki_jpeg.txt* za pomocą polecenia `grep`. Polecenie

`grep` służy właśnie do wyszukiwania określonych przez Ciebie słów lub wzorców w plikach tekstowych. Wywołuje się je w następujący sposób:

```
grep wzorzec plik1 [plik2 ...]
```

Wzorzec to łańcuch tekstowy lub wzorzec, który chcesz znaleźć, a *plik1*, *plik2* itd. to pliki, które należy przeszukać. Wiersze, które zawierają tekst pasujący do łańcucha lub wzorca, zostaną wypisane na standardowym wyjściu (to znaczy na konsoli). Spróbuj teraz wyszukać obrazy piłek (ang. *ball*):

```
[ty@fedora2 ty]$ grep ball moje_pliki_jpeg.txt
/usr/share/apps/klines/balls.jpg
/usr/lib/ooo-1.1/share/gallery/bullets/bluball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/gryball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/golfball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/redball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/poliball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/darkball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/ylwball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/orgball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/grnball.gif
[...]
/usr/share/doc/HTML/en/kdevelop/reference/C/PROBLEMS/whiteball.gif
/usr/share/doc/HTML/en/kdevelop/reference/C/blueball.gif
/usr/local/apache2/icons/ball.red.gif
/usr/local/apache2/icons/ball.gray.gif
/var/www/icons/ball.red.gif
/var/www/icons/ball.gray.gif
[ty@fedora2 ty]$
```

Polecenie `grep` znalazło w pliku *moje_pliki_jpeg.txt* kilka nazw plików, które zawierają słowo `ball`. Polecenie `grep` to doskonałe narzędzie do wydobywania danych z długich list.

Wyszukiwanie plików, które zawierają określone słowa

Za pomocą polecenia `grep` można również łatwo znaleźć pliki, które zawierają określone słowo. Na przykład w Twoim katalogu domowym obecnie znajdują się trzy pliki tekstowe: *moje_pliki_jpeg.txt*, *moj_nowy_plik_vi.txt* oraz *moj_plik_vi.txt*. Przypuśćmy, że chcesz się dowiedzieć, który z tych plików zawiera słowo `Unix`.

Polecenie `grep` z opcją `-l` (lista plików) pozwala szybko i łatwo uzyskać wyniki:

```
[ty@fedora2 ty]$ grep -l Unix *
moj_nowy_plik_vi.txt
moj_plik_vi.txt
[ty@fedora2 ty]$
```

W tym przypadku program `grep` poinformował, że dwa pliki w katalogu roboczym — *moj_nowy_plik_vi.txt* oraz *moj_plik_vi.txt* — zawierają słowo `Unix`.

**Uwaga na wielkość liter**

Kiedy używasz polecenia `grep -l`, przeprowadzasz wyszukiwanie z uwzględnieniem wielkości liter; aby polecenie `grep` znalazło w pliku pasujące słowo, jego litery muszą mieć taką samą wielkość, jak litery w kryterium wyszukiwania.

Aby polecenie `grep` przeprowadziło wyszukiwanie bez uwzględniania wielkości liter, oprócz opcji `-l` podaj opcję `-i`:

```
grep -il unix *
```

Używanie wyników poleceń do złożonych zadań

Jedną z największych zalet wiersza poleceń Linuksa jest możliwość wiązania wielu poleceń i ich danych wyjściowych na wiele różnorodnych sposobów. Dzięki wiązaniu poleceń możesz wykonywać skomplikowane zadania, które składają się z wielu etapów albo wymagają specyficznego zawężenia danych wyjściowych.

Dwie najważniejsze techniki wiązania wielu poleceń to potoki, dzięki którym dane wyjściowe jednego polecenia mogą zostać wykorzystane jako dane wejściowe drugiego, oraz podstawianie poleceń, które pozwala zmodyfikować działanie jednego polecenia w zależności od danych wyjściowych drugiego.

Łączenie poleceń za pomocą potoków

Czasem trzeba użyć danych wyjściowych jednego polecenia jako danych wejściowych drugiego. Służą do tego **potoki**, które w powłoce definiuje się za pomocą pionowej kreski (`|`).

Oto przykład. W poprzednim podrozdziale wyświetliłeś listę wszystkich obrazów plików (w formacie JPEG i GIF), przechowywanych w systemie; utworzyłeś tę listę, przekierowując wyniki dwóch poleceń `locate` do pliku *moje_pliki.jpeg.txt*, a następnie szukając w tym pliku słowa `ball` za pomocą polecenia `grep`. Przypuśćmy, że teraz chcesz się dowiedzieć, czy w systemie plików Linuksa są jakieś obrazy plików zapisane w formatach `.png` lub `.tif`. Czy nie byłoby wygodnie wysłać wyników polecenia `locate` bezpośrednio do polecenia `grep`, aby przeszukało ono dane „w locie”?

Możesz to zrobić za pomocą potoku (`|`).

```
[ty@fedora2 ty]$ locate '*.gif' '*.jpg' '*.tif' '*.png' | grep ball
/usr/lib/ooo-1.1/share/gallery/bullets/bluball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/gryball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/golfball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/redball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/poliball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/darkball.gif
```

```

/usr/lib/ooo-1.1/share/gallery/bullets/ylwball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/orgball.gif
/usr/lib/ooo-1.1/share/gallery/bullets/grnball.gif
/usr/lib/ooo-1.1/share/gallery/rulers/grnballs.gif
/usr/lib/ooo-1.1/share/gallery/rulers/gldballs.gif
/usr/lib/ooo-1.1/share/gallery/rulers/whtballs.gif
/usr/lib/ooo-1.1/share/gallery/rulers/blkballs.gif
/usr/lib/python2.3/site-packages/Ft/Share/Data/Icons/ball.red.gif
/usr/lib/python2.3/site-packages/Ft/Share/Data/Icons/ball.gray.gif
/usr/share/latex2html/icons/purpleball.gif
/usr/share/latex2html/icons/redball.gif
/usr/share/latex2html/icons/yellowball.gif
/usr/share/latex2html/icons/whiteball.gif
/usr/share/latex2html/icons/blueball.gif
/usr/share/latex2html/icons/orangeball.gif
/usr/share/latex2html/icons/pinkball.gif
/usr/share/latex2html/icons/greenball.gif
/usr/share/doc/gettext/javadoc1/images/cyan-ball.gif
/usr/share/doc/gettext/javadoc1/images/yellow-ball-small.gif
/usr/share/doc/gettext/javadoc1/images/magenta-ball.gif
/usr/share/doc/gettext/javadoc1/images/cyan-ball-small.gif
/usr/share/doc/gettext/javadoc1/images/green-ball.gif
[...]
/usr/share/apps/kbounce/pics/ball0022.png
/usr/share/apps/kbounce/pics/ball0002.png
/usr/share/apps/kbounce/pics/ball0016.png
/usr/share/apps/kbounce/pics/ball0013.png
/usr/share/apps/kbounce/pics/ball0020.png
/usr/share/apps/kbounce/pics/ball0023.png
/usr/share/apps/kbounce/pics/ball0017.png
/usr/share/apps/kbounce/pics/ball0000.png
/usr/share/apps/kbounce/pics/ball0010.png
/usr/share/apps/kbounce/pics/ball0001.png
/usr/share/apps/kbounce/pics/ball0019.png
/usr/local/apache2/icons/ball1.red.png
/usr/local/apache2/icons/ball1.gray.png
/var/www/icons/ball1.red.png
/var/www/icons/ball1.gray.png
[ty@fedora2 ty]$

```

Utworzyłeś listę wszystkich obrazów piłek, przechowywanych w czterech różnych formatach w całym systemie plików Linuksa. Wyniki polecenia `locate` (któremu podałeś cztery argumenty) zostały przesłane bezpośrednio do polecenia `grep`, które przeszukało je pod kątem słowa `ball`.

Bardziej przyziemnym, ale również użytecznym zastosowaniem potoków jest stronicowanie wyników poleceń. Spróbuj na przykład wpisać poniższe polecenie:

```

[ty@fedora2 ty]$ ls -l /usr/bin
[...]

```

Listing jest bardzo długi; znaczna jego część zniknie za górną krawędzią ekranu, zanim będziesz miał okazję obejrzeć wyniki. Jeśli jednak wyślesz wyniki tego polecenia przez potok do innego polecenia o nazwie `more`, problem zostanie rozwiązany. Polecenie `more`

to program stronicujący: wyświetla plik albo dane wejściowe stroną po stronie, zatrzymując się i oczekując na naciśnięcie klawisza spacji przed pokazaniem następnego ekranu informacji. Spróbuj teraz połączyć wyjście polecenia `ls` z wejściem polecenia `more`:

```
[ty@fedora2 ty]$ ls -l /usr/bin | more
[...]
```

Listing nadal zawiera wiele informacji, ale teraz możesz oglądać je w dowolnie wybranym tempie. Nie przejmuj się, jeśli zastosowania potoków wydają Ci się niejasne; przyzwyczaisz się do nich, kiedy będziesz pracował z powłoką.

Używanie wyników jednego polecenia jako argumentów drugiego

Innym zaawansowanym mechanizmem powłoki jest **podstawianie poleceń**. Dzięki podstawianiu poleceń wyniki jednego polecenia mogą zostać wykorzystane jako zbiór argumentów drugiego. Procedura ta umożliwia modyfikowanie działania drugiego polecenia w zależności od wyników pierwszego, co wpływa na wyniki drugiego polecenia i pozwala *dostosować* je do określonej sytuacji.

Przypuśćmy, że chcesz utworzyć katalog o nazwie *pliki_jpeg* i zgromadzić w nim wszystkie obrazy JPEG przechowywane w systemie, aby mieć do nich łatwy dostęp. Wiesz już, jak użyć polecenia `locate` w celu znalezienia nazw plików zakończonych na `.jpg` w całym systemie plików Linuksa. Dysponując tymi wynikami, możesz posłużyć się podstawianiem poleceń, aby zgromadzić pliki w jednym miejscu:

```
[ty@fedora2 ty]$ mkdir pliki_jpeg
[ty@fedora2 ty]$ ln -s $(locate '*.jpg') pliki_jpeg
[ty@fedora2 ty]$
```

Umieszczenie polecenia `locate '*.jpg'` w nawiasie i poprzedzenie go znakiem dolara (\$) sprawiło, że nazwy plików wygenerowane przez polecenie `locate` zostały potraktowane tak, jakby użytkownik wpisał je jedna po drugiej jako argumenty polecenia `ln -s` (które służy do tworzenia dowiązań symbolicznych).



Dowiązania symboliczne

Dowiązania symboliczne i polecenie `ln -s` omówiono w rozdziale 4.

Użyj polecenia `ls`, aby wyświetlić długi listing plików w katalogu *pliki_jpeg*. Przekonasz się, że utworzone zostały dowiązania symboliczne do każdego pliku `.jpg` w całym systemie; wszystkie obrazy JPEG są zgromadzone w jednym miejscu i masz do nich łatwy dostęp.

Podobnie jak w przypadku potoków, szeroka gama zastosowań podstawiania poleceń prawdopodobnie nie będzie od razu oczywista. Nie martw się tym — będziesz miał okazję użyć go ponownie w następnych rozdziałach.



Dwa sposoby podstawiania poleceń

Podstawienie poleceń w powłocie `bash` może zachodzić na dwa sposoby. Pierwszy, pokazany w powyższym przykładzie, polega na umieszczeniu podstawianego polecenia w nawiasie poprzedzonym znakiem dolara:

```
$(locate moj_plik)
```

Drugi, tradycyjny sposób podstawiania poleceń polega na umieszczeniu polecenia między znakami odwróconego apostrofu:

```
`locate moj_plik`
```

W tej książce korzystamy z mniej tradycyjnej metody, ponieważ odwrócone apostrofy często trudno odróżnić od zwykłych, co może prowadzić do nieporozumień.

Sterowanie programami powłoki

W następnych rozdziałach będziemy uruchamiać z poziomu konsoli dość skomplikowane aplikacje. Zanim dotrzemy do bardziej zaawansowanych zagadnień, musisz poznać metody sterowania zadaniami, działającymi pod kontrolą powłoki.

Możesz zalogować się na kilku konsolach wirtualnych, aby jednocześnie uruchomić wiele pełnoekranowych aplikacji — na przykład edytor `emacs` na konsoli pierwszej, a `vi` na konsoli drugiej. Przełączanie się między konsolami wirtualnymi bywa jednak dezorientujące i niewygodne. Co więcej, kiedy nauczysz się logować zdalnie za pomocą programów `telnet` lub `ssh`, będziesz musiał używać wiersza poleceń, nie dysponując wieloma konsolami wirtualnymi.

Aby uniknąć związanych z tym problemów, musisz nauczyć się technik wstrzymywania i wznowiania wielu zadań z pojedynczego wiersza poleceń. Dzięki nim będziesz mógł przełączać aplikacje, przenosić zadania na pierwszy plan lub do tła oraz przerywać niepotrzebne zadania. Poniższe podrozdziały nauczą Cię, jak używać omawianych technik w celu usprawnienia pracy nad wieloma projektami na pojedynczej konsoli Linuksa.

Przełączanie otwartych aplikacji

Przypuśćmy, że chcesz uruchomić jednocześnie edytory `emacs` i `vi` oraz przełączać się między nimi, nie zmieniając bieżącego stanu żadnej z tych aplikacji. Może pracujesz nad artykułem porównującym oba edytory, a może po prostu edytujesz plik w `vi`, a od czasu do czasu robisz sobie przerwę na grę `dunnet`, wspomnianą przy okazji omawiania edytora `emacs`.

Za pomocą kombinacji klawiszy `Ctrl + Z` możesz wstrzymać pracę bieżącej aplikacji, aby uruchomić nowy program. Polecenie `jobs` podaje listę wszystkich otwartych aplikacji.

Wczytaj plik *moj_plik_vi.txt*, utworzony wcześniej w tym rozdziale, do edytora *vi*:

```
[ty@fedora2 ty]$ vi moj_plik_vi.txt
```

Kiedy na ekranie pojawi się edytor *vi*, monit poleceń zniknie i nie będziesz miał do niego dostępu. Możesz teraz edytować plik, dokonując w nim niezbędnych zmian. Co jednak zrobić, gdy przyjdzie czas na sesję gry *dunnet*?

Wstrzymaj proces *vi*, naciskając klawisze *Ctrl + Z*:

```
[1]+ Stopped          vim moj_plik_vi.txt
[ty@fedora2 ty]$
```

Proces *vi* został wstrzymany. Zwróć uwagę na liczbę 1 w nawiasach kwadratowych; jest to **numer zadania** wstrzymanego procesu *vi*. Teraz możesz uruchomić edytor *emacs* i wpisać polecenie *M-x dunnet*, aby wczytać grę *dunnet*:

```
[ty@fedora2 ty]$ emacs
```

Po chwili relaksu trzeba będzie wrócić do dokumentu w edytorze *vi*; nie można grać w nieskończoność. Nie chcesz jednak zamykać edytora *emacs* i tracić bieżącej gry. Możesz wstrzymać proces *emacs* w taki sam sposób, w jaki wcześniej wstrzymałeś proces *vi* — naciskając klawisze *Ctrl + Z*:

```
[2]+ Stopped          vim moj_plik_vi.txt
[ty@fedora2 ty]$
```

Teraz na liście zadań masz dwa wstrzymane edytory tekstu. Aby zobaczyć listę bieżących zadań, wpisz polecenie *jobs*:

```
[ty@fedora2 ty]$ jobs
[1]- Stopped          vim moj_plik_vi.txt
[2]+ Stopped          emacs
[ty@fedora2 ty]$
```

Zadania mogą pozostawać wstrzymane dowolnie długo; możesz kontynuować pracę w powłoce i wykonywać inne zadania, a *vi* i *emacs* będą gotowe do wznowienia pracy dokładnie w tym punkcie, w którym je zostawiłeś.



Przerywanie działającego zadania

Jeśli chcesz przerwać, a nie tylko wstrzymać działające zadanie, naciśnij klawisze *Ctrl + C*. Czasem kombinacja ta nie zadziała. Nie uda Ci się na przykład przerwać w ten sposób pracy edytora *emacs* ani *vi*, choć będziesz mógł zatrzymać wyświetlanie szczególnie długiego listingu polecenia *locate*.

Wznawianie zadania za pomocą polecenia *fg*

Aby wrócić do edytowania dokumentu w programie *vi*, użyj polecenia *fg*, przekazując mu argument w postaci znaku procentu (%) oraz właściwego numeru zadania:

```
[ty@fedora2 ty]$ fg %1
```

Jak pokazało polecenie `jobs`, proces `vi` ma numer zadania 1. Kiedy wpiszesz polecenie `fg` (od ang. *foreground* — pierwszy plan), edytor `vi` wróci na ekran dokładnie w takim samym stanie, w jakim był, kiedy wstrzymałeś jego działanie.

Możesz powtarzać tę procedurę tak często, jak to jest potrzebne, wstrzymując i wznowiając pracę programów w dowolnej kolejności i kombinacji.

Uruchamianie zadania w tle za pomocą polecenia `bg`

Niektóre polecenia Linuksa wykonują się dość długo, zwłaszcza w starszych systemach. W takich przypadkach powłoka oferuje idealne rozwiązanie. Polecenia, które mogą działać bez interwencji użytkownika, można uruchomić **w tle** za pomocą polecenia `bg`. Po wydaniu tego polecenia możesz wrócić do pracy w innej otwartej aplikacji.

Przypuśćmy, że podczas edycji dokumentu w `vi` i gry w programie `emacs` postanawiasz wygenerować listę wszystkich plików, które może wyszukać polecenie `locate`. Aby zapisać tę listę w pliku o nazwie `lista_plikow.txt`, musisz najpierw wstrzymać działanie edytora za pomocą kombinacji klawiszy `Ctrl + Z`, a następnie wprowadzić poniższe polecenie:

```
[ty@fedora2 ty]$ locate '*' >lista_plikow.txt
```

Polecenie `locate` otrzymuje wzorzec, który pasuje do każdego pliku w jego bazie danych. Wyjście zostało przekierowane do pliku `lista_plikow.txt`. Zadanie jest nieskomplikowane, ale w większości komputerów jego wykonanie zajmie sporo czasu. Wstrzymajmy je więc i przenieśmy w tło. Naciśnij klawisze `Ctrl + Z`, aby wstrzymać działanie polecenia:

```
[3]+ Stopped          locate '*' >lista_plikow.txt
[ty@fedora2 ty]$
```

Zauważ, że polecenie otrzymało numer zadania 3 i że na ekranie ponownie pojawił się monit poleceń. Polecenie jest teraz wstrzymane; jeśli go nie wznowimy, nigdy nie zakończy pracy. Aby wznowić pracę polecenia w tle, użyj polecenia `bg` (od ang. *background* — tło):

```
[ty@fedora2 ty]$ bg %3
[3]+ locate '*' >lista_plikow.txt &
[ty@fedora2 ty]$
```

Polecenie działa teraz w tle; Linux będzie je wykonywał, dopóki zadanie nie zostanie zrealizowane. Ty tymczasem możesz wrócić do edytora albo do gry `dunnet` za pomocą polecenia `fg`. Aby najpierw zobaczyć zaktualizowaną listę zadań, wpisz polecenie `jobs`:

```
[ty@fedora2 ty]$ jobs
[1]- Stopped          vim moj_plik_vi.txt
[2]+ Stopped          emacs
[3]  Running          locate '*' >lista_plikow.txt &
[ty@fedora2 ty]$
```

W pewnym momencie komunikat o zakończeniu procesu działającego w tle wygląda tak:

```
[3] Done              locate '*' >lista_plikow.txt &
```

Zakończony proces nie będzie się pojawiał na liście zadań wraz z działającymi lub wstrzymanymi procesami.

Ostatnie uwagi o sterowaniu zadaniami

Istnieją jeszcze dwa polecenia związane ze sterowaniem zadaniami, które mogą uprościć Ci pracę w wierszu poleceń. Pierwsze z nich to `kill`. Polecenie `kill` z argumentem w postaci numeru zadania przerywa zadanie, które jest już niepotrzebne:

```
[ty@fedora2 ty]$ kill %2
[1]- Stopped          vim moj_plik_vi.txt
[2]+ Stopped          emacs
   [2] Stopped          emacs
[ty@fedora2 ty]$
```

Polecenie `kill` pomaga w usuwaniu zadań, natomiast znak ampersand (&) pomaga w ich tworzeniu. Jeśli chcesz od początku uruchomić zadanie w tle, zamiast naciskać *Ctrl* + *Z* i wpisywać polecenie `bg` wystarczy, że zakończysz wiersz polecenia znakiem ampersand. We wcześniejszym przykładzie mógłbyś użyć polecenia:

```
[ty@fedora2 ty]$ locate '*' > lista_plikow.txt &
[2] 14159
[ty@fedora2 ty]$
```

Pierwsza liczba zwracana po uruchomieniu zadania w tle to znany Ci już numer zadania. Druga liczba to systemowy numer procesu; szczegółowe informacje o systemowej tabeli procesów znajdziesz w rozdziale 18.



Zadania działające po wylogowaniu użytkownika

W zwykłych okolicznościach zadania, które działają w tle, są przerywane, kiedy się wylogujesz — tak, jakbyś usunął je własnoręcznie.

Aby temu zapobiec i pozwolić na kontynuowanie zadania nawet po wylogowaniu, użyj polecenia `nohup`:

```
nohup %zadanie
```

Parametr *zadanie* zastąp numerem zadania, które ma działać nawet wówczas, kiedy się wylogujesz. Pamiętaj jednak, że jeśli komputera używa wiele osób, pozostawianie zadań działających w tle jest niekulturalne, ponieważ spowalnia to działanie komputera.

Podsumowanie

W tym rozdziale nabyłeś umiejętności, dzięki którym z czasem zaczniesz sprawnie posługiwać się wierszem poleceń Linuksa. Utworzyłeś pliki tekstowe za pomocą dwóch popularnych edytorów i nauczyłeś się grupować pliki, aby efektywnie nimi zarządzać. Poznałeś też polecenia do szybkiego wyszukiwania nazw plików, typów plików oraz

słów lub wzorców w plikach. Nauczyłeś się dwóch sposobów łączenia poleceń, które pozwalają oszczędzić czas, wysiłek i miejsce na dysku. Dowiedziałeś się wreszcie, jak przełączać działające programy powłoki, nie zamykając ich ani nie korzystając z wielu konsoli.

Przy okazji nauczyłeś się wielu poleceń, kombinacji klawiszy i znaków specjalnych, używanych w wierszu poleceń powłoki. Pamiętaj, że:

- ♦ znak `>` przekierowuje standardowe wyjście do nowego pliku;
- ♦ znak `>>` dołącza standardowe wyjście do istniejącego pliku;
- ♦ znak `&` uruchamia proces w tle;
- ♦ kombinacja klawiszy `Ctrl + Z` wstrzymuje działający proces;
- ♦ polecenia `bg`, `fg` i `jobs` służą — odpowiednio — do wznawiania zadania w tle, wznawiania zadania na pierwszym planie oraz wyświetlania wszystkich bieżących zadań;
- ♦ polecenie `find` wyszukuje pliki w czasie rzeczywistym;
- ♦ polecenie `grep` szuka w pliku tekstowym albo w zbiorze plików określonego tekstu lub wzorca i zwraca nazwy plików, które go zawierają;
- ♦ polecenie `locate` wyszukuje lokalizację pliku za pomocą często aktualizowanej bazy danych;
- ♦ polecenie `more` wyświetla strona po stronie pliki utworzone w programach takich, jak `vi` lub `emacs`.

Poznałeś powłokę, polecenia i konsolę na tyle dobrze, że możesz przystąpić do rzeczywistej pracy. Dopóki w rozdziale 9. („Ujarzmianie potęgi powłoki”) nie zajmiemy się zaawansowanymi aspektami powłoki, nie będziesz już miał do czynienia z tego rodzaju technicznymi szczegółami — czas zająć się informacjami i aplikacjami.

Pytania i odpowiedzi

- P:** Kiedy przekierowuję wyjście niektórych poleceń do pliku za pomocą znaku wielkości, niektóre komunikaty i tak pojawiają się na konsoli. Dlaczego?
- O:** Komunikaty te nie są wysyłane na standardowe wyjście, lecz na standardowe wyjście błędów; informują one użytkownika, że stało się coś nieoczekiwanego. Aby przekierować standardowe wyjście błędów, wpisz na końcu polecenia symbol `>>`, a po nim nazwę pliku, w którym mają być zapisywane komunikaty o błędach.
- P:** Czy da się po prostu odrzucić komunikaty kierowane na standardowe wyjście lub standardowe wyjście błędów — kiedy na przykład chcę, żeby polecenie działało „cicho”?
- O:** Tak, przekieruj jedno albo oba wyjścia do specjalnego pliku urządzenia `/dev/null`, potocznie nazywanego „koszem na bity”. System odrzuca wszystkie dane kierowane do pliku `/dev/null`.

Warsztat

Warsztat ma pomóc Ci w przewidywaniu możliwych pytań, powtarzaniu nabytej wiedzy oraz przekładaniu teorii na praktykę.

Quiz

1. Jakiego wzorca użyłbyś w poleceniu `rm`, aby usunąć pliki `cena_domu.txt`, `ocena.gif` i `samochod.cena`?
2. Jak wyszukałbyś wiersze, zawierające słowo czerwone w pliku `kolory_butow.txt`?
3. Jak wyświetliłbyś listę działających zadań edytora `emacs`? (Wskazówka: użyj potoku).
4. Jak szybko przeszukać cały system plików, aby odnaleźć plik `ReleaseNotes.html`?

Odpowiedzi

1. `*cena*`
2. `grep czerwone kolory_butow.txt`
3. `jobs | grep emacs`
4. `locate ReleaseNotes.html`

Ćwiczenia

1. Poświęć trochę czasu na przestudiowanie samouczków w edytorach `emacs` oraz `vi` i zapoznaj się z możliwościami obu programów.
2. Użyj poleceń `locate` i `more`, aby uzyskać listę wszystkich plików typu `.txt`, `.gif`, `.jpg` i `.gz` oraz wyświetlić ją na ekranie strona po stronie.
3. Uruchamiaj wiele zadań i kolejno je wstrzymuj, dopóki wyniki polecenia `jobs` nie wypełnią całego ekranu. Następnie zakończ lub usuń każde zadanie po kolei.
4. Uruchom edytor `emacs` i zagraj w `dunnet`.