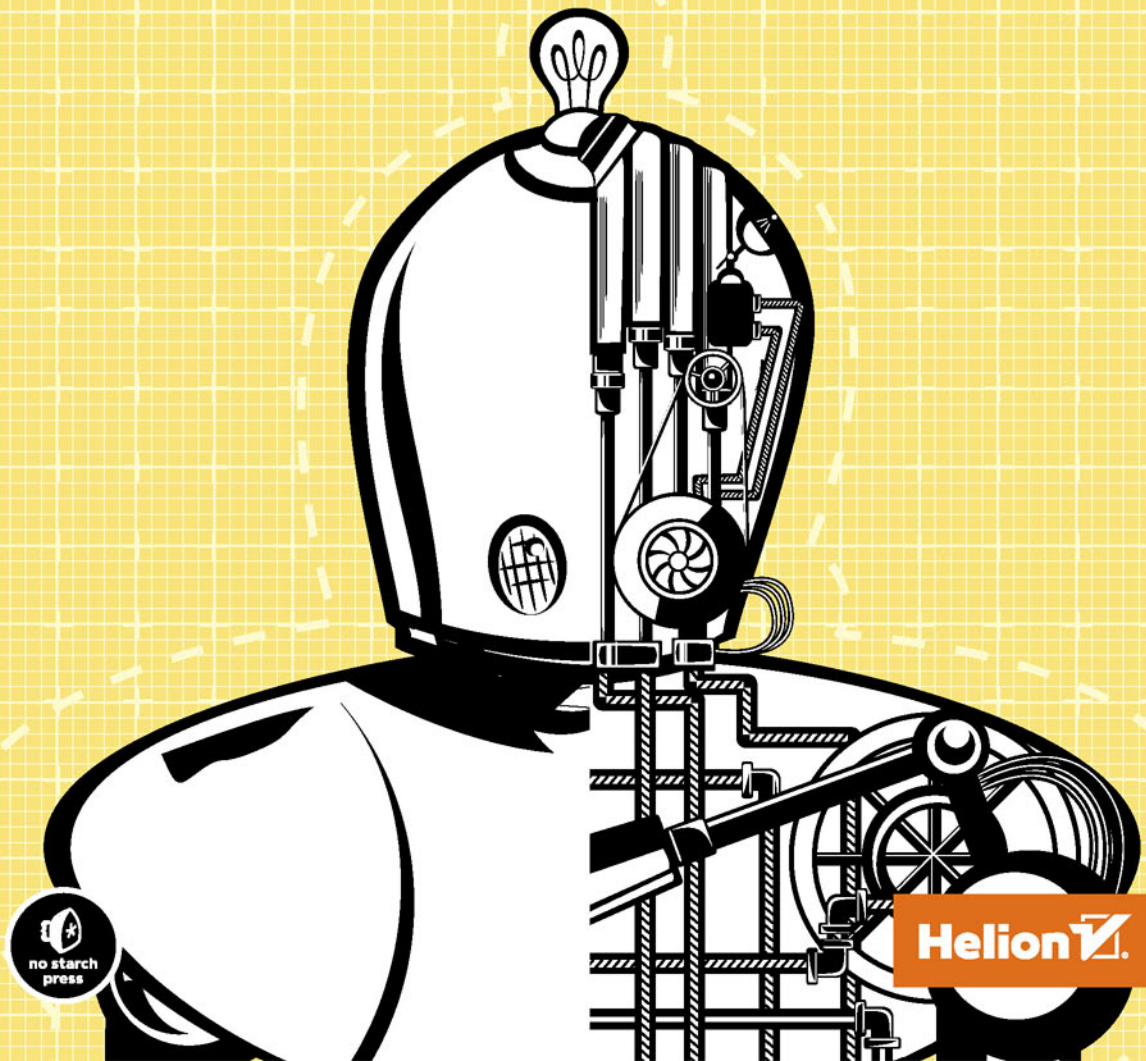


WYDANIE II

# JAK DZIAŁA LINUX

PODRĘCZNIK ADMINISTRATORA

BRIAN WARD



no starch  
press

Helion Z.

Tytuł oryginału: How Linux Works: What Every Superuser Should Know, Second Edition

Tłumaczenie: Piotr Pilch

ISBN: 978-83-283-0980-7

Copyright © 2015 by Brian Ward. Title of English-language original: How Linux Works, 2nd Edition, ISBN 978-1-59327-567-9, published by No Starch Press.

Polish-language edition copyright © 2015 by Helion SA.  
All rights reserved.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiegokolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz Wydawnictwo HELION dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz Wydawnictwo HELION nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Wydawnictwo HELION  
ul. Kościuszki 1c, 44-100 GLIWICE  
tel. 32 231 22 19, 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/jakli2>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

- [Kup książkę](#)
- [Poleć książkę](#)
- [Oceń książkę](#)

- [Księgarnia internetowa](#)
- [Lubię to! » Nasza społeczność](#)

# Spis treści

<b>PODZIĘKOWANIA .....</b>	<b>17</b>
----------------------------	-----------

<b>WSTĘP .....</b>	<b>19</b>
--------------------	-----------

Kto powinien przeczytać tę książkę? .....	19
Wymagania .....	20
Jak czytać tę książkę? .....	20
Ćwiczenia .....	21
Podział tej książki .....	21
Co nowego w drugim wydaniu? .....	22
Kilka słów o terminologii .....	22

## I

<b>INFORMACJE OGÓLNE .....</b>	<b>23</b>
--------------------------------	-----------

1.1. Poziomy i warstwy abstrakcji w systemie Linux .....	24
1.2. Sprzęt: pamięć operacyjna .....	26
1.3. Jądro systemu .....	26
1.3.1. Zarządzanie procesami .....	27
1.3.2. Zarządzanie pamięcią .....	28
1.3.3. Sterowniki urządzeń i zarządzanie urządzeniami .....	29
1.3.4. Wywołania systemowe .....	29
1.4. Przestrzeń użytkownika .....	30
1.5. Użytkownicy .....	32
1.6. Spojrzenie w przyszłość .....	33

## 2

### **PODSTAWOWE POLECENIA I HIERARCHIA KATALOGÓW ..... 35**

2.1.	Powłoka Bourne'a: /bin/sh .....	36
2.2.	Korzystanie z powłoki .....	37
2.2.1.	Okno powłoki .....	37
2.2.2.	Polecenie cat .....	37
2.2.3.	Standardowe wejście i wyjście .....	38
2.3.	Podstawowe polecenia .....	39
2.3.1.	Polecenie ls .....	39
2.3.2.	Polecenie cp .....	40
2.3.3.	Polecenie mv .....	40
2.3.4.	Polecenie touch .....	40
2.3.5.	Polecenie rm .....	41
2.3.6.	Polecenie echo .....	41
2.4.	Polecenia działające na katalogach .....	41
2.4.1.	Polecenie cd .....	42
2.4.2.	Polecenie mkdir .....	42
2.4.3.	Polecenie rmdir .....	42
2.4.4.	Rozwijanie nazw (nazwy wieloznaczne) .....	42
2.5.	Polecenia pośredniczące .....	44
2.5.1.	grep .....	44
2.5.2.	Polecenie less .....	45
2.5.3.	Polecenie pwd .....	45
2.5.4.	Polecenie diff .....	46
2.5.5.	Polecenie file .....	46
2.5.6.	Polecenia find i locate .....	46
2.5.7.	Polecenia head i tail .....	47
2.5.8.	Polecenie sort .....	47
2.6.	Zmianianie hasła i powłoki .....	47
2.7.	Pliki z kropką .....	47
2.8.	Zmienne środowiskowe i powłoki .....	48
2.9.	Ścieżka poleceń .....	49
2.10.	Znaki specjalne .....	50
2.11.	Edycja wiersza poleceń .....	50
2.12.	Edytory tekstu .....	51
2.13.	Uzyskiwanie pomocy .....	52
2.14.	Wejście i wyjście powłoki .....	54
2.14.1.	Standardowy strumień błędów .....	55
2.14.2.	Przekierowywanie standardowego wejścia .....	56
2.15.	Prawidłowe odczytywanie komunikatów o błędach .....	56
2.15.1.	Anatomia uniksowych komunikatów o błędach .....	56
2.15.2.	Typowe błędy .....	57
2.16.	Przeglądanie procesów i manipulowanie nimi .....	59
2.16.1.	Opcje polecenia .....	59
2.16.2.	Przerywanie działania procesów .....	60

2.16.3.	Kontrola zadań .....	61
2.16.4.	Procesy działające w tle .....	61
2.17.	Tryby plików i uprawnienia .....	62
2.17.1.	Modyfikowanie uprawnień .....	64
2.17.2.	Dowiązania symboliczne .....	65
2.17.3.	Tworzenie dowiązań symbolicznych .....	66
2.18.	Archiwizowanie i kompresowanie plików .....	67
2.18.1.	Program gzip .....	67
2.18.2.	Program tar .....	67
2.18.3.	Archiwa skompresowane (.tar.gz) .....	69
2.18.4.	Program zcat .....	69
2.18.5.	Inne narzędzia kompresujące .....	70
2.19.	Hierarchia katalogów .....	70
2.19.1.	Pozostałe katalogi główne .....	72
2.19.2.	Katalog /usr .....	73
2.19.3.	Umieszczenie jądra systemu .....	73
2.20.	Uruchamianie poleceń przez superużytkownika .....	74
2.20.1.	Polecenie sudo .....	74
2.20.2.	Plik /etc/sudoers .....	74
2.21.	Podsumowanie .....	75

### 3

<b>URZĄDZENIA .....</b>	<b>77</b>	
3.1.	Pliki urządzeń .....	78
3.2.	Ścieżka urządzeń sysfs .....	79
3.3.	Polecenie dd i urządzenia .....	81
3.4.	Podsumowanie nazewnictwa urządzeń .....	82
3.4.1.	Dyski twarde — /dev/sd* .....	82
3.4.2.	Napędy CD i DVD: /dev/sr* .....	83
3.4.3.	Dyski twarde PATA: /dev/hd* .....	84
3.4.4.	Terminale: /dev/tty*, /dev/pts/* i /dev/tty .....	84
3.4.5.	Porty szeregowy — /dev/ttyS* .....	85
3.4.6.	Porty równoległe — /dev/lp0 i /dev/lp1 .....	85
3.4.7.	Urządzenia audio — /dev/dsp, /dev/audio, /dev/snd/* i inne .....	86
3.4.8.	Tworzenie plików urządzeń .....	86
3.5.	System udev .....	87
3.5.1.	System plików devtmpfs .....	87
3.5.2.	Konfiguracja i działanie procesu udevd .....	88
3.5.3.	Program udevadm .....	90
3.5.4.	Monitorowanie urządzeń .....	91
3.6.	Szczegóły: SCSI i jądro Linuksa .....	92
3.6.1.	Pamięci masowe USB i protokół SCSI .....	95
3.6.2.	SCSI i ATA .....	96
3.6.3.	Ogólne urządzenia SCSI .....	97
3.6.4.	Wiele metod dostępu do jednego urządzenia .....	98

## 4

### **DYSKI I SYSTEMY PLIKÓW ..... 99**

4.1.	Partycjonowanie urządzeń dyskowych .....	102
4.1.1.	Przeglądanie tablicy partycji .....	102
4.1.2.	Modyfikowanie tablicy partycji .....	104
4.1.3.	Geometria dysku i partycji .....	105
4.1.4.	Dyski SSD .....	107
4.2.	Systemy plików .....	107
4.2.1.	Typy systemów plików .....	108
4.2.2.	Tworzenie systemu plików .....	109
4.2.3.	Montowanie systemu plików .....	110
4.2.4.	Identyfikator UUID systemu plików .....	112
4.2.5.	Buforowanie dysku i systemu plików .....	113
4.2.6.	Opcje montowania systemów plików .....	114
4.2.7.	Ponownie montowanie systemu plików .....	115
4.2.8.	Tabela systemów plików /etc/fstab .....	116
4.2.9.	Rozwiązania konkurencyjne dla pliku /etc/fstab .....	118
4.2.10.	Pojemność systemu plików .....	118
4.2.11.	Sprawdzanie i naprawianie systemów plików .....	119
4.2.12.	Systemy plików o specjalnym znaczeniu .....	122
4.3.	Przestrzeń wymiany .....	123
4.3.1.	Wykorzystywanie partycji jako przestrzeni wymiany .....	123
4.3.2.	Wykorzystywanie pliku jako przestrzeni wymiany .....	124
4.3.3.	Jak wielkiej przestrzeni wymiany potrzebuję? .....	124
4.4.	Spojrzenie w przyszłość: dyski i przestrzeń użytkownika .....	125
4.5.	Tradycyjny system plików .....	126
4.5.1.	Przeglądanie szczegółów węzłów inode .....	128
4.5.2.	Praca z systemami plików w przestrzeni użytkownika .....	130
4.5.3.	Ewolucja systemów plików .....	130

## 5

### **JAK URUCHAMIA SIĘ LINUX? ..... 133**

5.1.	Komunikaty rozruchowe .....	134
5.2.	Inicjowanie jądra i opcje rozruchu .....	135
5.3.	Parametry jądra .....	136
5.4.	Programy rozruchowe .....	137
5.4.1.	Zadania programu rozruchowego .....	138
5.4.2.	Przegląd programów rozruchowych .....	138
5.5.	Wprowadzenie do programu GRUB .....	139
5.5.1.	Przeszukiwanie urządzeń i partycji za pomocą wiersza poleceń programu GRUB .....	142
5.5.2.	Konfigurowanie programu GRUB .....	144
5.5.3.	Instalowanie programu GRUB .....	146
5.6.	Problemy z bezpiecznym rozruchem UEFI .....	148
5.7.	Ładowanie innych systemów operacyjnych .....	149

5.8.	Szczegóły programu rozruchowego .....	150
5.8.1.	Rozruch MBR .....	150
5.8.2.	Rozruch UEFI .....	150
5.8.3.	Jak działa GRUB? .....	151

## 6

### **URUCHAMIANIE PRZESTRZENI UŻYTKOWNIKA ..... 153**

6.1.	Wprowadzenie do procesu init .....	154
6.2.	Poziomy uruchomienia System V .....	155
6.3.	Identyfikowanie rodzaju procesu init .....	156
6.4.	systemd .....	156
6.4.1.	Jednostki i typy jednostek .....	157
6.4.2.	Zależności systemd .....	158
6.4.3.	Konfiguracja systemd .....	160
6.4.4.	Praca z systemd .....	163
6.4.5.	Dodawanie jednostek systemd .....	166
6.4.6.	Śledzenie i synchronizacja procesów systemd .....	167
6.4.7.	Uruchamianie na żądanie i zrównoleglenie zasobów .....	168
6.4.8.	Zgodność systemd z System V .....	173
6.4.9.	Programy pomocnicze systemd .....	173
6.5.	Upstart .....	174
6.5.1.	Procedura inicjowania procesu Upstart .....	175
6.5.2.	Zadania w procesie Upstart .....	176
6.5.3.	Konfiguracja Upstart .....	178
6.5.4.	Działanie procesu Upstart .....	183
6.5.5.	Protokoły procesu Upstart .....	184
6.5.6.	Poziomy uruchomienia procesu Upstart i zgodność z System V .....	185
6.6.	Proces init System V .....	186
6.6.1.	Proces init w stylu System V: sekwencja poleceń rozruchowych .....	188
6.6.2.	Farma dowiązań procesu init w stylu System V .....	189
6.6.3.	run-parts .....	190
6.6.4.	Sterowanie procesem init w stylu System V .....	191
6.7.	Wyłączanie systemu .....	192
6.8.	Początkowy system plików w pamięci RAM .....	193
6.9.	Rozruch awaryjny i tryb pojedynczego użytkownika .....	195

## 7

### **KONFIGURACJA SYSTEMU: REJESTROWANIE, CZAS SYSTEMOWY, ZADANIA WSADOWE I UŻYTKOWNICY ..... 197**

7.1.	Struktura katalogu /etc .....	198
7.2.	Rejestrowanie dzienników systemowych .....	199
7.2.1.	Rejestrator systemowy .....	199
7.2.2.	Pliki konfiguracyjne .....	199
7.3.	Pliki związane z zarządzaniem użytkownikami .....	202
7.3.1.	Plik /etc/passwd .....	202
7.3.2.	Użytkownicy specjalni .....	203

7.3.3.	Plik /etc/shadow .....	204
7.3.4.	Manipulowanie użytkownikami i hasłami .....	204
7.3.5.	Praca z grupami .....	205
7.4.	Programy getty i login .....	206
7.5.	Ustawianie czasu .....	206
7.5.1.	Reprezentacja czasu jądra i strefy czasowe .....	207
7.5.2.	Czas sieciowy .....	208
7.6.	Planowanie powtarzalnych zadań w programie cron .....	209
7.6.1.	Instalowanie plików crontab .....	210
7.6.2.	Systemowe pliki crontab .....	210
7.6.3.	Przyszłość narzędzia cron .....	211
7.7.	Planowanie jednorazowych zadań w programie at .....	211
7.8.	Identyfikatory użytkowników i przełączanie ich .....	212
7.8.1.	Prawo właściciela procesu, efektywny identyfikator użytkownika, rzeczywisty identyfikator użytkownika i zapisany identyfikator użytkownika .....	212
7.9.	Identyfikowanie i uwierzytelnianie użytkowników .....	215
7.9.1.	Użycie bibliotek do uzyskiwania informacji o użytkownikach .....	216
7.10.	System PAM .....	217
7.10.1.	Konfiguracja systemu PAM .....	217
7.10.2.	Uwagi dotyczące systemu PAM .....	221
7.10.3.	System PAM i hasła .....	222
7.11.	Spojrzenie w przyszłość .....	223

## 8

### **WYKORZYSTANIE PROCESÓW I ZASOBÓW ..... 225**

8.1.	Śledzenie procesów .....	226
8.2.	Wyszukiwanie otwartych plików programem lsof .....	226
8.2.1.	Analizowanie danych wyjściowych polecenia lsof .....	227
8.2.2.	Użycie polecenia lsof .....	228
8.3.	Śledzenie działania programu i wywołań systemowych .....	228
8.3.1.	Polecenie strace .....	229
8.3.2.	Polecenie ltrace .....	230
8.4.	Wątki .....	231
8.4.1.	Procesy jednowątkowe i wielowątkowe .....	231
8.4.2.	Wyświetlanie wątków .....	231
8.5.	Wprowadzenie do monitorowania zasobów .....	233
8.6.	Pomiar czasu procesora .....	233
8.7.	Nadawanie procesom priorytetów .....	234
8.8.	Średnie obciążenia .....	235
8.8.1.	Użycie polecenia uptime .....	235
8.8.2.	Wysokie obciążenia .....	236
8.9.	Pamięć .....	237
8.9.1.	Zasady działania pamięci .....	237
8.9.2.	Błędy stron .....	238
8.10.	Monitorowanie wydajności procesora i pamięci za pomocą polecenia vmstat .....	239



8.11.	Monitorowanie operacji wejścia-wyjścia .....	241
8.11.1.	Użycie narzędzia iostat .....	241
8.11.2.	Wykorzystanie i monitorowanie urządzeń wejścia-wyjścia dla poszczególnych procesów — narzędzie iotop .....	243
8.12.	Monitorowanie poszczególnych procesów za pomocą narzędzia pidstat .....	244
8.13.	Dodatkowe zagadnienia .....	244

## 9

### **SIEĆ I JEJ KONFIGURACJA .....** **247**

9.1.	Podstawy dotyczące sieci .....	248
9.1.1.	Pakiety .....	248
9.2.	Warstwy sieciowe .....	249
9.3.	Warstwa internetowa .....	250
9.3.1.	Wyświetlanie adresu IP używanego komputera .....	252
9.3.2.	Podsieci .....	252
9.3.3.	Typowe maski podsieci i notacja CIDR .....	253
9.4.	Trasy i tabela routingu jądra .....	254
9.4.1.	Brama domyślna .....	255
9.5.	Podstawowe narzędzia protokołu ICMP i systemu DNS .....	256
9.5.1.	ping .....	256
9.5.2.	Program traceroute .....	257
9.5.3.	DNS i host .....	258
9.6.	Warstwa fizyczna i Ethernet .....	258
9.7.	Interfejsy sieciowe jądra .....	259
9.8.	Wprowadzenie do konfiguracji interfejsów sieciowych .....	260
9.8.1.	Ręczne dodawanie i usuwanie tras .....	261
9.9.	Konfiguracja sieci aktywowana podczas rozruchu .....	261
9.10.	Problemy z konfiguracją sieci ręczną i aktywowaną podczas rozruchu .....	262
9.11.	Menedżery konfiguracji sieciowych .....	263
9.11.1.	Działanie narzędzia NetworkManager .....	263
9.11.2.	Interakcja z narzędziem NetworkManager .....	264
9.11.3.	Konfiguracja narzędzia NetworkManager .....	265
9.12.	Rozpoznawanie nazw hostów .....	267
9.12.1.	Plik /etc/hosts .....	268
9.12.2.	Plik resolv.conf .....	268
9.12.3.	Buforowanie i system DNS bez konfiguracji .....	268
9.12.4.	Plik /etc/nsswitch.conf .....	269
9.13.	Host lokalny .....	270
9.14.	Warstwa transportowa: protokoły TCP i UDP oraz usługi .....	271
9.14.1.	Porty TCP i połączenia .....	271
9.14.2.	Ustanawianie połączeń TCP .....	272
9.14.3.	Numery portów i plik /etc/services .....	273
9.14.4.	Właściwości protokołu TCP .....	273
9.14.5.	Protokół UDP .....	274
9.15.	Ponowna analiza prostej sieci lokalnej .....	276

9.16.	Protokół DHCP .....	276
9.16.1.	Klient DHCP w systemie Linux .....	277
9.16.2.	Serwery DHCP w systemie Linux .....	277
9.17.	Konfigurowanie systemu Linux jako routera .....	277
9.17.1.	Łącza internetowe .....	279
9.18.	Sieci prywatne .....	279
9.19.	Translacja adresów sieciowych (maskarada IP) .....	280
9.20.	Routery i system Linux .....	282
9.21.	Zapory sieciowe .....	283
9.21.1.	Podstawy dotyczące linuksowych zapór sieciowych .....	283
9.21.2.	Konfigurowanie reguł zapory sieciowej .....	285
9.21.3.	Strategie tworzenia zapór sieciowych .....	287
9.22.	Ethernet, IP i ARP .....	289
9.23.	Ethernet bezprzewodowy .....	291
9.23.1.	iw .....	292
9.23.2.	Zabezpieczenia sieci bezprzewodowych .....	293
9.24.	Podsumowanie .....	293

## 10

### **USŁUGI I APLIKACJE SIECIOWE ..... 295**

10.1.	Podstawy usług .....	296
10.1.1.	Dokładniejsza analiza .....	296
10.2.	Serwery sieciowe .....	298
10.3.	Secure Shell (SSH) .....	299
10.3.1.	Serwer SSHD .....	300
10.3.2.	Klient SSH .....	303
10.4.	Demony inetd i xinetd .....	305
10.4.1.	Wrapper TCP: tcpd, /etc/hosts.allow, /etc/hosts.deny .....	306
10.5.	Narzędzia diagnostyczne .....	306
10.5.1.	Isof .....	306
10.5.2.	tcpdump .....	308
10.5.3.	netcat .....	310
10.5.4.	Skanowanie portów .....	310
10.6.	Zdalne wywoływanie procedur (RPC) .....	311
10.7.	Zabezpieczenie sieci .....	312
10.7.1.	Typowe słabości .....	314
10.7.2.	Źródła danych o zabezpieczeniach .....	314
10.8.	Spojrzenie w przyszłość .....	315
10.9.	Gniazda: sposób komunikacji procesów z siecią .....	315
10.10.	Gniazda domenowe systemu Unix .....	317
10.10.1.	Korzyści dla projektantów .....	317
10.10.2.	Wyszczególnianie gniazd domenowych systemu Unix .....	318

## II

<b>WPROWADZENIE DO SKRYPTÓW POWŁOKI .....</b>	<b>319</b>
11.1. Podstawy skryptów powłoki .....	319
11.1.1. Ograniczenia skryptów powłoki .....	320
11.2. Cudzysłowy i literały .....	321
11.2.1. Literały .....	321
11.2.2. Pojedyncze cudzysłowy .....	322
11.2.3. Podwójne cudzysłowy .....	323
11.2.4. Przekazywanie literału w postaci znaku pojedynczego cudzysłowu .....	323
11.3. Zmienne specjalne .....	324
11.3.1. Pojedyncze argumenty: \$1, \$2... .....	324
11.3.2. Liczba argumentów: \$# .....	325
11.3.3. Wszystkie argumenty: \$@ .....	325
11.3.4. Nazwa skryptu: \$0 .....	326
11.3.5. Identyfikator procesu: \$\$ .....	326
11.3.6. Kod wyjścia: \$? .....	327
11.4. Kody wyjścia .....	327
11.5. Wyrażenia warunkowe .....	328
11.5.1. Obsługa list pustych parametrów .....	329
11.5.2. Użycie innych poleceń do testów .....	329
11.5.3. Słowo kluczowe elif .....	329
11.5.4. Konstrukcje logiczne && i    .....	330
11.5.5. Sprawdzanie warunków .....	330
11.5.6. Porównywanie ciągów znaków instrukcją case .....	333
11.6. Pętle .....	334
11.6.1. Pętle for .....	334
11.6.2. Pętle while .....	335
11.7. Podmiana poleceń .....	336
11.8. Zarządzanie plikami tymczasowymi .....	337
11.9. Dokumenty miejscowe .....	338
11.10. Ważne narzędzia skryptów powłoki .....	338
11.10.1. Polecenie basename .....	339
11.10.2. Polecenie awk .....	339
11.10.3. Polecenie sed .....	340
11.10.4. Polecenie xargs .....	341
11.10.5. Polecenie expr .....	342
11.10.6. Polecenie exec .....	342
11.11. Podpowłoki .....	342
11.12. Włączanie do skryptów innych plików .....	343
11.13. Pobieranie danych od użytkowników .....	344
11.14. Kiedy (nie)używać skryptów powłoki? .....	344

## 12

### **PRZENOSZENIE PLIKÓW W SIECI ..... 345**

12.1.	Szybkie wykonywanie kopii .....	345
12.2.	rsync .....	346
12.2.1.	Podstawy dotyczące narzędzia rsync .....	346
12.2.2.	Tworzenie dokładnych kopii struktury katalogów .....	348
12.2.3.	Jak używać końcowego ukośnika? .....	348
12.2.4.	Pomijanie plików i katalogów .....	350
12.2.5.	Integralność transferu, sumy kontrolne i tryby informacyjne .....	351
12.2.6.	Kompresja .....	352
12.2.7.	Ograniczanie przepustowości .....	352
12.2.8.	Przesyłanie plików do naszego komputera .....	352
12.2.9.	Więcej informacji o programie rsync .....	353
12.3.	Wprowadzenie do współużytkowania plików .....	353
12.4.	Współużytkowanie plików za pomocą pakietu Samba .....	354
12.4.1.	Konfigurowanie serwera .....	354
12.4.2.	Kontrola dostępu do serwera .....	355
12.4.3.	Hasła .....	356
12.4.4.	Uruchamianie serwera .....	358
12.4.5.	Diagnostyka i pliki dziennika .....	358
12.4.6.	Konfigurowanie udziału plikowego .....	358
12.4.7.	Katalogi domowe .....	359
12.4.8.	Współużytkowanie drukarek .....	359
12.4.9.	Korzystanie z klientów Samby .....	360
12.4.10.	Dostęp do plików jako klient .....	361
12.5.	Klienci NFS .....	362
12.6.	Dodatkowe ograniczenia i opcje sieciowych usług plikowych .....	363

## 13

### **ŚRODOWISKA UŻYTKOWNIKÓW ..... 365**

13.1.	Wytyczne dotyczące tworzenia plików uruchomieniowych .....	366
13.2.	Kiedy należy modyfikować pliki uruchomieniowe? .....	366
13.3.	Elementy plików uruchamiających powłokę .....	367
13.3.1.	Ścieżka wyszukiwania poleceń .....	367
13.3.2.	Ścieżka stron podręcznika man .....	368
13.3.3.	Symbol zachęty .....	369
13.3.4.	Alias .....	370
13.3.5.	Maska uprawnień .....	370
13.4.	Kolejność plików uruchomieniowych i przykłady .....	371
13.4.1.	Powłoka bash .....	371
13.4.2.	Powłoka tcsh .....	374
13.5.	Domyślne ustawienia użytkownika .....	375
13.5.1.	Domyślne ustawienia powłoki .....	375
13.5.2.	Edytor .....	376
13.5.3.	Program stronicujący .....	376

13.6.	Pułapki w plikach uruchomieniowych .....	376
13.7.	Dalsze informacje .....	377

## 14

### OGÓLNY PRZEGLĄD INTERFEJSÓW UŻYTKOWNIKA

<b>SYSTEMU LINUX .....</b>	<b>379</b>	
14.1.	Komponenty interfejsów użytkownika .....	380
14.1.1.	Menedżery okien .....	380
14.1.2.	Pakiety narzędziowe .....	381
14.1.3.	Środowiska interfejsów użytkownika .....	381
14.1.4.	Aplikacje .....	382
14.2.	System X Window System .....	382
14.2.1.	Menedżery wyświetlaczy .....	383
14.2.2.	Przezroczystość sieci .....	383
14.3.	Eksplorowanie klientów serwera X .....	384
14.3.1.	Zdarzenia serwera X .....	384
14.3.2.	Ustawianie preferencji i dane wejściowe serwera X .....	385
14.4.	Przyszłość serwera X .....	388
14.5.	Usługa D-Bus .....	389
14.5.1.	Instancja sesji i instancja systemowa .....	389
14.5.2.	Monitorowanie komunikatów usługi D-Bus .....	390
14.6.	Drukowanie .....	390
14.6.1.	CUPS .....	391
14.6.2.	Konwersja formatów i filtry wydruku .....	392
14.7.	Inne zagadnienia związane z interfejsami użytkownika .....	392

## 15

<b>NARZĘDZIA PROGRAMISTYCZNE .....</b>	<b>393</b>	
15.1.	Kompilator języka C .....	394
15.1.1.	Wiele plików źródłowych .....	395
15.1.2.	Pliki i katalogi nagłówkowe .....	396
15.1.3.	Konsolidacja z bibliotekami .....	398
15.1.4.	Biblioteki współużytkowane .....	399
15.2.	Narzędzie make .....	403
15.2.1.	Przykładowy plik Makefile .....	404
15.2.2.	Wbudowane reguły .....	405
15.2.3.	Końcowe budowanie programu .....	406
15.2.4.	Aktualizowanie .....	406
15.2.5.	Argumenty i opcje wiersza poleceń .....	407
15.2.6.	Standardowe makra i zmienne .....	408
15.2.7.	Typowe cele kompilacji .....	408
15.2.8.	Organizowanie pliku Makefile .....	409
15.3.	Debuggery .....	411
15.4.	Lex i Yacc .....	412

15.5.	Języki skryptowe .....	412
15.5.1.	Python .....	413
15.5.2.	Perl .....	414
15.5.3.	Pozostałe języki skryptowe .....	414
15.6.	Java .....	415
15.7.	Spojrzenie w przyszłość: kompilowanie pakietów .....	416

## 16

### WPROWADZENIE DO KOMPILOWANIA OPROGRAMOWANIA

<b>Z KODU ŹRÓDŁOWEGO C .....</b>	<b>417</b>	
16.1.	Systemy do tworzenia oprogramowania .....	418
16.2.	Rozpakowywanie pakietów kodu źródłowego języka C .....	419
16.2.1.	Od czego zacząć? .....	420
16.3.	GNU autoconf .....	420
16.3.1.	Przykład użycia systemu GNU autoconf .....	421
16.3.2.	Instalacja za pomocą narzędzia do tworzenia pakietów .....	422
16.3.3.	Opcje skryptu configure .....	423
16.3.4.	Zmienne środowiskowe .....	424
16.3.5.	Cele tworzone przez system autoconf .....	425
16.3.6.	Pliki dziennika systemu autoconf .....	426
16.3.7.	pkg-config .....	426
16.4.	Praktyki instalacyjne .....	428
16.4.1.	Gdzie instalować? .....	428
16.5.	Stosowanie poprawek .....	429
16.6.	Rozwiązywanie problemów z kompilowaniem i instalowaniem .....	430
16.6.1.	Częste błędy .....	431
16.7.	Spojrzenie w przyszłość .....	433

## 17

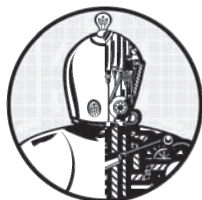
<b>BUDOWANIE NA FUNDAMENTACH .....</b>	<b>435</b>	
17.1.	Serwery WWW i aplikacje internetowe .....	435
17.2.	Bazy danych .....	436
17.2.1.	Typy baz danych .....	437
17.3.	Wirtualizacja .....	437
17.4.	Obliczenia rozproszone i na żądanie .....	438
17.5.	Systemy wbudowane .....	438
17.6.	Końcowe uwagi .....	440

<b>BIBLIOGRAFIA .....</b>	<b>441</b>
---------------------------	------------

<b>SKOROWIDZ .....</b>	<b>445</b>
------------------------	------------

# 5

## Jak uruchamia się Linux?



ZNASZ JUŻ FIZYCZNĄ I LOGICZNĄ STRUKTURĘ SYSTEMU LINUKSOWEGO, WIESZ, CZYM JEST JĄDRO SYSTEMU I JAK WSPÓŁPRACUJE Z PROCESAMI. W TYM ROZDZIALE ZAPREZENTUJĘ PROCEDURĘ uruchamiania (lub rozruchu) jądra. Innymi słowy, dowiesz się teraz, w jaki sposób jądro przenosi się do pamięci komputera do momentu, w którym uruchomiony zostaje pierwszy proces użytkownika.

Uproszczona procedura uruchamiania systemu wygląda następująco.

1. BIOS komputera lub jego firmware ładuje i uruchamia program rozruchowy (ang. *boot loader*).
2. Program rozruchowy odszukuje na dysku obraz jądra, ładuje go do pamięci i uruchamia.
3. Jądro inicjuje wszystkie urządzenia wraz z ich sterownikami.
4. Jądro montuje podstawowy system plików.
5. Jądro uruchamia program o nazwie `init` związany z procesem o identyfikatorze 1. To w tym momencie *uruchamiana jest przestrzeń użytkownika*.
6. Program `init` wprawia w ruch pozostałe elementy systemu.
7. W pewnym momencie program `init` uruchamia proces pozwalający na zalogowanie się użytkownika. Zazwyczaj dzieje się to pod sam koniec procesu uruchamiania systemu.

W tym rozdziale zajmę się czterema pierwszymi krokami; skoncentruję się na jądrze i programie rozruchowym. W rozdziale 6. podejmę ten temat od momentu uruchomienia przestrzeni użytkownika.

Umiejętność zidentyfikowania poszczególnych etapów uruchamiania systemu okazuje się niezastąpiona podczas rozwiązywania problemów ze startem systemu, ale też ułatwia zrozumienie zasad funkcjonowania systemu jako całości. Niestety domyślne zachowanie w wielu dystrybucjach Linuksa często bardzo utrudnia, a czasem i uniemożliwia zidentyfikowanie pierwszych etapów rozruchu. W efekcie możesz przejrzeć ten proces dopiero po zakończeniu uruchamiania i zalogowaniu się.

## 5.1. Komunikaty rozruchowe

Tradycyjne systemy uniksowe podczas uruchamiania generują wiele komunikatów diagnostycznych, które dokładnie opisują cały ten proces. Początkowo komunikaty te pochodzą wyłącznie z jądra, ale później pojawiają się też takie, które pochodzą z procesów i procedur inicjujących uruchamianych przez program `init`. Komunikaty te nie są jednak ani ładne, ani spójne, a w niektórych przypadkach nie noszą nawet czytelnych informacji. Większość aktualnych dystrybucji Linuksa stara się jak najlepiej ukrywać je za ekranami startowymi i innymi obrazkami. Dodatkowo rozwijany ciągle sprzęt sprawia, że jądro uruchamia się dzisiaj znacznie szybciej niż dawniej, przez co wszystkie te komunikaty przewijane są tak szybko przez ekran, że nie da się ich przeczytać.

Istnieją dwie metody przeglądania komunikatów diagnostycznych rozruchu jądra i działających później programów. Możemy:

- przejrzeć plik protokołu systemowego jądra; często znajduje się on w pliku `/var/log/kern.log`, ale w zależności od konfiguracji systemu może też znaleźć się wśród innych plików protokołów systemowych, w katalogu `/var/log/messages` lub innym,
- użyć polecenia `dmesg`, przy czym jego wyjście koniecznie trzeba przepuścić przez program `less`, ponieważ komunikatów na pewno będzie więcej, niż może pomieścić jeden ekran. Polecenie to wykorzystuje bufor cykliczny jądra, który ma ograniczoną wielkość, ale w większości nowoczesnych jąder jest na tyle duży, że przez długi czas może przechowywać komunikaty rozruchowe.

Oto przykład tego, czego można się spodziewać po uruchomieniu polecenia `dmesg`:

---

```
$ dmesg
[ 0.000000] Initializing cgroup subsys cpu
[ 0.000000] Linux version 3.2.0-67-generic-pae (buildd@toyol) (gcc version 4.6.3 (Ubuntu/Linaro
↳4.6.3-1ubuntu5) ) #101-Ubuntu SMP Tue Jul 15 18:04:54 UTC 2014(Ubuntu 3.2.0-67.101-generic-pae
↳3.2.60)
```



```
[ 0.000000] KERNEL supported cpus:
--ciach--
[ 2.986148] sr0: scsi3-mmc drive: 24x/8x writer dvd-ram cd/rw xa/form2 cdda tray
[ 2.986153] cdrom: Uniform CD-ROM driver Revision: 3.20
[ 2.986316] sr 1:0:0:0: Attached scsi CD-ROM sr0
[ 2.986416] sr 1:0:0:0: Attached scsi generic sgl type 5
[ 3.007862] sda: sda1 sda2 < sda5 >
[ 3.008658] sd 0:0:0:0: [sda] Attached SCSI disk
--ciach--
```

---

Po zakończeniu uruchamiania jądra komunikaty generowane są przez procedurę uruchamiania przestrzeni użytkownika. Te komunikaty będą znacznie trudniejsze do przejrzania i oceny, ponieważ w większości systemów nie są zapisywane do jednego pliku protokołów. Skrypty startowe zazwyczaj wypisują na konsolę swoje komunikaty, a te są zwykle kasowane po zakończeniu procedury uruchamiania. Zazwyczaj nie stanowi to problemu, ponieważ każdy skrypt prowadzi też swój własny protokół. Niektóre wersje programu `init`, takie jak `Upstart` i `systemd`, zarówno w czasie rozruchu, jak i w trakcie normalnej pracy systemu mogą przechwytywać komunikaty, które trafiają jedynie na konsolę.

## 5.2. Inicjowanie jądra i opcje rozruchu

Podczas rozruchu jądro Linuksa inicjuje się w następującej kolejności.

1. Sprawdzenie procesora.
2. Sprawdzenie pamięci.
3. Rozpoznawanie magistrali urządzeń.
4. Rozpoznawanie urządzeń.
5. Konfigurowanie uzupełniających podsystemów jądra (sieć i tym podobne).
6. Montowanie podstawowego systemu plików.
7. Uruchamianie przestrzeni użytkownika.

Pierwsze kroki nie są szczególnie spektakularne, ale gdy jądro dotrze do rozpoznawania urządzeń, pojawia się pytanie o zależności między nimi. Przykładowo sterowniki urządzenia dyskowego mogą zależeć od funkcji obsługi magistrali oraz obsługi podsystemu SCSI.

Na dalszym etapie procesu inicjacji jądro musi zamontować podstawowy system plików, żeby później uruchomić program `init`. Zwykle nie musimy się przejmować tymi sprawami. Wyjątek może stanowić sytuacja, kiedy niektóre niezbędne komponenty nie są częścią samego jądra, ale dostępne są jako zewnętrzne moduły. Na niektórych komputerach konieczne może być załadowanie tych modułów jeszcze przed zamontowaniem samego systemu plików. Tym problemem oraz jego rozwiązaniem w postaci początkowego systemu plików w pamięci RAM zajmę się w podrozdziale 6.8.

W czasie pisania tej książki jądro nie generowało żadnych komunikatów w związku z przygotowaniem do uruchomienia pierwszego procesu przestrzeni użytkownika. Jednak poniższe komunikaty związane z zarządzaniem pamięcią całkiem dobrze wskazują na to, że zaraz nastąpi przekazanie kontroli do przestrzeni użytkownika, ponieważ w ten sposób jądro zaczyna chronić własną pamięć przed działającymi w niej procesami.

---

```
Freeing unused kernel memory: 740k freed
Write protecting the kernel text: 5820k
Write protecting the kernel read-only data: 2376k
NX-protecting the kernel data: 4420k
```

---

W tym miejscu możesz też zobaczyć też komunikat informujący o montowaniu podstawowego systemu plików.

**UWAGA** *Jeżeli interesujesz się dalszym ciągiem procesu uruchamiania przestrzeni użytkownika realizowanej przez program `init` uruchomiony przez jądro, od razu możesz przejść do rozdziału 6. W pozostałej części tego rozdziału będę opisywał dokładnie kolejne etapy uruchamiania jądra.*

## 5.3. Parametry jądra

Podczas uruchamiania jądra Linuksa program rozruchowy przekazuje do niego zestaw tekstowych *parametrów jądra*, które decydują o tym, w jaki sposób ma ono zostać uruchomione. Parametry te definiują wiele różnych typów zachowań, takich jak ilość komunikatów diagnostycznych wypisywanych przez jądro, albo podają opcje właściwe dla różnych sterowników.

Parametry jądra użyte przy uruchamianiu swojego systemu można przejrzeć w pliku `/proc/cmdline`:

---

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-3.2.0-67-generic-pae root=UUID=70ccd6e7-6ae6-44f6-812c-
↳ 51aab8036d29 ro quiet splash vt.handoff=7
```

---

Parametrami mogą być znaczniki składające się z pojedynczego słowa, takie jak `ro` lub `quiet`, ale mogą być nimi również pary *klucz=wartość*, takie jak `vt.handoff=7`. Wiele parametrów nie ma większego znaczenia, na przykład parametr `splash` pozwalający wyświetlić ekran startowy, ale jeden z nich jest absolutnie niezbędny — `root`. Definiuje on lokalizację podstawowego systemu plików. Bez niego jądro nie będzie mogło odnaleźć programu `init` i nie uruchomi przestrzeni użytkownika.

Podstawowy system plików możesz zdefiniować za pomocą pliku urządzenia, tak jak w poniższym przykładzie:

---

```
root=/dev/sda1
```

---

Jednak w większości nowoczesnych systemów najczęściej używany jest w tym celu identyfikator UUID systemu plików (omawiany w punkcie 4.2.4):

---

```
root=UUID=70ccd6e7-6ae6-44f6-812c-51aab8036d29
```

---

Użycie tu parametru `ro` jest całkiem normalne. Nakazuje on, by jądro zamontowało podstawowy system plików w trybie tylko do odczytu w momencie uruchamiania przestrzeni użytkownika. Tryb tylko do odczytu sprawia, że program `fsck` może bezpiecznie skontrolować system plików. Po zakończeniu takiej kontroli proces uruchamiania systemu może ponownie zamontować podstawowy system plików w trybie pełnego dostępu.

Jeżeli jądro Linuksa natknie się na parametr, którego nie rozumie, mimo wszystko go zapamiętuje, a następnie przekazuje jako parametr programu `init` podczas uruchamiania przestrzeni użytkownika. Jeśli na przykład wśród parametrów jądra znajdzie się wartość `-s`, jądro przekaze ją do programu `init`, w wyniku czego ten uruchomi się w trybie pojedynczego użytkownika.

Przyjrzyjmy się teraz mechanice uruchamiania jądra przez program rozruchowy.

## 5.4. Programy rozruchowe

Na początku procesu rozruchu systemu program rozruchowy (ang. *boot loader*) musi uruchomić jądro. Zadanie programu rozruchowego wydaje się całkiem proste: załadować do pamięci jądro systemu, a następnie uruchomić je, przekazując odpowiednie parametry. Zastanówmy się jednak, z jakimi problemami musi zmierzyć się ten program.

- Gdzie znajduje się jądro?
- Jakie parametry należy przekazać do jądra w momencie jego uruchamiania?

Odpowiedź na te pytania (zazwyczaj) brzmi tak, że jądro oraz niezbędne parametry znajdują się gdzieś w podstawowym systemie plików. Wydaje się, że odszukanie parametrów jądra nie powinno być kłopotliwe, z tym że samo jądro jeszcze nie działa, dlatego nie można przeszukać systemu plików, żeby odnaleźć niezbędne pliki. Co gorsza, niedostępne są też sterowniki urządzeń, z których normalnie korzysta jądro. Wydaje się, że mamy tu do czynienia z problemem typu „jajko czy kura”.

Rozważania zacznę od sterowników. W komputerach PC programy rozruchowe uzyskują dostęp do dysków za pośrednictwem interfejsów BIOS (ang. *Basic Input/Output System*) lub UEFI (ang. *Unified Extensible Firmware Interface*). Niemal wszystkie dyski wyposażone są w oprogramowanie pozwalające na dostęp do danych za pomocą adresowania LBA (ang. *Linear Block Addressing*). Mimo że to rozwiązanie ma bardzo niską wydajność, ten tryb adresowania pozwala na uniwersalny dostęp do zawartości dysku. Programy rozruchowe są chyba jedynymi programami, które odwołują się do dysków za pośrednictwem BIOS-u. Samo jądro korzysta już z własnych sterowników o znacznie lepszej wydajności.

Z systemem plików sprawa nie jest już tak prosta. Większość nowoczesnych programów rozruchowych może odczytać tablicę partycji i ma wbudowaną obsługę systemów plików w trybie tylko do odczytu, a to oznacza, że mogą one odczytywać pliki z dysku. Takie możliwości sprawiają, że znacznie łatwiejsze staje się dynamiczne konfigurowanie i rozbudowywanie programu rozruchowego. Starsze linuksowe programy rozruchowe nie miały takich możliwości, przez co ich konfigurowanie sprawiało znacznie większe problemy.

### 5.4.1. Zadania programu rozruchowego

Oto zadania linuksowego programu rozruchowego.

- Wybieranie spośród kilku różnych jąder.
- Przełączanie pomiędzy różnymi zestawami parametrów jądra.
- Umożliwienie użytkownikowi ręcznej edycji nazw i parametrów jądra systemu (na przykład w celu przejścia do trybu pojedynczego użytkownika).
- Umożliwienie uruchamiania innych systemów operacyjnych.

Od czasu gdy powstało jądro Linuksa, programy rozruchowe stawały się coraz bardziej rozbudowane i przejmowały takie funkcje jak historia i system menu. Jednak największą przewagą ich nowoczesnych wersji jest możliwość elastycznego wyboru jądra i jego parametrów. Co ciekawe, przy okazji okazało się, że pewne wymagania zostały mocno zredukowane. Przykładowo dzięki temu, że możemy teraz wykonać awaryjne uruchomienie systemu częściowo lub całkowicie z napędu podłączonego do portu USB, raczej nie musimy zajmować się ręcznym wprowadzaniem parametrów jądra albo przechodzeniem do trybu pojedynczego użytkownika. Nowoczesne programy rozruchowe dają więcej różnych możliwości niż było to kiedykolwiek możliwe, co bardzo przydaje się osobom tworzącym własne jądra systemu albo próbującym zmieniać ich parametry.

### 5.4.2. Przegląd programów rozruchowych

Oto najważniejsze programy rozruchowe, z którymi możesz się zetknąć, według ich popularności.

**GRUB** — praktycznie standardowy program w systemach linuksowych.

**LILO** — jeden z pierwszych linuksowych programów rozruchowych.

ELILO to wersja współpracująca z UEFI.

**SYSLINUX** — możesz go skonfigurować tak, żeby działał z wieloma różnymi systemami plików.

**LOADLIN** — uruchamia jądro w systemie MS-DOS.

**efilinux** — program rozruchowy działający z UEFI, który ma być wzorem i modelem dla innych programów rozruchowych.

**coreboot** (dawniej **LinuxBIOS**) — zamiennik BIOS-u komputera o wysokiej wydajności, który może zawierać jądro systemu.

**Linux Kernel EFISTUB** — moduł jądra pozwalający na załadowanie jądra bezpośrednio z partycji systemowej EFI/UEFI. Pojawia się w najnowszych systemach.

W tej książce będziemy się zajmować wyłącznie programem GRUB. Wynika to z faktu, że inne programy rozruchowe są albo łatwiejsze do skonfigurowania niż GRUB, albo są od niego szybsze.

Aby wprowadzić nazwę jądra lub jego parametry, musisz najpierw włączyć wiersz poleceń programu. Niestety czasami nie do końca wiadomo, jak to zrobić, ponieważ dystrybucje Linuksa starają się dopasować do siebie zachowanie i wygląd programu rozruchowego.

W kolejnym podrozdziale opiszę, jak przejść do wiersza poleceń programu rozruchowego w celu wprowadzenia nazwy jądra i jego parametrów. Gdy już będzie wiadomo, jak to zrobić, możesz spróbować samodzielnie skonfigurować i zainstalować swój program rozruchowy.

## 5.5. Wprowadzenie do programu GRUB

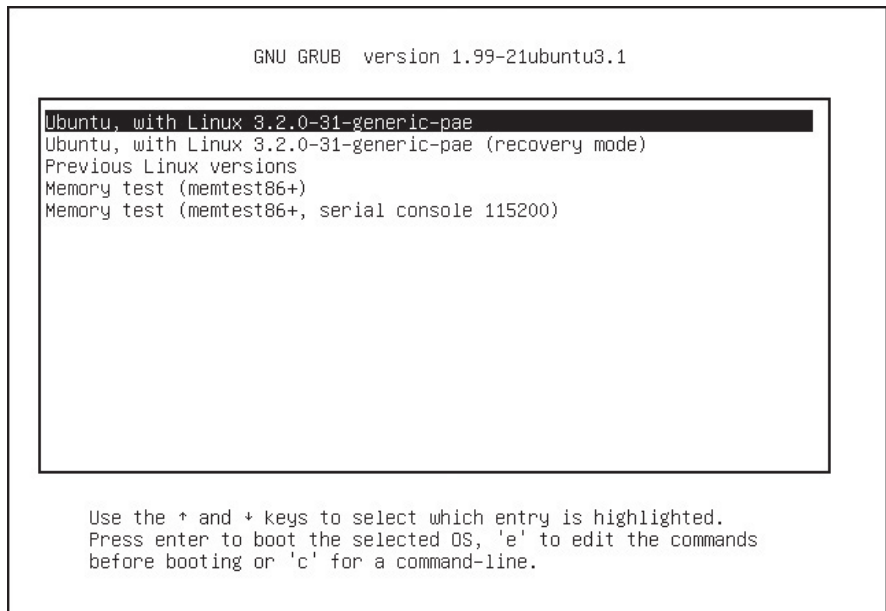
Nazwa *GRUB* jest skrótem od angielskiego *Grand Unified Boot Loader*, czyli „wielki zunifikowany program rozruchowy”. Omówię tutaj wersję GRUB 2. Czasami używana jest jeszcze starsza wersja nazywana teraz GRUB Legacy, która jednak pojawia się bardzo rzadko.

Jedną z najważniejszych funkcji tego programu jest możliwość nawigowania w systemie plików, która umożliwi łatwiejsze wybranie obrazu jądra i jego konfigurację. Przejrzenie menu programu jest dobrą metodą na poznanie rządzących nim zasad. W udostępnianym interfejsie łatwo się poruszać nawet wtedy, kiedy widzisz go po raz pierwszy, co jest całkiem prawdopodobne. W końcu twórcy dystrybucji Linuksa starają się jak mogą, żeby ukryć program rozruchowy przed użytkownikami.

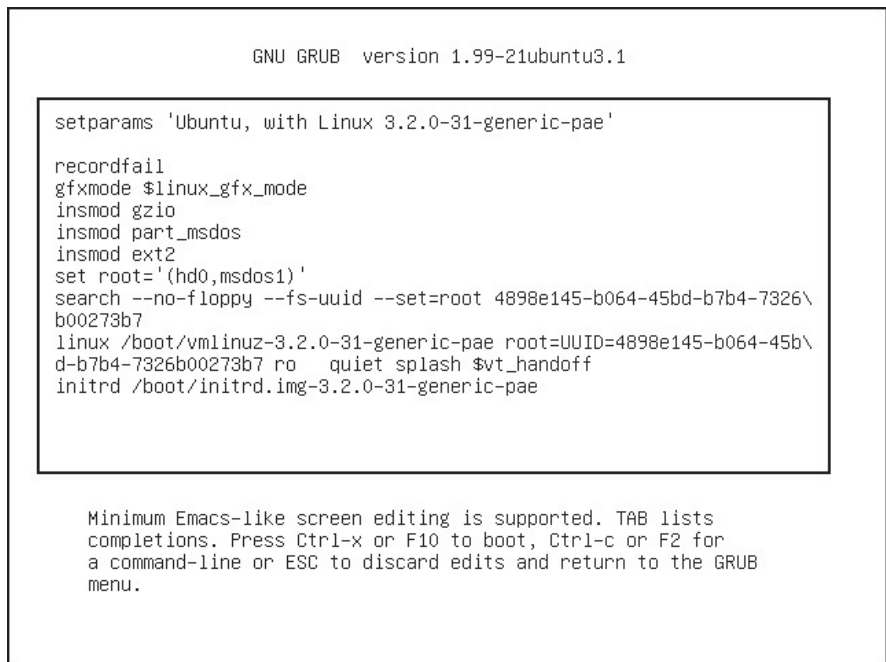
Aby dostać się do menu programu GRUB, należy przytrzymać klawisz *Shift* w czasie, gdy na ekranie widoczne są informacje wyświetlane przez BIOS lub firmware komputera. Jeżeli tego nie zrobisz, program rozruchowy nie zatrzyma się przed załadowaniem jądra do pamięci. Na rysunku 5.1 przedstawiam menu programu GRUB. Po pojawieniu się tego menu można nacisnąć klawisz *Esc*, żeby tymczasowo wyłączyć opóźnienie automatycznego rozruchu.

W ramach poznawania programu rozruchowego proponuję wykonać poniższe kroki.

1. Uruchom (ponownie) swój system linuksowy.
2. Podczas autotestu BIOS-u lub firmware'u komputera przytrzymaj klawisz *Shift*, żeby wywołać menu GRUB-a.
3. Naciśnij klawisz *E*, żeby przejrzeć polecenia konfiguracyjne programu rozruchowego domyślnej opcji rozruchu. Na ekranie powinno pojawić się coś podobnego do przedstawionego na rysunku 5.2.



*Rysunek 5.1. Menu programu GRUB*



*Rysunek 5.2. Edytor konfiguracji programu GRUB*

Z tego ekranu dowiesz się, że w tej konfiguracji podstawowy system plików wybierany jest za pomocą identyfikatora UUID, obraz jądra umieszczony jest w pliku `/boot/vmlinuz-3.2.0.31-generic-pae`, a wśród parametrów jądra znajdują się opcje `ro`, `quiet`, i `splash`. Jeżeli nie zdarzyło Ci się wcześniej przeglądać tego rodzaju konfiguracji, możesz czuć się trochę nieswojo. Dlaczego odwołanie do podstawowego systemu plików pojawia się tu wielokrotnie i na dodatek z różnymi wartościami? Co oznacza opcja `insmod`? Czy nie jest to funkcja jądra normalnie uruchamiana przy użyciu demona `udev`?

Takie podwójne zapisy są całkowicie uzasadnione, ponieważ GRUB *nie korzysta* z jądra, a jedynie je *uruchamia*. Widoczna tu konfiguracja składa się wyłącznie z poleceń wewnętrznych samego programu GRUB, a ten jest całkowicie niezależnym światem.

Powstałe zamieszanie wynika z faktu, że GRUB zapożycza terminologię z wielu różnych źródeł. Ma swoje własne „jądro” oraz własne polecenie `insmod` pozwalające na dynamiczne ładowanie modułów programu, które są całkowicie niezależne od jądra Linuksa. Wiele poleceń GRUB-a bardzo przypomina polecenia uniksowej powłoki. Dostępne jest tu nawet polecenie `ls` do wypisywania plików.

Największe zamieszanie powoduje jednak sposób użycia słowa `root`. W ramach wyjaśnienia zaznaczę, że szukając podstawowego systemu plików swojego komputera, należy stosować się do prostej zasady: interesujący nas system plików podawany jest *wyłącznie* w ramach parametrów uruchamianego *jądra*.

W konfiguracji GRUB-a parametry jądra zapisywane są zaraz za nazwą obrazu w poleceniu `linux`. Każde inne wystąpienie słowa `root` dotyczyć będzie jedynie partycji podstawowej GRUB-a. Dla tego programu słowo „`root`” oznacza system plików, GRUB poszukuje jądra oraz plików obrazów systemów plików umieszczonych w pamięci RAM.

Na rysunku 5.2 partycja podstawowa GRUB-a najpierw ustawiana jest na specjalnym urządzeniu (`hd0,msdos1`). W następnym poleceniu GRUB poszukuje partycji z określonym identyfikatorem UUID. Jeżeli ją znajdzie, właśnie ona stanie się katalogiem podstawowym.

Podsumowując, można stwierdzić, że pierwszy parametr polecenia `linux (/boot/vmlinuz-...)` podaje lokalizację pliku z obrazem jądra Linuksa. GRUB załaduje ten plik ze swojej partycji podstawowej. Podobnie działa polecenie `initrd`, z tym że najpierw określa lokalizację wstępnego systemu plików dla pamięci RAM.

Oczywiście możemy dowolnie edytować taką konfigurację w samym programie GRUB. Jest to najprostsza metoda tymczasowego naprawiania uszkodzonego procesu rozruchu systemu. Aby taki problem usunąć trwale, konieczna jest zmiana zapisów konfiguracyjnych (więcej o tym w punkcie 5.5.2), ale na razie przyjrzymy się wewnętrznym elementom GRUB-a; użyjemy przy tym interfejsu wiersza poleceń.

## 5.5.1. Przeszukiwanie urządzeń i partycji za pomocą wiersza poleceń programu GRUB

Jak można było zobaczyć na rysunku 5.2, GRUB ma własny sposób adresowania urządzeń. Przykładowo pierwszy znaleziony dysk twardy otrzymuje nazwę `hd0`, następny to `hd1` i tak dalej. Niestety przypisanie nazw do urządzeń może podlegać zmianom. Na szczęście GRUB może przeszukać partycje, sprawdzając ich identyfikator UUID, w celu odnalezienia tej, na której znajduje się plik jądra systemu; do tego służy polecenie `search`.

### Wypisywanie urządzeń

Aby poznać sposób odwoływania się do urządzeń w systemie przez GRUB, możesz otworzyć jego wiersz poleceń, naciskając klawisz `C` w menu rozruchowym albo w edytorze konfiguracji. Powinien się wtedy pojawić znak zachęty:

---

```
grub>
```

---

Możesz tu wprowadzić dowolne z poleceń, które widziałeś w konfiguracji. Na początek jednak proponuję użyć polecenia diagnostycznego `ls`. Bez żadnych parametrów wypisuje ono listę urządzeń znanych GRUB-owi:

---

```
grub> ls
(hd0) (hd0,msdos1) (hd0,msdos5)
```

---

W tym przypadku na liście znajduje się jedno urządzenie dyskowe oznaczone jako `(hd0)` oraz dwie partycje — `(hd0,msdos1)` i `(hd0,msdos5)`. Przedrostek `msdos` w nazwie partycji oznacza, że na dysku znajduje się tablica partycji typu MBR. Jeżeli byłaby to tablica partycji typu GPT, nazwy partycji miałyby przedrostek `gpt`. Możliwe są jeszcze kolejne kombinacje z trzecim identyfikatorem, kiedy wewnątrz partycji znajduje się mapa etykiet dysku BSD. Zazwyczaj nie trzeba się tym przejmować, o ile nie pracujesz z kilkoma różnymi systemami operacyjnymi na jednym komputerze.

Bardziej szczegółowe informacje uzyskasz po wprowadzeniu polecenia `ls -l`. Polecenie to może być szczególnie użyteczne, ponieważ podaje też identyfikatory UUID wszystkich partycji na dysku. Oto przykład.

---

```
grub> ls -l
Device hd0: Not a known filesystem - Total size 426743808 sectors
  Partition hd0,msdos1: Filesystem type ext2 - Last modification time
    2015-09-18 20:45:00 Friday, UUID 4898e145-b064-45bd-b7b4-7326b00273b7 -
Partition start at 2048 - Total size 424644608 sectors
  Partition hd0,msdos5: Not a known filesystem - Partition start at
    424648704 - Total size 2093056 sectors
```

---



Ten konkretny dysk na pierwszej partycji MBR zawiera linuksowy system plików ext2/3/4, natomiast 5. partycja nosi sygnaturę linuksowej przestrzeni wymiany. Oznacza to dość typową konfigurację dysku. Niestety, patrząc na podany wyżej wydruk, nie da się stwierdzić, że partycja (hd0,msdos5) została przeznaczona na przestrzeń wymiany.

## Nawigacja wśród plików

Przyjrzyjmy się teraz funkcjom GRUB-a, przy użyciu których możliwe jest poruszanie się w systemie plików. Za pomocą polecenia `echo` możesz sprawdzić podstawową partycję (pamiętaj, że to na niej GRUB spodziewa się znaleźć jądro systemu):

---

```
grub> echo $root
hd0,msdos1
```

---

Polecenie `ls` pozwala też na wypisanie listy plików i katalogów w partycji podstawowej. Wystarczy jako parametr wpisać jej nazwę i zakończyć znakiem ukośnika:

---

```
grub> ls (hd0,msdos1)/
```

---

Niestety zapamiętanie i późniejsze wpisywanie pełnej nazwy partycji podstawowej może być kłopotliwe, dlatego zastosowanie zmiennej `$root` pozwala na sporą oszczędność czasu i nerwów:

---

```
grub> ls ($root)/
```

---

W efekcie zobaczysz krótką listę plików i katalogów umieszczonych w systemie plików tej partycji. Znajdą się na niej takie pozycje jak `etc/`, `bin/` lub `dev/`. Pamiętaj, że to zupełnie osobna funkcja polecenia `ls` z GRUB-a. Wcześniej wypisywało ono urządzenia, tablice partycji i czasami informacje nagłówkowe systemów plików. Teraz pozwala na faktyczne przeglądanie zawartości tych systemów plików.

W podobny sposób możesz też zajrzeć głębiej w hierarchię plików i katalogów danej partycji. Gdy na przykład chcesz zobaczyć zawartość katalogu `/boot`, możesz użyć następującego polecenia:

---

```
grub> ls ($root)/boot
```

---

### UWAGA

Za pomocą klawiszy strzałek w górę i w dół możesz przeglądać zawartość historii poleceń, natomiast klawisze strzałek w lewo i w prawo umożliwiają edycję aktualnie wybranego polecenia. Podobnie działają standardowe kombinacje klawiszy Ctrl+N i Ctrl+P.

Polecenie `set` pozwala na przejrzanie wszystkich zdefiniowanych w danym momencie zmiennych programu GRUB:

---

```
grub> set
?=0
color_highlight=black/white
color_normal=white/black
--ciach--
prefix=(hd0,msdos1)/boot/grub
root=hd0,msdos1
```

---

Jedną z najważniejszych zmiennych programu jest zmienna `$prefix` definiująca system plików i katalog, w którym GRUB powinien szukać swojej konfiguracji oraz dodatkowych plików. Więcej informacji na ten temat podam już w następnym punkcie.

Po zapoznaniu się z interfejsem wiersza poleceń GRUB-a możesz wprowadzić polecenie `boot`, żeby uruchomić system z zastosowaniem aktualnej konfiguracji, albo nacisnąć klawisz *Esc*, żeby wrócić do menu programu. Jednak teraz uruchom system, ponieważ zajmiemy się konfigurowaniem GRUB-a, a to najlepiej robić z wykorzystaniem narzędzi oferowanych przez system operacyjny.

## 5.5.2. Konfigurowanie programu GRUB

Katalog konfiguracji GRUB-a zawiera główny plik konfiguracyjny (*grub.cfg*) oraz wiele modułów ładowanych o rozszerzeniu *.mod*. Wraz z powstawaniem kolejnych wersji programu moduły te były przenoszone do podkatalogów, takich jak *i386-pc*. Zazwyczaj pliki te znajdują się w katalogu */boot/grub* lub */boot/grub2*. Nie należy bezpośrednio modyfikować zawartości pliku *grub.cfg*. Znacznie lepiej będzie, jeżeli użyjesz polecenia `grub-mkconfig` (a w dystrybucji Fedora — `grub2-mkconfig`).

### Przeglądanie pliku *grub.cfg*

Na początek przejrzymy zawartość pliku *grub.cfg*, żeby sprawdzić, w jaki sposób GRUB inicjuje swoje menu oraz opcje jądra. Plik ten zawiera różne polecenia GRUB-a, których lista zazwyczaj zaczyna się od pewnych operacji inicjujących. Następnie pojawia się lista pozycji menu dla różnych konfiguracji jądra i procesu jego rozruchu. Sama inicjacja nie jest bardzo skomplikowana. To zaledwie kilka poleceń definicji zmiennych oraz konfiguracji trybu wyświetlania, na przykład takich:

---

```
if loadfont /usr/share/grub/unicode.pf2 ; then
  set gfxmode=auto
  load_video
  insmod gfxterm
--ciach--
```

---

Dalej w pliku powinniśmy zobaczyć dostępne konfiguracje rozruchu systemu, przy czym każda z nich będzie zaczynać się od polecenia `menuentry`. Korzystając z tego, czego nauczyłeś się w poprzednim punkcie, z pewnością uda Ci się odczytać i zrozumieć zawartość poniższego przykładu:

---

```
menuentry 'Ubuntu, with Linux 3.2.0-34-generic-pae' --class ubuntu --class gnu-linux --class gnu
↳--class os {
    recordfail
    gfxmode $linux_gfx_mode
    insmod gzio
    insmod part_msdos
    insmod ext2
    set root='(hd0,msdos1)'
```

---

Przyjrzyj się poleceniom submenu. Jeżeli w pliku `grub.cfg` znajduje się wiele poleceń `menuentry`, większość z nich będzie zapewne umieszczona wewnątrz polecenia submenu, gromadzącego konfiguracje starszych wersji jądra, tak żeby nie zaśmiecały głównego menu programu.

## Generowanie nowego pliku konfiguracyjnego

Jeżeli chcesz wprowadzić zmiany do konfiguracji swojego programu GRUB, raczej nie polecam ręcznego edytowania pliku `grub.cfg`, ponieważ jest on generowany automatycznie i od czasu do czasu system nadpisuje jego zawartość. Nową konfigurację lepiej wprowadzić w innym miejscu, a następnie uruchomić polecenie `grub-mkconfig`, żeby wygenerować nowy plik konfiguracyjny.

Poznanie metod generowania pliku konfiguracyjnego najlepiej rozpocząć od przyjrzenia się początkowi pliku `grub.cfg`. Powinien tam znajdować się komentarz podobny do poniższego:

---

```
### BEGIN /etc/grub.d/00_header ###
```

---

Po dokładniejszym sprawdzeniu tego tropu przekonasz się, że każdy plik w katalogu `/etc/grub.d` jest skryptem powłoki, który generuje część zawartości pliku `grub.cfg`. Samo polecenie `grub-mkconfig` jest kolejnym skryptem powłoki, który uruchamia poszczególne pliki z katalogu `/etc/grub.d`.

Proponuję wypróbować działanie tego polecenia (nie zapomnij o uprawnieniach superużytkownika). Nie trzeba się przy tym obawiać o zniszczenie aktualnej konfiguracji. Poniższe polecenie wypisuje jedynie wygenerowaną konfigurację na standardowe wyjście:

---

```
# grub-mkconfig
```

---

A co zrobić, jeżeli chcesz dodać nowe pozycje do menu albo kolejne polecenia do konfiguracji programu GRUB? Krótko mówiąc, wszystkie takie zmiany umieścić w nowym pliku *custom.cfg* znajdującym się w katalogu konfiguracyjnym programu, takim jak */boot/grub/custom.cfg*.

Dłuższa odpowiedź będzie już trochę bardziej złożona. Katalog konfiguracji */etc/grub.d* daje do wyboru dwie opcje: *40\_custom* oraz *41\_custom*. Pierwsza z nich jest skryptem, który możesz samodzielnie edytować, ale to rozwiązanie mniej stabilne. Każda aktualizacja pakietu może zniszczyć wszystkie zmiany, jakie tu wprowadzisz. Skrypt *41\_custom* jest znacznie prostszy i zawiera jedynie serię poleceń ładujących plik *custom.cfg* w momencie uruchamiania GRUB-a. Pamiętaj, że jeżeli zdecydujesz się na tę drugą opcję, Twoje modyfikacje nie pojawią się podczas generowania pliku konfiguracyjnego.

Obie opcje własnych plików konfiguracyjnych nie dają szczególnie wielkich możliwości. W poszczególnych dystrybucjach w katalogu */etc/grub.d* można zobaczyć różne dodatkowe elementy. Przykładowo Ubuntu dodaje do konfiguracji opcję uruchomienia testu pamięci (*memtest86+*).

Aby wygenerować i zapisać nowy plik konfiguracji GRUB-a, należy zastosować opcję `-o` podczas uruchamiania polecenia `grub-mkconfig`, podając lokalizację wynikowego pliku, na przykład tak:

---

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

---

Użytkownicy Ubuntu mogą po prostu skorzystać z polecenia `install-grub`. W każdym przypadku uprzednio pamiętaj o wykonaniu kopii bezpieczeństwa dotychczasowego pliku konfiguracyjnego, upewnij się, że instalujesz nowy plik we właściwym katalogu i tym podobne.

Teraz możesz przejść do bardziej technicznych szczegółów GRUB-a i ogólnie programów rozruchowych. Jeżeli czujesz już znużenie tym tematem, możesz od razu przejść do rozdziału 6.

### 5.5.3. Instalowanie programu GRUB

Instalowanie GRUB-a jest bardziej złożone od jego konfigurowania. Na szczęście zazwyczaj nie musisz przejmować się tym procesem, ponieważ zadanie to przejmuje dystrybucja. Jeżeli jednak próbujesz wykonać kopię dysku z systemem lub odtworzyć taki dysk albo przygotować własną sekwencję rozruchową, całkiem możliwe, że nie obejdziesz się bez samodzielnej instalacji programu rozruchowego.

Zanim przejdziemy dalej, proponuję przeczytać punkt 5.8.3, żeby zorientować się w sposobie rozruchu komputera PC i sprawdzić, czy masz do czynienia z procedurą MBR, czy raczej z EFI. Następnie musisz skompilować GRUB-a i określić, w jakim katalogu ma zostać umieszczony. Domyślnie stosuje się katalog */boot/grub*. Jeżeli dystrybucja sama przejmuje to zadanie, możliwe, że samodzielne kompilowanie programu nie będzie konieczne, ale jeżeli będzie trzeba, możesz zajrzeć do rozdziału 16., żeby poznać metody kompilowania oprogramowania z kodu źródło-

wego. Warto się przy tym upewnić, że podczas kompilacji używany jest właściwy cel, ponieważ dla procedur MBR i EFI stosowane są różne cele kompilacji. Dodatkową różnicą jest wersja procesu EFI — może być 32-bitowa lub 64-bitowa.

## Instalowanie GRUB-a w swoim systemie

Instalowanie programu rozruchowego wymaga zdefiniowania elementów, takich jak:

- docelowy katalog programu widzianego przez aktualnie pracujący system; zazwyczaj jest to katalog `/boot/grub`, ale jeżeli instalujesz program rozruchowy na innym dysku albo używasz innego systemu, katalog ten może być inny,
- aktualne urządzenie docelowego dysku dla GRUB-a,
- aktualny punkt montowania partycji rozruchowej EFI, w przypadku procedury rozruchowej EFI.

Pamiętaj, że GRUB jest systemem modułowym, ale sam program musi odczytać system plików z tym programem, żeby mógł załadować odpowiednie w danej chwili moduły. Twoim zadaniem jest tu skonstruowanie takiej wersji programu, która będzie w stanie odczytać ten system plików, co pozwoli jej na załadowanie pozostałej części swojej konfiguracji (plik `grub.cfg`) oraz wszystkich wymaganych modułów. W Linuksie oznacza to zwykle konieczność zbudowania wersji programu z załadowanym już modulem `ext2.mod`. Po przygotowaniu takiej wersji pozostaje już tylko umieszczenie jej w części dysku przeznaczony do rozruchu systemu i skopiowanie pozostałych plików do katalogu `/boot/grub`.

Na szczęście w zestawie z programem GRUB dostarczane jest narzędzie o nazwie `grub-install` (nie należy go mylić z poleceniem `install-grub` z Ubuntu), które wykonuje większość prac związanych z instalacją plików i konfigurowaniem GRUB-a. Jeśli na przykład aktualny dysk to `/dev/sda` i na nim chcesz zainstalować GRUB-a z zastosowaniem standardowego katalogu `/boot/grub`, poniższe polecenie zainstaluje program rozruchowy w MBR:

---

```
# grub-install /dev/sda
```

---

**OSTRZEŻENIE** *Nieprawidłowa instalacja GRUB-a może spowodować uszkodzenie sekwencji rozruchu systemu operacyjnego, dlatego nie wykonuj tej operacji pochopnie. Jeżeli masz wątpliwości, lepiej najpierw dowiedz się, jak można wykonać kopię zapasową MBR za pomocą polecenia `dd`, wykonaj kopię katalogu aktualnie zainstalowanej wersji GRUB-a i upewnij się, że masz plan na wypadek awarii.*

## Instalowanie GRUB-a na zewnętrznym nośniku

Aby zainstalować GRUB na nośniku danych niezależnym od aktualnego systemu, należy zdefiniować katalog GRUB-a na tym urządzeniu, tak jak widzi go aktualnie pracujący system. Załóżmy, że docelowe urządzenie to `/dev/sdc`, a katalog podstawowy (lub rozruchowy) systemu plików z tego urządzenia (na przykład `/dev/sdc1`)

zamontowany jest w aktualnym systemie w katalog `/mnt`. Oznacza to, że podczas instalowania GRUB-a aktualny system będzie widział pliki programu rozruchowego w katalogu `/mnt/boot/grub`. A zatem uruchamiając polecenie `grub-install`, musisz zastosować poniższe parametry:

---

```
# grub-install --boot-directory=/mnt/boot /dev/sdc
```

---

## Instalowanie GRUB-a w systemach z UEFI

Procedura instalacji w systemach z UEFI powinna być prostsza, ponieważ w takiej sytuacji trzeba tylko skopiować program rozruchowy we właściwe miejsce. Oprócz tego konieczne jest jednak „zaprezentowanie” programu rozruchowego firmware’owi komputera za pomocą polecenia `efibootmgr`. Jeżeli polecenie to jest dostępne, program `grub-install` uruchomi je automatycznie, dlatego w teorii instalacja GRUB-a na partycji UEFI wymagać będzie jedynie wydania poniższego polecenia:

---

```
# grub-install --efi-directory=katalog_efi --bootloader-id=nazwa
```

---

W poleceniu tym parametr `katalog_efi` opisuje umiejscowienie katalogu UEFI w aktualnym systemie (zazwyczaj jest to katalog `/boot/efi/efi`, ponieważ partycja UEFI jest najczęściej montowana w katalogu `/boot/efi`), natomiast parametr `nazwa` jest prostym identyfikatorem programu rozruchowego, który omówię w punkcie 5.8.2.

Niestety podczas instalowania programu rozruchowego dla UEFI może pojawić się wiele różnych problemów. Przykładowo podczas instalowania go na dysku, który ma się znaleźć w innym systemie, musimy znaleźć sposób na zaprezentowanie nowego programu rozruchowego firmware’owi nowego komputera. Poza tym procedura instalacji na zewnętrznych nośnikach danych wygląda zupełnie inaczej.

Jednak najpoważniejszym problemem jest procedura bezpiecznego rozruchu UEFI (ang. *secure boot*).

## 5.6. Problemy z bezpiecznym rozruchem UEFI

Jednym z najnowszych problemów, z którymi muszą się zmagać systemy linuksowe, jest funkcja bezpiecznego rozruchu pojawiająca się w najnowszych komputerach PC. Gdy jest aktywna, UEFI wymaga od programu rozruchowego cyfrowego podpisu wystawionego przez zaufaną organizację. Microsoft wymógł na producentach sprzętu dostarczających system Windows 8, żeby zawsze używali funkcji bezpiecznego rozruchu. W efekcie każda próba uruchomienia zainstalowanego, niepodpisanego programu rozruchowego (a jest to powszechne w linuksowych dystrybucjach) skazana jest na niepowodzenie.

Najprostszą metodą obejścia tego problemu dla osób niezainteresowanych używaniem systemów Windows jest wyłączenie funkcji bezpiecznego rozruchu w ustawieniach UEFI. Niestety nie sprawdzi się to w komputerach z dwoma systemami, a poza tym nie jest to rozwiązanie odpowiednie dla wszystkich użytkowników. W związku z tym dystrybucje Linuksa coraz częściej dostarczają podpisane programy rozruchowe. Niektóre rozwiązania są tylko programami wstępnymi dla GRUB-a, ale niektóre to w pełni podpisane sekwencje rozruchu systemu (od programu rozruchowego aż po jądro). Inne rozwiązanie polega na zastosowaniu zupełnie nowego rodzaju programu rozruchowego (programy te w większości bazują na programie *efilinux*).

## 5.7. Ładowanie innych systemów operacyjnych

UEFI bardzo ułatwia obsługę innych systemów operacyjnych, ponieważ na partycji EFI można zainstalować kilka różnych programów rozruchowych. Niestety takie rozwiązania nie są obsługiwane przez stary dobry rekord MBR, dlatego nawet w komputerach z UEFI możemy być zmuszeni do używania programu rozruchowego współpracującego z MBR. Możliwe jest natomiast takie skonfigurowanie GRUB-a, żeby załadował i uruchomił inny program rozruchowy umieszczony na wyznaczonej partycji. Taki proces nazywany jest *rozruchem łańcuchowym* (ang. *chainloading*).

Aby zastosować rozruch łańcuchowy, należy utworzyć nową pozycję menu w konfiguracji GRUB-a (stosując metody opisane w podpunkcie „Generowanie nowego pliku konfiguracyjnego”). Oto przykładowa konfiguracja uruchamiająca system Windows zainstalowany na trzeciej partycji dysku:

---

```
menuentry "Windows" {
    insmod chain
    insmod ntfs
    set root=(hd0,3)
    chainloader +1
}
```

---

Wartość +1 w opcji `chainloader` nakazuje załadować dowolne dane, jakie znajdują się w pierwszym sektorze tej partycji. Możliwe jest też bezpośrednio załadowanie całego pliku. Przykładowo poniższy wariant umożliwia załadowanie pliku ładującego system MS-DOS, czyli *io.sys*:

---

```
menuentry "DOS" {
    insmod chain
    insmod fat
    set root=(hd0,3)
    chainloader /io.sys
}
```

---

## 5.8. Szczegóły programu rozruchowego

Teraz przyjrzymy się pokrótce wewnętrznym mechanizmom programu rozruchowego. Jeżeli nie interesuje Cię ta tematyka, możesz spokojnie przejść do następnego rozdziału.

Jeśli chcesz zrozumieć zasadę działania takich programów jak GRUB, musisz przyrzeć się sposobom rozruchu komputera PC po włączeniu zasilania. Z powodu różnych braków tradycyjnych mechanizmów rozruchu komputerów PC istnieje kilka wariantów tego procesu, ale łączą się one w dwa główne schematy — MBR i UEFI.

### 5.8.1. Rozruch MBR

Oprócz informacji o partycjach opisywanych już w podrozdziale 4.1, rekord rozruchowy MBR (ang. *Master Boot Record*) zawiera niewielki obszar, który jest ładowany do pamięci przez BIOS i uruchamiany zaraz po zakończeniu wstępnego testu (POST — ang. *Power-On Self-Test*). Niestety jest to zbyt mała przestrzeń, żeby zmieścić w niej jakikolwiek program rozruchowy, dlatego wykorzystywane jest dodatkowe miejsce, co tworzy tak zwany *wieloetapowy program rozruchowy*. W tym przypadku pierwszy kawałek kodu z rekordu MBR zajmuje się jedynie załadowaniem pozostałej części kodu programu rozruchowego, który najczęściej zostaje upchnięty pomiędzy rekordem MBR a pierwszą partycją na dysku.

Oczywiście takie rozwiązanie nie jest zbyt bezpieczne, ponieważ każdy program może nadpisać znajdujący się w tym miejscu kod, ale z metody tej korzysta większość programów rozruchowych, w tym i sam GRUB. Co więcej, ta metoda rozruchu nie współpracuje z partycjami GPT, ponieważ tablica GPT umieszczana jest zaraz za rekordem MBR. Mechanizm GPT nie narusza jednak samego MBR w celu zachowania wstecznej zgodności.

Obejściem tego problemu w przypadku partycji GPT jest przygotowanie niewielkiej partycji nazywanej partycją rozruchową o specjalnym identyfikatorze UUID, która daje programowi rozruchowemu dość miejsca na dysku. Partycje GPT są jednak zazwyczaj używane w połączeniu ze schematem UEFI, a nie z tradycyjnym BIOS-em.

### 5.8.2. Rozruch UEFI

Producenci komputerów PC oraz firmy tworzące oprogramowanie zauważyli, że tradycyjny proces rozruchu związany z BIOS-em jest bardzo ograniczony, dlatego zdecydowano się przygotować nowsze rozwiązanie o angielskiej nazwie *Extensible Firmware Interface (EFI)*. Trochę czasu upłynęło, zanim EFI przyjęło się na komputerach PC, ale dzisiaj jest już całkiem powszechne. Aktualnym standardem jest UEFI (*Unified EFI*), które udostępnia takie funkcje jak wbudowana powłoka oraz możliwość odczytywania tablic partycji i nawigowania w systemach plików. Sposób partycjonowania GPT jest częścią standardu UEFI.

W systemach z UEFI rozruch wygląda zupełnie inaczej i jest zdecydowanie łatwiejszy do zrozumienia. W tym przypadku wykonywalny kod rozruchowy nie



musi znajdować się poza systemem plików. Na dysku tworzony jest specjalny system plików nazywany *systemową partycją EFI* (ang. *EFI System Partition* — ESP), który zawiera katalog o nazwie *efi*. Każdy program rozruchowy ma swój własny identyfikator oraz podkatalog, taki jak *efi/microsoft*, *efi/apple* albo *efi/grub*. Plik programu rozruchowego musi mieć rozszerzenie *.efi* i znajdować się w jednym z tych katalogów, razem ze wszystkimi plikami wspomagającymi.

**UWAGA** *Partycja ESP różni się od partycji rozruchowej BIOS opisywanej w punkcie 5.8.1 i ma inny identyfikator UUID.*

I tutaj pojawia się mały problem. Nie można po prostu umieścić kodu starego programu rozruchowego w takim katalogu, ponieważ kod ten został przystosowany do współpracy z interfejsem BIOS-u. Konieczne staje się stosowanie programów rozruchowych przystosowanych do pracy z UEFI. Kiedy przykładowo stosujesz GRUB, musisz użyć specjalnej wersji dla UEFI, a nie dla BIOS-u. Dodatkowo konieczne jest „przedstawienie” takiego programu rozruchowego firmware’owi komputera.

Jak już wspominałem w podrozdziale 5.6, pojawia się też problem „bezpiecznego rozruchu”.

### 5.8.3. Jak działa GRUB?

Podsumuję ten rozdział opisaniem sposobów działania programu GRUB.

1. BIOS lub firmware komputera PC inicjuje elementy sprzętowe i poszukuje kodu rozruchowego na dyskach w kolejności ustalonej w konfiguracji.
2. Po odnalezieniu kodu rozruchowego BIOS lub firmware ładuje go do pamięci i uruchamia. To w tym momencie GRUB ożywa.
3. Ładowane są główne funkcje GRUB-a.
4. Następuje inicjacja. W tym momencie GRUB może już uzyskać dostęp do dysków i systemów plików.
5. GRUB odszukuje swoją partycję podstawową i pobiera z niej konfigurację.
6. GRUB daje użytkownikowi szansę na zmianę tej konfiguracji.
7. Po upływie czasu oczekiwania GRUB wykonuje instrukcje z konfiguracji (sekwencję poleceń opisywaną w punkcie 5.5.2).
8. W trakcie wykonywania instrukcji konfiguracyjnych GRUB może załadować dodatkowy kod (*moduły*) z partycji podstawowej.
9. GRUB wykonuje polecenie boot, które ładuje i uruchamia jądro zgodnie z instrukcjami podawanymi w poleceniu `linux`.

Kroki 3. i 4. podanej wyżej sekwencji, w których ładowany jest podstawowy program GRUB-a, mogą być bardzo skomplikowane, co wynika z różnorodności tradycyjnego mechanizmu rozruchu komputerów PC. Podstawowe pytanie brzmi w tym momencie: „Gdzie *jest* jądro GRUB-a?”. Można na nie odpowiedzieć na trzy różne sposoby.

- Częściowo może być upchnięte pomiędzy rekordem MBR a początkiem pierwszej partycji.
- Znajduje się na zwyczajnej partycji na dysku.
- Umieszczony jest na specjalnej partycji rozruchowej: partycji rozruchowej GPT, partycji systemowej EFI albo gdzieś indziej.

We wszystkich przypadkach, z wyjątkiem użycia partycji ESP, BIOS komputera załaduje 512 bajtów rekordu MBR i to właśnie od tego zaczyna się praca GRUB-a. Ten niewielki wycinek kodu (pobrany z pliku *boot.img* z katalogu GRUB-a) nie jest jeszcze jego jądrem, ale przechowuje dane o jego lokalizacji i zajmuje się jego załadowaniem do pamięci.

Jeżeli jednak korzystasz z partycji ESP, cały kod GRUB-a umieszczony jest w jednym pliku. Firmware komputera może wtedy przeszukać partycję ESP i bezpośrednio uruchomić jądro GRUB-a albo dowolny inny program rozruchowy, jaki się na niej znajdzie.

Niestety w przypadku większości systemów nie jest to jeszcze pełny obraz sytuacji. Programy rozruchowe mogą wymagać załadowania obrazu wstępnego systemu plików w pamięci RAM i dopiero potem przystąpić do ładowania i uruchamiania jądra systemu operacyjnego. To właśnie takie zachowanie definiuje parametr konfiguracji `ini trd` opisywany w podrozdziale 6.8. Zanim jednak zaczniesz poznawać wstępny system plików w pamięci RAM, powinieneś dowiedzieć się, w jaki sposób uruchamiana jest przestrzeń użytkownika. I tym zajmiemy się w następnym rozdziale.

# Skorowidz

## A

- abstrakcje, 24
- adres
  - IP, 252
  - MAC, 258
- akcja
  - ctrlaltdel, 188
  - respawn, 187
  - sysinit, 188
- aktywowanie jednostki, 160
- aliasy, 370
- analiza sieci lokalnej, 276
- aplikacje, 382
  - internetowe, 435
  - sieciowe, 295
- archiwa skompresowane, 69
- archiwizowanie, 67, 68
- argumenty
  - kontrolne, 218
  - modułu, 221
  - pojedyncze, 324
- ARP, Address Resolution Protocol, 289
- ATA, 96
- ataki DoS, 313

## B

- bazy danych, 436
- biblioteka, 398
  - gobject, 399
  - libata, 96
- biblioteki współużytkowane, 399–402
- BIOS, 137, 150
- bit setuid, 212–215
- błędy, 56
  - drugorzędne strony, 238
  - podstawowe strony, 238
  - procesu budowania, 431
  - stron, 238
  - typowe, 57
- brama domyślna, 255
- buforowanie, 268
  - dysku, 113
  - systemu plików, 113

## C

- cele
  - kompilacji, 408
  - systemu autoconf, 425

CIDR, Classless Inter-Domain Routing, 253

cron

pliki crontab, 210

cudzysłowy, 321

podwójne, 323

pojedyncze, 322

CUPS, 391

czas, 206

sieciowy, 208

systemowy, 197, 234

trwania, 234

użytkownika, 234

## D

debuggery, 411

debugowanie, 55

definicje makr, 397

demon

inetd, 305

xinetd, 305

DHCP, 276

DNS, 256, 258

dodawanie

do ścieżki, 368

tras, 261

użytkowników, 357

dokumenty miejscowe, 338

domyślne ustawienia

powłoki, 375

użytkownika, 375

dostęp do

dysków, 101

jednego urządzenia, 98

plików, 361

serwera, 355

dowiązania symboliczne, 65

drukowanie, 390

dyrektywy warunkowe, 398

dysk twardy, 82, 99, 106

PATA, 84

SATA, 83, 96

SSD, 107

działanie

GRUB, 151

pamięci, 237

procesu udevd, 88

procesu Upstart, 183

programu pkg-config, 427

dziennik systemowy, 199

## E

edycja wiersza poleceń, 50

edytor, 376

edytor tekstu, 51

eksplorowanie klientów serwera X, 384

elementy pierwotne, 309

Ethernet, 258, 289

Ethernet bezprzewodowy, 291

ewolucja systemów plików, 130

ext3, 108

ext4, 108

## F

FAT, 109

FFS, Fas File System, 108

filtrowanie

według protokołu i portu, 307

według statusu połączenia, 308

filtry wydruku, 392

FTP, File Transfer Protocol, 249

funkcja, 200

## G

generowanie pliku konfiguracyjnego, 145

geometria

dysku, 105

partycji, 105

globbing, 42

gniazda, 315

gniazda domenowe systemu, 317

GNU autoconf, 420

GRUB

działanie, 151

instalowanie, 146

konfigurowanie, 140, 144

menu, 140

przeszukiwanie urządzeń, 142

wiersz poleceń, 142

grupa, 205

## H

hasła, 222, 356

hierarchia katalogów, 35, 70

host, 258

host lokalny, 270

HTTP, 296

**I**

- identyfikator
  - procesu, 326
  - UUID, 112, 113
  - użytkownika
    - efektywny, 212
    - rzeczywisty, 212
    - zapisany, 212
- identyfikowanie użytkowników, 215
- informacje o użytkownikach, 216
- inicjowanie jądra, 135
- init System V
  - farma dowiązań, 189
  - identyfikowanie rodzaju procesu, 156
  - run-parts, 190
  - sekwencja poleceń rozruchowych, 188
  - sterowanie procesem, 191
- instalowanie
  - oprogramowania, 428
  - plików crontab, 210
  - programu GRUB, 146
- instancja, 172
  - sesji, 389
  - systemowa, 389
- instrukcja case, 333
- interfejs
  - niezarządzany, 265
  - SCSI, 82
  - sieciowy jądra, 259
  - użytkownika, 379
    - komponenty, 380
- IP, 289
- IPC, Interprocess Communication, 316

**J**

- Java, 415
- jądro, kernel, 24, 26, 73
- jednostka echo@.service, 172
- jednostki, 157
- jednostki pomocnicze, 169
- język
  - Emacs Lisp, 414
  - Java, 415
  - JavaScript, 414
  - m4, 415
  - Mathematica, 415
  - Matlab, 414
  - Octave, 414

- Perl, 414
- PHP, 414
- Python, 413
- Ruby, 414
- Tcl, 415
- języki skryptowe, 412

**K**

- katalog
  - /bin, 71
  - /boot, 73
  - /dev, 71
  - /etc, 71, 198
  - /home, 71
  - /include, 73
  - /info, 73
  - /lib, 71
  - /local, 73
  - /man, 73
  - /media, 73
  - /opt, 73
  - /proc, 72
  - /sbin, 72
  - /share, 73
  - /sys, 72
  - /tmp, 72
  - /usr, 72
  - /var, 72
- katalogi
  - domowe, 359
  - główne, 72
  - kompilacji, 423
  - nagłówkowe, 396
- klawiatura, 387
- klient
  - DHCP, 277
  - NFS, 362
  - SSH, 303
- klucze hostów, 301
- kod wyjścia, 327
- kolejność plików uruchomieniowych, 371
- kompilacja jądra systemu, 434
- kompilator języka C, 394
- kompilowanie
  - oprogramowania, 417
  - pakietów, 416
- kompresja, 352
- kompresowanie plików, 67
- komunikacja procesów z siecią, 315

- komunikat o błędzie, 56, 431–433
- komunikaty rozruchowe, 134
- konfigurowanie
  - interfejsów sieciowych, 260
  - narzędzia NetworkManager, 265
  - procesu udevd, 88
  - programu GRUB, 140, 144
  - reguł, 285
  - sieci, 247
    - aktywowana podczas rozruchu, 261
    - menedżery konfiguracji, 263
    - ręczna, 262
  - systemd, 160
  - systemu, 197
  - systemu PAM, 217
  - udziału plikowego, 358
  - Upstart, 178
  - zależności, 159
- konsekwencje dwóch rodzajów powłok, 372
- konsole wirtualne, 84
- konsolidacja z bibliotekami, 398, 402
- konstrukcja
  - &&, 330
  - ||, 330
- konstrukcje logiczne, 330
- kontrola zadań, 61
- konwersja formatów, 392
- kopia struktury katalogów, 348
- kopiowanie pliku, 345
- korzystanie z powłoki, 37

## L

- liczba argumentów, 325
- literały, 321

## Ł

- ładowanie innych systemów operacyjnych, 149
- łącza internetowe, 279

## M

- MAC, Media Access Control, 258
- magistrala SCSI, 93
- make
  - aktualizowanie, 406
  - argumenty, 407
  - budowanie programu, 406
  - makra, 408

- opcje wiersza poleceń, 407
- wbudowane reguły, 405
- wiersz poleceń, 407
- zmiennie, 408
- manipulowanie
  - hasłami, 204
  - użytkownikami, 204
- maska
  - uprawnień, 370
  - podsieci, 253
- maskarada IP, 280
- MBR, 103
- menedżery
  - konfiguracji sieciowych, 263
  - okien, 380
  - wyświetlaczy, 383
- metody dostępu do urządzenia, 98
- MMU, Memory Management Unit, 237
- moduł, 24
- moduły systemu PAM, 221
- modyfikowanie
  - plików uruchomieniowych, 366
  - sekwencji uruchamiania, 190
  - tablicy partycji, 104
  - uprawnień, 64
- monitorowanie
  - operacji wejścia-wyjścia, 241
  - poszczególnych procesów, 244
  - urządzeń, 91
  - urządzeń wejścia-wyjścia, 243
  - wydajności procesora, 239
  - zasobów, 233
- montowanie
  - ponowne systemu plików, 115
  - systemu plików, 110, 114
- mysz, 387

## N

- nadawanie procesom priorytetów, 234
- napęd DVD, 83
- naprawianie systemów plików, 119
- narzędzia
  - diagnostyczne, 306
  - do zarządzania partycjami, 102
  - kompresujące, 70
  - programistyczne, 393
  - skryptów powłoki, 338
- narzędzie, *Patrz* program
- NAT, 280
- nazewnictwo urządzeń, 82

- nazwa
  - hosta, 267
  - skryptu, 326
- nazwy wieloznaczne, 42, 43
- notacja CIDR, 253
- numery portów, 273



- obliczenia
  - na żądanie, 438
  - rozproszone, 438
- obsługa list pustych parametrów, 329
- odczyt jądra, 103
- ograniczenia skryptów powłoki, 320
- okno powłoki, 37
- opcje
  - długie, 115
  - krótkie, 114
  - montowania systemów plików, 114
  - polecenia, 59
  - rozruchu, 135
  - sieciowych usług plikowych, 363
  - skryptu configure, 423
- operacja kopiowania, 349
- operacje wejścia-wyjścia, 241
- operatory, 310
  - typu plików, 332
  - uprawnień plików, 332
- optymalizacja uruchamiania, 169
- organizacja systemu, 25
- organizowanie pliku Makefile, 409

## P

- pakiet, 24, 248
- pakiet Samba, 354
- pakiety
  - kodu źródłowego, 419
  - narzędziowe, 381
  - źródłowe, 434
- PAM, Pluggable Authentication Modules, 217
  - argumenty kontrolne, 218
  - argumenty modułu, 221
  - hasła, 222
  - konfiguracja systemu, 217
  - reguły zestawiane, 218
  - typy funkcji, 218
- pamięć, 237
  - Flash, 83
  - masowa USB, 95
  - operacyjna, 26

- parametry jądra, 136
- partycja, 99
  - logiczna, 103
  - podstawowa, 103
  - rozszerzona, 103
- partycjonowanie urządzeń dyskowych, 102
- pełny dostęp, 313
- pętla
  - for, 334
  - while, 335
- plik
  - .bashrc, 372
  - .cshrc, 374
  - /etc/fstab, 118
  - /etc/hosts, 268
  - /etc/nsswitch.conf, 269
  - /etc/passwd, 202, 204
  - /etc/services, 273
  - /etc/shadow, 204
  - /etc/shells, 218
  - /etc/sudoers, 74
  - config.log, 426
  - echo@.service, 173
  - grub.cfg, 144
  - Makefile, 404, 409
  - resolv.conf, 268
- pliki
  - crontab, 210
  - dziennika, 108, 358, 426
  - jednostek, 161
  - konfiguracyjne, 199
  - kopie, 345
  - nagłówkowe, 396, 397
  - otwarte, 226
  - przenoszenie w sieci, 345
  - tymczasowe, 337
  - uruchomieniowe, 366
  - urządzeń, 78, 86
  - współużytkowane, 345, 353
  - z kropką, 47
  - źródłowe, 395
- pobieranie danych, 344
- podmiana poleceń, 336
- podpowłoki, 342
- podręcznik systemowy, man, 53, 368
- podsieci, 252
- podsystem, 24
- podsystem SCSI, 94
- pojemność systemu plików, 118

- polecenia, 35
  - działające na katalogach, 41
  - pośredniczące, 44
  - SCSI, 93
- polecenie
  - awk, 339
  - basename, 339
  - cat, 37
  - cd, 42
  - cp, 40
  - dd, 81
  - diff, 46
  - echo, 41
  - exec, 342
  - expr, 342
  - file, 46
  - find, 46
  - grep, 44
  - head, 47
  - less, 45
  - locate, 46
  - ls, 39
  - lsof, 227, 228
  - ltrace, 230
  - mkdir, 42
  - mv, 40
  - ping, 256
  - pwd, 45
  - rm, 41
  - rmdir, 42
  - sed, 340
  - sort, 47
  - strace, 229
  - sudo, 74
  - tail, 47
  - tcpdump, 308
  - touch, 40
  - uptime, 235
  - vmstat, 239
  - xargs, 341
  - xset, 388
- połączenia, 271
- połączenia TCP, 272
- pomiar czasu procesora, 233
- pomoc, 52
- poprawki, 429
- porównywanie ciągów znaków, 333
- porty
  - równoległe, 85
  - szeregowo, 85
  - TCP, 271

- POST, Power-On Self-Test, 150
- powłoka, 54
  - bash, 371
  - Bourne'a, 36, 320
  - tcsh, 374
- powłoki
  - bez logowania, 372
  - interaktywne, 373
  - logowania, 371
- poziomy uruchomienia System V, 155
- praktyki instalacyjne, 428
- prawo właściciela procesu, 212
- preprocesor, 397
- priorytet, 200
- priorytet procesu, 234
- problemy z
  - bibliotekami, 402
  - instalowaniem, 430
  - kompilowaniem, 430
- proces
  - init, 154, 156
  - init System V, 186
  - systemd, 167
  - udevd, 88
  - Upstart, 175
- procesy, 59
  - działające w tle, 61
  - jednowątkowe, 231
  - wielowątkowe, 231
- program
  - at, 211
  - cron, 209
  - coreboot, 138
  - efilinux, 138
  - fdisk, 102
  - gdisk, 102
  - getty, 206
  - gparted, 102
  - GRUB, 138–140
  - gzip, 67
  - init, 154
  - iostat, 241
  - iotop, 243
  - iw, 292
  - lex, 412
  - LILO, 138
  - Linux Kernel EFISTUB, 139
  - LOADLIN, 138
  - login, 206
  - lsof, 226, 306
  - make, 403



- netcat, 310
- NetworkManager, 263–265
- parted, 102
- pidstat, 244
- pkg-config, 426
- rsync, 346, 348
- stronicujący, 376
- SYSLINUX, 138
- systemd, 154
- tar, 67
- tcpdump, 310
- traceroute, 257
- udevadm, 90
- Upstart, 154
- yacc, 412
- zcat, 69
- programy
  - pomocnicze systemd, 173
  - rozruchowe, 137, 138
- protokoły procesu Upstart, 184
- protokół
  - ATA, 96
  - DHCP, 276
  - ICMP, 256
  - SCSI, 95
  - SSH, 299
  - TCP, 271, 273
  - UDP, 271, 274
- przeglądanie
  - procesów, 59
  - tablicy partycji, 102
- przekazywanie, 172
- przekazywanie literału, 323
- przekierowywanie standardowego wejścia, 56
- przełączanie użytkowników, 212
- przenoszenie plików, 345
- przerywanie działania procesów, 60
- przestrzeń
  - jądra, kernel space, 25
  - użytkownika, user space, 24, 30, 125, 130, 153
  - wymiany, 123
  - rozmiar, 124
  - wykorzystywanie partycji, 123
  - wykorzystywanie pliku, 124
- przesyłanie plików, 352
- przeszukiwanie
  - partycji, 142
  - urządzeń, 142
- przezroczystość sieci, 383
- pałapki w plikach uruchomieniowych, 376
- punkt montowania, 111

## R

- ramka, 258
- reguły
  - zapory sieciowej, 285
  - zestawiane, 218
- rejestrwanie dzienników systemowych, 197, 199
- reprezentacja czasu jądra, 207
- router, 277, 282
- rozpakowywanie
  - pakietów, 419
  - plików tar, 68
- rozpoznawanie nazw hostów, 267
- rozruch
  - awaryjny, 195
  - MBR, 150
  - UEFI, 148, 150
- rozsyłanie, 266
- rozwiązywanie problemów, 201
- rozwijanie nazw, 42
- RPC, Remote Procedure Call, 311
- rsync
  - integralność transferu, 351
  - kompresja, 352
  - kopiowanie struktury katalogów, 348
  - ograniczanie przepustowości, 352
  - pomijanie plików, 350
  - przesyłanie plików, 352
  - sumy kontrolne, 351
  - tryby informacyjne, 351

## S

- Samba, 354
  - diagnostyka, 358
  - dodawanie użytkowników, 357
  - dostęp do plików, 361
  - hasła, 356
  - katalogi domowe, 359
  - konfigurowanie serwera, 354
  - konfigurowanie udziału plikowego, 358
  - kontrola dostępu, 355
  - korzystanie z klientów, 360
  - pliki dziennika, 358
  - uruchamianie serwera, 358
  - usuwanie użytkowników, 357
- SATA, 96
- schemat
  - dysku, 100
  - jądra, 101
  - podsystemu SCSI, 94
  - sterowników urządzeń, 98

SCSI, Small Computer System Interface, 82, 92, 96  
 Secure Shell, 299  
 sekcja [Install], 162  
 sekwencja poleceń rozruchowych, 188  
 serwer
 

- DHCP, 277
- SSH, 302
- SSHD, 300
- X
  - dane wejściowe, 385
  - ustawianie preferencji, 385
  - zdarzenia, 384

 serwery
 

- sieciowe, 298
- WWW, 435

 sieć, 247
 

- bezprzewodowa, 291
- lokalna, 248
- prywatna, 279

 skanowanie portów, 310  
 składnia rozszerzona, 200  
 skróty klawiaturowe, 51  
 skrypt configure, 423  
 skrypty powłoki, 36, 319  
 słowo kluczowe elif, 329  
 SMB, Server Message Block, 354  
 specyfikatory, 163  
 sprawdzanie
 

- systemu plików, 121
- warunków, 330
- zalogowania, 373

 SSD, solid-state disk, 107  
 SSH, Secure Shell, 299  
 SSHD, 300  
 SSL, Secure Socket Layer, 249  
 standardowe wejście i wyjście, 38  
 standardowy strumień błędów, 55  
 stdin, 38  
 stdout, 38  
 sterowniki urządzeń, 29  
 stosowanie poprawek, 429  
 strefy czasowe, 207  
 strofa expect, 182  
 struktura
 

- katalogu /etc, 198
- węzłów inode, 127

 strumień błędów, 55  
 suma kontrolna, 351  
 superużytkownik, 32, 74, 204  
 symbol
 

- aktualnego katalogu, 368
- zachęty, 369

 synchronizacja procesów systemd, 167  
 system
 

- autoconf, 425
- DNS, 256, 268
- drukowania, 391
- GNU autoconf, 421
- NFS, 362
- PAM, 217, 221
- plików, 99, 107
  - buforowanie, 113
  - CIFS, 361
  - devtmpfs, 87
  - ewolucja, 130
  - ext2, 109
  - ext3, 108
  - ext4, 108
  - FAT, 109
  - FFS, 108
  - HSF+, 109
  - identyfikator UUID, 112
  - initramfs, 193
  - montowanie, 110
  - naprawianie, 119
  - o specjalnym znaczeniu, 122
  - opcje montowania, 114
  - początkowy, 193
  - pojemność, 118
  - ponownie montowanie, 115
  - sprawdzanie, 119
  - struktura węzłów inode, 127
  - tabela, 116
  - tradycyjny, 126
  - tworzenie, 109
  - typy, 108
  - UFS, 108
    - wbudowany, 438
  - udev, 87
  - V, 155, 173
  - X Window System, 382
 systemd, 156, 163
  - dodawanie jednostek, 166
  - jednostki, 157
  - konfiguracja, 160
  - programy pomocnicze, 173
  - specyfikatory, 163
  - typy jednostek, 157
  - usuwanie jednostek, 167
  - włączanie jednostek, 162
  - zależności, 158

- zgodność z System V, 173
- zmiennie, 163
- systemowe pliki crontab, 210
- systemy do tworzenia oprogramowania, 418

## Ś

- ścieżka
  - polecień, 49
  - stron podręcznika, 368
  - urządzeń sysfs, 79
  - wyszukiwania poleceń, 367
- śledzenie
  - działania programu, 228
  - procesów, 226
  - procesów systemd, 167
- średnie obciążenia, 235
- środowiska
  - interfejsów użytkownika, 381
  - użytkowników, 365

## T

- tabela
  - routingu jądra, 254
  - systemów plików, 116
- tablica
  - MBR, 103
  - partycji, 99, 102, 104
- terminale, 84
- testy
  - arytmetyczne, 333
  - ciągów znaków, 332
  - plików, 331
- tło wyświetlacza, 388
- translacja adresów sieciowych, 280
- trasy, 254
- tryb
  - informacyjny, 351
  - jądra, kernel mode, 25
  - pojedynczego użytkownika, 195
  - wypisywania zawartości archiwum, 68
- tryby
  - plików, 62
  - wyświetlania, 84
- tworzenie
  - dokładnych kopii, 348
  - dowiązań symbolicznych, 66
  - oprogramowania, 418
  - pakietów, 422
  - plików uruchomieniowych, 366

- plików urządzeń, 86
- systemu plików, 109
- zapór sieciowych, 287

typy

- baz danych, 437
- funkcji, 218
- jednostek, 157
- systemów plików, 108

## U

- UEFI, 137, 148
- UFS, Unix File System, 108
- ukośnik, 348
- uprawnienia, 62
- Upstart, 174
  - działanie procesu, 183
  - inicjowanie procesu, 175
  - konfiguracja, 178
  - poziomy uruchomienia, 185
  - protokoły, 184
  - przeglądanie zadań, 177
  - przejścia między stanami, 178
  - strofa expect, 182
  - śledzenie procesów, 182
  - zadania, 176
  - zgodność z System V, 185
- uruchamianie
  - na żądanie, 168
  - nowego procesu, 30
  - przestrzeni użytkownika, 153
  - sekwencyjne usług, 170
  - serwera, 358
  - serwera SSH, 302
  - skryptów, 190
  - systemu, 133
  - usług, 190
- urządzenia, 77
  - audio, 86
  - blokowe, 78
  - gniazdkowe, 79
  - optyczne, 98
  - potokowe, 79
  - SCSI, 97
  - wejściowe, 386
  - znakowe, 79
- usługa, 271, 295, 296
  - D-Bus, 389
    - monitorowanie komunikatów, 390
  - gniazdka sieciowego, 171
  - syslog, 201
  - tty1, 180

- ustanawianie połączeń TCP, 272
- ustawianie czasu, 206
- usuwanie
  - tras, 261
  - użytkowników, 357
- UTC, Universal Coordinated Time, 207
- UUID, Universally Unique Identifier, 112
- uwierzytelnianie użytkowników, 215
- użytkownik, 32, 197
  - root, 204
  - specjalny, 203
- używanie
  - końcowego ukośnika, 349
  - narzędzia iostat, 241
  - polecenia lsof, 228
  - polecenia uptime, 235
  - skryptów powłoki, 344
  - systemu GNU autoconf, 421

## W

- warstwa
  - aplikacji, 249
  - fizyczna, 249, 258
  - internetowa, 249, 250
  - transportowa, 249, 271
- warstwy
  - abstrakcji, 24
  - sieciowe, 249
- wątki, 231
- węzły
  - inode, 128
  - urządzeń, 78
- widok systemu plików, 126
- wieloetapowy program rozruchowy, 150
- wiersz poleceń, 50
- Wi-Fi, 291
- wirtualizacja, 437
- właściwości protokołu TCP, 273
- włączanie
  - jednostek, 162
  - plików, 343
- wrapper TCP, 306
- współużytkowanie
  - drukarek, 359
  - plików, 345, 353
- wstępny odczyt jądra, 103
- wydajność procesora, 239
- wyjście powłoki, 54

- wykorzystanie
  - procesów, 225
  - zasobów, 225
- wyłączanie systemu, 192
- wyrażenia warunkowe, 328
- wysokie obciążenia, 236
- wyszukiwanie otwartych plików, 226
- wyświetlanie
  - adresu IP, 252
  - wątków, 231
- wywołania systemowe, 29, 228

## X

- X Window System
  - menedżery wyświetlaczy, 383
  - przezroczystość sieci, 383

## Z

- zabezpieczenia, 214
  - sieci, 312, 314
  - sieci bezprzewodowych, 293
- zadania
  - jednorazowe, 211
  - programu rozruchowego, 138
  - wsadowe, 197
- zależności
  - systemd, 158
  - warunkowe, 160
- zależność
  - Conflicts, 159
  - Requires, 158
  - Requisite, 159
  - Wants, 159
- zapory sieciowe, 283
- zarządzanie
  - pamięcią, 28
  - plikami tymczasowymi, 337
  - procesami, 27
  - urządzeniami, 29
  - użytkownikami, 202
- zatrzymywanie usług, 190
- zdalne wywoływanie procedur, RPC, 311
- zdarzenia serwera X, 384
- złośliwe programy, 313
- zmienianie
  - hasła, 47, 357
  - powłoki, 47
  - urządzenia, 89

zmiennie, 163  
powłoki, 48  
specjalne, 324  
środowiskowe, 48, 424

znak  
krzyżyka, 320  
ukośnika, 348  
zapytania, 43  
znaki specjalne, 50  
zrównoleganie zasobów, 168



# PROGRAM PARTNERSKI

GRUPY WYDAWNICZEJ HELION

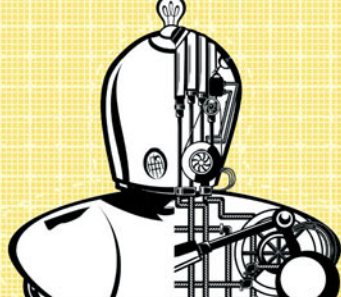


- 1. ZAREJESTRUJ SIĘ**
- 2. PREZENTUJ KSIĄŻKI**
- 3. ZBIERAJ PROWIZJĘ**

Zmień swoją stronę WWW  
w działający bankomat!

**Dowiedz się więcej i dołącz już dzisiaj!**

<http://program-partnerski.helion.pl>



## WYKORZYSTAJ W PEŁNI SWÓJ KOMPUTER DZIĘKI SYSTEMOWI LINUX!

Linux to system operacyjny będący solą w oku giganta z Redmond. Możliwości, wydajność, elastyczność i bezpieczeństwo Linuksa zostały docenione przez zaawansowanych użytkowników. Natomiast różnorodność powłok graficznych oraz łatwość ich użytkowania przypadły do gustu osobom rozpoczynającym przygodę z tym systemem. Jeżeli chcesz mieć pełną kontrolę nad swoim komputerem i wykorzystać potencjał Linuksa, potrzebujesz tego podręcznika.

Dzięki niemu poznasz wszystkie elementy systemu — począwszy od czeluści jądra, a skończywszy na powłokach użytkownika. W kolejnych rozdziałach znajdziesz informacje na temat podstawowych poleceń, pozwalających na wydajną nawigację w systemie plików oraz wyświetlanie zawartości tych plików, a także na temat narzędzi do kompresji oraz sposobów zarządzania procesami. Ponadto zobaczysz, jak uzyskać dostęp do urządzeń i zarządzać twardymi dyskami. Na tym etapie będziesz już gotów porównać systemy plików ext3 i ext4 oraz poznać kierunki rozwoju systemów plików. Zdobędziesz także ciekawe wiadomości dotyczące startu systemu, konfiguracji harmonogramu zadań, dostępu do sieci oraz tworzenia skryptów ułatwiających pracę. Ta książka będzie pasjonującą lekturą dla każdego użytkownika chcącego zgłębić tajniki pracy z systemem Linux.

### Dzięki tej książce:

- poznasz podstawowe polecenia dostępne w systemie Linux
- nauczysz się zarządzać dostępnymi zasobami i procesami
- opanujesz narzędzia programistyczne
- zaznajomisz się z podstawami tworzenia skryptów powłoki
- poznasz system Linux od podszewki

**Brian Ward** — przygodę z systemem Linux rozpoczął w 1993 roku na komputerze z procesorem 386. Posiada tytuł doktora informatyki, uzyskany na Uniwersytecie Chicagowskim. Jest autorem książek poświęconych Linuksowi. Aktualnie mieszka w San Francisco i pracuje jako konsultant oraz trener.

**Helion**

37365 numer katalogowy  
księgarnia internetowa

<http://helion.pl>

zamówienia telefoniczne

0 801 339900

0 601 339900

Informatyka w najlepszym wydaniu

Sprawdź najnowsze promocje:  
• <http://helion.pl/promocje>  
Książki najchętniej czytane:  
• <http://helion.pl/bestsellery>  
Zamów informacje o nowościach:  
• <http://helion.pl/nowosci>

Helion SA  
ul. Kościuszki 1c, 44-100 Gliwice  
tel.: 32 230 98 63  
e-mail: [helion@helion.pl](mailto:helion@helion.pl)  
<http://helion.pl>

sięgnij po WIĘCEJ



KOD KORZYŚCI

ISBN 978-83-283-0980-7



9 788328 309807

cena: 69,00 zł

