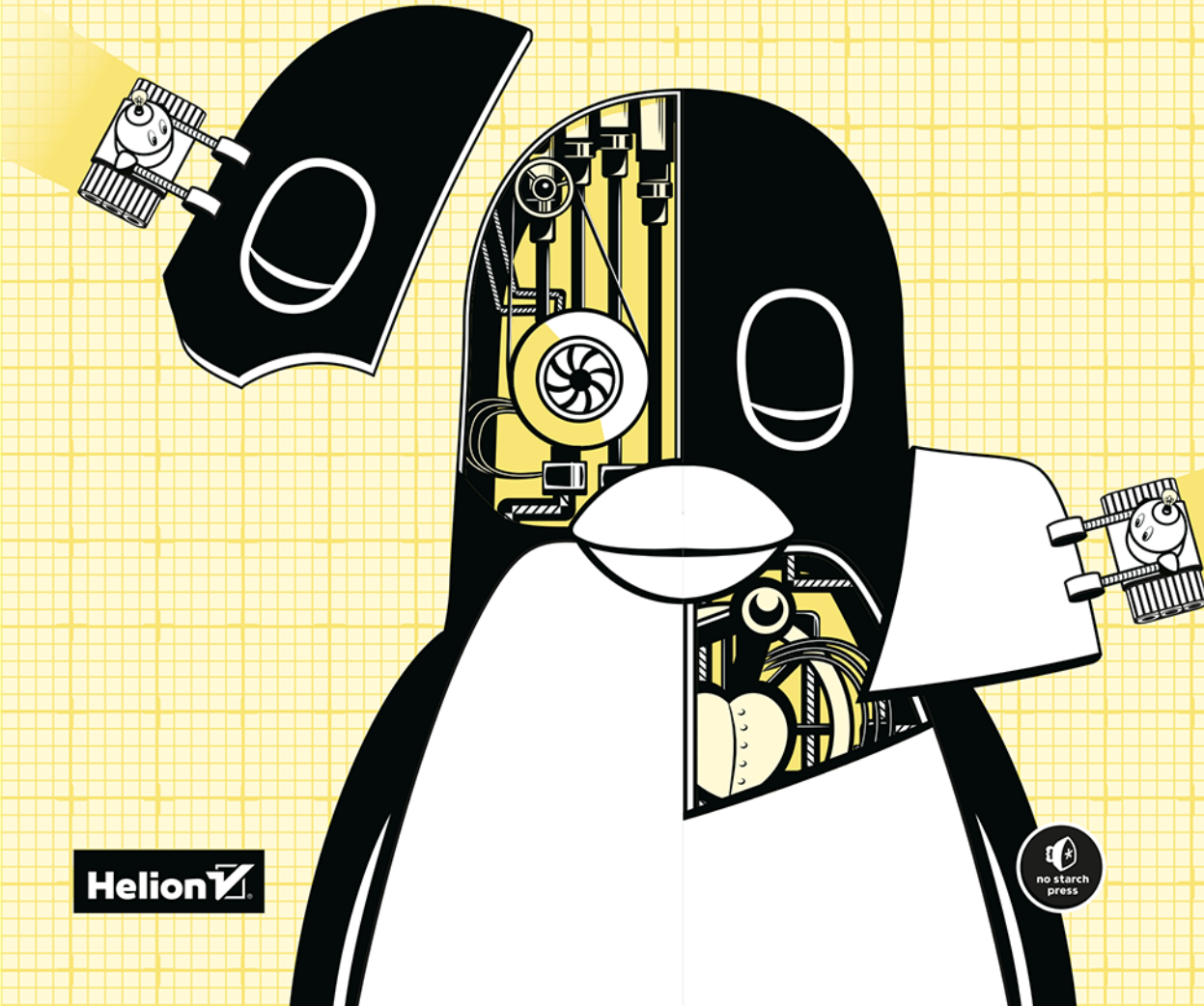
Wydanie III

JAK DZIAŁA LINUX

PODRĘCZNIK ADMINISTRATORA

BRIAN WARD



Helion

no starch press

Tytuł oryginału: How Linux Works, 3rd Edition What Every Superuser Should Know

Tłumaczenie: Piotr Pilch na podstawie „Jak działa Linux" w przekładzie Wojciecha Mocha

ISBN: 978-83-283-8863-5

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *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/jakli3
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

PODZIĘKOWANIA **18**

WSTĘP **19**

1
INFORMACJE OGÓLNE **23**
1.1. Poziomy i warstwy abstrakcji w systemie Linux 24
1.2. Sprzęt: pamięć operacyjna 26
1.3. Jądro systemu 27
1.3.1. Zarządzanie procesami 27
1.3.2. Zarządzanie pamięcią 29
1.3.3. Sterowniki urządzeń i zarządzanie urządzeniami 29
1.3.4. Wywołania systemowe 30
1.4. Przestrzeń użytkownika 31
1.5. Użytkownicy 32
1.6. Spojrzenie w przyszłość 33

2
PODSTAWOWE POLECENIA I HIERARCHIA KATALOGÓW **34**
2.1. Powłoka Bourne'a: /bin/sh 35
2.2. Korzystanie z powłoki 36
2.2.1. Okno powłoki 36
2.2.2. Polecenie cat 37
2.2.3. Standardowe wejście i wyjście 37
2.3. Podstawowe polecenia 38
2.3.1. Polecenie ls 38
2.3.2. Polecenie cp 39
2.3.3. Polecenie mv 39
2.3.4. Polecenie touch 40
2.3.5. Polecenie rm 40
2.3.6. Polecenie echo 40
2.4. Polecenia działające na katalogach 41
2.4.1. Polecenie cd 41
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. Polecenie 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. Zmienianie hasła i powłoki ... 47
2.7. Pliki z kropką ... 48
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 ... 52
2.13. Uzyskiwanie pomocy ... 53
2.14. Wejście i wyjście powłoki ... 55
2.14.1. Standardowy strumień błędów ... 56
2.14.2. Przekierowywanie standardowego wejścia ... 56
2.15. Prawidłowe odczytywanie komunikatów o błędach ... 57
2.15.1. Anatomia uniksowych komunikatów o błędach ... 57
2.15.2. Typowe błędy ... 58
2.16. Przeglądanie procesów i manipulowanie nimi ... 60
2.16.1. Opcje polecenia ps ... 61
2.16.2. Kończenie działania procesów ... 61
2.16.3. Kontrola zadań ... 62
2.16.4. Procesy działające w tle ... 63
2.17. Tryby plików i uprawnienia ... 64
2.17.1. Modyfikowanie uprawnień ... 65
2.17.2. Dowiązania symboliczne ... 67
2.18. Archiwizowanie i kompresowanie plików ... 69
2.18.1. Program gzip ... 69
2.18.2. Program tar ... 69
2.18.3. Archiwa skompresowane (.tar.gz) ... 71
2.18.4. Program zcat ... 71
2.18.5. Inne narzędzia kompresujące ... 72
2.19. Hierarchia katalogów ... 72
2.19.1. Pozostałe katalogi główne ... 75
2.19.2. Katalog /usr ... 75
2.19.3. Umiejscowienie jądra systemu ... 75

2.20. Uruchamianie poleceń przez superużytkownika ... 76
2.20.1. Polecenie sudo ... 76
2.20.2. Plik /etc/sudoers ... 76
2.20.3. Dzienniki programu sudo ... 77
2.21. Podsumowanie ... 78

3
URZĄDZENIA ... 79
3.1. Pliki urządzeń ... 80
3.2. Ścieżka urządzeń sysfs ... 81
3.3. Polecenie dd i urządzenia ... 83
3.4. Podsumowanie nazewnictwa urządzeń ... 84
3.4.1. Dyski twarde: /dev/sd* ... 84
3.4.2. Dyski wirtualne: /dev/xvd*, /dev/vd* ... 86
3.4.3. Urządzenia pamięci nieulotnej: /dev/nvme* ... 86
3.4.4. Mapowanie urządzeń: /dev/dm-*, /dev/mapper/* ... 86
3.4.5. Napędy CD i DVD: /dev/sr* ... 86
3.4.6. Dyski twarde PATA: /dev/hd* ... 86
3.4.7. Terminale: /dev/tty*, /dev/pts/* i /dev/tty ... 87
3.4.8. Porty szeregowe: /dev/ttyS*, /dev/ttyUSB*, /dev/ttyACM* ... 88
3.4.9. Porty równoległe: /dev/lp0 i /dev/lp1 ... 88
3.4.10. Urządzenia audio: /dev/dsp, /dev/audio, /dev/snd/* i inne ... 89
3.4.11. Tworzenie plików urządzeń ... 89
3.5. System udev ... 90
3.5.1. System plików devtmpfs ... 91
3.5.2. Konfiguracja i działanie procesu udevd ... 92
3.5.3. Program udevadm ... 94
3.5.4. Monitorowanie urządzeń ... 95
3.6. Szczegóły: SCSI i jądro Linuksa ... 96
3.6.1. Pamięci masowe USB i protokół SCSI ... 100
3.6.2. SCSI i ATA ... 100
3.6.3. Ogólne urządzenia SCSI ... 101
3.6.4. Wiele metod dostępu do jednego urządzenia ... 102

4
DYSKI I SYSTEMY PLIKÓW ... 104
4.1. Partycjonowanie urządzeń dyskowych ... 107
4.1.1. Przeglądanie tablicy partycji ... 108
4.1.2. Modyfikowanie tablicy partycji ... 111
4.1.3. Tworzenie tablicy partycji ... 112
4.1.4. Geometria dysku i partycji ... 114
4.1.5. Odczyt z dysków SSD ... 116
4.2. Systemy plików ... 117
4.2.1. Typy systemów plików ... 118
4.2.2. Tworzenie systemu plików ... 119
4.2.3. Montowanie systemu plików ... 120

4.2.4. Identyfikator UUID systemu plików ... 122
4.2.5. Buforowanie dysku i systemu plików ... 123
4.2.6. Opcje montowania systemów plików ... 124
4.2.7. Ponowne montowanie systemu plików ... 126
4.2.8. Tabela systemów plików /etc/fstab ... 126
4.2.9. Rozwiązania konkurencyjne dla pliku /etc/fstab ... 128
4.2.10. Pojemność systemu plików ... 129
4.2.11. Sprawdzanie i naprawianie systemów plików ... 130
4.2.12. Systemy plików o specjalnym znaczeniu ... 133
4.3. Przestrzeń wymiany ... 134
4.3.1. Wykorzystywanie partycji jako przestrzeni wymiany ... 135
4.3.2. Wykorzystywanie pliku jako przestrzeni wymiany ... 135
4.3.3. Jak wielkiej przestrzeni wymiany potrzebuję? ... 136
4.4. Menedżer LVM ... 137
4.4.1. Obsługa menedżera LVM ... 138
4.4.2. Implementacja menedżera LVM ... 151
4.5. Spojrzenie w przyszłość: dyski i przestrzeń użytkownika ... 155
4.6. Tradycyjny system plików ... 155
4.6.1. Szczegóły węzłów inode i licznik dowiązań ... 158
4.6.2. Alokowanie bloków ... 159
4.6.3. Praca z systemami plików w przestrzeni użytkownika ... 160

5
JAK URUCHAMIA SIĘ LINUX? ... 162
5.1. Komunikaty rozruchowe ... 163
5.2. Inicjowanie jądra i opcje rozruchu ... 164
5.3. Parametry jądra ... 165
5.4. Programy rozruchowe ... 167
5.4.1. Zadania programu rozruchowego ... 168
5.4.2. Przegląd programów rozruchowych ... 169
5.5. Wprowadzenie do programu GRUB ... 169
5.5.1. Przeszukiwanie urządzeń i partycji za pomocą wiersza poleceń programu GRUB ... 172
5.5.2. Konfigurowanie programu GRUB ... 175
5.5.3. Instalowanie programu GRUB ... 177
5.6. Problemy z bezpiecznym rozruchem UEFI ... 179
5.7. Ładowanie innych systemów operacyjnych ... 180
5.8. Szczegóły programu rozruchowego ... 181
5.8.1. Rozruch MBR ... 181
5.8.2. Rozruch UEFI ... 182
5.8.3. Jak działa GRUB? ... 183

6
URUCHAMIANIE PRZESTRZENI UŻYTKOWNIKA ... 185
6.1. Wprowadzenie do procesu init ... 186
6.2. Identyfikowanie rodzaju procesu init ... 187

6.3. systemd ... 187
6.3.1. Jednostki i typy jednostek ... 188
6.3.2. Rozruch i diagramy zależności jednostek ... 189
6.3.3. Konfiguracja demona systemd ... 190
6.3.4. Praca z systemd ... 192
6.3.5. Śledzenie i synchronizacja procesów systemd ... 197
6.3.6. Zależności demona systemd ... 198
6.3.7. Uruchamianie na żądanie i zrównoleglanie zasobów przez demon systemd ... 202
6.3.8. Składniki pomocnicze demona systemd ... 207
6.4. Poziomy uruchomienia System V ... 208
6.5. Proces init w stylu System V ... 209
6.5.1. Proces init w stylu System V: sekwencja poleceń rozruchowych ... 210
6.5.2. Farma dowiązań procesu init w stylu System V ... 211
6.5.3. run-parts ... 213
6.5.4. Sterowanie procesem init w stylu System V ... 213
6.5.5. Zgodność demona systemd z System V ... 214
6.6. Wyłączanie systemu ... 214
6.7. Początkowy system plików w pamięci RAM ... 216
6.8. Rozruch awaryjny i tryb pojedynczego użytkownika ... 218
6.9. Spojrzenie w przyszłość ... 219

7
KONFIGURACJA SYSTEMU: REJESTROWANIE, CZAS SYSTEMOWY, ZADANIA WSADOWE I UŻYTKOWNICY ... 220
7.1. Rejestrowanie dzienników systemowych ... 221
7.1.1. Sprawdzanie konfiguracji dzienników ... 222
7.1.2. Wyszukiwanie i monitorowanie dzienników ... 222
7.1.3. Rotacja pliku dziennika ... 227
7.1.4. Utrzymywanie dziennika ... 228
7.1.5. Proces rejestrowania w systemie ... 228
7.2. Struktura katalogu /etc ... 231
7.3. Pliki związane z zarządzaniem użytkownikami ... 232
7.3.1. Plik /etc/passwd ... 232
7.3.2. Użytkownicy specjalni ... 234
7.3.3. Plik /etc/shadow ... 234
7.3.4. Manipulowanie użytkownikami i hasłami ... 234
7.3.5. Praca z grupami ... 235
7.4. Programy getty i login ... 236
7.5. Ustawianie czasu ... 237
7.5.1. Reprezentacja czasu jądra i strefy czasowe ... 238
7.5.2. Czas sieciowy ... 239
7.6. Planowanie powtarzalnych zadań w programie cron ... 239
7.6.1. Instalowanie plików crontab ... 241
7.6.2. Systemowe pliki crontab ... 241

7.6.3. Jednostki licznika czasu ... 242
7.6.4. Narzędzie cron a jednostki licznika czasu ... 244
7.7. Planowanie jednorazowych zadań w programie at ... 244
7.7.1. Odpowiedniki jednostki licznika czasu ... 245
7.8. Jednostki licznika czasu działające z uprawnieniami zwykłych użytkowników ... 246
7.9. Zagadnienia dotyczące dostępu uzyskiwanego przez użytkowników ... 246
7.9.1. Identyfikatory użytkowników i przełączanie ich ... 246
7.9.2. Prawo właściciela procesu, efektywny identyfikator użytkownika, rzeczywisty identyfikator użytkownika i zapisany identyfikator użytkownika ... 247
7.9.3. Identyfikowanie, uwierzytelnianie i autoryzowanie użytkowników ... 250
7.9.4. Użycie bibliotek do uzyskiwania informacji o użytkownikach ... 251
7.10. System PAM ... 252
7.10.1. Konfiguracja systemu PAM ... 252
7.10.2. Wskazówki dotyczące składni konfiguracji systemu PAM ... 256
7.10.3. System PAM i hasła ... 257
7.11. Spojrzenie w przyszłość ... 258

8
WYKORZYSTANIE PROCESÓW I ZASOBÓW ... 259
8.1. Śledzenie procesów ... 260
8.2. Wyszukiwanie otwartych plików programem lsof ... 261
8.2.1. Analizowanie danych wyjściowych polecenia lsof ... 261
8.2.2. Użycie polecenia lsof ... 262
8.3. Śledzenie działania programu i wywołań systemowych ... 263
8.3.1. Polecenie strace ... 263
8.3.2. Polecenie ltrace ... 265
8.4. Wątki ... 265
8.4.1. Procesy jednowątkowe i wielowątkowe ... 266
8.4.2. Wyświetlanie wątków ... 266
8.5. Wprowadzenie do monitorowania zasobów ... 268
8.5.1. Pomiar czasu procesora ... 268
8.5.2. Nadawanie procesom priorytetów ... 269
8.5.3. Pomiar wydajności procesora na podstawie średnich obciążeń ... 270
8.5.4. Monitorowanie statusu pamięci ... 272
8.5.5. Monitorowanie wydajności procesora i pamięci za pomocą polecenia vmstat ... 274
8.5.6. Monitorowanie operacji wejścia-wyjścia ... 276
8.5.7. Monitorowanie poszczególnych procesów za pomocą narzędzia pidstat ... 279
8.6. Grupy kontrolne (cgroup) ... 280
8.6.1. Rozróżnianie wersji grup kontrolnych ... 280
8.6.2. Wyświetlanie grup kontrolnych ... 283
8.6.3. Tworzenie i modyfikowanie grup kontrolnych ... 284
8.6.4. Wyświetlanie informacji o wykorzystaniu zasobów ... 285
8.7. Dodatkowe zagadnienia ... 285

9
SIEĆ I JEJ KONFIGURACJA 287
9.1. Podstawy dotyczące sieci 288
9.2. Pakiety 288
9.3. Warstwy sieciowe 289
9.4. Warstwa internetowa 291
9.4.1. Wyświetlanie adresów IP 292
9.4.2. Podsieci 293
9.4.3. Typowe maski podsieci i notacja CIDR 294
9.5. Trasy i tabela routingu jądra 295
9.6. Brama domyślna 296
9.7. Adresy i sieci IPv6 296
9.7.1. Wyświetlanie w systemie konfiguracji protokołu IPv6 298
9.7.2. Konfigurowanie sieci z dwoma stosami 299
9.8. Podstawowe narzędzia protokołu ICMP i systemu DNS 299
9.8.1. ping 299
9.8.2. DNS i host 300
9.9. Warstwa fizyczna i Ethernet 301
9.10. Interfejsy sieciowe jądra 302
9.11. Wprowadzenie do konfiguracji interfejsów sieciowych 303
9.11.1. Ręczne konfigurowanie interfejsów 304
9.11.2. Ręczne dodawanie i usuwanie tras 304
9.12. Konfiguracja sieci aktywowana podczas rozruchu 305
9.13. Problemy z ręczną i aktywowaną podczas rozruchu konfiguracją sieci 306
9.14. Menedżery konfiguracji sieciowych 307
9.14.1. Działanie narzędzia NetworkManager 307
9.14.2. Interakcja z narzędziem NetworkManager 308
9.14.3. Konfiguracja narzędzia NetworkManager 309
9.15. Rozpoznawanie nazw hostów 311
9.15.1. Plik /etc/hosts 312
9.15.2. Plik resolv.conf 312
9.15.3. Buforowanie i system DNS bez konfiguracji 313
9.15.4. Plik /etc/nsswitch.conf 314
9.16. Host lokalny 315
9.17. Warstwa transportowa: protokoły TCP i UDP oraz usługi 316
9.17.1. Porty TCP i połączenia 316
9.17.2. Protokół UDP 319
9.18. Ponowna analiza prostej sieci lokalnej 321
9.19. Protokół DHCP 322
9.19.1. Klient DHCP w systemie Linux 322
9.19.2. Serwery DHCP w systemie Linux 323
9.20. Automatyczna konfiguracja sieci z protokołem IPv6 323
9.21. Konfigurowanie systemu Linux jako routera 324
9.22. Sieci prywatne (IPv4) 326

9.23. Translacja adresów sieciowych (maskarada IP) ... 327
9.24. Routery i system Linux ... 328
9.25. Zapory sieciowe ... 329
9.25.1. Podstawy dotyczące linuksowych zapór sieciowych ... 330
9.25.2. Konfigurowanie reguł zapory sieciowej ... 331
9.25.3. Strategie tworzenia zapór sieciowych ... 334
9.26. Ethernet, IP, ARP i NDP ... 336
9.27. Ethernet bezprzewodowy ... 338
9.27.1. iw ... 339
9.27.2. Zabezpieczenia sieci bezprzewodowych ... 340
9.28. Podsumowanie ... 340

10
USŁUGI I APLIKACJE SIECIOWE ... 342
10.1. Podstawy usług ... 342
10.2. Dokładniejsza analiza ... 344
10.3. Serwery sieciowe ... 346
10.3.1. Secure Shell (SSH) ... 346
10.3.2. Serwer SSHD ... 348
10.3.3. fail2ban ... 351
10.3.4. Klient SSH ... 351
10.4. Serwery połączeń sieciowych zastąpione przez demon systemd: inetd i xinetd ... 353
10.5. Narzędzia diagnostyczne ... 354
10.5.1. lsof ... 355
10.5.2. tcpdump ... 357
10.5.3. netcat ... 359
10.5.4. Skanowanie portów ... 359
10.6. Zdalne wywoływanie procedur (RPC) ... 360
10.7. Zabezpieczenie sieci ... 361
10.7.1. Typowe słabości ... 363
10.7.2. Źródła danych o zabezpieczeniach ... 364
10.8. Spojrzenie w przyszłość ... 364
10.9. Gniazda sieciowe ... 365
10.10. Gniazda domenowe systemu Unix ... 366

11
WPROWADZENIE DO SKRYPTÓW POWŁOKI ... 368
11.1. Podstawy skryptów powłoki ... 368
11.1.1. Ograniczenia skryptów powłoki ... 370
11.2. Cudzysłowy i literały ... 370
11.2.1. Literały ... 371
11.2.2. Pojedyncze cudzysłowy ... 372
11.2.3. Podwójne cudzysłowy ... 372
11.2.4. Literały w postaci znaku pojedynczego cudzysłowu ... 373

11.3. Zmienne specjalne ... 373
11.3.1. Pojedyncze argumenty: $1, $2... ... 374
11.3.2. Liczba argumentów: $# ... 375
11.3.3. Wszystkie argumenty: $@ ... 375
11.3.4. Nazwa skryptu: $0 ... 375
11.3.5. Identyfikator procesu: $$ ... 376
11.3.6. Kod wyjścia: $? ... 376
11.4. Kody wyjścia ... 376
11.5. Wyrażenia warunkowe ... 377
11.5.1. Obsługa list pustych parametrów ... 378
11.5.2. Użycie innych poleceń do testów ... 379
11.5.3. Słowo kluczowe elif ... 379
11.5.4. Konstrukcje logiczne ... 379
11.5.5. Sprawdzanie warunków ... 380
11.5.6. Instrukcja case ... 383
11.6. Pętle ... 384
11.6.1. Pętle for ... 384
11.6.2. Pętle while ... 385
11.7. Podmiana poleceń ... 386
11.8. Zarządzanie plikami tymczasowymi ... 387
11.9. Dokumenty miejscowe ... 388
11.10. Ważne narzędzia skryptów powłoki ... 389
11.10.1. Polecenie basename ... 389
11.10.2. Polecenie awk ... 390
11.10.3. Polecenie sed ... 390
11.10.4. Polecenie xargs ... 391
11.10.5. Polecenie expr ... 392
11.10.6. Polecenie exec ... 392
11.11. Podpowłoki ... 393
11.12. Włączanie do skryptów innych plików ... 394
11.13. Pobieranie danych od użytkowników ... 394
11.14. Kiedy (nie)używać skryptów powłoki? ... 394

12
UDOSTĘPNIANIE I PRZESYŁANIE PLIKÓW W SIECI ... 396
12.1. Szybkie wykonywanie kopii ... 397
12.2. rsync ... 397
12.2.1. Podstawy dotyczące narzędzia rsync ... 398
12.2.2. Tworzenie dokładnych kopii struktury katalogów ... 400
12.2.3. Jak używać końcowego ukośnika? ... 400
12.2.4. Pomijanie plików i katalogów ... 402
12.2.5. Sprawdzanie transferów, dodawanie sum kontrolnych i użycie trybu informacyjnego ... 402
12.2.6. Kompresja danych ... 404
12.2.7. Ograniczanie przepustowości ... 404

12.2.8. Przesyłanie plików do naszego komputera ... 404
12.2.9. Więcej informacji o programie rsync ... 405
12.3. Wprowadzenie do współużytkowania plików ... 405
12.3.1. Współużytkowanie plików i jego wydajność ... 405
12.3.2. Bezpieczeństwo współużytkowanych plików ... 406
12.4. Współużytkowanie plików za pomocą pakietu Samba ... 407
12.4.1. Konfigurowanie serwera ... 407
12.4.2. Kontrola dostępu do serwera ... 408
12.4.3. Hasła ... 409
12.4.4. Ręczne uruchamianie serwera ... 411
12.4.5. Diagnostyka i pliki dziennika ... 411
12.4.6. Konfigurowanie udziału plikowego ... 411
12.4.7. Katalogi domowe ... 412
12.4.8. Współużytkowanie drukarek ... 412
12.4.9. Korzystanie z klientów Samby ... 413
12.5. SSHFS ... 415
12.6. NFS ... 416
12.7. Magazynowanie danych w chmurze ... 417
12.8. Stan rozwoju technologii współużytkowania plików w sieci ... 418

13
ŚRODOWISKA UŻYTKOWNIKÓW ... 420
13.1. Wytyczne dotyczące tworzenia plików uruchomieniowych ... 421
13.2. Kiedy należy modyfikować pliki uruchomieniowe? ... 421
13.3. Elementy plików uruchamiających powłokę ... 422
13.3.1. Ścieżka wyszukiwania poleceń ... 422
13.3.2. Ścieżka stron podręcznika man ... 423
13.3.3. Symbol zachęty ... 423
13.3.4. Aliasy ... 424
13.3.5. Maska uprawnień ... 425
13.4. Kolejność plików uruchomieniowych i przykłady ... 426
13.4.1. Powłoka bash ... 426
13.4.2. Powłoka tcsh ... 429
13.5. Domyślne ustawienia użytkownika ... 430
13.5.1. Domyślne ustawienia powłoki ... 430
13.5.2. Edytor ... 431
13.5.3. Program stronicujący ... 431
13.6. Pułapki w plikach uruchomieniowych ... 432
13.7. Dalsze informacje ... 432

14
OGÓLNY PRZEGLĄD INTERFEJSÓW UŻYTKOWNIKA SYSTEMU LINUX I OBSŁUGI DRUKOWANIA ... 433
14.1. Komponenty interfejsów użytkownika ... 434
14.1.1. Bufory ramki ... 434
14.1.2. X Window System ... 435
14.1.3. Wayland ... 435
14.1.4. Menedżery okien ... 436
14.1.5. Pakiety narzędziowe ... 436
14.1.6. Środowiska interfejsów użytkownika ... 437
14.1.7. Aplikacje ... 437
14.2. Czy używasz systemu Wayland czy X? ... 437
14.3. Coś więcej o protokole Wayland ... 438
14.3.1. Menedżer kompozycji ... 438
14.3.2. libinput ... 439
14.3.3. Zgodność protokołu Wayland z systemem X ... 441
14.4. X Window System ... 442
14.4.1. Menedżery wyświetlaczy ... 443
14.4.2. Przezroczystość sieci ... 443
14.4.3. Eksplorowanie klientów serwera X ... 444
14.4.4. Zdarzenia serwera X ... 444
14.4.5. Ustawianie preferencji i dane wejściowe serwera X ... 445
14.5. Usługa D-Bus ... 448
14.5.1. Instancja sesji i instancja systemowa ... 448
14.5.2. Monitorowanie komunikatów usługi D-Bus ... 449
14.6. Drukowanie ... 450
14.6.1. CUPS ... 450
14.6.2. Konwersja formatów i filtry wydruku ... 451
14.7. Inne zagadnienia związane z interfejsami użytkownika ... 451

15
NARZĘDZIA PROGRAMISTYCZNE ... 453
15.1. Kompilator języka C ... 454
15.1.1. Kompilowanie wielu plików źródłowych ... 455
15.1.2. Konsolidacja z bibliotekami ... 456
15.1.3. Biblioteki współużytkowane ... 458
15.1.4. Pliki i katalogi nagłówkowe ... 462
15.2. Narzędzie make ... 465
15.2.1. Przykładowy plik Makefile ... 466
15.2.2. Wbudowane reguły ... 467
15.2.3. Końcowe budowanie programu ... 467
15.2.4. Aktualizowanie zależności ... 468
15.2.5. Argumenty i opcje wiersza poleceń ... 468
15.2.6. Standardowe makra i zmienne ... 469
15.2.7. Typowe cele kompilacji ... 471
15.2.8. Organizowanie pliku Makefile ... 471

15.3. Lex i Yacc ... 473
15.4. Języki skryptowe ... 473
15.4.1. Python ... 474
15.4.2. Perl ... 475
15.4.3. Pozostałe języki skryptowe ... 475
15.5. Java ... 476
15.6. Spojrzenie w przyszłość: kompilowanie pakietów ... 477

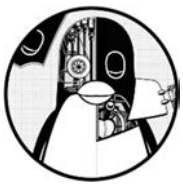
16
WPROWADZENIE DO KOMPILOWANIA OPROGRAMOWANIA Z KODU ŹRÓDŁOWEGO C ... 478
16.1. Systemy do tworzenia oprogramowania ... 479
16.2. Rozpakowywanie pakietów kodu źródłowego języka C ... 480
16.3. GNU autoconf ... 481
16.3.1. Przykład użycia systemu GNU autoconf ... 482
16.3.2. Instalacja za pomocą narzędzia do tworzenia pakietów ... 483
16.3.3. Opcje skryptu configure ... 484
16.3.4. Zmienne środowiskowe ... 485
16.3.5. Cele tworzone przez system autoconf ... 486
16.3.6. Pliki dziennika systemu autoconf ... 487
16.3.7. pkg-config ... 487
16.4. Praktyki instalacyjne ... 489
16.4.1. Gdzie instalować? ... 490
16.5. Stosowanie poprawek ... 490
16.6. Rozwiązywanie problemów z kompilowaniem i instalowaniem ... 492
16.6.1. Częste błędy ... 493
16.7. Spojrzenie w przyszłość ... 495

17
WIRTUALIZACJA ... 496
17.1. Maszyny wirtualne ... 497
17.1.1. Hipernadzorcy ... 497
17.1.2. Sprzęt maszyny wirtualnej ... 498
17.1.3. Typowe zastosowania maszyn wirtualnych ... 500
17.1.4. Mankamenty maszyn wirtualnych ... 500
17.2. Kontenery ... 501
17.2.1. Docker, Podman i przywileje ... 502
17.2.2. Przykład użycia Dockera ... 503
17.2.3. LXC ... 512
17.2.4. Kubernetes ... 513
17.2.5. Pułapki związane z kontenerami ... 513
17.3. Wirtualizacja oparta na środowisku uruchomieniowym ... 516

BIBLIOGRAFIA ... 518

1

Informacje ogólne



NA PIERWSZY RZUT OKA NOWOCZESNY SYSTEM OPERACYJNY, TAKI JAK LINUX, JEST BARDZO SKOMPLIKOWANY, PONIEWAŻ SKŁADA SIĘ Z ZAWROTNEJ LICZBY DZIAŁAJĄCYCH JEDNOCZEŚNIE i komunikujących się elementów. Przykładowo serwer WWW może wymieniać dane z serwerem bazy danych, który z kolei może wykorzystywać współdzieloną bibliotekę używaną też przez wiele innych programów. Ale jak to wszystko naprawdę działa i jak możesz odnaleźć w tym sens?

Najskuteczniejszą metodą zrozumienia mechanizmów rządzących systemem operacyjnym jest wykorzystanie *abstrakcji*, czyli ładnej nazwy dla prostego ignorowania większości szczegółów tworzących element, który próbujesz zrozumieć. Abstrakcja pozwala skoncentrować się na jego podstawowym przeznaczeniu i sposobie działania. Jadąc samochodem, zazwyczaj nie musisz zastanawiać się nad takimi detalami jak śruby mocujące silnik pod maską albo ludzie zajmujący się konserwacją drogi, po której się poruszasz. Musisz wiedzieć tylko, co to urządzenie robi (transportuje Cię w jakieś miejsce) oraz jak z niego skorzystać (jak otworzyć drzwi albo zapiąć pasy bezpieczeństwa).

Taki poziom abstrakcji może się sprawdzić, jeśli jesteś w samochodzie tylko pasażerem. Jeżeli jednak masz również zamiar kierować samochodem, musisz wiedzieć już znacznie więcej i dokonać podziału abstrakcji na kilka części. Niezbędne jest poszerzenie wiedzy w trzech obszarach: informacje o samym pojeździe (np. jego wielkość i osiągi), sposób posługiwania się elementami sterującymi (takimi jak kierownica lub pedał gazu) oraz oznakowanie drogi.

Abstrakcja może okazać się naprawdę pomocna, gdy próbujesz znaleźć i usunąć problem. Załóżmy na przykład, że droga, po której jedziemy, jest bardzo wyboista. Możesz szybko ocenić trzy właśnie wspomniane podstawowe abstrakcje związane z samochodem, aby ustalić źródło problemu. Dość proste powinno być wyeliminowanie pierwszych dwóch abstrakcji (posiadany samochód i sposób jego prowadzenia). Jeśli żadna z nich nie jest kłopotliwa, możesz zawęzić problem do stanu samej drogi. Prawdopodobnie okaże się ona wyboista. W razie potrzeby możesz zagłębić się w kwestię abstrakcji drogi i ustalić, dlaczego jej stan się pogorszył, a jeśli droga jest nowa, dlaczego budujący ją robotnicy wykonali kiepską pracę.

Twórcy oprogramowania wykorzystują abstrakcje jako narzędzia do tworzenia systemu operacyjnego oraz działających w nim aplikacji. Istnieje wiele pojęć pozwalających na opisanie takich abstrakcyjnych podziałów w oprogramowaniu komputerowym; są to *podsystemy*, *moduły* lub *pakiety*. W tym rozdziale używał będę jednak pojęcia *komponent*, ponieważ wydaje mi się najprostsze. Tworząc komponent programowy, programiści zazwyczaj nie myślą o wewnętrznych strukturach innych komponentów, ale za to rozmyślają o tym, do czego mogą używać innych komponentów (dzięki temu nie muszą tworzyć żadnego dodatkowego i zbędnego oprogramowania) oraz jak można z nich skorzystać.

W tym rozdziale zaprezentuję ogólny przegląd komponentów, z których składa się system operacyjny Linux. Co prawda, każdy z nich składa się z wielu drobnych szczegółów technicznych, ale na razie będziemy te szczegóły ignorować i skoncentrujemy się na roli poszczególnych komponentów w ramach całego systemu. Szczegółami zajmiemy się w kolejnych rozdziałach.

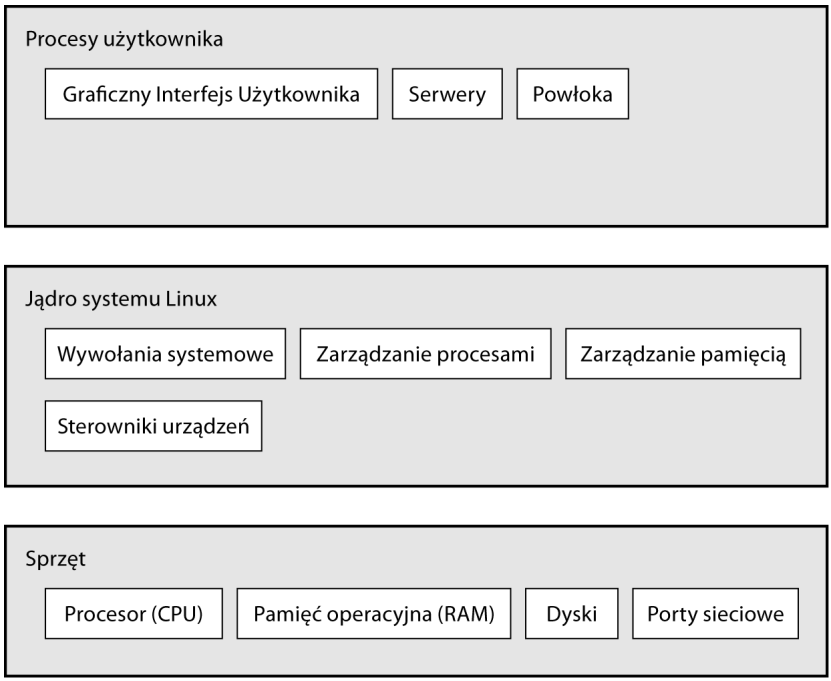
1.1. Poziomy i warstwy abstrakcji w systemie Linux

Wykorzystanie abstrakcji do podziału systemów komputerowych na komponenty sprawia, że całość jest łatwiejsza do zrozumienia, ale bez odpowiedniej organizacji na nic się nie przyda. Poszczególne komponenty trzeba pogrupować na warstwy lub poziomy. *Warstwa* lub *poziom* to sposób klasyfikacji (albo grupowania) komponentów, zależny od tego, gdzie dany komponent znajduje się pomiędzy sprzętem a użytkownikiem. Przeglądarki internetowe albo gry znajdują się w najwyższej warstwie, natomiast najniższą warstwę stanowi pamięć zamontowana fizycznie w komputerze, czyli same zera i jedynki. System operacyjny zajmuje większość warstw znajdujących się pomiędzy tymi dwoma.

System Linux składa się z trzech głównych poziomów. Na rysunku 1.1 przedstawiłem te poziomy wraz z wybranymi komponentami, które się w nich znajdują. *Sprzęt* jest podstawą całości. Do sprzętu można zaliczyć zarówno pamięć, jak i jeden lub kilka procesorów (CPU) wykonujących różne obliczenia oraz odczytujących i zapisujących dane w pamięci. Takie urządzenia jak dyski i karty sieciowe również zaliczane są do sprzętu.

Piętro wyżej znajduje się *jądro* (ang. *kernel*) stanowiące główny element systemu operacyjnego. Jądro jest oprogramowaniem rezydującym w pamięci komputera, które instruuje procesor, gdzie ma szukać swojego następnego zadania. Pełniąc funkcję mediatora, jądro zajmuje się obsługą sprzętu (a zwłaszcza pamięci głównej) i działa jako główny interfejs pomiędzy sprzętem a działającymi w komputerze programami.

Procesy — działające programy zarządzane przez jądro systemu — wspólnie składają się na najwyższy poziom systemu operacyjnego, czyli *przestrzeń użytkownika*. (Dokładniejszym terminem opisującym proces byłby *proces użytkownika* i to niezależnie od tego, czy użytkownik wchodzi w jakąkolwiek interakcję z danym procesem. Przykładowo każdy serwer WWW działa jako jeden z procesów użytkownika).



Rysunek 1.1. Ogólna organizacja systemu Linux

Istnieje poważna różnica między procesami działającymi jako procesy jądra i procesy użytkownika. Jądro pracuje w *trybie jądra* (ang. *kernel mode*), natomiast procesy użytkownika działają w *trybie użytkownika* (ang. *user mode*). Kod działający w trybie jądra ma nieograniczony dostęp do procesora oraz pamięci głównej. To bardzo użyteczny, ale i niebezpieczny przywilej pozwalający procesom jądra na łatwe uszkodzenie całego systemu. Przestrzeń pamięci, do którego dostęp ma jedynie jądro systemu, nazywany jest *przestrzenią jądra* (ang. *kernel space*).

Z drugiej strony, tryb użytkownika ogranicza procesom dostęp do (zazwyczaj bardzo ograniczonego) podzbioru pamięci i bezpiecznych poleceń procesora.

Przestrzeń użytkownika (ang. *user space*) odnosi się do tych części pamięci operacyjnej, do których dostęp mają procesy użytkownika. Jeżeli dany proces wykona nieprawidłową operację, jej konsekwencje są ograniczone i mogą zostać usunięte przez jądro systemu. Oznacza to, że w przypadku błędu w przeglądarce nie musimy się obawiać o naukowe obliczenia wykonywane w tle od kilku dni.

Teoretycznie źle działający proces użytkownika nie jest w stanie wyrządzić poważnych szkód w systemie operacyjnym. W rzeczywistości zależy to od definicji „poważnej szkody" oraz konkretnych uprawnień, jakie taki proces może mieć, ponieważ niektórym procesom pozwala się na więcej niż innym. Czy na przykład proces użytkownika może całkowicie zniszczyć dane zapisane na dysku? Owszem, jeżeli ma odpowiednie uprawnienia, a to można uznać za bardzo niebezpieczne działanie. Oczywiście istnieją mechanizmy zabezpieczające przed takimi sytuacjami, a większość procesów nie ma uprawnień, żeby siać aż takie zniszczenia.

UWAGA *Jądro systemu Linux może uruchamiać własne wątki, które bardzo przypominają procesy, ale mają też dostęp do obszaru jądra. Przykładami takich wątków są demony* `kthreadd` *i* `kblockd`*.*

1.2. Sprzęt: pamięć operacyjna

Spośród wszystkich elementów sprzętowych komputera *pamięć operacyjna* jest chyba elementem najważniejszym. W swojej najprostszej postaci pamięć jest tylko gigantycznym zbiorem przechowującym zera i jedynki. Każde z tych zer i każda jedynka nazywane są *bitami*. To właśnie w tym miejscu przechowywane są jądro i procesy użytkownika — one również są jedynie zbiorami zer i jedynek. Wszystkie sygnały wejścia i wyjścia z urządzeń peryferyjnych przepływają przez pamięć operacyjną, a zatem i one są jedynie zbiorem bitów. Sam procesor jest tylko elementem pracującym na pamięci. Odczytuje on instrukcje oraz dane z pamięci, a następnie zapisuje do niej dane wynikowe.

W odniesieniu do pamięci, procesów, jądra i innych elementów systemu komputerowego często pojawia się pojęcie *stanu*. W ścisłym znaczeniu stan jest określonym ułożeniem bitów. Przykładowo dla czterech bitów w pamięci wartości 0110, 0001 i 1011 oznaczają trzy różne stany.

Jeżeli przyjmiemy, że pojedynczy proces może składać się z milionów bitów pamięci, podczas dyskusji o stanie procesu zwykle łatwiej stosować jakieś abstrakcyjne pojęcie. Zamiast opisywać stan za pomocą pojedynczych bitów, mówimy o tym, co proces w danym momencie robił lub robi. Możemy na przykład powiedzieć, że „proces oczekuje na wejście" albo „proces wykonuje drugi etap procedury uruchomieniowej".

UWAGA *Zazwyczaj o stanie mówi się z wykorzystaniem pojęć abstrakcyjnych, a nie faktycznych bitów, dlatego pojęcie* obrazu *(ang.* image*) odnosi się do fizycznego ułożenia bitów w pamięci.*

1.3. Jądro systemu

Dlaczego ciągle mówimy o pamięci operacyjnej i stanach? Po prostu niemal wszystko, co robi jądro systemu, wiąże się z pamięcią komputera. Jednym z zadań jądra jest dzielenie pamięci na wiele części, dlatego cały czas musi ono przechowywać informację o stanie każdej z tych części. Każdy proces otrzymuje swoją porcję pamięci, a jądro systemu musi zapewnić procesom wyłączność na przydzieloną im pamięć.

Zadaniem jądra systemu jest zarządzanie zadaniami w czterech głównych obszarach systemowych. Oto one.

Procesy — jądro musi określić, który z procesów może w danym momencie korzystać z procesora.

Pamięć — jądro musi odpowiednio zarządzać pamięcią, przydzielać ją poszczególnym procesom, wyznaczać części współdzielone przez procesy i organizować wolną pamięć.

Sterowniki urządzeń — jądro działa jako interfejs pomiędzy sprzętem (na przykład dyskiem) a procesami. Zazwyczaj operowanie urządzeniami to zadanie jądra.

Wywołania systemowe — procesy zwykle komunikują się z jądrem za pomocą wywołań systemowych.

Teraz pokrótce omówię każdy z tych obszarów.

UWAGA *Jeżeli interesują Cię szczegóły działania jądra systemu, polecam lekturę dwóch dobrych książek: pierwsza to dziesiąte wydanie* Operating System Concepts*, autorstwa Abrahama Silberschatza, Petera B. Galvina i Grega Gagne'go (Wiley, 2018), a druga to czwarte wydanie* Systemów operacyjnych *autorstwa Andrew S. Tanenbauma i Herberta Bosa (Helion, 2015).*

1.3.1. Zarządzanie procesami

Zarządzanie procesami to ogół zadań związanych z uruchamianiem, wstrzymywaniem, ponownym uruchamianiem, planowaniem i kończeniem pracy procesów. Koncepcje związane z uruchamianiem i kończeniem procesu są względnie proste, ale już opisanie, w jaki sposób proces wykorzystuje procesor w trakcie swojej normalnej pracy, jest zdecydowanie bardziej skomplikowane.

W każdym nowoczesnym systemie operacyjnym wiele procesów może działać „jednocześnie". Przykładowo w tym samym momencie na swoim komputerze możesz mieć uruchomioną przeglądarkę stron WWW i arkusz kalkulacyjny. Niestety nie wszystko wygląda tutaj tak, jak można by się spodziewać. Poszczególne procesy kryjące się za aplikacjami zazwyczaj nie pracują *dokładnie* w tym samym czasie.

Załóżmy, że dysponujemy systemem z jednordzeniowym procesorem. Z takiego procesora *może* korzystać wiele różnych procesów, ale tylko jeden z nich będzie

w danym momencie rzeczywiście wykonywany przez ten procesor. W praktyce każdy proces używa procesora przez pewien krótki czas, a potem wstrzymuje swoją pracę. Następnie kolejny proces zaczyna używać procesora, co trwa ułamek sekundy, po czym przez następny ułamek sekundy z procesora może korzystać kolejny proces i tak dalej. Operacja przekazania kontroli nad procesorem od jednego procesu do drugiego nazywana jest *przełączaniem kontekstu* (ang. *context switch*).

Każdy taki *wycinek czasu* (ang. *time slice*) pozwala procesowi na wykonanie całkiem sporej ilości obliczeń i rzeczywiście większość procesów może zakończyć swoje aktualne prace w ramach pojedynczego wycinka. A dzięki temu, że wycinki czasu są tak małe, ludzie nie są w stanie ich zauważyć, przez co system sprawia wrażenie, jakby działało w nim jednocześnie wiele różnych procesów — to cecha nazywana *wielozadaniowością* (ang. *multitasking*).

Za przełączanie kontekstów odpowiedzialne jest jądro systemu. Aby lepiej poznać zasadę działania tego mechanizmu, wyobraźmy sobie sytuację, w której proces działa w trybie użytkownika i właśnie zakończył się przydzielony mu wycinek czasu. W tym momencie wykonywane są opisane niżej operacje.

1. W reakcji na sygnał z wewnętrznego zegara procesor (rzeczywisty sprzęt) przerywa pracę aktualnego procesu, przełącza się w tryb jądra i przekazuje kontrolę do jądra systemu.
2. Jądro systemu zapisuje aktualny stan procesora w pamięci, co pozwoli mu później wznowić pracę procesu, który właśnie został przerwany.
3. Jądro wykonuje wszelkie zadania, jakie mogły się pojawić w trakcie zakończonego właśnie wycinka czasu, takie jak zbieranie danych z wejść lub wyjść lub wykonywanie operacji wejścia i wyjścia.
4. Teraz jądro jest gotowe do wznowienia pracy kolejnego procesu. Zaczyna ono analizę listy procesów działających aktualnie w systemie i wybiera z niej jeden proces.
5. Jądro przygotowuje pamięć do pracy nowego procesu, a następnie odtwarza stan procesora.
6. Jądro informuje procesor, jak długo będzie trwał wycinek czasu przydzielony wznawianemu procesowi.
7. Jądro przełącza procesor w tryb użytkownika i przekazuje kontrolę nad procesorem wznawianemu procesowi.

Operacja przełączania kontekstu odpowiada na pytanie, *kiedy* działa jądro systemu? Okazuje się, że działa ono *pomiędzy* wycinkami czasu przydzielanymi procesom, w trakcie przełączania kontekstów.

W przypadku systemu wieloprocesorowego (dotyczy to większości dostępnych obecnie komputerów) całość nieco się komplikuje, ponieważ jądro nie musi już zrzekać się kontroli nad procesorem w celu wznowienia pracy dowolnego procesu na innym procesorze, a ponadto jednocześnie może działać więcej niż jeden proces. Mimo to, jądro zwalnia każdy z procesorów dostępnych w systemie

w celu jak najlepszego wykorzystania wszystkich dostępnych zasobów. Oczywiście może też wykorzystać kilka prostych sztuczek, żeby przejąć na swoje potrzeby nieco więcej czasu procesora.

1.3.2. Zarządzanie pamięcią

Jądro systemu musi zarządzać pamięcią podczas przełączania kontekstu, co może być złożonym zadaniem. Konieczne jest zapewnienie następujących warunków:

- Jądro musi mieć własną, prywatną przestrzeń w pamięci, do której nie mają dostępu procesy użytkownika.
- Każdy proces użytkownika musi otrzymać własny wycinek pamięci.
- Żaden z procesów użytkownika nie może uzyskać dostępu do prywatnej pamięci innego procesu.
- Procesy użytkownika mogą współdzielić pamięć.
- Część pamięci procesu użytkownika może być pamięcią tylko do odczytu.
- System może wykorzystywać więcej pamięci niż fizycznie zainstalowana, używając zamiennie przestrzeni na dysku twardym.

Na szczęście jądro systemu może skorzystać z pomocy. W nowoczesnych procesorach znajduje się specjalny *moduł zarządzania pamięcią* (MMU — ang. *Memory Management Unit*), który pozwala na korzystanie ze schematu dostępu do pamięci nazywanego *pamięcią wirtualną* (ang. *virtual memory*). Podczas wykorzystania pamięci wirtualnej procesor nie adresuje jej bezpośrednio, podając fizyczny adres w sprzętowym układzie scalonym. W takiej konfiguracji jądro przygotowuje każdy proces tak, jakby sam miał do swojej dyspozycji całą maszynę. Gdy proces próbuje uzyskać dostęp do przydzielonej mu pamięci, moduł MMU przechwytuje taką próbę i wykorzystuje mechanizm mapowania adresów, żeby przekształcić lokalizację w pamięci procesu na fizyczną lokalizację w pamięci komputera. Oczywiście jądro systemu musi odpowiednio zainicjować, a potem ciągle aktualizować zawartość mapy adresów. Przykładowo podczas przełączania kontekstów jądro musi zmienić aktualną mapę, dopasowując ją do wznawianego właśnie procesu.

UWAGA *Implementacja mapy adresów pamięci nazywana jest* tablicą stron *(ang.* page table*).*

Więcej informacji na temat działania pamięci komputera znajdziesz w rozdziale 8.

1.3.3. Sterowniki urządzeń i zarządzanie urządzeniami

Zadania jądra w odniesieniu do urządzeń pracujących w komputerze są stosunkowo proste. Dostęp do każdego urządzenia możliwy jest jedynie w trybie jądra, ponieważ niewłaściwa ich obsługa (na przykład proces użytkownika proszący o wyłączenie zasilania) może doprowadzić do zablokowania komputera. Godną

uwagi trudność stanowi fakt, że poszczególne urządzenia bardzo rzadko korzystają z tego samego interfejsu programowania, nawet jeżeli mają takie same zadania. Przykładem mogą być karty sieciowe. Z tego właśnie powodu sterowniki tradycyjnie już budowane są jako część jądra i starają się zaprezentować procesom użytkownika jednolity interfejs, żeby ułatwić pracę twórcom aplikacji.

1.3.4. Wywołania systemowe

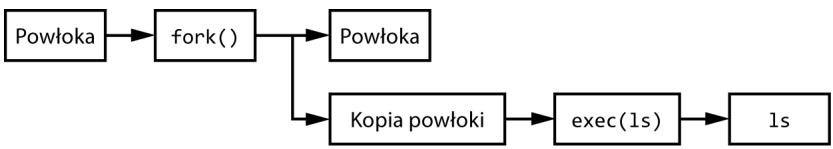
Istnieje też kilka innych rodzajów funkcji udostępnianych przez jądro procesom użytkownika. Przykładowo *wywołania systemowe* (ang. *system calls*) spełniają określone zadania, których procesy użytkownika nie są w stanie wykonać dobrze lub nie mogą ich wykonać w ogóle. Przykładem takich wywołań systemowych mogą być operacje związane z otwieraniem, odczytywaniem i zapisywaniem plików.

Dwa wywołania systemowe — `fork()` i `exec()` — są szczególnie istotne przy poznawaniu sposobu uruchamiania procesów w systemie.

`fork()` — gdy dany proces wywołuje funkcję `fork()`, jądro systemu tworzy niemal identyczną kopię tego procesu.

`exec()` — gdy proces wywołuje funkcję `exec(program)`, jądro systemu ładuje i uruchamia *program*, który zastępuje aktualny proces.

W systemach linuksowych *wszystkie* nowe procesy użytkownika z wyjątkiem procesu *init* (więcej o nim w rozdziale 6.) powstają w wyniku wywołania systemowego `fork()`. Dodatkowo, w większości przypadków wywoływana jest funkcja `exec()` pozwalająca na uruchomienie nowego programu bez tworzenia kopii istniejącego już procesu. Bardzo prostym przykładem mogą być programy uruchamiane w wierszu poleceń, takie jak polecenie `ls` wyświetlające zawartość katalogu. Po wprowadzeniu polecenia `ls` w oknie terminala działająca w nim powłoka wywołuje funkcję `fork()` w celu utworzenia nowego procesu powłoki, a następnie ta nowa kopia procesu wywołuje funkcję `exec(ls)`, uruchamiając tym samym program `ls`. Na rysunku 1.2 można zobaczyć schemat interakcji procesów i wywołań systemowych niezbędnych do uruchomienia takiego programu jak `ls`.



Rysunek 1.2. Uruchamianie nowego procesu

UWAGA *Wywołania systemowe są zazwyczaj wyróżniane za pomocą nawiasów. W poprzednim przykładzie z rysunku 1.2 proces proszący jądro o utworzenie nowego procesu musi skorzystać z wywołania systemowego `fork()`. Taki zapis wywodzi się ze sposobu zapisywania wywołań funkcji w języku C. Nie musisz znać tego języka w celu poznania zawartości tej książki. Wystarczy zapamiętać, że wywołanie*

systemowe jest formą interakcji między procesem a jądrem systemu. Dodatkowo w tej książce będę upraszczał niektóre grupy wywołań systemowych. Przykładowo zapis `exec()` *odnosić się będzie do całej grupy wywołań systemowych wykonujących podobne zadania, ale różniących się drobnymi szczegółami. W przypadku procesu istnieje również wariant nazywany* wątkiem, *który zostanie omówiony w rozdziale 8.*

Jądro obsługuje też procesy użytkownika, udostępniając im funkcje inne niż tradycyjne wywołania systemowe, takie jak *pseudourządzenia* (ang. *pseudodevices*). Dla procesu użytkownika pseudourządzenie wygląda jak normalne urządzenie, ale istnieje wyłącznie w postaci oprogramowania. W związku z tym takie urządzenia nie muszą znajdować się w samym jądrze, ale ze względów praktycznych zazwyczaj się je tam umieszcza. Przykładowo urządzenie do generowania liczb losowych (`/dev/random`) zaimplementowane jako proces użytkownika nie zapewniałoby odpowiedniego poziomu bezpieczeństwa.

UWAGA *Pod względem technicznym proces użytkownika, który chce skorzystać z pseudourządzenia, musi użyć wywołań systemowych, żeby to urządzenie otworzyć. Oznacza to, że procesy nie mogą całkowicie unikać korzystania z tych wywołań.*

1.4. Przestrzeń użytkownika

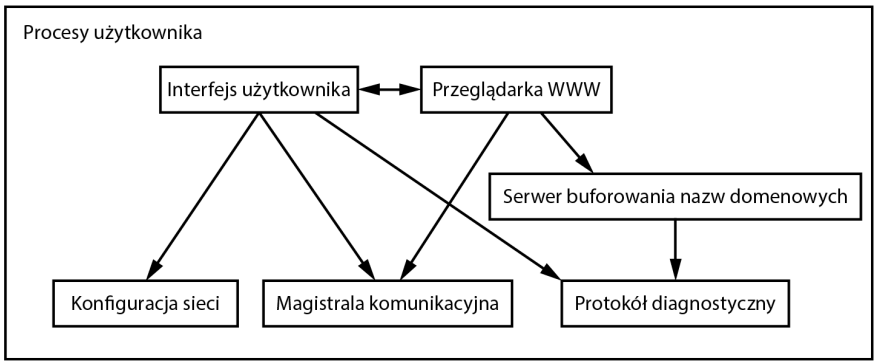
Jak już wspominałem wcześniej, pamięć komputera przygotowywana przez jądro na potrzeby procesów użytkownika nazywana jest *przestrzenią użytkownika* (ang. *user space*). Ze względu na to, że proces jest jedynie stanem (lub obrazem) w pamięci, przestrzeń użytkownika opisuje w pamięci całą kolekcję działających w niej procesów.

Większość faktycznych działań w systemach Linux wykonywana jest w przestrzeni użytkownika. Co prawda, z punktu widzenia jądra systemu wszystkie procesy są sobie równe, jednak mogą one wykonywać różne zadania na rzecz użytkowników. Istnieje ogólna struktura poziomów (warstw) opisująca rodzaje komponentów systemu reprezentowanych przez procesy. Na rysunku 1.3 przedstawiłem przykładowy zestaw komponentów współpracujących w ramach systemu. Podstawowe usługi zlokalizowane są na dolnym poziomie (najbliżej jądra), usługi pomocnicze znajdują się na środkowym poziomie, a na samej górze umieszczone są aplikacje, z którymi styka się użytkownik. Oczywiście rysunek 1.3 został bardzo uproszczony, ponieważ przedstawiłem na nim zaledwie sześć komponentów, ale już tutaj widać, że komponenty umieszczone na górze znajdują się najbliżej użytkownika (interfejs użytkownika i przeglądarka WWW). Wśród komponentów wymieniam serwer buforowania nazw domenowych, z którego korzysta przeglądarka, natomiast na samym dole umieściłem kilka niewielkich elementów systemu.

Najniższy poziom najczęściej składa się z niewielkich komponentów, które wykonują proste, nieskomplikowane zadania. Poziom środkowy zwykle zawiera większe komponenty, takie jak serwery poczty, wydruku lub baz danych. I w końcu

komponenty z najwyższego poziomu wykonują złożone zadania, które użytkownik często sam kontroluje. Każdy komponent może też korzystać z innych komponentów. Zazwyczaj gdy jeden komponent chce użyć innego, ten drugi znajduje się albo na tym samym, albo na niższym poziomie.

Pamiętaj, że rysunek 1.3 jest tylko ogólnym przybliżeniem ułożenia komponentów w przestrzeni użytkownika. W rzeczywistości w przestrzeni użytkownika nie ma żadnych reguł. Przykładowo większość aplikacji i usług zapisuje komunikaty diagnostyczne nazywane *protokołami* (ang. *logs*). Zazwyczaj programy wykorzystują do tego standardową usługę syslog, ale część może samodzielnie wykonywać zadania związane z protokołowaniem.



Rysunek 1.3. Rodzaje procesów i ich interakcje

Co więcej, kategoryzacja części komponentów z przestrzeni użytkownika jest bardzo kłopotliwa. Komponenty serwerowe, takie jak serwery WWW lub baz danych, można traktować jak aplikacje bardzo wysokiego poziomu, ponieważ wykonywane przez nie zadania są tak złożone, że na rysunku 1.3 powinny się znaleźć na najwyższym poziomie. Mimo to, aplikacje użytkownika mogą używać tych serwerów w celu wykonywania zadań, którymi nie chcą się zajmować samodzielnie, w związku z czym należałoby je umieścić na środkowym poziomie.

1.5. Użytkownicy

Jądro Linuksa obsługuje tradycyjną koncepcję użytkownika systemu Unix. *Użytkownik* (ang. *user*) jest encją, która może uruchamiać procesy i być właścicielem plików. Każdemu użytkownikowi przypisana jest najczęściej *nazwa użytkownika* (ang. *username*). Przykładowo w dowolnym systemie może istnieć użytkownik o nazwie *marian*. Jądro systemu nie rozpoznaje użytkowników po nazwach, ale identyfikuje za pomocą specjalnych liczbowych identyfikatorów o nazwie *userid*. (W rozdziale 7. opowiem, w jaki sposób nazwy użytkownika wiązane są z ich identyfikatorami).

Użytkownicy istnieją w systemie głównie po to, żeby wyznaczać uprawnienia i ich granice. Każdy proces w przestrzeni użytkownika ma swojego *właściciela* (ang. *owner*), dlatego mówi się, że procesy uruchamiane są *jako* dany użytkownik. Użytkownik może kończyć lub zmieniać zachowanie swoich własnych procesów (w pewnych granicach), ale nie może w żaden sposób wpływać na procesy innych użytkowników. Oprócz tego użytkownicy mogą być właścicielami plików i decydować o tym, czy chcą się nimi dzielić z innymi użytkownikami.

W systemie Linux zazwyczaj zdefiniowanych jest kilku użytkowników, oprócz tych przypisanych konkretnym osobom korzystającym z tego systemu. Więcej na ten temat przeczytasz w rozdziale 3., a tutaj zaznaczę, że najważniejszy użytkownik nosi nazwę *root*. Ten właśnie użytkownik stanowi wyjątek od opisywanych wcześniej reguł, ponieważ może zakończyć lub zmodyfikować proces dowolnego użytkownika oraz odczytywać zawartość każdego pliku w systemie. Z tego właśnie powodu root znany jest też jako *superużytkownik* (ang. *superuser*). Mówi się, że osoba pracująca w systemie jako użytkownik root ma *uprawnienia roota* (ang. *root access*) i jest administratorem tradycyjnego systemu uniksowego.

UWAGA *Praca z uprawnieniami superużytkownika może być niebezpieczna. Naprawdę trudne może być identyfikowanie i poprawianie ewentualnych błędów, ponieważ system pozwoli na wykonanie każdej operacji, nawet jeżeli może ona uszkodzić system. Z tego powodu projektanci systemów starają się jak najbardziej ograniczyć konieczność używania konta root, na przykład nie wymagają go do przełączania się w laptopie pomiędzy różnymi sieciami bezprzewodowymi. Musisz też pamiętać, że mimo swoich ogromnych uprawnień, użytkownik root nadal działa w trybie użytkownika systemu operacyjnego, a nie w trybie jądra.*

Grupy są zbiorami użytkowników. Podstawowym zadaniem grup jest umożliwianie współdzielenia plików między innymi członkami grupy.

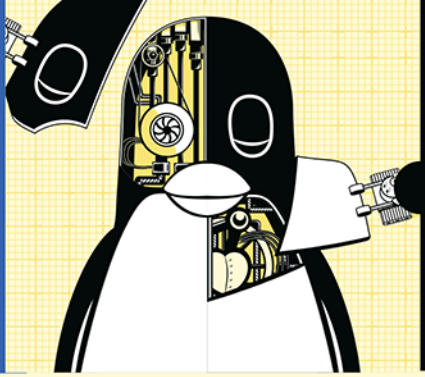
1.6. Spojrzenie w przyszłość

Na razie zaprezentowałem elementy niezbędne do *działania* systemu Linux. Procesy użytkownika tworzą środowisko, z którym możesz wchodzić w bezpośrednią interakcję, natomiast jądro systemu zajmuje się obsługą procesów i sprzętu. Zarówno jądro, jak i wszystkie procesy przechowywane są w pamięci.

To bardzo cenne informacje, ale nie poznasz tajników Linuksa, czytając wyłącznie o samych procesach. Powoli trzeba zacząć brudzić sobie ręce. Następny rozdział zacznę od zaprezentowania kilku podstaw związanych z przestrzenią użytkownika. Przy okazji poznasz ważne części systemu, o których nawet nie wspomniałem w tym rozdziale, czyli dyski, pliki i tym podobne. W końcu wszystkie te programy i dane musimy gdzieś zapisywać.


PROGRAM
PARTNERSKI
GRUPY 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
GRUPA
Helion

NIGDY WIĘCEJ
WALKI Z WŁASNYM
KOMPUTEREM!



System Linux umożliwia uzyskanie pełnej kontroli nad komputerem, pozwala bowiem na łatwy dostęp do jego ważnych elementów. Konfiguracja większości składników systemu jest zapisana w plikach tekstowych, które można bez trudu odczytać. Uzyskana w ten sposób wiedza przydaje się nie tylko programistom i administratorom, ale i użytkownikom, którzy chcą dobrze zrozumieć działanie swojego komputera, a także dowiedzieć się, jak pracują wewnętrzne mechanizmy systemu, jak funkcjonuje sieć i jakie zadania realizuje jądro systemu.

To trzecie wydanie bestsellerowego podręcznika dla administratorów systemów Linux. Zostało zaktualizowane i uzupełnione materiałem dotyczącym menedżera LVM, wirtualizacji i kontenerów. Znajdziesz tu informacje o sposobie pracy poszczególnych elementów systemu Linux i o sekwencji jego rozruchu. W książce omówiono też jądro i przybliżono kluczowe procesy przestrzeni użytkowników, w tym wywołania systemowe, operacje wejścia-wyjścia i utrzymywanie systemów plików. Nie zabrakło także dokładnych instrukcji dotyczących narzędzi używanych przez administratorów i programistów, praktycznych przykładów i ćwiczeń opatrzonych szczegółowymi objaśnieniami. W efekcie lektury zrozumiesz, w jaki sposób działa Twój komputer, i poznasz tajniki zaawansowanej konfiguracji systemu Linux!

Dzięki książce dowiesz się:

- jak przebiega proces uruchamiania Linuksa i jak działa demon systemd
- w jaki sposób jądro systemu zarządza urządzeniami, sterownikami urządzeń i procesami
- jak działają sieci, interfejsy, zapory sieciowe i serwery
- jak korzystać z narzędzi do projektowania
- jak tworzyć efektywne skrypty powłoki
- czym jest menedżer LVM, system rejestrowania demona journald i jak działa protokół IPv6
- na czym polega wirtualizacja z uwzględnieniem kontenerów i grup kontrolnych cgroup

Dr Brian Ward pracuje jako konsultant i instruktor w San Francisco. Jest autorem kilku cenionych książek dotyczących systemów linuksowych. Linuksem zainteresował się w 1993 roku, kiedy udało mu się uzbierać dość pieniędzy na używany 386.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej! ▶

ISBN 978-83-283-8863-5

Cena: 99,00 zł

no starch press