

Wszystko, co chcesz wiedzieć o systemach operacyjnych!

SYSTEMY OPERACYJNE

WYDANIE III

Jak działają algorytmy szeregowania?
Jakie możliwości stoją za wirtualizacją?
Jak systemy operacyjne zarządzają pamięcią?

System
wielowątkowy

Błąd

Karuzela

Wątek

Mobilny system
operacyjny

Zakleszczenie

Cienki
klient

Bezpieczeństwo

Sekcja krytyczna

Mechanizm
zabezpieczający

Wieżenie

Równoważenie
obciążenia

Niebieski
krzyk śmierci

Spyware

Intruz

Wyjście

Koń trojański

Wejście

Podsystem
zarządzania
pamięcią

Program
szeregujący

System
wbudowany

Algorytm
strusia

Virtualizacja

Multimedia

System
wieloprocesorowy

Uruchamianie
komputera

Kompresja
video

Zarządzanie
energiją

Przerwanie

Wyścig

Serwer

Przepelnienie
bufora

Klient

Dwurdzeniowy
system Linux

Unix

ANDREW S.
TANENBAUM

PEARSON
Prentice
Hall



Helion

Tytuł oryginału: Modern Operating Systems, 3rd Edition

Tłumaczenie: Radosław Meryk (rozdz. 1–8),
Mikołaj Szczepaniak (rozdz. 9 – 14)

ISBN: 978-83-246-7115-1

Authorized translation from the English language edition, entitled:
Modern Operating Systems, Third Edition, ISBN 0136006639,
by Andrew S. Tanenbaum, published by Pearson Education, Inc.,
publishing as Prentice Hall. Copyright © 2008 by Pearson Education, Inc.

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 Pearson Education Inc.

Polish language edition published by Helion S.A.
Copyright © 2013.

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
WWW: <http://helion.pl> (księgarnia internetowa, katalog książek)

Drogi Czytelniku!

Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres

<http://helion.pl/user/opinie/syso3v>

Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

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

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

SPIS TREŚCI

PRZEDMOWA	23
O AUTORZE	27
1 WPROWADZENIE	29
1.1. CZYM JEST SYSTEM OPERACYJNY?	32
1.1.1. System operacyjny jako rozszerzona maszyna	32
1.1.2. System operacyjny jako menedżer zasobów	34
1.2. HISTORIA SYSTEMÓW OPERACYJNYCH	36
1.2.1. Pierwsza generacja (1945 – 1955)	
— lampy elektronowe	37
1.2.2. Druga generacja (1955 – 1965)	
— tranzystory i systemy wsadowe	37
1.2.3. Trzecia generacja (1965 – 1980)	
— układy scalone i wieloprogramowość	40
1.2.4. Czwarta generacja (1980 – czasy współczesne)	
— komputery osobiste	45
1.3. SPRZĘT KOMPUTEROWY — PRZEGLĄD	50
1.3.1. Procesory	50
1.3.2. Pamięć	54
1.3.3. Dyski	58
1.3.4. Taśmy	59

1.3.5.	Urządzenia wejścia-wyjścia	59
1.3.6.	Magistrale	63
1.3.7.	Uruchamianie komputera	66
1.4.	PRZEGLĄD SYSTEMÓW OPERACYJNYCH	67
1.4.1.	Systemy operacyjne komputerów mainframe	67
1.4.2.	Systemy operacyjne serwerów	68
1.4.3.	Wieloprocessorowe systemy operacyjne	68
1.4.4.	Systemy operacyjne komputerów osobistych	68
1.4.5.	Systemy operacyjne komputerów podręcznych	69
1.4.6.	Wbudowane systemy operacyjne	69
1.4.7.	Systemy operacyjne węzłów sensorowych	70
1.4.8.	Systemy operacyjne czasu rzeczywistego	70
1.4.9.	Systemy operacyjne kart elektronicznych	71
1.5.	POJĘCIA DOTYCZĄCE SYSTEMÓW OPERACYJNYCH	72
1.5.1.	Procesy	72
1.5.2.	Przestrzenie adresowe	74
1.5.3.	Pliki	75
1.5.4.	Wejście-wyjście	79
1.5.5.	Zabezpieczenia	79
1.5.6.	Powłoka	80
1.5.7.	Ontogeneza jest rekapitulacją filogenezy	81
1.6.	WYWOŁANIA SYSTEMOWE	85
1.6.1.	Wywołania systemowe do zarządzania procesami	90
1.6.2.	Wywołania systemowe do zarządzania plikami	93
1.6.3.	Wywołania systemowe do zarządzania katalogami	94
1.6.4.	Różne wywołania systemowe	96
1.6.5.	Interfejs Win32 API systemu Windows	97
1.7.	STRUKTURA SYSTEMÓW OPERACYJNYCH	99
1.7.1.	Systemy monolityczne	100
1.7.2.	Systemy warstwowe	101
1.7.3.	Mikrojądra	102
1.7.4.	Model klient-serwer	105
1.7.5.	Maszyny wirtualne	106
1.7.6.	Egzojądra	110
1.8.	ŚWIAT WEDŁUG JĘZYKA C	111
1.8.1.	Język C	111
1.8.2.	Pliki nagłówkowe	112
1.8.3.	Duże projekty programistyczne	113
1.8.4.	Model fazy działania	114

1.9.	BADANIA DOTYCZĄCE SYSTEMÓW OPERACYJNYCH	115
1.10.	PLAN POZOSTAŁEJ CZĘŚCI KSIĄŻKI	117
1.11.	JEDNOSTKI MIAR	118
1.12.	PODSUMOWANIE	119

2 PROCESY I WĄTKI

123

2.1.	PROCESY	123
2.1.1.	Model procesów	124
2.1.2.	Tworzenie procesów	126
2.1.3.	Kończenie działania procesów	129
2.1.4.	Hierarchie procesów	130
2.1.5.	Stany procesów	131
2.1.6.	Implementacja procesów	133
2.1.7.	Modelowanie wieloprogramowości	135
2.2.	WĄTKI	137
2.2.1.	Wykorzystanie wątków	137
2.2.2.	Klasyczny model wątków	143
2.2.3.	Wątki POSIX	147
2.2.4.	Implementacja wątków w przestrzeni użytkownika	150
2.2.5.	Implementacja wątków w jądrze	153
2.2.6.	Implementacje hybrydowe	154
2.2.7.	Mechanizm aktywacji zarządcy	155
2.2.8.	Wątki pop-up	157
2.2.9.	Przystosowywanie kodu jednowątkowego do obsługi wielu wątków	158
2.3.	KOMUNIKACJA MIĘDZY PROCESAMI	161
2.3.1.	Wyścig	162
2.3.2.	Regiony krytyczne	163
2.3.3.	Wzajemne wykluczanie z wykorzystaniem aktywnego oczekiwania	165
2.3.4.	Wywołania sleep i wakeup	171
2.3.5.	Semafor	174
2.3.6.	Muteksy	177
2.3.7.	Monitory	182
2.3.8.	Przekazywanie komunikatów	188
2.3.9.	Bariery	191

2.4.	SZEREGOWANIE	193
2.4.1.	Wprowadzenie do szeregowania	193
2.4.2.	Szeregowanie w systemach wsadowych	201
2.4.3.	Szeregowanie w systemach interaktywnych	203
2.4.4.	Szeregowanie w systemach czasu rzeczywistego	210
2.4.5.	Oddzielenie strategii od mechanizmu	212
2.4.6.	Szeregowanie wątków	212
2.5.	KLASYCZNE PROBLEMY KOMUNIKACJI MIĘDZY PROCESAMI	214
2.5.1.	Problem pięciu filozofów	214
2.5.2.	Problem czytelników i pisarzy	218
2.6.	PRACE BADAWCZE NAD PROCESAMI I WĄTKAMI	219
2.7.	PODSUMOWANIE	220

3 ZARZĄDZANIE PAMIĘCIĄ

227

3.1.	BRAK ABSTRAKCJI PAMIĘCI	228
3.2.	ABSTRAKCJA PAMIĘCI: PRZESTRZENIE ADRESOWE	232
3.2.1.	Pojęcie przestrzeni adresowej	232
3.2.2.	Wymiana pamięci	235
3.2.3.	Zarządzanie wolną pamięcią	237
3.3.	PAMIĘĆ WIRTUALNA	241
3.3.1.	Stronicowanie	243
3.3.2.	Tabele stron	247
3.3.3.	Przyspieszenie stronicowania	249
3.3.4.	Tabele stron dla pamięci o dużej objętości	253
3.4.	ALGORYTMY ZASTĘPOWANIA STRON	257
3.4.1.	Optymalny algorytm zastępowania stron	258
3.4.2.	Algorytm NRU	258
3.4.3.	Algorytm FIFO	260
3.4.4.	Algorytm drugiej szansy	260
3.4.5.	Algorytm zegarowy	261
3.4.6.	Algorytm LRU	262
3.4.7.	Programowa symulacja algorytmu LRU	263
3.4.8.	Algorytm bazujący na zbiorze roboczym	265

3.4.9.	Algorytm WSClock	269
3.4.10.	Podsumowanie algorytmów zastępowania stron	271
3.5.	PROBLEMY PROJEKTOWE SYSTEMÓW STRONICOWANIA	273
3.5.1.	Lokalne i globalne strategie alokacji pamięci	273
3.5.2.	Zarządzanie obciążeniem	276
3.5.3.	Rozmiar strony	277
3.5.4.	Osobne przestrzenie instrukcji i danych	278
3.5.5.	Strony współdzielone	279
3.5.6.	Biblioteki współdzielone	281
3.5.7.	Pliki odwzorowane w pamięci	283
3.5.8.	Strategia czyszczenia	284
3.5.9.	Interfejs pamięci wirtualnej	284
3.6.	PROBLEMY IMPLEMENTACJI	285
3.6.1.	Zadania systemu operacyjnego w zakresie stronicowania	286
3.6.2.	Obsługa błędów braku strony	287
3.6.3.	Wznawianie instrukcji	288
3.6.4.	Blokowanie stron w pamięci	289
3.6.5.	Magazyn stron	290
3.6.6.	Oddzielenie strategii od mechanizmu	292
3.7.	SEGMENTACJA	294
3.7.1.	Implementacja klasycznej segmentacji	298
3.7.2.	Segmentacja ze stronicowaniem: MULTICS	298
3.7.3.	Segmentacja ze stronicowaniem: Intel Pentium	302
3.8.	BADANIA NAD ZARZĄDZANIEM PAMIĘCIĄ	307
3.9.	PODSUMOWANIE	308

4 SYSTEMY PLIKÓW

317

4.1.	PLIKI	319
4.1.1.	Nazwy plików	319
4.1.2.	Struktura pliku	321
4.1.3.	Typy plików	323
4.1.4.	Dostęp do plików	325
4.1.5.	Atrybuty plików	325

4.1.6.	Operacje na plikach	327
4.1.7.	Przykładowy program wykorzystujący wywołania obsługi systemu plików	328
4.2.	KATALOGI	331
4.2.1.	Jednopoziomowe systemy katalogów	331
4.2.2.	Hierarchiczne systemy katalogów	332
4.2.3.	Nazwy ścieżek	333
4.2.4.	Operacje na katalogach	335
4.3.	IMPLEMENTACJA SYSTEMU PLIKÓW	337
4.3.1.	Układ systemu plików	337
4.3.2.	Implementacja plików	338
4.3.3.	Implementacja katalogów	344
4.3.4.	Pliki współdzielone	347
4.3.5.	Systemy plików o strukturze dziennika	349
4.3.6.	Księgujące systemy plików	352
4.3.7.	Wirtualne systemy plików	354
4.4.	ZARZĄDZANIE SYSTEMEM PLIKÓW I OPTYMALIZACJA	357
4.4.1.	Zarządzanie miejscem na dysku	357
4.4.2.	Kopie zapasowe systemu plików	365
4.4.3.	Spójność systemu plików	371
4.4.4.	Wydajność systemu plików	375
4.4.5.	Defragmentacja dysków	380
4.5.	PRZYKŁADOWE SYSTEMY PLIKÓW	381
4.5.1.	Systemy plików na płytach CD-ROM	381
4.5.2.	System plików MS-DOS	387
4.5.3.	System plików V7 systemu UNIX	391
4.6.	BADANIA DOTYCZĄCE SYSTEMÓW PLIKÓW	394
4.7.	PODSUMOWANIE	394

5 WEJŚCIE-WYJŚCIE

399

5.1.	WARUNKI, JAKIE POWINIEN SPEŁNIAĆ SPRZĘT WEJŚCIA-WYJŚCIA	400
5.1.1.	Urządzenia wejścia-wyjścia	400
5.1.2.	Kontrolery urządzeń	402

5.1.3.	Urządzenia wejścia-wyjścia odwzorowane w pamięci	403
5.1.4.	Bezpośredni dostęp do pamięci (DMA)	407
5.1.5.	O przerwaniach raz jeszcze	410
5.2.	WARUNKI, JAKIE POWINNO SPEŁNIAĆ OPROGRAMOWANIE WEJŚCIA-WYJŚCIA	415
5.2.1.	Cele oprogramowania wejścia-wyjścia	415
5.2.2.	Programowane wejście-wyjście	417
5.2.3.	Wejście-wyjście sterowane przerwaniami	419
5.2.4.	Wejście-wyjście z wykorzystaniem DMA	420
5.3.	WARSTWY OPROGRAMOWANIA WEJŚCIA-WYJŚCIA	420
5.3.1.	Procedury obsługi przerw	421
5.3.2.	Sterowniki urządzeń	422
5.3.3.	Oprogramowanie wejścia-wyjścia niezależne od urządzeń	426
5.3.4.	Oprogramowanie wejścia-wyjścia w przestrzeni użytkownika	432
5.4.	DYSKI	435
5.4.1.	Sprzęt	435
5.4.2.	Formatowanie dysków	452
5.4.3.	Algorytmy szeregowania żądań dostępu do dysku	456
5.4.4.	Obsługa błędów	459
5.4.5.	Stabilna pamięć masowa	462
5.5.	ZEGARY	466
5.5.1.	Sprzęt obsługi zegara	466
5.5.2.	Oprogramowanie obsługi zegara	468
5.5.3.	Zegary programowe	471
5.6.	INTERFEJSY UŻYTKOWNIKÓW: Klawiatura, mysz, monitor	473
5.6.1.	Oprogramowanie do wprowadzania danych	473
5.6.2.	Oprogramowanie do generowania wyjścia	479
5.7.	CIEŃKIE KLIENTY	496
5.8.	ZARZĄDZANIE ENERGIĄ	499
5.8.1.	Problemy sprzętowe	500
5.8.2.	Problemy po stronie systemu operacyjnego	501
5.8.3.	Problemy do rozwiązania w programach aplikacyjnych	507

5.9.	BADANIA DOTYCZĄCE WEJŚCIA-WYJŚCIA	509
5.10.	PODSUMOWANIE	510

6 Zakleszczenia

517

6.1.	ZASOBY	518
6.1.1.	Zasoby z możliwością wywłaszczenia i bez niej	518
6.1.2.	Zdobywanie zasobu	520
6.2.	WPROWADZENIE W TEMATYKĘ ZAKLESZCZEŃ	521
6.2.1.	Warunki powstawania zakleszczeń zasobów	522
6.2.2.	Modelowanie zakleszczeń	523
6.3.	ALGORYTM STRUSIA	526
6.4.	WYKRYWANIE ZAKLESZCZEŃ I ICH USUWANIE	526
6.4.1.	Wykrywanie zakleszczeń z jednym zasobem każdego typu	527
6.4.2.	Wykrywanie zakleszczeń dla przypadku wielu zasobów każdego typu	529
6.4.3.	Usuwanie zakleszczeń	532
6.5.	UNIKANIE ZAKLESZCZEŃ	534
6.5.1.	Trajektorie zasobów	534
6.5.2.	Stany bezpieczne i niebezpieczne	535
6.5.3.	Algorytm bankiera dla pojedynczego zasobu	537
6.5.4.	Algorytm bankiera dla wielu zasobów	538
6.6.	PRZECIWDZIAŁANIE ZAKLESZCZENIOM	540
6.6.1.	Atak na warunek wzajemnego wykluczania	540
6.6.2.	Atak na warunek wstrzymania i oczekiwania	541
6.6.3.	Atak na warunek braku wywłaszczenia	541
6.6.4.	Atak na warunek cyklicznego oczekiwania	542
6.7.	INNE PROBLEMY	543
6.7.1.	Blokowanie dwufazowe	543
6.7.2.	Zakleszczenia komunikacyjne	544
6.7.3.	Uwięzienia	546
6.7.4.	Zagłodzenia	548

- 6.8. BADANIA NA TEMAT ZAKLESZCZEŃ 548
- 6.9. PODSUMOWANIE 549

7 MULTIMEDIALNE SYSTEMY OPERACYJNE 555

- 7.1. WPROWADZENIE W TEMATYKĘ MULTIMEDIÓW 556
- 7.2. PLIKI MULTIMEDIALNE 561
 - 7.2.1. Kodowanie wideo 562
 - 7.2.2. Kodowanie audio 565
- 7.3. KOMPRESJA WIDEO 567
 - 7.3.1. Standard JPEG 568
 - 7.3.2. Standard MPEG 571
- 7.4. KOMPRESJA AUDIO 574
- 7.5. SZEREGOWANIE PROCESÓW MULTIMEDIALNYCH 577
 - 7.5.1. Szeregowanie procesów homogenicznych 577
 - 7.5.2. Szeregowanie w czasie rzeczywistym — przypadek ogólny 578
 - 7.5.3. Szeregowanie monotoniczne w częstotliwości 580
 - 7.5.4. Algorytm szeregowania EDF 581
- 7.6. WZORCE MULTIMEDIALNYCH SYSTEMÓW PLIKÓW 584
 - 7.6.1. Funkcje sterujące VCR 585
 - 7.6.2. Wideo niemal na życzenie 587
 - 7.6.3. Usługa wideo niemal na życzenie z funkcjami magnetowidu 589
- 7.7. ROZMIESZCZENIE PLIKÓW 591
 - 7.7.1. Umieszczanie pliku na pojedynczym dysku 591
 - 7.7.2. Dwie alternatywne strategie organizacji plików 592
 - 7.7.3. Rozmieszczenie plików w usłudze wideo niemal na życzenie 596
 - 7.7.4. Rozmieszczenie wielu plików na jednym dysku 598
 - 7.7.5. Rozmieszczenie plików na wielu dyskach 600

7.8.	BUFOROWANIE	603
7.8.1.	Buforowanie bloków	603
7.8.2.	Buforowanie plików	605
7.9.	SZEREGOWANIE OPERACJI DYSKOWYCH W SYSTEMACH MULTIMEDIALNYCH	606
7.9.1.	Statyczne szeregowanie operacji dyskowych	606
7.9.2.	Dynamiczne szeregowanie operacji dyskowych	608
7.10.	BADANIA NA TEMAT MULTIMEDIÓW	610
7.11.	PODSUMOWANIE	610

8 SYSTEMY WIELOPROCESOROWE 617

8.1.	SYSTEMY WIELOPROCESOROWE	620
8.1.1.	Sprzęt wieloprocessorowy	620
8.1.2.	Typy wieloprocessorowych systemów operacyjnych	630
8.1.3.	Synchronizacja w systemach wieloprocessorowych	634
8.1.4.	Szeregowanie w systemach wieloprocessorowych	640
8.2.	WIELOKOMPUTERY	646
8.2.1.	Sprzęt wielokomputerów	647
8.2.2.	Niskopoziomowe oprogramowanie komunikacyjne	651
8.2.3.	Oprogramowanie komunikacyjne poziomu użytkownika	654
8.2.4.	Zdalne wywołania procedur	657
8.2.5.	Rozproszona współdzielona pamięć	660
8.2.6.	Szeregowanie systemów wielokomputerowych	665
8.2.7.	Równoważenie obciążenia	666
8.3.	WIRTUALIZACJA	669
8.3.1.	Wymagania dla wirtualizacji	671
8.3.2.	Hipernadzorcy typu 1	672
8.3.3.	Hipernadzorcy typu 2	673
8.3.4.	Parawirtualizacja	675
8.3.5.	Wirtualizacja pamięci	678
8.3.6.	Wirtualizacja wejścia-wyjścia	679
8.3.7.	Urządzenia wirtualne	681
8.3.8.	Maszyny wirtualne na procesorach wielordzeniowych	681
8.3.9.	Problemy licencji	682

- 8.4. SYSTEMY ROZPROSZONE 683
 - 8.4.1. Sprzęt sieciowy 685
 - 8.4.2. Usługi i protokoły sieciowe 689
 - 8.4.3. Warstwa middleware bazująca na dokumentach 693
 - 8.4.4. Warstwa middleware bazująca na systemie plików 694
 - 8.4.5. Warstwa middleware bazująca na obiektach 700
 - 8.4.6. Warstwa middleware bazująca na koordynacji 701
 - 8.4.7. Siatki 707
- 8.5. BADANIA DOTYCZĄCE
SYSTEMÓW WIELOPROCESOROWYCH 708
- 8.6. PODSUMOWANIE 710

9 Bezpieczeństwo

717

- 9.1. ŚRODOWISKO BEZPIECZEŃSTWA 719
 - 9.1.1. Zagrożenia 720
 - 9.1.2. Intruzi 721
 - 9.1.3. Przypadkowa utrata danych 723
- 9.2. PODSTAWY KRYPTOGRAFII 723
 - 9.2.1. Kryptografia z kluczem tajnym 725
 - 9.2.2. Kryptografia z kluczem publicznym 726
 - 9.2.3. Funkcje jednokierunkowe 727
 - 9.2.4. Podpisy cyfrowe 727
 - 9.2.5. Moduł TPM 729
- 9.3. MECHANIZMY OCHRONY 730
 - 9.3.1. Domeny ochrony 730
 - 9.3.2. Listy kontroli dostępu 733
 - 9.3.3. Uprawnienia 736
 - 9.3.4. Systemy zaufane 740
 - 9.3.5. Zaufana baza obliczeniowa 742
 - 9.3.6. Modele formalne bezpiecznych systemów 743
 - 9.3.7. Bezpieczeństwo wielopoziomowe 745
 - 9.3.8. Ukryte kanały 748
- 9.4. UWIERZYTELNIANIE 753
 - 9.4.1. Uwierzytelnianie z wykorzystaniem haseł 755

9.4.2.	Uwierzytelnianie z wykorzystaniem obiektu fizycznego	765
9.4.3.	Uwierzytelnianie z wykorzystaniem technik biometrycznych	769
9.5.	ATAKI Z WEWNĄTRZ	772
9.5.1.	Bomby logiczne	773
9.5.2.	Tylne drzwi	773
9.5.3.	Podszywanie się pod ekran logowania	774
9.6.	WYKORZYSTYWANIE BŁĘDÓW W KODZIE	776
9.6.1.	Ataki z wykorzystaniem przepełnienia bufora	777
9.6.2.	Ataki z wykorzystaniem łańcuchów formatujących	780
9.6.3.	Ataki powrotu do biblioteki libc	782
9.6.4.	Ataki z wykorzystaniem przepełnień liczb całkowitych	783
9.6.5.	Ataki polegające na wstrzykiwaniu kodu	784
9.6.6.	Ataki polegające na rozszerzaniu uprawnień	786
9.7.	ZŁOŚLIWE OPROGRAMOWANIE	786
9.7.1.	Konie trojańskie	790
9.7.2.	Wirusy	793
9.7.3.	Robaki	805
9.7.4.	Oprogramowanie szpiegujące	808
9.7.5.	Rootkity	813
9.8.	ŚRODKI OBRONY	819
9.8.1.	Firewalle	820
9.8.2.	Techniki antywirusowe i antyantyvirusowe	822
9.8.3.	Podpisywanie kodu	830
9.8.4.	Wtrącanie do więzienia	832
9.8.5.	Wykrywanie włamań z użyciem modeli	833
9.8.6.	Izolowanie kodu mobilnego	835
9.8.7.	Bezpieczeństwo Javy	840
9.9.	BADANIA DOTYCZĄCE BEZPIECZEŃSTWA	843
9.10.	PODSUMOWANIE	844

10 Pierwsze studium przypadku: Linux 851

10.1.	HISTORIA SYSTEMÓW UNIX I LINUX	852
10.1.1.	UNICS	852
10.1.2.	PDP-11 UNIX	853
10.1.3.	Przenośny UNIX	855
10.1.4.	Berkeley UNIX	856
10.1.5.	Standard UNIX	857
10.1.6.	MINIX	858
10.1.7.	Linux	860
10.2.	PRZEGLĄD SYSTEMU LINUX	863
10.2.1.	Cele Linuksa	863
10.2.2.	Interfejsy systemu Linux	865
10.2.3.	Powłoka	867
10.2.4.	Programy użytkowe systemu Linux	870
10.2.5.	Struktura jądra	872
10.3.	PROCESY W SYSTEMIE LINUX	876
10.3.1.	Podstawowe pojęcia	876
10.3.2.	Wywołania systemowe Linuksa związane z zarządzaniem procesami	879
10.3.3.	Implementacja procesów i wątków w systemie Linux	884
10.3.4.	Szeregowanie w systemie Linux	892
10.3.5.	Uruchamianie systemu Linux	896
10.4.	ZARZĄDZANIE PAMIĘCIĄ W SYSTEMIE LINUX	899
10.4.1.	Podstawowe pojęcia	899
10.4.2.	Wywołania systemowe Linuksa odpowiedzialne za zarządzanie pamięcią	903
10.4.3.	Implementacja zarządzania pamięcią w systemie Linux	904
10.4.4.	Stronicowanie w systemie Linux	911
10.5.	OPERACJE WEJŚCIA-WYJŚCIA W SYSTEMIE LINUX	916
10.5.1.	Podstawowe pojęcia	916
10.5.2.	Obsługa sieci	917
10.5.3.	Wywołania systemowe wejścia-wyjścia w systemie Linux	919
10.5.4.	Implementacja wejścia-wyjścia w systemie Linux	921
10.5.5.	Moduły w systemie Linux	925

10.6. SYSTEM PLIKÓW LINUKSA	925
10.6.1. Podstawowe pojęcia	926
10.6.2. Wywołania systemu plików w Linuksie	931
10.6.3. Implementacja systemu plików Linuksa	936
10.6.4. NFS — sieciowy system plików	945
10.7. BEZPIECZEŃSTWO W SYSTEMIE LINUX	953
10.7.1. Podstawowe pojęcia	953
10.7.2. Wywołania systemowe Linuksa związane z bezpieczeństwem	956
10.7.3. Implementacja bezpieczeństwa w systemie Linux	957
10.8. PODSUMOWANIE	958

11 Drugie studium przypadku:

Windows Vista

965

11.1. HISTORIA SYSTEMU WINDOWS VISTA	965
11.1.1. Lata osiemdziesiąte: MS-DOS	966
11.1.2. Lata dziewięćdziesiąte: Windows na bazie MS-DOS-a	967
11.1.3. Lata dwutysięczne: Windows na bazie NT	967
11.1.4. Windows Vista	971
11.2. PROGRAMOWANIE SYSTEMU WINDOWS VISTA	972
11.2.1. Rdzenny interfejs programowania aplikacji (API) systemu NT	975
11.2.2. Interfejs programowania aplikacji Win32	980
11.2.3. Rejestr systemu Windows	984
11.3. STRUKTURA SYSTEMU	987
11.3.1. Struktura systemu operacyjnego	988
11.3.2. Uruchamianie systemu Windows Vista	1006
11.3.3. Implementacja menedżera obiektów	1007
11.3.4. Podsystemy, biblioteki DLL i usługi trybu użytkownika	1019
11.4. PROCESY I WĄTKI SYSTEMU WINDOWS VISTA	1023
11.4.1. Podstawowe pojęcia	1023
11.4.2. Wywołania API związane z zarządzaniem zadaniami, procesami, wątkami i włóknami	1029
11.4.3. Implementacja procesów i wątków	1036

11.5.	ZARZĄDZANIE PAMIĘCIĄ	1045
11.5.1.	Podstawowe pojęcia	1045
11.5.2.	Wywołania systemowe związane z zarządzaniem pamięcią	1051
11.5.3.	Implementacja zarządzania pamięcią	1052
11.6.	PAMIĘĆ PODRĘCZNA SYSTEMU WINDOWS VISTA	1063
11.7.	OPERACJE WEJŚCIA-WYJŚCIA W SYSTEMIE WINDOWS VISTA	1066
11.7.1.	Podstawowe pojęcia	1067
11.7.2.	Wywołania API związane z operacjami wejścia-wyjścia	1069
11.7.3.	Implementacja systemu wejścia-wyjścia	1072
11.8.	SYSTEM PLIKÓW NT SYSTEMU WINDOWS	1079
11.8.1.	Podstawowe pojęcia	1079
11.8.2.	Implementacja systemu plików NTFS	1081
11.9.	BEZPIECZEŃSTWO W SYSTEMIE WINDOWS VISTA	1093
11.9.1.	Podstawowe pojęcia	1095
11.9.2.	Wywołania API związane z bezpieczeństwem	1097
11.9.3.	Implementacja bezpieczeństwa	1098
11.10.	PODSUMOWANIE	1102

12 Trzecie studium przypadku: Symbian OS

1107

12.1.	HISTORIA SYSTEMU SYMBIAN OS	1108
12.1.1.	Korzenie systemu operacyjnego Symbian OS: Psion i EPOC	1108
12.1.2.	Symbian OS 6	1110
12.1.3.	Symbian OS 7	1111
12.1.4.	Współczesna wersja systemu operacyjnego Symbian OS	1111
12.2.	PRZEGLĄD SYSTEMU SYMBIAN OS	1111
12.2.1.	Obiektowość	1112
12.2.2.	Projekt mikrojądra	1113
12.2.3.	Nanojądro systemu Symbian OS	1114
12.2.4.	Dostęp do zasobów w trybie klient-serwer	1115

12.2.5.	Funkcje znane z większych systemów operacyjnych	1116
12.2.6.	Komunikacja i multimedia	1117
12.3.	PROCESY I WĄTKI W SYSTEMIE SYMBIAN OS	1117
12.3.1.	Wątki i nanowątki	1118
12.3.2.	Procesy	1119
12.3.3.	Obiekty aktywne	1120
12.3.4.	Komunikacja międzyprocesowa	1121
12.4.	ZARZĄDZANIE PAMIĘCIĄ	1121
12.4.1.	Systemy pozbawione pamięci wirtualnej	1122
12.4.2.	Adresowanie pamięci w systemie Symbian OS	1124
12.5.	WEJŚCIE I WYJŚCIE	1127
12.5.1.	Sterowniki urządzeń	1127
12.5.2.	Rozszerzenia jądra	1128
12.5.3.	Bezpośredni dostęp do pamięci (DMA)	1128
12.5.4.	Przypadek specjalny: nośniki pamięci	1129
12.5.5.	Blokujące operacje wejścia-wyjścia	1129
12.5.6.	Nośniki wymienne	1130
12.6.	SYSTEMY PRZECHOWYWANIA DANYCH	1130
12.6.1.	Systemy plików dla urządzeń mobilnych	1131
12.6.2.	Systemy plików systemu operacyjnego Symbian OS	1132
12.6.3.	Bezpieczeństwo i ochrona systemu plików	1132
12.7.	BEZPIECZEŃSTWO W SYSTEMIE SYMBIAN OS	1133
12.8.	KOMUNIKACJA W SYSTEMIE SYMBIAN OS	1136
12.8.1.	Podstawowa infrastruktura	1136
12.8.2.	Bardziej szczegółowa analiza infrastruktury komunikacji	1137
12.9.	PODSUMOWANIE	1141

13 Projekt systemu operacyjnego

1143

13.1.	ISTOTA PROBLEMÓW ZWIĄZANYCH Z PROJEKTOWANIEM SYSTEMÓW	1144
13.1.1.	Cele	1144
13.1.2.	Dlaczego projektowanie systemów operacyjnych jest takie trudne?	1146

13.2.	PROJEKT INTERFEJSU	1148
13.2.1.	Zalecenia projektowe	1148
13.2.2.	Paradygmaty	1150
13.2.3.	Interfejs wywołań systemowych	1155
13.3.	IMPLEMENTACJA	1158
13.3.1.	Struktura systemu	1158
13.3.2.	Mechanizm kontra strategia	1163
13.3.3.	Ortogonalność	1164
13.3.4.	Nazewnictwo	1165
13.3.5.	Czas kojarzenia nazw	1167
13.3.6.	Struktury statyczne kontra struktury dynamiczne	1168
13.3.7.	Implementacja z góry na dół kontra implementacja z dołu do góry	1170
13.3.8.	Przydatne techniki	1171
13.4.	WYDAJNOŚĆ	1177
13.4.1.	Dlaczego systemy operacyjne są powolne?	1177
13.4.2.	Co należy optymalizować?	1179
13.4.3.	Dylemat przestrzeni – czas	1180
13.4.4.	Buforowanie	1183
13.4.5.	Wskazówki	1185
13.4.6.	Wykorzystywanie efektu lokalności	1185
13.4.7.	Optymalizacja z myślą o typowych przypadkach	1186
13.5.	ZARZĄDZANIE PROJEKTEM	1186
13.5.1.	Mityczny osobomiesiąc	1187
13.5.2.	Struktura zespołu	1189
13.5.3.	Znaczenie doświadczenia	1191
13.5.4.	Nie istnieje jedno cudowne rozwiązanie	1192
13.6.	TRENDY W ŚWIECIE PROJEKTÓW SYSTEMÓW OPERACYJNYCH	1192
13.6.1.	Wirtualizacja	1193
13.6.2.	Układy wielordzeniowe	1193
13.6.3.	Systemy operacyjne z wielkimi przestrzeniami adresowymi	1194
13.6.4.	Komunikacja sieciowa	1195
13.6.5.	Systemy równoległe i rozproszone	1196
13.6.6.	Multimedia	1196
13.6.7.	Komputery zasilane bateriami	1197

13.6.8. Systemy wbudowane	1197
13.6.9. Węzły czujników	1198
13.7. PODSUMOWANIE	1198

14 Lista publikacji i bibliografia 1203

14.1. SUGEROWANE PUBLIKACJE DODATKOWE	1203
14.1.1. Publikacje wprowadzające i ogólne	1204
14.1.2. Procesy i wątki	1204
14.1.3. Zarządzanie pamięcią	1205
14.1.4. Wejście-wyjście	1205
14.1.5. Systemy plików	1206
14.1.6. Zakleszczenia	1206
14.1.7. Multimedialne systemy operacyjne	1207
14.1.8. Systemy wieloprocesorowe	1208
14.1.9. Bezpieczeństwo	1209
14.1.10. System Linux	1211
14.1.11. System Windows Vista	1212
14.1.12. System Symbian OS	1213
14.1.13. Zasady projektowe	1213
14.2. BIBLIOGRAFIA W PORZĄDKU ALFABETYCZNYM	1214

Skorowidz 1247

2

PROCESY I WĄTKI

Zanim rozpocznie się szczegółowe studium tego, w jaki sposób systemy operacyjne są zaprojektowane i skonstruowane, warto przypomnieć, że kluczowym pojęciem we wszystkich systemach operacyjnych jest **proces**: abstrakcja działającego programu. Wszystkie pozostałe elementy systemu operacyjnego bazują na pojęciu procesu, dlatego jest bardzo ważne, aby projektant systemu operacyjnego (a także student) jak najszybciej dobrze zapoznał się z pojęciem procesu.

Procesy to jedne z najstarszych i najważniejszych abstrakcji występujących w systemach operacyjnych. Zapewniają one możliwość wykonywania (pseudo-) współbieżnych operacji nawet wtedy, gdy dostępny jest tylko jeden procesor. Przekształcają one pojedynczy procesor CPU w wiele wirtualnych procesorów. Bez abstrakcji procesów istnienie współczesnej techniki komputerowej byłoby niemożliwe. W niniejszym rozdziale przedstawimy szczegółowe informacje na temat tego, czym są procesy oraz ich pierwsi kuzynowie — wątki.

2.1. PROCESY

Wszystkie nowoczesne komputery bardzo często wykonują wiele operacji jednocześnie. Osoby przyzwyczajone do pracy z komputerami osobistymi mogą nie być do końca świadome tego faktu, zatem kilka przykładów pozwoli przybliżyć to zagadnienie. Na początek rozważmy serwer WWW. Żądania stron WWW mogą nadchodzić z wielu miejsc. Kiedy przychodzi żądanie, serwer sprawdza, czy potrzebna strona znajduje się w pamięci podręcznej. Jeśli tak, jest przesyłana do klienta. Jeśli nie, inicjowane jest żądanie dyskowe w celu jej pobrania. Jednak z perspektywy procesora

obsługa żądań dyskowych zajmuje wieczność. W czasie oczekiwania na zakończenie obsługi żądania na dysk może nadejść wiele kolejnych żądań. Jeśli w systemie jest wiele dysków niektóre z żądań może być skierowanych na inne dyski na długo przed obsłużeniem pierwszego żądania. Oczywiście, że potrzebny jest sposób zamodelowania i zarządzania tą współbieżnością. Do tego celu można wykorzystać procesy (a w szczególności wątki).

Teraz rozważmy komputer osobisty użytkownika. Podczas rozruchu systemu następuje start wielu procesów. Często użytkownik nie jest tego świadomy. Na przykład może być uruchomiony proces oczekujący na wchodzące wiadomości e-mail. Inny uruchomiony proces może działać w imieniu programu antywirusowego i sprawdzać okresowo, czy są dostępne jakieś nowe definicje wirusów. Dodatkowo mogą działać jawne procesy użytkownika — na przykład drukujące pliki lub wypalające płytę CD — podczas gdy użytkownik przegląda strony WWW. Działaniami tymi trzeba zarządzać. W tym przypadku bardzo przydaje się system z obsługą wieloprogramowości, obsługujący wiele procesów jednocześnie.

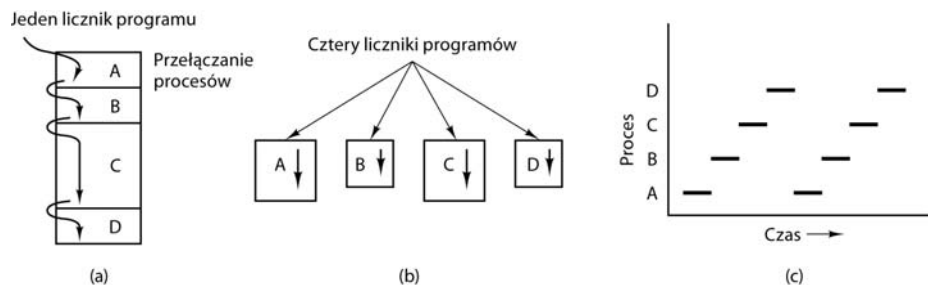
W każdym systemie wieloprogramowym procesor szybko przełącza się pomiędzy procesami, poświęcając każdemu z nich po kolei dziesiątki bądź setki milisekund. Chociaż ściśle rzecz biorąc w dowolnym momencie procesor realizuje tylko jeden proces, w ciągu sekundy może obsłużyć ich wiele, co daje iluzję współbieżności. Czasami w tym kontekście mówi się o **pseudowspółbieżności**, dla odróżnienia jej od rzeczywistej, sprzętowej współbieżności systemów **wieloprocessorowych** (wyposażonych w dwa procesory współdzielące tę samą fizyczną pamięć lub większą liczbę takich procesorów). Śledzenie wielu równoległych działań jest bardzo trudne. Z tego powodu projektanci systemów operacyjnych w ciągu wielu lat opracowali model pojęciowy (procesów sekwencyjnych), które ułatwiają obsługę współbieżności. Ten model, jego zastosowania oraz kilka innych konsekwencji stanowią temat niniejszego rozdziału.

2.1.1. Model procesów

W tym modelu całe oprogramowanie możliwe do uruchomienia w komputerze — czasami włącznie z systemem operacyjnym — jest zorganizowane w postaci zbioru **procesów sekwencyjnych** (lub w skrócie **procesów**). Proces jest egzemplarzem uruchomionego programu włącznie z bieżącymi wartościami licznika programu, rejestrów i zmiennych. Pojęciowo każdy proces ma własny wirtualny procesor CPU. Oczywiście w rzeczywistości procesor fizyczny przełącza się od procesu do procesu. Aby jednak zrozumieć system, znacznie łatwiej jest myśleć o kolekcji procesów działających (pseudo) współbieżnie, niż próbować śledzić to, jak procesor przełącza się od programu do programu. To szybkie przełączanie się procesora jest określane jako **wieloprogramowość**, o czy mówiliśmy w rozdziale 1.

Na rysunku 2.1(a) pokazaliśmy komputer, w którym w pamięci działają w trybie wieloprogramowym cztery programy. Na rysunku 2.1(b) widać cztery procesy —

każdy ma własny przepływ sterowania (tzn. własny logiczny licznik programu) i każdy działa niezależnie od pozostałych. Oczywiście jest tylko jeden fizyczny licznik programu, dlatego kiedy działa wybrany proces, jego logiczny licznik programu jest kopiowany do rzeczywistego licznika programu. Kiedy proces kończy działanie (na pewien czas), jego fizyczny licznik programu jest zapisywany w logicznym liczniku programu umieszczonym w pamięci. Na rysunku 2.1(c) widać, że w dłuższym przedziale czasu nastąpił postęp we wszystkich procesach, jednak w danym momencie działa tylko jeden proces.



Rysunek 2.1. (a) Cztery programy uruchomione w trybie wieloprogramowym;
(b) pojęciowy model czterech niezależnych od siebie procesów sekwencyjnych;
(c) w wybranym momencie jest aktywny tylko jeden program

W tym rozdziale założymy, że jest tylko jeden procesor CPU. Coraz częściej jednak takie założenie okazuje się nieprawdziwe. Nowe układy często są wielordzeniowe — mają dwa procesory, cztery lub większą ich liczbę. O układach wielordzeniowych i systemach wieloprocessorowych powiemy więcej w rozdziale 8. Na razie będzie prościej, jeśli przyjmiemy, że maszyna wykorzystuje jednorazowo tylko jeden procesor. Jeśli zatem mówimy, że procesor w danym momencie może wykonywać tylko jeden proces, to jeśli zawiera dwa rdzenie (lub dwa procesory), na każdym z nich w określonym momencie może działać jeden proces.

Ze względu na szybkie przełączanie się procesora pomiędzy procesami, tempo, w jakim proces wykonuje obliczenia, nie jest jednolite, a nawet trudne do powtórzenia w przypadku ponownego uruchomienia tego samego procesu. A zatem nie można programować procesów z wbudowanymi założeniami dotyczącymi czasu działania. Rozważmy dla przykładu proces wejścia-wyjścia, który uruchamia taśmę streamera w celu odtworzenia plików z kopii zapasowej, następnie wykonuje 10 000 iteracji pustej pętli w celu rozpędzenia streamera do właściwej szybkości i, na koniec, wydaje polecenie przeczytania pierwszego rekordu. Jeśli procesor zdecyduje się na przełączenie do innego procesu podczas trwania pętli, w której streamer się rozpędza, proces obsługi taśmy nie będzie mógł ponownie się uruchomić do momentu, kiedy pierwszy rekord znajdzie się za głowicą czytającą. Kiedy proces obowiązują tak ścisłe wymagania działania w czasie rzeczywistym — to znaczy określone zdarzenia *muszą* wystąpić w ciągu określonej liczby milisekund — trzeba przedsięwziąć

specjalne środki w celu zapewnienia, że tak się stanie. Zazwyczaj jednak większości procesów nie dotyczą ograniczenia wieloprogramowości procesora czy też względne szybkości działania różnych procesów.

Różnica pomiędzy procesem a programem jest subtelna, ale ma kluczowe znaczenie. Do wyjaśnienia tej różnicy posłużymy się analogią. Załóżmy, że pewien informatyk o zdolnościach kulinarnych piecze urodzinowy tort dla swojej córki. Ma do dyspozycji przepis na tort urodzinowy oraz kuchnię dobrze wyposażoną we wszystkie składniki: mąkę, jajka, cukier, aromat waniliowy itp. W tym przykładzie przepis spełnia rolę programu (tzn. algorytmu wyrażonego w odpowiedniej notacji), informatyk jest procesorem (CPU), natomiast składniki ciasta odgrywają rolę danych wejściowych. Proces jest operacją, w której informatyk czyta przepis, dodaje składniki i piecze ciasto.

Wyobraźmy sobie teraz, że z krzykiem wbiega syn informatyka i mówi, że użądliła go pszczoła. Informatyk zapamiętuje, w którym miejscu przepisu się znajdował (zapisuje bieżący stan procesu), bierze książkę o pierwszej pomocy i zaczyna postępować zgodnie z zapisanymi w niej wskazówkami. W tym momencie widzimy przełączenie się procesora z jednego procesu (pieczenie) do procesu o wyższym priorytecie (udzielanie pomocy medycznej). Przy czym każdy z procesów ma inny program (przepis na ciasto, książka pierwszej pomocy medycznej). Kiedy informatyk poradzi sobie z opatrzeniem użądlenia, powraca do pieczenia ciasta i kontynuuje od miejsca, w którym skończył.

Kluczowe znaczenie ma uświadomienie sobie, że proces jest pewnym działaniem. Charakteryzuje się programem, wejściem, wyjściem i stanem. Jeden procesor może być współdzielony przez kilka procesów za pomocą algorytmu szeregowania. Algorytm ten decyduje, w którym zatrzymać pracę nad jednym programem i rozpocząć obsługę innego.

Warto zwrócić uwagę na to, że jeśli program uruchomi się dwa razy, liczy się jako dwa procesy. Na przykład często istnieje możliwość dwukrotnego uruchomienia edytora tekstu lub jednoczesnego drukowania dwóch plików, jeśli system komputerowy jest wyposażony w dwie drukarki. Fakt, że dwa działające procesy korzystają z tego samego programu, nie ma znaczenia — są to oddzielne procesy. System operacyjny może mieć możliwość współdzielenia kodu pomiędzy nimi w taki sposób, że w pamięci znajduje się jedna kopia. Jest to jednak szczegół techniczny, który nie zmienia faktu działania dwóch procesów.

2.1.2. Tworzenie procesów

Systemy operacyjne wymagają sposobu tworzenia procesów. W bardzo prostych systemach lub w systemach zaprojektowanych do uruchamiania tylko jednej aplikacji (na przykład kontrolera w kuchence mikrofalowej), bywa możliwe zainicjowanie wszystkich potrzebnych procesów natychmiast po uruchomieniu systemu. Jednak

w systemach ogólnego przeznaczenia potrzebny jest sposób tworzenia i niszczenia procesów podczas ich działania. W tym punkcie przyjrzymy się niektórym spośród tych mechanizmów.

Są cztery podstawowe zdarzenia, które powodują tworzenie procesów:

1. Inicjalizacja systemu.
2. Uruchomienie wywołania systemowego tworzącego proces przez działający proces.
3. Żądanie użytkownika utworzenia nowego procesu.
4. Zainicjowanie zadania wsadowego.

W momencie rozruchu systemu operacyjnego zwykle tworzonych jest kilka procesów. Niektóre z nich są procesami pierwszego planu — to znaczy są to procesy, które komunikują się z użytkownikami i wykonują dla nich pracę. Inne są procesami drugoplanowymi, które nie są powiązane z określonym użytkownikiem, ale spełniają pewną specyficzną funkcję. Na przykład jeden proces drugoplanowy może być zaprojektowany do akceptacji wchodzących wiadomości e-mail. Taki proces może być uśpiony przez większość dnia i nagle się uaktywnić, kiedy nadchodzi wiadomość e-mail. Inny proces drugoplanowy może być zaprojektowany do akceptacji wchodzących żądań stron WWW zapisanych na serwerze. Proces ten budzi się w momencie odebrania żądania strony WWW w celu jego obsłużenia. Procesy działające na drugim planie, które są przeznaczone do obsługi pewnych operacji, takich jak odbiór wiadomości e-mail, serwowanie stron WWW, aktualności, drukowanie itp., są określane jako **demony**. W dużych systemach zwykle działają dziesiątki takich procesów. W systemie UNIX, aby wyświetlić listę działających procesów, można skorzystać z programu ps. W systemie Windows można skorzystać z menedżera zadań.

Procesy mogą być tworzone nie tylko w czasie rozruchu, ale także później. Działający proces często wydaje wywołanie systemowe w celu utworzenia jednego lub kilku nowych procesów mających pomóc w realizacji zadania. Tworzenie nowych procesów jest szczególnie przydatne, kiedy pracę do wykonania można łatwo sformułować w kontekście kilku związanych ze sobą, ale poza tym niezależnych, współdziałających ze sobą procesów. Jeśli na przykład przez sieć jest pobierana duża ilość danych w celu ich późniejszego przetwarzania, to można utworzyć jeden proces, który pobiera dane i umieszcza je we współdzielonym buforze, oraz drugi proces, który usuwa dane z bufora i je przetwarza. W systemie wieloprocessorowym, w którym każdy z procesów może działać na innym procesorze, zadanie może być wykonane w krótszym czasie.

W systemach interaktywnych użytkownicy mogą uruchomić program poprzez wpisanie polecenia lub kliknięcie (ewentualnie dwukrotne kliknięcie) ikony. Wykonanie dowolnej z tych operacji inicjuje nowy proces i uruchamia w nim wskazany

program. W systemach uniksowych bazujących na systemie X Window nowy proces przejmie okno, w którym został uruchomiony. W systemie Microsoft Windows po uruchomieniu procesu nie ma on przypisanego okna. Może on jednak stworzyć jedno (lub więcej) okien i większość systemów to robi. W obydwu systemach użytkownicy mają możliwość jednoczesnego otwarcia wielu okien, w których działają jakieś procesy. Za pomocą myszy użytkownik może wybrać okno i komunikować się z procesem, na przykład podawać dane wejściowe wtedy, kiedy są potrzebne.

Ostatnia sytuacja, w której są tworzone procesy, dotyczy tylko systemów wsadowych w dużych komputerach mainframe. W systemach tego typu użytkownicy mogą przysyłać do systemu zadania wsadowe (czasami zdalnie). Kiedy system operacyjny zdecyduje, że ma zasoby wystarczające do uruchomienia innego zadania, tworzy nowy proces i uruchamia następne zadanie z kolejki.

Z technicznego punktu widzenia we wszystkich tych sytuacjach proces tworzy się poprzez zlecenie istniejącemu procesowi wykonania wywołania systemowego tworzenia procesów. Może to być działający proces użytkownika, proces systemowy, wywołany z klawiatury lub za pomocą myszy, albo proces zarządzania zadaniami systemowymi. Proces ten wykonuje wywołanie systemowe tworzące nowy proces. To wywołanie systemowe zleca systemowi operacyjnemu utworzenie nowego procesu i wskazuje, w sposób pośredni lub bezpośredni, jaki program należy w nim uruchomić.

W systemie UNIX istnieje tylko jedno wywołanie systemowe do utworzenia nowego procesu: `fork`. Wywołanie to tworzy dokładny klon procesu wywołującego. Po wykonaniu instrukcji `fork` procesy rodzic i dziecko mają ten sam obraz pamięci, te same zmienne środowiskowe oraz te same otwarte pliki. Po prostu są identyczne. Wtedy zazwyczaj proces-dziecko uruchamia wywołanie `execve` lub podobne wywołanie systemowe w celu zmiany obrazu pamięci i uruchomienia nowego programu. Kiedy użytkownik wpisze polecenie w środowisku powłoki, na przykład `sort`, powłoka najpierw tworzy proces-dziecko za pomocą wywołania `fork`, a następnie proces-dziecko wykonuje polecenie `sort`. Powodem, dla którego dokonuje się ten dwuetapowy proces, jest umożliwienie procesowi-dziecku manipulowania deskryptorami plików po wykonaniu wywołania `fork`, ale przed wywołaniem `execve` w celu przekierowania standardowego wejścia, standardowego wyjścia oraz standardowego urządzenia błędów.

Dla odróżnienia w systemie Windows jedna funkcja interfejsu Win32 — `CreateProcess` — jest odpowiedzialna zarówno za utworzenie procesu, jak i załadowanie odpowiedniego programu do nowego procesu. Wywołanie to ma 10 parametrów. Są to program do uruchomienia, parametry wiersza polecenia przekazywane do programu, różne atrybuty zabezpieczeń, bity decydujące o tym, czy otwarte pliki będą dziedziczone, informacje dotyczące priorytetów, specyfikacja okna, jakie ma być utworzone dla procesu (jeśli proces ma mieć okno), oraz wskaźnik do struktury, w której są zwracane do procesu wywołującego informacje o nowo tworzonym procesie. Oprócz wywołania `CreateProcess` interfejs Win32 zawiera około 100 innych funkcji do zarządzania i synchronizowania procesów oraz wykonywania powiązanych z tym operacji.

Zarówno w systemie UNIX, jak i Windows po utworzeniu procesu rodzic i dziecko mają osobne przestrzenie adresowe. Jeśli dowolny z procesów zmieni słowo w swojej przestrzeni adresowej, zmiana nie jest widoczna dla drugiego procesu. W systemie UNIX początkowa przestrzeń adresowa procesu-dziecka jest *kopią* przestrzeni adresowej procesu-rodzica. Są to jednak całkowicie odrębne przestrzenie adresowe. Zapisywalna pamięć nie jest współdzielona pomiędzy procesami (w niektórych implementacjach Uniksa tekst programu jest współdzielony pomiędzy procesami rodzica i dziecka, ponieważ nie może on być modyfikowany). Nowo utworzony proces może jednak współdzielić niektóre inne zasoby procesu swojego twórcy — na przykład otwarte pliki. W systemie Windows przestrzenie adresowe procesów rodzica i dziecka od samego początku są różne.

2.1.3. Kończenie działania procesów

Po utworzeniu proces zaczyna działanie i wykonuje swoje zadania. Nic jednak nie trwa wiecznie — nawet procesy. Prędzej czy później nowy proces zakończy swoje działanie. Zwykle dzieje się to z powodu jednego z poniższych warunków:

1. Normalne zakończenie pracy (dobrowolnie).
2. Zakończenie pracy w wyniku błędu (dobrowolnie).
3. Błąd krytyczny (przymusowo).
4. Zniszczenie przez inny proces (przymusowo).

Większość procesów kończy działanie dlatego, że wykonały swoją pracę. Kiedy kompilator skompiluje program, wykonuje wywołanie systemowe, które informuje system operacyjny o zakończeniu pracy. Tym wywołaniem jest `exit` w systemie UNIX oraz `ExitProcess` w systemie Windows. W programach wyposażonych w interfejs ekranowy zwykle są mechanizmy pozwalające na dobrowolne zakończenie działania. W edytorach tekstu, przeglądarkach internetowych i podobnych im programach zawsze jest ikona lub polecenie menu, które użytkownik może kliknąć, aby zlecić procesowi usunięcie otwartych plików tymczasowych i zakończenie działania.

Innym powodem zakończenia pracy jest sytuacja, w której proces wykryje błąd krytyczny.

Jeśli na przykład użytkownik wpisze polecenie:

```
cc foo.c
```

w celu skompilowania programu `foo.c`, a taki plik nie istnieje, to kompilator po prostu skończy działanie. Procesy interaktywne wyposażone w interfejsy ekranowe zwykle nie kończą działania, jeśli zostaną do nich przekazane błędne parametry. Zamiast tego wyświetlają okno dialogowe z prośbą do użytkownika o ponowienie próby.

Trzecim powodem zakończenia pracy jest błąd spowodowany przez proces — często wynikający z błędów w programie. Może to być uruchomienie niedozwolonej instrukcji, odwołanie się do nieistniejącego obszaru pamięci lub dzielenie przez zero. W niektórych systemach (na przykład w Uniksie) proces może poinformować system operacyjny, że sam chce obsłużyć określone błędy. W takim przypadku, jeśli wystąpi błąd, proces otrzymuje sygnał (przerwanie), zamiast zakończyć pracę.

Czwartym powodem, dla którego proces może zakończyć działanie, jest wykonanie wywołania systemowego, które zleca systemowi operacyjnemu zniszczenie innego procesu. W Uniksie można to zrobić za pomocą wywołania systemowego `kill`. Odpowiednikiem tego wywołania w interfejsie Win32 API jest `TerminateProcess`. W obu przypadkach proces niszczący musi posiadać odpowiednie uprawnienia do niszczenia innych procesów. W niektórych systemach zakończenie procesu — niezależnie od tego, czy jest wykonywane dobrowolnie, czy przymusowo — wiąże się z zakończeniem wszystkich procesów utworzonych przez ten proces. Jednak w taki sposób nie działa ani system UNIX, ani Windows.

2.1.4. Hierarchie procesów

W niektórych systemach, kiedy proces utworzy inny proces, to proces-rodzic jest w pewien sposób związany z procesem-dzieckiem. Proces-dziecko sam może tworzyć kolejne procesy, co formuje hierarchię procesów. Zwróćmy uwagę, że w odróżnieniu od roślin i zwierząt rozmnażających się płciowo proces może mieć tylko jednego rodzica (ale zero, jedno dziecko lub więcej dzieci).

W Uniksie proces wraz z wszystkimi jego dziećmi i dalszymi potomkami tworzy grupę procesów. Kiedy użytkownik wysła sygnał z klawiatury, sygnał ten jest dostarczany do wszystkich członków grupy procesów, które w danym momencie są powiązane z klawiaturą (zwykle są to wszystkie aktywne procesy utworzone w bieżącym oknie). Każdy proces może indywidualnie przechwycić sygnał, zignorować go lub podjąć działanie domyślne — to znaczy zostać zniszczonym przez sygnał.

W celu przedstawienia innego przykładu sytuacji, w której hierarchia procesów odgrywa rolę, przyjrzyjmy się sposobowi, w jaki system UNIX inicjuje się podczas rozruchu. W obrazie rozruchowym występuje specjalny proces o nazwie `init`. Kiedy rozpoczyna działanie, odczytuje plik i informuje o liczbie dostępnych terminali. Następnie tworzy po jednym nowym procesie na terminal. Procesy te czekają, aż ktoś się zaloguje. Kiedy logowanie zakończy się pomyślnie, proces logowania uruchamia powłokę, która jest gotowa na przyjmowanie poleceń. Polecenia te mogą uruchamiać nowe procesy itd. Tak więc wszystkie procesy w całym systemie należą do tego samego drzewa — jego korzeniem jest proces `init`.

Dla odróżnienia w systemie Windows nie występuje pojęcie hierarchii procesów. Wszystkie procesy są sobie równe. Jedyną oznaką hierarchii procesu jest to, że podczas tworzenia procesu rodzic otrzymuje specjalny znacznik (nazywany **uchwytem** — ang. *handle*), który może wykorzystać do zarządzania dzieckiem. Może jednak

swobodnie przekazać ten znacznik do innego procesu i w ten sposób zdezaktualizować hierarchię. Procesy w Unikse nie mają możliwości „wydziedziczenia” swoich dzieci.

2.1.5. Stany procesów

Chociaż każdy proces jest niezależnym podmiotem, posiadającym własny licznik programu i wewnętrzny stan, procesy często muszą się komunikować z innymi procesami. Jeden proces może generować wyjście, które inny proces wykorzysta jako wejście. W poleceniu powłoki:

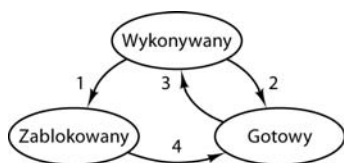
```
cat rozdzial1 rozdzial2 rozdzial3 | grep drzewo
```

pierwszy proces uruchamia polecenie `cat`, łączy i wyprowadza trzy pliki. Drugi proces uruchamia polecenie `grep`, wybiera wszystkie wiersze zawierające słowo „drzewo”. W zależności od względnej szybkości obu procesów (co z kolei zależy zarówno od względnej złożoności programów, jak i tego, ile czasu procesora każdy z nich ma do dyspozycji), może się zdarzyć, że polecenie `grep` będzie gotowe do działania, ale nie będą na nie czekały żadne dane wejściowe. Proces będzie się musiał zablokować do czasu, aż będą one dostępne.

Proces blokuje się, ponieważ z logicznego punktu widzenia nie może kontynuować działania. Zazwyczaj dzieje się tak dlatego, że oczekuje na dane wejściowe, które jeszcze nie są dostępne. Jest również możliwe, że proces, który jest gotowy i zdolny do działania, zostanie zatrzymany ze względu na to, że system operacyjny zdecydował się przydzielić procesor na pewien czas jakiemuś innemu procesowi. Te dwie sytuacje diametralnie różnią się od siebie. W pierwszym przypadku wstrzymanie pracy jest ściśle związane z charakterem problemu (nie można przetworzyć wiersza poleceń wprowadzanego przez użytkownika do czasu, kiedy użytkownik go nie wprowadzi). W drugim przypadku to techniczne aspekty systemu (niewystarczająca liczba procesorów do tego, aby każdy proces otrzymał swój prywatny procesor). Na rysunku 2.2 pokazano diagram stanów pokazujący trzy stany, w jakich może znajdować się proces:

1. Działanie (rzeczywiste korzystanie z procesora w tym momencie).
2. Gotowość (proces może działać, ale jest tymczasowo wstrzymany, aby inny proces mógł działać).
3. Blokada (proces nie może działać do momentu, w którym wydarzy się jakieś zewnętrzne zdarzenie).

Z logicznego punktu widzenia pierwsze dwa stany są do siebie podobne. W obu przypadkach proces chce działać, ale w drugim przypadku chwilowo brakuje dla niego czasu procesora. Trzeci stan różni się od pierwszych dwóch w tym sensie, że proces nie może działać nawet wtedy, gdy procesor w tym czasie nie ma innego zajęcia.



1. Proces blokuje się w oczekiwaniu na dane wejściowe
2. Program szeregujący przydzielił procesor innemu procesowi
3. Program szeregujący przydzielił procesor temu procesowi
4. Dane wejściowe stają się dostępne

Rysunek 2.2. Proces może być w stanie działania, blokady lub gotowości. Na rysunku pokazano przejścia pomiędzy tymi stanami

Tak jak pokazano na rysunku, pomiędzy tymi trzema stanami możliwe są cztery przejścia. Przejście nr 1 występuje wtedy, kiedy system operacyjny wykryje, że proces nie może kontynuować działania. W niektórych systemach proces może wywołać wywołanie systemowe, na przykład pause, w celu przejścia do stanu zablokowania. W innych systemach, w tym w Uniksie, kiedy proces czyta dane z potoku lub pliku specjalnego (na przykład terminala) i dane wejściowe są niedostępne, jest automatycznie blokowany.

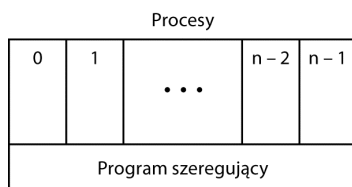
Przejścia nr 2 i nr 3 są realizowane przez program szeregujący (ang. *process scheduler*) — część systemu operacyjnego, a procesy nie są o tym nawet informowane. Przejście nr 2 zachodzi wtedy, gdy program szeregujący zdecyduje, że działający proces działał wystarczająco długo i nadszedł czas, by przydzielić czas procesora jakiemuś innemu procesowi. Przejście nr 3 zachodzi wtedy, gdy wszystkie inne procesy skorzystały ze swojego udziału i nadszedł czas na to, by pierwszy proces otrzymał procesor i wznowił działanie. Zadanie szeregowania procesów — to znaczy decydowania o tym, który proces powinien się uruchomić, kiedy i na jak długo — jest bardzo ważne. Przyjrzymy się mu bliżej w dalszej części tego rozdziału. Opracowano wiele algorytmów mających na celu zapewnienie równowagi pomiędzy wymaganiami wydajności systemu jako całości oraz sprawiedliwego przydziału procesora do indywidualnych procesów. Niektóre z tych algorytmów omówimy w dalszej części niniejszego rozdziału.

Przejście nr 4 występuje wtedy, gdy zachodzi zewnętrzne zdarzenie, na które proces oczekiwał (na przykład nadejście danych wejściowych). Jeśli w tym momencie nie działa żaden inny proces, zajdzie przejście nr 3 i proces rozpocznie działanie. W innym przypadku może być zmuszony do oczekiwania w stanie *gotowości* przez pewien czas, aż procesor stanie się dostępny i nadejdzie jego kolejka.

Wykorzystanie modelu procesów znacznie ułatwia myślenie o tym, co dzieje się wewnątrz systemu. Niektóre procesy uruchamiają programy realizujące polecenia wprowadzane przez użytkownika. Inne procesy są częścią systemu i obsługują takie zadania, jak obsługa żądań usług plikowych lub zarządzanie szczegółami dotyczącymi uruchamiania napędu dysku lub taśm. Kiedy zachodzi przerwanie dyskowe, system podejmuje decyzję o zatrzymaniu działania bieżącego procesu i uruchamia proces dyskowy, który był zablokowany w oczekiwaniu na to przerwanie. Tak więc zamiast myśleć o przerwaniach, możemy myśleć o procesach użytkownika, procesach dysku, procesach terminala itp., które blokują się w czasie oczekiwania, aż coś

się wydarzy. Kiedy nastąpi próba czytania danych z dysku albo użytkownik przycisnie klawisz, proces oczekujący na to zdarzenie jest odblokowywany i może wznowić działanie.

Ten stan rzeczy jest podstawą modelu pokazanego na rysunku 2.3. W tym przypadku na najniższym poziomie systemu operacyjnego znajduje się program szeregujący, zarządzający zbiorem procesów występujących w warstwie nad nim. Cały mechanizm obsługi przerwań i szczegółów związanych z właściwym uruchamianiem i zatrzymywaniem procesów jest ukryty w elemencie nazwanym tu zarządcą procesów. Element ten w rzeczywistości nie zawiera zbyt wiele kodu. Pozostała część systemu operacyjnego ma strukturę procesów. W praktyce jednak istnieje bardzo niewiele systemów operacyjnych, które miałyby tak przejrzystą strukturę.



Rysunek 2.3. Najniższa warstwa systemu operacyjnego o strukturze procesów zarządza przerwaniami i szeregowaniem. Powyżej tej warstwy znajdują się sekwencyjne procesy

2.1.6. Implementacja procesów

W celu zaimplementowania modelu procesów w systemie operacyjnym występuje tabela (tablica struktur), zwana **tabelą procesów**, w której każdemu z procesów odpowiada jedna pozycja — niektórzy autorzy nazywają te pozycje **blokami zarządzania procesami**. W blokach tych są zapisane ważne informacje na temat stanu procesu. Zawierają one wartości licznika programu, wskaźnika stosu, dane dotyczące przydziału pamięci, statusu otwartych procesów, rozliczeń i szeregowania oraz wszystkie inne informacje, które trzeba zapisać w czasie przełączania procesu ze stanu *wykonujący* do stanu *gotowy* lub *zablokowany*. Dzięki nim proces może być później wznowiony, tak jakby nigdy nie został zatrzymany.

W tabeli 2.1. pokazano kilka kluczowych pól w typowym systemie. Pola w pierwszej kolumnie są związane z zarządzaniem procesami. Pozostałe dwa łączą się odpowiednio z zarządzaniem pamięcią oraz zarządzaniem plikami. Należy zwrócić uwagę na to, że obecność poszczególnych pól w tabeli procesów w dużym stopniu zależy od systemu. Poniższa tabela daje jednak ogólny obraz rodzajów potrzebnych informacji.

Teraz, kiedy przyjrzelśmy się tabeli procesów, możemy wyjaśnić nieco dokładniej to, w jaki sposób iluzja wielu sekwencyjnych procesów jest utrzymywana w jednym procesorze (lub każdym z procesorów). Z każdą klasą wejścia-wyjścia wiąże się lokalizacja (zwykle pod ustalonym adresem w dolnej części pamięci) zwana **wektorem przerwań**. Jest w niej zapisany adres procedury obsługi przerwania. Załóżmy,

Tabela 2.1. Przykładowe pola typowego wpisu w tabeli procesów

Zarządzanie procesami	Zarządzanie pamięcią	Zarządzanie plikami
Rejestry	Wskaźnik do informacji segmentu tekstu	Katalog główny
Licznik programu	Wskaźnik do informacji segmentu danych	Katalog roboczy
Słowo stanu programu	Wskaźnik do informacji segmentu stosu	Deskryptory plików
Wskaźnik stosu		Identyfikator użytkownika
Stan procesu		Identyfikator grupy
Priorytet		
Parametry szeregowania		
Identyfikator procesu		
Proces-rodzic		
Grupa procesów		
Sygnały		
Czas rozpoczęcia procesu		
Wykorzystany czas CPU		
Czas CPU procesów-dzieci		
Godzina następnego alarmu		

że w momencie wystąpienia przerwania związanego z dyskiem ma działać proces użytkownika nr 3. Sprzęt obsługujący przerwanie odkłada na stos licznik programu procesu użytkownika nr 3, słowo stanu programu i czasami jeden lub kilka rejestrów. Następnie sterowanie przechodzi pod adres określony w wektorze przerwań. To jest wszystko, co robi sprzęt. Od tego momentu obsługą przerwania zajmuje się oprogramowanie — w szczególności procedura obsługi przerwania.

Obsługa każdego przerwania rozpoczyna się od zapisania rejestrów — często pod pozycją tabeli procesów odpowiadającą bieżącemu procesowi. Następnie informacje odłożone na stos przez mechanizm obsługi przerwania są z niego zdejmowane, a wskaźnik stosu jest ustawiany na adres tymczasowego stosu używanego przez procedurę obsługi procesu. Takich działań, jak zapisanie rejestrów i ustawienie wskaźnika stosu, nawet nie można wyrazić w językach wysokopoziomowych, takich jak C. W związku z tym operacje te są wykonywane przez niewielką procedurę w języku asemblera. Zazwyczaj jest to ta sama procedura dla wszystkich przerw, ponieważ zadanie zapisania rejestrów jest identyczne, niezależnie od tego, co było przyczyną przerwania.

Kiedy ta procedura zakończy działanie, wywołuje procedurę w języku C, która wykonuje resztę pracy dla tego konkretnego typu przerwania (zakładamy, że system operacyjny został napisany w języku C — w tym języku napisana jest większość systemów operacyjnych). Kiedy procedura ta wykona swoje zadanie (co może spowodować, że pewne procesy uzyskają gotowość do działania), wywoływany jest program szeregujący, który ma sprawdzić, jaki proces powinien zostać uruchomiony w następnej kolejności. Następnie sterowanie jest przekazywane z powrotem do

kodu w asemblerze, który ładuje rejestry i mapę pamięci nowego bieżącego procesu oraz rozpoczyna jego działanie. Obsługę przerwania i szeregowanie podsumowano w tabeli 2.2. Warto zwrócić uwagę, że różne systemy nieco się różnią pewnymi szczegółami.

Tabela 2.2. Szkielet działań wykonywanych przez najniższy poziom systemu operacyjnego w momencie wystąpienia przerwania

1. Sprzęt odkłada na stos licznik programu itp.
2. Sprzęt ładuje nowy licznik programu z wektora przerwania.
3. Procedura w języku asemblera zapisuje rejestry.
4. Procedura w języku asemblera ustawia nowy stos.
5. Uruchamia się procedura obsługi przerwania w C (zazwyczaj czyta i buforuje dane wejściowe).
6. Program szeregujący decyduje o tym, który proces ma być uruchomiony w następnej kolejności.
7. Procedura w języku C zwraca sterowanie do kodu w asemblerze.
8. Procedura w języku asemblera uruchamia nowy bieżący proces.

Kiedy proces zakończy działanie, system operacyjny wyświetla symbol zachęty i oczekuje na nowe polecenie. Po otrzymaniu polecenia ładuje do pamięci nowy program, nadpisując starą zawartość pamięci.

2.1.7. Modelowanie wieloprogramowości

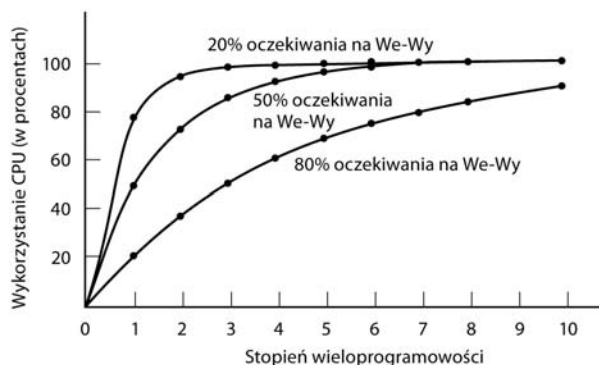
Zastosowanie wieloprogramowości pozwala na poprawę wykorzystania procesora. Z grubsza rzecz biorąc, jeśli przeciętny proces jest przetwarzany przez 20% czasu rezydowania w pamięci, to w przypadku gdy w pamięci jest jednocześnie pięć procesów, procesor powinien być zajęty przez cały czas. Ten model jest jednak nierealistycznie optymistyczny, ponieważ zakłada, że w żadnym momencie nie zdarzy się sytuacja, w której wszystkie pięć procesów będzie jednocześnie oczekiwało na operację wejścia-wyjścia.

Lepszym modelem jest spojrzenie na wykorzystanie procesora z probabilistycznego punktu widzenia. Załóżmy, że proces spędza fragment p swojego czasu na zakończeniu operacji wejścia-wyjścia. Przy n procesach znajdujących się jednocześnie w pamięci prawdopodobieństwo tego, że wszystkie n procesów będzie jednocześnie oczekiwało na obsługę wejścia-wyjścia (wtedy procesor pozostanie bezczynny), wynosi p^n . W takim przypadku wykorzystanie procesora można opisać za pomocą wzoru:

$$\text{Wykorzystanie procesora} = 1 - p^n$$

Na rysunku 2.4 pokazano procent wykorzystania procesora w funkcji n — co określa się jako **stopień wieloprogramowości**.

Z rysunku jasno wynika, że jeśli procesy spędzają 80% czasu w oczekiwaniu na operację wejścia-wyjścia, to aby współczynnik marnotrawienia procesora utrzymać



Rysunek 2.4. Wykorzystanie procesora w funkcji liczby procesów w pamięci

na poziomie poniżej 10%, w pamięci musi być jednocześnie co najmniej 10 procesów. Kiedy zdamy sobie sprawę ze stanu, w którym proces interaktywny oczekuje, aż użytkownik wpisze na terminalu jakieś dane, stanie się oczywiste, że czasy oczekiwania na wejścia-wyjścia rzędu 80% i więcej nie są niczym niezwykłym. Nawet na serwerach procesy wykonujące wiele dyskowych operacji wejścia-wyjścia często charakteryzują się tak wysokim procentem.

Dla ścisłości należy dodać, że model probabilistyczny opisany przed chwilą jest tylko przybliżeniem. Zakłada on niejawnie, że wszystkie n procesów jest niezależnych. Oznacza to, że w przypadku systemu z pięcioma procesami w pamięci dopuszczalnym stanem jest to, aby trzy z nich działały, a dwa czekały. Jednak przy jednym procesorze nie ma możliwości jednoczesnego działania trzech procesów. W związku z tym proces, który osiąga gotowość w czasie, gdy procesor jest zajęty, będzie musiał czekać. Tak więc procesy nie są niezależne. Dokładniejszy model można stworzyć z wykorzystaniem teorii kolejokowania, jednak teza, którą sformułowaliśmy — wieloprogramowość pozwala procesom wykorzystywać procesor w czasie, gdy w innej sytuacji byłby on bezczynny — jest oczywiście w dalszym ciągu prawdziwa. Faktu tego nie zmieniałaby nawet sytuacja, w której rzeczywiste krzywe stopnia wieloprogramowości nieco odbiegałyby od tych pokazanych na rysunku 2.4.

Mimo że model z rysunku 2.4 jest uproszczony, można go wykorzystywać w celu tworzenia specyficznych, jednak przybliżonych prognoz dotyczących wydajności procesora. Przypuśćmy na przykład, że komputer ma 512 MB pamięci, przy czym system operacyjny zajmuje 128 MB, a każdy z programów użytkownika również zajmuje do 128 MB. Te rozmiary pozwalają na to, aby w pamięci jednocześnie znajdowały się trzy programy użytkownika. Przy średnim czasie oczekiwania na operacje wejścia-wyjścia, wynoszącym 80%, mamy procent wykorzystania procesora na poziomie $1 - 0,8^3$ czyli około 49%. Dodanie kolejnych 512 MB pamięci operacyjnej umożliwia przejście systemu z trójstopniowej wieloprogramowości do siedmiostopniowej, co przyczyni się do wzrostu wykorzystania procesora do 79%. Mówiąc inaczej, dodatkowe 512 MB pamięci podniesie przepustowość o 30%.

Dodanie kolejnych 512 MB spowodowałoby zwiększenie stopnia wykorzystania procesora z 79% do 91%, a zatem podniosłoby przepustowość tylko o kolejne 12%. Korzystając z tego modelu, właściciel komputera może zdecydować, że pierwsza rozbudowa systemu jest dobrą inwestycją, natomiast druga nie.

2.2. WĄTKI

W tradycyjnych systemach operacyjnych każdy proces ma przestrzeń adresową i jeden wątek sterowania. W rzeczywistości prawie tak wygląda definicja procesu. Niemniej jednak często występują sytuacje, w których korzystne jest posiadanie wielu wątków sterowania w tej samej przestrzeni adresowej, działających quasi-równoległe — tak jakby były (niemal) oddzielnymi procesami (z wyjątkiem współdzielonej przestrzeni adresowej). Sytuacje te oraz wynikające z tego implikacje omówiono w kolejnych punktach.

2.2.1. Wykorzystanie wątków

Do czego może służyć rodzaj procesu wewnątrz innego procesu? Okazuje się, że istnieją powody istnienia tych miniprocessów zwanych **wątkami**. Spróbujmy przyrzeć się kilku z nich. Głównym powodem występowania wątków jest to, że w wielu aplikacjach jednocześnie wykonywanych jest wiele działań. Niektóre z nich mogą być zablokowane od czasu do czasu. Dzięki dekompozycji takiej aplikacji na wiele sekwencyjnych wątków działających quasi-równoległe model programowania staje się prostszy.

Taką samą dyskusję przedstawiliśmy już wcześniej. Dokładnie te same argumenty przemawiają za istnieniem procesów. Zamiast myśleć o przerwaniach, licznikach czasu i przełączaniu kontekstu, możemy myśleć o równoległych procesach. Tyle że teraz, przy pojęciu wątków, dodajemy nowy element: zdolność równoległych podmiotów do współdzielenia pomiędzy sobą przestrzeni adresowej oraz wszystkich swoich danych. Zdolność ta ma kluczowe znaczenie dla niektórych aplikacji, dlatego właśnie obecność wielu procesów (z oddzielnymi przestrzeniami adresowymi) w tym przypadku nie wystarczy.

Drugi argument, który przemawia za istnieniem wątków, jest taki, że — ponieważ są one mniejsze od procesów — w porównaniu z procesami łatwiej (tzn. szybciej) się je tworzy i niszczy. W wielu systemach tworzenie wątku trwa 10 – 100 razy krócej od tworzenia procesu. Ponieważ liczba potrzebnych wątków zmienia się dynamicznie i gwałtownie, szybkość nabiera dużego znaczenia.

Trzecim powodem istnienia wątków są względy wydajności. Istnienie wątków nie poprawi wydajności, jeśli wszystkie one będą związane z procesorem. Jednak w przypadku wykonywania intensywnych obliczeń i jednocześnie znaczącej liczby operacji wejścia-wyjścia występowanie wątków pozwala na nakładanie się na siebie tych działań, co w efekcie końcowym przyczynia się do przyspieszenia aplikacji.

Na koniec — wątki przydają się w systemach wyposażonych w wiele procesorów, gdzie możliwa jest rzeczywista współbieżność. Do tego zagadnienia powrócimy w rozdziale 8.

Najłatwiej przekonać się o przydatności wątków, analizując konkretne przykłady. W roli pierwszego przykładu rozważmy edytor tekstu. Edytory tekstu zazwyczaj wyświetlają na ekranie tworzony dokument sformatowany dokładnie w takiej postaci, w jakiej będzie on wyglądał na drukowanej stronie. Zwłaszcza wszystkie znaki podziału wierszy i stron znajdują się na prawidłowych i ostatecznych pozycjach. Dzięki temu użytkownik ma możliwość przeglądania i poprawienia dokumentu, jeśli zajdzie taka potrzeba (na przykład w celu wyeliminowania sierot i wdów — niekompletnych wierszy na początku i na końcu strony, które uważa się za nieestetyczne).

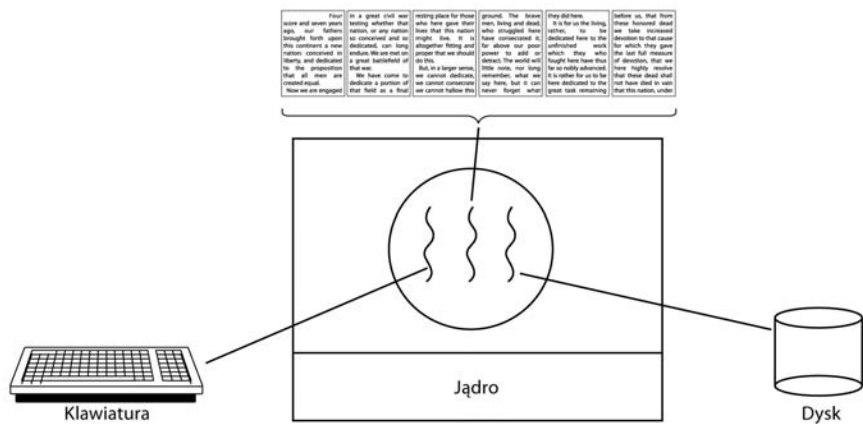
Załóżmy, że użytkownik pisze książkę. Z punktu widzenia autora najłatwiej umieścić całą książkę w pojedynczym pliku, tak by łatwiej było wyszukiwać tematy, wykonywać globalne operacje zastępowania itp. Alternatywnie można umieścić każdy z rozdziałów w osobnym pliku. Jednak umieszczenie każdego podrozdziału i punktu w osobnym pliku, na przykład gdyby zaszła potrzeba globalnego zastąpienia jakiegoś terminu w całej książce, byłoby prawdziwym utrapieniem. W takim przypadku trzeba by było bowiem indywidualnie edytować każdy z kilkuset plików. Jeśli na przykład zaproponowany termin „standard xxxx” zostałby zatwierdzony tuż przed oddaniem książki do druku, trzeba by było w ostatniej chwili zastąpić wszystkie wystąpienia terminu „roboczo: standard xxxx” na „standard xxxx”. Jeśli książka znajduje się w jednym pliku, taką operację można wykonać za pomocą jednego polecenia. Dla odróżnienia, gdyby książka składała się z 300 plików, każdy z nich trzeba by osobno otworzyć w edytorze.

Rozważmy teraz, co się zdarzy, kiedy użytkownik nagle usunie jedno zdanie z pierwszej strony 800-stronicowego dokumentu. Po sprawdzeniu poprawności zmodyfikowanej strony zdecydował, że chce wykonać inną zmianę na stronie 600 i wpisuje polecenie zlecające edytorowi przejście do tej strony (na przykład poprzez wyszukanie frazy, która znajduje się tylko tam). Edytor tekstu jest w tej sytuacji zmuszony do natychmiastowego przeformatowania całej książki do strony 600, ponieważ nie będzie wiedział, jaką treść ma pierwszy wiersz na stronie 600, dopóki nie przetworzy wszystkich poprzednich stron. Zanim będzie można wyświetlić stronę 600, może powstać znaczące opóźnienie, co doprowadzi do niezadowolenia użytkownika.

W takim przypadku może pomóc wykorzystanie wątków. Załóżmy, że edytor tekstu jest napisany jako program składający się z dwóch wątków. Jeden wątek zajmuje się komunikacją z użytkownikiem, a drugi przeprowadza w tle korektę formatowania. Natychmiast po usunięciu zdania ze strony 1 wątek komunikacji z użytkownikiem informuje wątek formatujący o konieczności przeformatowania całej książki. Tymczasem wątek komunikacji z użytkownikiem kontynuuje nasłuchiwanie klawiatury i myszy i odpowiada na proste polecenia, takie jak przeglądanie strony 1. W tym samym czasie drugi z wątków w tle wykonuje intensywne obliczenia. Przy odrobinie

szczęścia zmiana formatu zakończy się, zanim użytkownik poprosi o przejście na stronę 600. Jeśli tak się stanie, przejście na stronę 600 będzie mogło się odbyć bezwzględnie.

Kiedy już jesteśmy przy edytorach, odpowiedzmy sobie na pytanie, dlaczego by nie dodać trzeciego wątku. Wiele edytorów tekstu jest wyposażonych w mechanizm automatycznego zapisywania całego pliku na dysk co kilka minut. Ma to zapobiec utracie całodniowej pracy w przypadku awarii programu, awarii systemu lub problemów z zasilaniem. Trzeci wątek może obsługiwać wykonywanie kopii zapasowych na dysku, nie przeszkadzając w działaniu pozostałym dwóm. Sytuację z trzema wątkami pokazano na rysunku 2.5.



Rysunek 2.5. Edytor tekstu składający się z trzech wątków

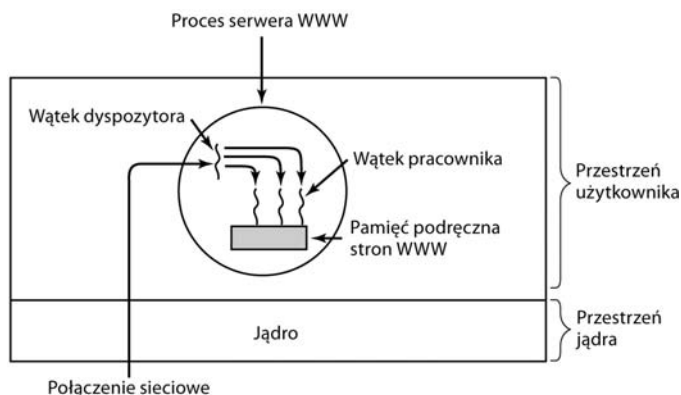
Gdyby program zawierał jeden wątek, to każde rozpoczęcie wykonywania kopii zapasowej na dysk powodowałoby, że polecenia z klawiatury i myszy byłyby ignorowane do czasu zakończenia wykonywania kopii zapasowej. Użytkownik z pewnością by to zauważył jako obniżoną wydajność. Alternatywnie zdarzenia związane z klawiaturą i myszą mogłyby przerwać wykonywanie kopii zapasowej na dysk, co pozwoliłoby na zachowanie dobrej wydajności, ale prowadziłoby do skomplikowanego modelu programowania bazującego na przerwaniach. W przypadku zastosowania trzech wątków model programowania jest znacznie prostszy. Pierwszy wątek zajmuje się jedynie interakcjami z użytkownikiem. Drugi wątek przeformatowuje dokument, kiedy otrzyma takie zlecenie. Trzeci wątek okresowo zapisuje zawartość pamięci RAM na dysk.

W tym przypadku powinno być jasne, że istnienie trzech oddzielnych procesów w tej sytuacji się nie sprawdzi, ponieważ wszystkie trzy wątki muszą operować na tym samym dokumencie. Dzięki występowaniu trzech wątków zamiast trzech procesów wątki współdzielą pamięć i w efekcie wszystkie mają dostęp do edytowanego dokumentu.

Analogiczna sytuacja występuje w przypadku wielu innych interaktywnych programów. Na przykład elektroniczny arkusz kalkulacyjny jest programem umożliwiającym użytkownikowi obsługę macierzy — niektóre z jej elementów są danymi wprowadzanymi przez użytkownika. Inne elementy są wyliczane na podstawie wprowadzonych danych i z wykorzystaniem potencjalnie skomplikowanych wzorów. Kiedy użytkownik zmodyfikuje jeden element, może zająć potrzeba obliczenia wielu innych elementów. Dzięki zdefiniowaniu działającego w tle wątku zajmującego się przeliczaniem wątek interaktywny pozwala użytkownikowi na wprowadzanie zmian w czasie, gdy są wykonywane obliczenia. Na podobnej zasadzie trzeci wątek może samodzielnie obsługiwać kopie zapasowe wykonywane na dysku.

Rozważmy teraz jeszcze jeden przykład zastosowania wątków: serwer ośrodka WWW. Przychodzą żądania stron, a w odpowiedzi żądane strony są przesyłane do klienta. W większości ośrodków WWW niektóre strony WWW są częściej odwiedzane niż inne. Na przykład główna strona serwisu Sony jest odwiedzana znacznie częściej od strony umieszczonej głęboko w drzewie katalogów i zawierającej specyfikację techniczną jakiegoś modelu kamery wideo. Serwery WWW wykorzystują ten fakt do poprawy wydajności. Utrzymują kolekcję często używanych stron w pamięci głównej, aby wyeliminować potrzebę odwoływania się do dysku w celu ich pobrania. Taka kolekcja jest nazywana pamięcią podręczną (ang. *cache*) i wykorzystuje się ją również także w wielu innych kontekstach. Na przykład w rozdziale 1. zetknęliśmy się z pamięciami podręcznymi procesora.

Jeden ze sposobów organizacji serwera WWW pokazano na rysunku 2.6(a). W tym przypadku jeden z wątków — **dyspozytor** — odczytuje z sieci przychodzące żądania. Po przeanalizowaniu żądania wybiera beczynny (tzn. zablokowany) **wątek pracownika** i przekazuje mu żądanie — na przykład poprzez zapisanie wskaźnika do komunikatu w specjalnym słowie powiązanim z każdym wątkiem. Następnie dyspozytor budzi uśpiony wątek pracownika — to znaczy zmienia jego stan z „zablokowany” na „gotowy”.



Rysunek 2.6. Serwer WWW z obsługą wielu wątków

Kiedy wątek się obudzi, sprawdza, czy jest w stanie spełnić żądanie z pamięci podręcznej strony WWW, do której mają dostęp wszystkie wątki. Jeśli tak nie jest, rozpoczyna operację odczytu w celu pobrania strony z dysku i przechodzi do stanu „zablokowany”, trwającego do chwili zakończenia operacji dyskowej. Kiedy wątek zablokuje się na operacji dyskowej, inny wątek zaczyna działanie, na przykład dyspozytor, którego zadaniem jest przyjęcie jak największej liczby żądań, albo inny pracownik, który jest gotowy do działania.

W tym modelu serwer może być zapisany w postaci kolekcji sekwencyjnych wątków. Program dyspozytora zawiera pętlę nieskończoną, w której jest pobierane żądanie pracy, później wręczane pracownikowi. Kod każdego pracownika zawiera pętlę nieskończoną, w której jest akceptowane żądanie od dyspozytora i następuje sprawdzenie, czy żądana strona jest dostępna w pamięci podręcznej serwera WWW. Jeśli tak, strona jest zwracana do klienta, a pracownik blokuje się w oczekiwaniu na nowe żądanie. Jeśli nie, pracownik pobiera stronę z dysku, zwraca ją do klienta i blokuje się w oczekiwaniu na nowe żądanie.

W uproszczonej formie kod przedstawiono na listingu 2.1. W tym przypadku, podobnie jak w pozostałej części tej książki, założono, że TRUE odpowiada stałej o wartości 1. Natomiast buf i strona są strukturami do przechowywania odpowiednio żądania pracy i strony WWW.

Listing 2.1. Uproszczona postać kodu dla struktury serwera z rysunku 2.6.

(a) Wątek dyspozytora; (b) wątek pracownika

(a)	(b)
<pre>while (TRUE){ pobierz_nast_zadanie(&buf); przekaz_prace(&buf); }</pre>	<pre>while (TRUE){ czekaj_na_prace(&buf) szukaj_strony_w_pamieci_cache(&buf, ↪ &strona); if ((&strona)) czytaj_strone_z_dysku(&buf, ↪ &strona); zwroc_strone(&strona); }</pre>

Zastanówmy się, jak mógłby być napisany serwer WWW, gdyby nie było wątków. Jedna z możliwości polega na zaimplementowaniu go jako pojedynczego wątku. W głównej pętli serwera WWW następowałyby pobieranie żądania, jego analiza i realizacja. Dopiero potem serwer WWW mógłby pobrać następne żądanie. Podczas oczekiwania na zakończenie operacji dyskowej serwer byłby bezczynny i nie przetwarzałby żadnych innych przychodzących żądań. Jeśli serwer WWW działa na dedykowanej maszynie, tak jak to zwykle bywa, w czasie oczekiwania serwera WWW na dysk procesor pozostałby bezczynny. W efekcie końcowym można by było przetworzyć znacznie mniej żądań na sekundę. A zatem skorzystanie z wątków pozwala na uzyskanie znaczącego zysku wydajności, ale każdy z wątków jest programowany sekwencyjnie — w standardowy sposób.

Do tej pory omówiliśmy dwa możliwe projekty: wielowątkowy serwer WWW i jednowątkowy serwer WWW. Załóżmy, że wątki nie są dostępne, ale projektanci systemu uznali obniżenie wydajności spowodowane istnieniem pojedynczego wątku za niedopuszczalne. Jeśli jest dostępna nieblokująca wersja wywołania systemowego read, możliwe staje się trzecie podejście. Kiedy przychodzi żądanie, analizuje go jeden i tylko jeden wątek. Jeżeli żądanie może być obsłużone z pamięci podręcznej, to dobrze, ale jeśli nie, inicjowana jest nieblokująca operacja dyskowa.

Serwer rejestruje stan bieżącego żądania w tabeli, a następnie pobiera następne zdarzenie do obsługi. Może to być żądanie nowej pracy albo odpowiedź dysku dotycząca poprzedniej operacji. Jeśli jest to żądanie nowej pracy, rozpoczyna się jego obsługa. Jeśli jest to odpowiedź z dysku, właściwe informacje są pobierane z tabeli i następuje przetwarzanie odpowiedzi. W przypadku nieblokujących dyskowych operacji wejścia-wyjścia odpowiedź zwykle ma postać sygnału lub przerwania.

W tym projekcie model „procesów sekwencyjnych” omawiany w pierwszych dwóch przypadkach nie występuje. Stan obliczeń musi być jawnie zapisany i odtworzony z tabeli, za każdym razem, kiedy serwer przełącza się z pracy nad jednym żądaniem do pracy nad kolejnym żądaniem. W rezultacie wątki i ich stosy są symulowane w trudniejszy sposób. W projektach takich jak ten wszystkie obliczenia mają zapisany stan. Ponadto istnieje zbiór zdarzeń, których wystąpienie może zmieniać określone stany. Takie systemy nazywa się **automatami o skończonej liczbie stanów** — pojęcie to jest powszechnie używane w branży komputerowej.

Teraz powinno być jasne, co oferują wątki. Pozwalają na utrzymanie idei procesów sekwencyjnych wykonujących blokujące wywołania systemowe (na przykład dotyczące dyskowych operacji wejścia-wyjścia) z jednoczesnym uzyskaniem efektu współbieżności. Blokujące wywołania systemowe ułatwiają programowanie, a współbieżność poprawia wydajność. Jednowątkowy serwer zachowuje prostotę blokujących wywołań systemowych, ale gwarantuje wydajność. Trzecie podejście pozwala na osiągnięcie wysokiej wydajności dzięki współbieżności, ale wykorzystuje nieblokujące wywołania i przerwania, dlatego jest trudne do zaprogramowania. Dostępne modele zestawiono w tabeli 2.3.

Tabela 2.3. Trzy sposoby konstrukcji serwera

Model	Charakterystyka
Wątki	Współbieżność, blokujące wywołania systemowe
Proces jednowątkowy	Brak współbieżności, blokujące wywołania systemowe
Automat o skończonej liczbie stanów	Współbieżność, nieblokujące wywołania systemowe, przerwania

Trzecim przykładem zastosowania wątków są aplikacje, które muszą przetwarzać duże ilości danych. Normalne podejście polega na przeczytaniu bloku danych, przetworzeniu go, a następnie ponownym zapisaniu. Problem w takim przypadku

polega na tym, że jeśli dostępne są tylko blokujące wywołania systemowe, proces blokuje się, kiedy dane przychodzą oraz kiedy są wysyłane na zewnątrz. Doprowadzenie do sytuacji, w której procesor jest bezczynny w czasie, gdy jest wiele obliczeń do wykonania, to oczywiście marnotrawstwo i w miarę możliwości należy unikać takiej sytuacji.

Rozwiązaniem problemu jest wykorzystanie wątków. Wewnątrz procesu można wydzielić wątek wejściowy, wątek przetwarzania danych i wątek wyprowadzania danych. Wątek wejściowy czyta dane do bufora wejściowego. Wątek przetwarzania danych pobiera dane z bufora wejściowego, przetwarza je i umieszcza wyniki w buforze wyjściowym. Wątek wyprowadzania danych zapisuje wyniki z bufora wyjściowego na dysk. W ten sposób wprowadzanie danych, ich wyprowadzanie i przetwarzanie mogą być realizowane w tym samym czasie. Oczywiście model ten działa tylko wtedy, kiedy wywołanie systemowe blokuje wyłącznie wątek wywołujący, a nie cały proces.

2.2.2. Klasyczny model wątków

Teraz, kiedy pokazaliśmy, do czego mogą się przydać wątki i jak ich można używać, spróbujmy przeanalizować to zagadnienie nieco dokładniej. Model procesów bazuje na dwóch niezależnych pojęciach: grupowaniu zasobów i uruchamianiu. Czasami wygodnie jest je rozdzielić — wtedy można skorzystać z wątków. Najpierw przyjrzemy się klasycznemu modelowi wątków. Następnie omówimy model wątków Linuksa, w którym linia pomiędzy wątkami i procesami jest rozmyta.

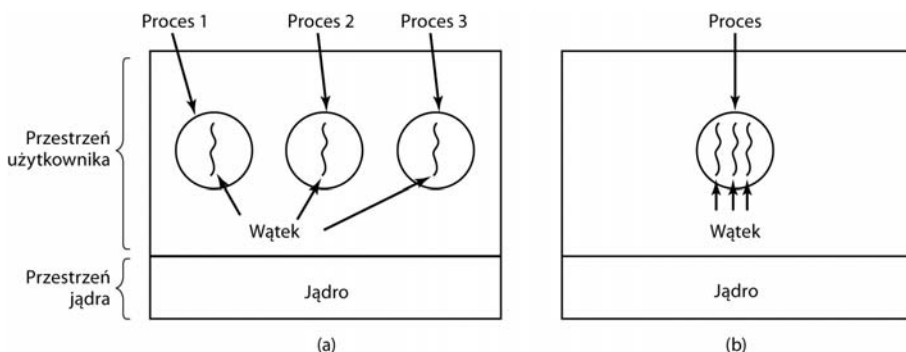
Jednym ze sposobów patrzenia na proces jest postrzeganie go jako sposobu grupowania powiązanych ze sobą zasobów. Proces dysponuje przestrzenią adresową zawierającą tekst programu i dane, a także inne zasoby. Do zasobów tych można zaliczyć otwarte pliki, procesy-dzieci, nieobsłużone alarmy, porcedury obsługi sygnałów, informacje rozliczeniowe i wiele innych. Dzięki pogrupowaniu ich w formie procesu można nimi łatwiej zarządzać.

W innym pojęciu proces zawiera wykonywany wątek — zwykle w skrócie używa się samego pojęcia wątku. **Wątek** zawiera licznik programu, który śledzi to, jaka instrukcja będzie wykonywana w następnej kolejności. Posiada rejestry zawierające jego bieżące robocze zmienne. Ma do dyspozycji stos zawierający historię działania — po jednej ramce dla każdej procedury, której wykonywanie się rozpoczęło, ale jeszcze się nie zakończyło. Chociaż wątek musi realizować jakiś proces, wątek i jego proces są pojęciami odrębnymi i można je traktować osobno. Procesy są wykorzystywane do grupowania zasobów, wątki są podmiotami zaplanowanymi do wykonania przez procesor.

Wątki dodają do modelu procesu możliwość realizacji wielu wykonań w tym samym środowisku procesu, w dużym stopniu w sposób wzajemnie od siebie niezależny. Równoległe działanie wielu wątków w obrębie jednego procesu jest analogiczne do równoległego działania wielu procesów w jednym komputerze. W pierwszym z tych przypadków wątki współdzielą przestrzeń adresową i inne zasoby.

W drugim przypadku procesy współdzielą pamięć fizyczną, dyski, drukarki i inne zasoby. Ponieważ wątki mają pewne właściwości procesów, czasami nazywa się je **lekkimi procesami**. Do opisanía sytuacji, w której w tym samym procesie może działać wiele wątków, używa się także terminu **wielowątkowość**. Jak widzieliśmy w rozdziale 1., niektóre procesory mają bezpośrednią obsługę sprzętową wielowątkowości i pozwalają na przełączanie wątków w skali czasowej rzędu nanosekund.

Na rysunku 2.7(a) widać trzy tradycyjne procesy. Każdy proces ma swoją własną przestrzeń adresową oraz pojedynczy wątek sterowania. Dla odmiany w układzie z rysunku 2.7(b) widzimy jeden proces z trzema wątkami sterowania. Chociaż w obu przypadkach mamy trzy wątki, w sytuacji z rysunku 2.7(a) każdy z nich działa w innej przestrzeni adresowej, podczas gdy w sytuacji z rysunku 2.7(b) wszystkie współdzielą tę samą przestrzeń adresową.



Rysunek 2.7. (a) Trzy procesy, z których każdy posiada jeden wątek; (b) jeden wątek z trzema wątkami

Kiedy wielowątkowy proces działa w jednoprocessorowym systemie, wątki działają po kolei. Na rysunku 2.1 widzieliśmy, jak działa wieloprogramowość procesów. Dzięki przełączaniu pomiędzy wieloma procesami system daje iluzję oddzielnych procesów sekwencyjnych działających współbieżnie. Wielowątkowość działa w taki sam sposób. Procesor przełącza się w szybkim tempie pomiędzy wątkami, dając iluzję, że wątki działają współbieżnie — chociaż na wolniejszym procesorze od fizycznego. Przy trzech wątkach obliczeniowych w procesie wątki będą sprawiały wrażenie równoległego działania, ale tak, jakby każdy z nich działał na procesorze o szybkości równej jednej trzeciej szybkości fizycznego procesora.

Różne wątki procesu nie są tak niezależne, jak różne procesy. Wszystkie wątki posługują się dokładnie tą samą przestrzenią adresową, co również oznacza, że współdzielą one te same zmienne globalne. Ponieważ każdy wątek może uzyskać dostęp do każdego adresu pamięci w obrębie przestrzeni adresowej procesu, jeden wątek może odczytać, zapisać, a nawet wyczyścić stos innego wątku. Pomiedzy wątkami nie ma zabezpieczeń, ponieważ (1) byłyby one niemożliwe do realizacji, a (2) nie powinny być potrzebne. W odróżnieniu od różnych procesów, które potencjalnie

należą do różnych użytkowników i które mogą być dla siebie wrogie, proces zawsze należy do jednego użytkownika, który przypuszczalnie utworzył wiele wątków, a zatem powinny one współpracować, a nie walczyć ze sobą. Oprócz przestrzeni adresowej wszystkie wątki mogą współdzielić ten sam zbiór otwartych plików, procesów-dzieci, alarmów, sygnałów itp., tak jak pokazano w tabeli 2.4. Tak więc organizacja pokazana na rysunku 2.7(a) mogłaby zostać użyta, jeśli trzy procesy są ze sobą niezwiązane, natomiast organizacja z rysunku 2.7(b) byłaby właściwa w przypadku, gdyby trzy wątki były częścią tego samego zadania i gdyby aktywnie i ściśle ze sobą współpracowały.

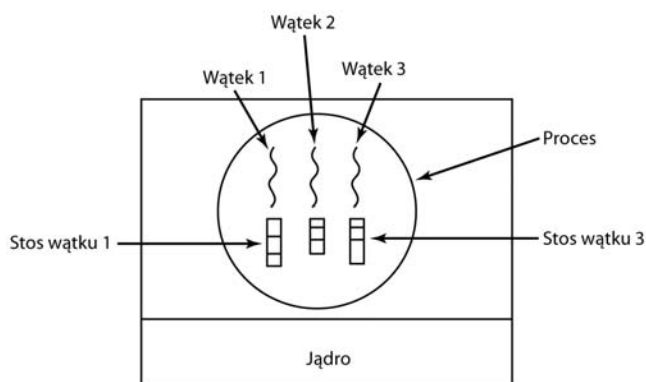
Tabela 2.4. W pierwszej kolumnie wyszczególniono cechy wspólne dla wszystkich wątków w procesie. W drugiej kolumnie zamieszczono niektóre elementy prywatne dla każdego wątku

Komponenty procesu	Komponenty wątku
Przestrzeń adresowa	Licznik programu
Zmienne globalne	Rejestry
Otwarte pliki	Stos
Procesy-dzieci	Stan
Zaległe alarmy	
Sygnały i procedury obsługi sygnałów	
Informacje dotyczące rozliczeń	

Elementy w pierwszej kolumnie są właściwościami procesu, a nie wątku. Jeśli na przykład jeden wątek otworzy plik, będzie on widoczny dla innych wątków w procesie. Wątki te będą mogły czytać dane z pliku i je zapisywać. To logiczne, ponieważ właśnie proces, a nie wątek jest jednostką zarządzania zasobami. Gdyby każdy wątek miał własną przestrzeń adresową, otwarte pliki, nieobsłużone alarmy itd., byłby osobnym procesem. Wykorzystując pojęcie wątków, chcemy, aby wiele wątków mogło współdzielić zbiór zasobów. Dzięki temu mogą one ze sobą ściśle współpracować w celu wykonania określonego zadania.

Podobnie jak tradycyjny proces (czyli taki, który zawiera tylko jeden wątek), wątek może znajdować się w jednym z kilku stanów: „działający”, „zablokowany”, „gotowy” lub „zakończony”. Działający wątek posiada dostęp do procesora i jest aktywny. Zablokowany wątek oczekuje na jakieś zdarzenie, by mógł się odblokować. Kiedy na przykład wątek realizuje wywołanie systemowe odczytujące dane z klawiatury, jest zablokowany do czasu, kiedy użytkownik wpisze dane wejściowe. Wątek może się blokować w oczekiwaniu na wystąpienie zdarzenia zewnętrznego lub może oczekiwać, aż odblokuje go inny wątek. Wątek gotowy jest zaplanowany do uruchomienia i zostanie uruchomiony, kiedy nadejdzie jego kolej. Przejścia pomiędzy stanami wątków są identyczne jak przejścia pomiędzy stanami procesów. Zilustrowano je na rysunku 2.2.

Istotne znaczenie ma zdanie sobie sprawy, że każdy wątek posiada własny stos, co zilustrowano na rysunku 2.8. Stos każdego wątku zawiera po jednej ramce dla każdej procedury, która została wywołana, a z której jeszcze nie nastąpił powrót. Ramka ta zawiera zmienne lokalne procedury oraz adres powrotu, który będzie wykorzystany po zakończeniu obsługi wywołania procedury. Jeśli na przykład procedura X wywoła procedurę Y, a procedura Y wywoła procedurę Z, to w czasie, kiedy działa procedura Z, na stosie będą ramki dla procedur X, Y i Z. Każdy wątek, ogólnie rzecz biorąc, będzie wywoływał inne procedury, a zatem będzie miał inną historię wywołań. Dlatego właśnie każdy wątek potrzebuje własnego stosu.



Rysunek 2.8. Każdy wątek ma własny stos

W przypadku gdy system obsługuje wielowątkowość, procesy zazwyczaj rozpoczynają działanie z jednym wątkiem. Wątek ten posiada zdolność do tworzenia nowych wątków za pomocą wywołania procedury, na przykład `thread_create`. Parametr procedury `thread_create` zwykle określa nazwę procedury, która ma się uruchomić dla nowego wątku. Nie jest konieczne (ani nawet możliwe) ustalenie czegośkolwiek na temat przestrzeni adresowej nowego wątku, ponieważ wątek automatycznie działa w przestrzeni adresowej wątku tworzącego. Czasami wątki są hierarchiczne i zachodzą pomiędzy nimi relacje rodzic – dziecko, często jednak takie relacje nie występują, a wszystkie wątki są sobie równe. Niezależnie od tego, czy pomiędzy wątkami zachodzi relacja hierarchii, wątek tworzący zwykle zwraca identyfikator wątku zawierający nazwę nowego wątku.

Kiedy wątek zakończy swoją pracę, może zakończyć działanie poprzez wywołanie procedury bibliotecznej, na przykład `thread_exit`. W tym momencie wątek znika i nie może być więcej zarządzany. W niektórych systemach obsługi wątków jeden wątek może czekać na zakończenie innego wątku poprzez wywołanie procedury, na przykład `thread_join`. Procedura ta blokuje wątek wywołujący do czasu zakończenia (specyficznego) wątku. Pod tym względem tworzenie i kończenie wątków przypomina tworzenie i kończenie procesów i wymaga w przybliżeniu tych samych opcji.

Innym popularnym wywołaniem dotyczącym wątków jest `thread_yield`. Umożliwia ono wątkowi dobrowolną rezygnację z procesora w celu umożliwienia działania innemu wątkowi. Takie wywołanie ma istotne znaczenie, ponieważ nie istnieje przerwanie zegara, które wymuszałoby wieloprogramowość, tak jak w przypadku procesów. W związku z tym istotne znaczenie ma to, aby wątki były „uprzejme” i od czasu do czasu dobrowolnie rezygnowały z procesora, tak by inne wątki miały szansę na działanie. Są również inne wywołania — na przykład pozwalające na to, aby jeden wątek poczekał, aż następny zakończy jakąś pracę, lub by ogłosił, że właśnie zakończył jakąś pracę itd.

Chociaż wątki często się przydają, wprowadzają także szereg komplikacji do modelu programowania. Na początek przeanalizujemy efekty na uniksowe wywołanie systemowej `fork`. Jeśli proces-rodzic ma wiele wątków, to czy proces-dziecko również powinien je mieć? Jeśli nie, to proces może nie działać prawidłowo, ponieważ wszystkie wątki mogą mieć istotne znaczenie.

Tymczasem gdy proces-dziecko otrzyma tyle samo wątków, co rodzic, to co się stanie, jeśli wątek należący do rodzica zostanie zablokowany przez wywołanie `read`, powiedzmy, z klawiatury? Czy teraz dwa wątki są zablokowane przez klawiaturę — jeden w procesie-rodzicu i drugi w dziecku? Kiedy użytkownik wpisze wiersz, to czy kopia pojawi się w obu wątkach? A może tylko w wątku rodzica? Lub tylko w wątku dziecka? Ten sam problem występuje dla twardych połączeń sieciowych.

Inna klasa problemów wiąże się z faktem współdzielenia przez wątki wielu struktur danych. Co się dzieje, jeśli jeden wątek zamknie plik, podczas gdy inny ciągle z niego czyta? Przypuśćmy, że jeden z wątków zauważa, że jest za mało pamięci, i rozpoczyna alokowanie większej ilości pamięci. W trakcie tego działania następuje przełączenie wątku. Nowy wątek również zauważa, że jest za mało pamięci i także rozpoczyna alokowanie dodatkowej pamięci. Pamięć prawdopodobnie będzie alokowana dwukrotnie. Przy odrobinie wysiłku można rozwiązać te problemy, jednak poprawna praca programów wykorzystujących wielowątkowość wymaga dokładnych przemyśleń i dokładnego projektowania.

2.2.3. Wątki POSIX

Aby było możliwe napisanie przenośnego programu z obsługą wielu wątków, organizacja IEEE zdefiniowała standard 1003.1c. Pakiet obsługi wątków, który tam zdefiniowano, nosi nazwę `Pthreads`. Jest on obsługiwany przez większość systemów uniksowych. W standardzie zdefiniowano ponad 60 wywołań funkcji. To o wiele za dużo, by można je było dokładnie omówić w tej książce. Omówimy zatem kilka najważniejszych. Dzięki temu Czytelnik uzyska obraz ich działania. Wywołania, które opiszemy, zostały wyszczególnione w tabeli 2.5.

Tabela 2.5. Niektóre wywołania funkcji należące do pakietu Pthreads

Wywołanie obsługi wątku	Opis
Pthread_create	Utworzenie nowego wątku
Pthread_exit	Zakończenie wątku wywołującego
Pthread_join	Oczekiwanie na zakończenie specyficznego wątku
Pthread_yield	Zwolnienie procesora w celu umożliwienia działania innemu wątkowi
Pthread_attr_init	Utworzenie i zainicjowanie struktury atrybutów wątku
Pthread_attr_destroy	Usunięcie struktury atrybutów wątku

Wszystkie wątki pakietu Pthreads mają określone właściwości. Każdy z nich posiada identyfikator, zbiór rejestrów (łącznie z licznikiem programu) oraz zbiór atrybutów zapisanych w pewnej strukturze. Do atrybutów tych należy rozmiar stosu, parametry szeregowania oraz inne elementy potrzebne do korzystania z wątku.

Nowy wątek tworzy się za pomocą wywołania `pthread_create`. Jako wartość funkcji zwracany jest identyfikator nowo utworzonego wątku. Wywołanie to nieprzypadkowo przypomina wywołanie systemowe `fork`. W tym przypadku identyfikator wątku spełnia rolę identyfikatora PID, głównie do celów identyfikacji wątków w innych wywołaniach.

Kiedy wątek zakończy pracę, która została do niego przydzielona, może zakończyć swoje działanie poprzez wywołanie funkcji `pthread_exit`. Wywołanie to zatrzymuje wątek i zwalnia jego stos.

Często wątek musi czekać, aż inny wątek zakończy swoją pracę. Dopiero później może kontynuować działanie. Wątek oczekujący na zakończenie specyficznego innego wątku wywołuje funkcję `pthread_join`. Identyfikator wątku, który ma się zakończyć, jest przekazywany jako parametr.

Czasami się zdarza, że wątek nie jest logicznie zablokowany, ale czuje, że działa już dość długo, i chce dać innemu wątkowi szansę działania. Cel ten można osiągnąć za pomocą wywołania `pthread_yield`. Nie ma takiego wywołania w przypadku procesów, ponieważ zakłada się, że procesy ze sobą rywalizują i każdy z nich chce uzyskać maksymalnie dużo czasu procesora. Ponieważ jednak wątki procesu współdziałają ze sobą, a ich kod jest pisany przez tego samego programistę, czasami programista chce, aby każdy z wątków otrzymał swoją szansę.

Następne dwa wywołania obsługi wątków dotyczą atrybutów wątku. Wywołanie `Pthread_attr_init` tworzy strukturę atrybutów powiązaną z wątkiem i inicjuje ją do wartości domyślnych. Wartości te (takie jak priorytet) można zmieniać poprzez modyfikowanie pól w strukturze atrybutów.

Na koniec — wywołanie `pthread_attr_destroy` usuwa strukturę atrybutów wątku i zwalnia pamięć. Wywołanie to nie ma wpływu na wątki korzystające z atrybutów. Wątki te w dalszym ciągu istnieją.

Aby uzyskać lepszy obraz tego, jak działa pakiet Pthreads, rozważmy prosty przykład z listingu 2.2. Główny program wykonuje się w pętli `NUMBER_OF_THREADS` razy. W każdej iteracji program wyświetla komunikat i tworzy nowy wątek. Jeśli tworzenie wątku nie powiedzie się, program wyświetla komunikat o błędzie i kończy działanie. Po utworzeniu wszystkich wątków program główny kończy działanie.

Listing 2.2. Przykładowy program wykorzystujący wątki

```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

#define NUMBER_OF_THREADS 10

void *print_hello_world(void *tid)
{
    /* Funkcja wyświetla identyfikator wątku i kończy działanie. */
    printf("Witaj, Świecie. Pozdrowienia od wątku %d0, tid);
    pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
    /* Program główny tworzy 10 wątków i kończy działanie. */
    pthread_t threads[NUMBER_OF_THREADS];
    int status, i;

    for(i=0; i < NUMBER_OF_THREADS; i++) {
        printf("Tu program główny. Tworzenie wątku %d0, i);
        status = pthread_create(&threads[i], NULL, print_hello_world,
            ↪(void *)i);

        if (status != 0) {
            printf("Oops. Funkcja pthread_create zwróciła kod błędu %d0, status);
            exit(-1);
        }
    }
    exit(NULL);
}
```

Podczas tworzenia wątek wyświetla jednowierszowy komunikat, w którym się przedstawia, a następnie kończy działanie. Kolejność, w jakiej będą się pojawiały poszczególne komunikaty, nie jest określona i może być różna w kolejnych uruchomieniach programu.

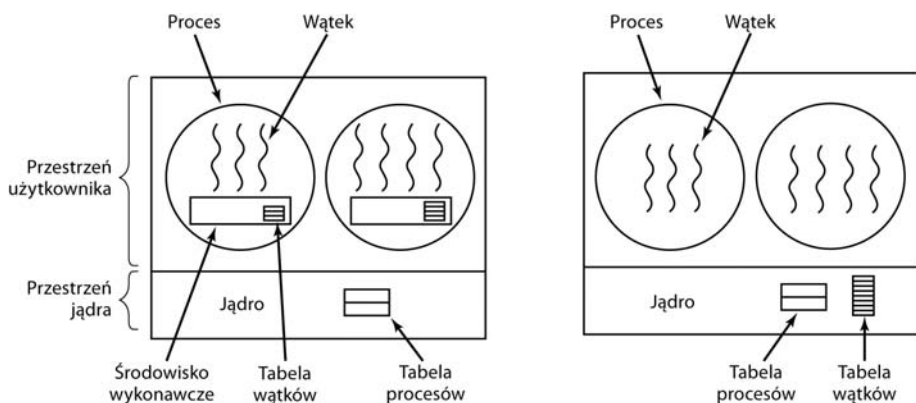
Pakiet Pthreads w żadnym razie nie ogranicza się do funkcji opisanych powyżej. Jest ich znacznie więcej. Niektóre z kolejnych wywołań opiszemy później, po omówieniu zagadnienia synchronizacji procesów i wątków.

2.2.4. Implementacja wątków w przestrzeni użytkownika

Istnieją dwa główne sposoby implementacji pakietu obsługi wątków: w przestrzeni użytkownika i w jądrze. Podział ten jest dość płynny. Możliwe są również implementacje hybrydowe. Poniżej opiszemy obie metody razem z ich zaletami i wadami.

Pierwsza metoda polega na umieszczeniu pakietu wątków w całości w przestrzeni użytkownika. Jądro nic o nich nie wie. Z jego punktu widzenia procesy, którymi zarządza, są standardowe — jednowątkowe. Pierwsza i najbardziej oczywista zaleta tego rozwiązania polega na tym, że pakiet obsługi wątków na poziomie przestrzeni użytkownika można zaimplementować w systemie operacyjnym, który nie obsługuje wątków. Do tej kategorii w przeszłości należały wszystkie systemy operacyjne i nawet dziś niektóre do niej należą. Przy takim podejściu wątki są implementowane za pomocą biblioteki.

Wszystkie tego rodzaju implementacje mają taką samą ogólną strukturę, co zilustrowano na rysunku 2.9(a). Wątki działają na bazie środowiska wykonawczego — kolekcji procedur, które nimi zarządzają. Do tej pory zapoznaliśmy się z czterema procedurami z tej grupy: `pthread_create`, `pthread_exit`, `pthread_join` oraz `pthread_yield`. Zwykle jednak jest ich więcej.



Rysunek 2.9. (a) Pakiet obsługi wątków na poziomie użytkownika; (b) pakiet obsługi wątków zarządzany przez jądro

Jeśli wątki są zarządzane w przestrzeni użytkownika, każdy proces potrzebuje swojej prywatnej **tabeli wątków**, która ma na celu śledzenie wątków w tym procesie. Tabela ta jest analogiczna do tabeli procesów w jądrze. Różnica polega na tym, że śledzi ona właściwości tylko na poziomie wątku — na przykład licznik programu każdego z wątków, wskaźnik stosu, rejestry, stan itp. Tabela wątków jest zarządzana przez środowisko wykonawcze. Kiedy wątek przechodzi do stanu gotowości lub zablokowania, informacje potrzebne do jego wznowienia są zapisywane w tabeli wątków, dokładnie w taki sam sposób, w jaki jądro zapisuje informacje o procesach w tabeli procesów.

Kiedy wątek wykona operację, która może spowodować jego lokalne zablokowanie, na przykład oczekuje, aż inny wątek w tym samym procesie wykona jakąś pracę, wykonuje procedurę ze środowiska wykonawczego. Procedura ta sprawdza, czy wątek musi być przełączony do stanu zablokowania. Jeśli tak, to zapisuje rejestry wątku (tzn. własne) w tabeli wątków, szuka w tabeli wątku gotowego do działania i ładuje rejestry maszyny zapisanymi wartościami odpowiadającymi nowemu wątkowi. Po przełączeniu wskaźnika stosu i licznika programu nowy wątek automatycznie powraca do życia. Jeśli maszyna posiada instrukcję zapisującą wszystkie rejestry oraz inną instrukcję, która je wszystkie ładuje, przełączenie wątku można przeprowadzić za pomocą zaledwie kilku instrukcji. Przeprowadzenie przełączania wątków w taki sposób jest co najmniej o jeden rząd wielkości szybsze od wykonywania rozkazu pułapki do jądra. To silny argument przemawiający za implementacją pakietu zarządzania wątkami na poziomie przestrzeni użytkownika.

Jest jednak jedna zasadnicza różnica w porównaniu z procesami. Kiedy wątek zakończy na chwilę działanie, na przykład kiedy wywoła funkcję `thread_yield`, kod funkcji `thread_yield` może zapisać informacje dotyczące wątku w samej tabeli wątków. Co więcej, może on następnie wywołać zarządcę wątków w celu wybrania innego wątku do uruchomienia. Procedura zapisująca stan wątku oraz program szeregujący są po prostu lokalnymi procedurami, zatem wywołanie ich jest znacznie bardziej wydajne od wykonania wywołania jądra. Między innymi nie jest potrzebny rozkaz pułapki, nie trzeba przełączać kontekstu, nie trzeba opróżniać pamięci podręcznej. W związku z tym zarządzanie wątkami odbywa się bardzo szybko.

Implementacja wątków na poziomie przestrzeni użytkownika ma także inne zalety. Dzięki temu każdemu procesowi można przypisać własny, spersonalizowany algorytm szeregowania. W przypadku niektórych aplikacji, na przykład zawierających wątek mechanizmu odświeżania, brak konieczności przejmowania się możliwością zatrzymania się wątku w nieodpowiednim momencie jest zaletą. Takie rozwiązanie okazuje się również łatwiejsze do skalowania, ponieważ wątki zarządzane na poziomie jądra niewątpliwie wymagają przestrzeni na tabelę i stos w jądrze, a to, w przypadku dużej liczby wątków, może być problemem.

Pomimo lepszej wydajności implementacja wątków na poziomie przestrzeni użytkownika ma również istotne wady. Pierwsza z nich dotyczy sposobu implementacji blokujących wywołań systemowych. Przypuśćmy, że wątek czyta z klawiatury, zanim zostanie wciśnięty jakikolwiek klawisz. Zezwolenie wątkowi na wykonanie wywołania systemowego jest niedopuszczalne, ponieważ spowoduje to zatrzymanie wszystkich wątków. Trzeba pamiętać, że jednym z podstawowych celów korzystania z wątków jest umożliwienie wszystkim wątkom używania wywołań blokujących, a przy tym niedopuszczenie do tego, by zablokowany wątek miał wpływ na inne. W przypadku blokujących wywołań systemowych trudno znaleźć łatwe rozwiązanie pozwalające na spełnienie tego celu.

Wszystkie wywołania systemowe można zmienić na nieblokujące (na przykład odczyt z klawiatury zwróciłby 0 bajtów, gdyby znaki nie były wcześniej zbuforowane),

ale wymaganie zmian w systemie operacyjnym jest nieatrakcyjne. Poza tym jednym z argumentów przemawiających za obsługą wątków na poziomie użytkownika była możliwość wykorzystania takiego mechanizmu w istniejących systemach operacyjnych. Co więcej, zmiana semantyki wywołania `read` wymagałaby modyfikacji wielu programów użytkowych.

Jedną z możliwych alternatyw można zastosować w przypadku, gdy można z góry powiedzieć, czy wywołanie jest blokujące. W niektórych wersjach Uniksa istnieje wywołanie systemowe `select`, które pozwala procesowi wywołującemu na sprawdzenie, czy wywołanie `read` będzie blokujące. Jeśli jest dostępne to wywołanie, można zastąpić procedurę biblioteczną `read` nową wersją, która najpierw wykonuje wywołanie `select`, a następnie wywołuje `read` tylko wtedy, gdy jest to bezpieczne (tzn. nie spowoduje zablokowania). Jeżeli wywołanie `read` ma doprowadzić do zablokowania, nie jest wykonywane. Zamiast wywołania `read` uruchamiany jest inny wątek. Następnym razem, kiedy środowisko wykonawcze otrzyma sterowanie, może sprawdzić ponownie, czy wykonanie wywołania `read` jest bezpieczne. Takie podejście wymaga zmiany implementacji części biblioteki wywołań systemowych, jest niewygodne i nieeleganckie, ale możliwości wyboru są ograniczone. Kod wokół wywołania systemowego, który wykonuje test, określa się **osłoną** lub **opakowaniem** (ang. *wrapper*).

W pewnym sensie podobnym problemem do blokujących wywołań systemowych jest problem braku stron w pamięci (ang. *page faults*). Zagadnienie to omówimy w rozdziale 3. Na razie wystarczy, jeśli powiemy, że komputery można skonfigurować w taki sposób, aby w danym momencie w głównej pamięci znajdowała się tylko część programu. Jeżeli program wywoła lub skoczy do instrukcji, której nie ma w pamięci, występuje warunek braku strony. Wtedy system operacyjny jest zmuszony do pobrania brakującej instrukcji (wraz z jej sąsiadami) z dysku. Na tym właśnie polega warunek braku strony. Podczas gdy potrzebna instrukcja jest wyszukiwana i wczytywana, proces pozostaje zablokowany. Jeśli wątek spowoduje warunek braku strony, jądro, które nawet nie wie o istnieniu wątków, blokuje cały proces do czasu zakończenia dyskowej operacji wejścia-wyjścia. Robi to, mimo że nie ma przeszkód, by inne wątki działały.

Inny problem z pakietami obsługi wątków na poziomie użytkownika polega na tym, że jeśli wątek zacznie działać, to żaden inny wątek w tym procesie nigdy nie zacznie działać, o ile pierwszy wątek dobrowolnie nie zrezygnuje z procesora. W obrębie pojedynczego procesu nie ma przerwań zegara, dlatego nie ma możliwości szeregowania procesów w trybie cyklicznym (tzn. po kolei). Jeśli wątek z własnej woli nie przekaże sterowania do środowiska wykonawczego, program szeregujący nigdy nie będzie miał szansy działania.

Jednym z możliwych rozwiązań problemu wątków działających bez przerwy jest zlecenie środowisku wykonawczemu żądania sygnału zegara (przerwania) co sekundę w celu przekazania mu kontroli. Takie rozwiązanie okazuje się jednak toporne i trudne do zaprogramowania. Okresowe przerwania zegara z wyższą częstotliwością nie

zawsze są możliwe, a nawet jeśli tak jest, koszt obliczeniowy takiej operacji może być wysoki. Co więcej, wątek również może potrzebować przerwania zegara, co przeszkadza wykorzystaniu zegara przez środowisko wykonawcze.

Innym, rzeczywiście druzgoczącym argumentem przeciwko wątkom zarządzanym na poziomie przestrzeni użytkownika jest fakt, że programiści, ogólnie rzecz biorąc, potrzebują wątków w aplikacjach, gdzie wątki blokują się często — na przykład w wielowątkowym serwerze WWW. Wątki te bezustannie wykonują wywołania systemowe. Kiedy zostanie wykonany rozkaz pułapki do jądra w celu realizacji wywołania systemowego, jądro nie ma nic więcej do roboty przy przełączaniu wątków, w przypadku gdy stary wątek się zablokował, a zlecenie jądra wykonania tej czynności eliminuje potrzebę ciągłego wykonywania wywołań systemowych `select` sprawdzających, czy wywołania systemowe `read` są bezpieczne. Jaki jest sens istnienia wątków w aplikacjach całkowicie powiązanych z procesorem, które rzadko się blokują? Nikt nie jest w stanie zaproponować sensownego rozwiązania problemu wyliczania liczb pierwszych lub grania w szachy z wykorzystaniem wątków, ponieważ realizacja tych programów w ten sposób nie przynosi istotnych korzyści.

2.2.5. Implementacja wątków w jądrze

Rozważmy teraz sytuację, w której jądro wie o istnieniu wątków i to ono nimi zarządza. Środowisko wykonawcze w każdym z procesów nie jest wymagane, co pokazano na rysunku 2.9(b). Tabela wątków również nie występuje w każdym procesie. Zamiast tego jądro dysponuje tabelą wątków, która śledzi wszystkie wątki w systemie. Kiedy wątek chce utworzyć nowy wątek lub zniszczyć istniejący, wykonuje wywołanie systemowe, które następnie realizuje utworzenie lub zniszczenie wątku poprzez aktualizację tabeli wątków na poziomie jądra.

W tabeli wątków w jądrze są zapisane rejestry, stan oraz inne informacje dla każdego wątku. Informacje są takie same, jak w przypadku wątków zarządzanych na poziomie użytkownika, z tą różnicą, że są one umieszczone w jądrze, a nie w przestrzeni użytkownika (wewnątrz środowiska wykonawczego). Informacje te stanowią podzbiór informacji tradycyjnie utrzymywanych przez jądro na temat jednowątkowych procesów — czyli stanu procesów. Oprócz tego jądro utrzymuje również tradycyjną tabelę procesów, która służy do śledzenia procesów.

Wszystkie wywołania, które mogą zablokować wątek, są implementowane jako wywołania systemowe znacząco większym kosztem niż wywołanie procedury środowiska wykonawczego. Kiedy wątek się zablokuje, jądro może uruchomić wątek z tego samego procesu (jeśli jakiś jest gotowy) lub wątek z innego procesu. W przypadku wątków zarządzanych na poziomie przestrzeni użytkownika środowisko wykonawcze uruchamia wątki z własnego procesu do czasu, aż jądro zabierze mu procesor (lub nie będzie wątków gotowych do działania).

Ze względu na relatywnie większy koszt tworzenia i niszczenia wątków na poziomie jądra niektóre systemy przyjmują rozwiązanie „ekologiczne” i ponownie wykorzystują

swoje wątki. W momencie niszczenia wątku jest on oznaczany jako niemożliwy do uruchomienia, ale poza tym struktury danych jądra pozostają bez zmian. Kiedy później trzeba utworzyć nowy wątek, stary wątek jest reaktywowany, co eliminuje konieczność wykonywania pewnych obliczeń. Recykling wątków jest również możliwy w przypadku wątków zarządzanych w przestrzeni użytkownika, ale ponieważ koszty zarządzania wątkami są znacznie niższe, motywacja do korzystania z tego mechanizmu jest mniejsza.

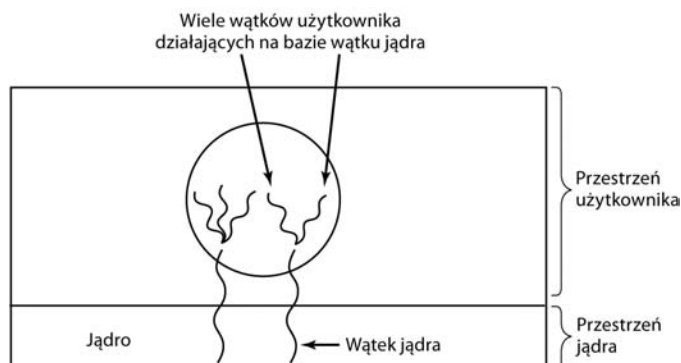
Wątki jądra nie wymagają żadnych nowych nieblokujących wywołań systemowych. Co więcej, jeśli jeden z wątków w procesie spowoduje warunek braku strony, jądro może łatwo sprawdzić, czy proces zawiera inne wątki możliwe do uruchomienia. Jeśli tak, uruchamia jeden z nich w oczekiwaniu na przesłanie z dysku wymaganej strony. Główną wadą tego rozwiązania jest fakt, że koszty wywołania systemowego są znaczące. W związku z tym, w przypadku dużej liczby operacji zarządzania wątkami (tworzenia, niszczenia itp.), ponoszone koszty obliczeniowe okazują się wysokie.

O ile wykorzystanie zarządzania wątkami na poziomie jądra rozwiązuje niektóre problemy, o tyle nie rozwiązuje ich wszystkich. Na przykład co się stanie, jeśli wielowątkowy proces wykona wywołanie `fork`? Czy nowy proces będzie miał tyle wątków, co stary, czy tylko jeden? W wielu przypadkach najlepszy wybór zależy od tego, do czego będzie służył nowy proces. Jeśli zamierza skorzystać z wywołania `exec` w celu uruchomienia nowego programu, prawdopodobnie właściwe będzie stworzenie procesu z jednym wątkiem, jeśli jednak ma on kontynuować działanie, reprodukcja wszystkich wątków wydaje się właściwsza.

Innym problemem są sygnały. Jak pamiętamy, sygnały są przesyłane do procesów, a nie do wątków (przynajmniej w modelu klasycznym). Który wątek ma obsłużyć nadchodzący sygnał? Można wyobrazić sobie rozwiązanie, w którym wątki rejestrują swoje zainteresowanie określonymi sygnałami. Dzięki temu w przypadku nadejścia sygnału mógłby on być skierowany do wątku, który na ten sygnał oczekuje. Co się jednak stanie, jeśli dwa wątki (lub większa liczba wątków) zarejestrują zainteresowanie tym samym sygnałem? To tylko dwa problemy, jakie stwarzają wątki. Jest ich jednak więcej.

2.2.6. Implementacje hybrydowe

Próbowano różnych rozwiązań mających na celu połączenie zalet zarządzania wątkami na poziomie użytkownika oraz zarządzania nimi na poziomie jądra. Jednym ze sposobów jest użycie wątków na poziomie jądra, a następnie zwielokrotnienie niektórych lub wszystkich wątków jądra na wątki na poziomie użytkownika. Sposób ten pokazano na rysunku 2.10. W przypadku skorzystania z takiego podejścia programista może określić, ile wątków jądra chce wykorzystać oraz na ile wątków poziomu użytkownika ma być zwielokrotniony każdy z nich. Taki model daje największą elastyczność.



Rysunek 2.10. Zwielokrotnianie wątków użytkownika na bazie wątków jądra

Przy tym podejściu jądro jest świadome istnienia *wyłącznie* wątków poziomu jądra i tylko nimi zarządza. Niektóre spośród tych wątków mogą zawierać wiele wątków poziomu użytkownika, stworzonych na bazie wątków jądra. Wątki poziomu użytkownika są tworzone, niszczone i zarządzane identycznie, jak wątki na poziomie użytkownika działające w systemie operacyjnym bez obsługi wielowątkowości. W tym modelu każdy wątek poziomu jądra posiada pewien zbiór wątków na poziomie użytkownika. Wątki poziomu użytkownika po kolei korzystają z wątku poziomu jądra.

2.2.7. Mechanizm aktywacji zarządcy

Chociaż zarządzanie wątkami na poziomie jądra jest lepsze od zarządzania nimi na poziomie użytkownika pod pewnymi istotnymi względami, jest ono również bezdyskusyjnie wolniejsze. W związku z tym poszukiwano sposobów poprawy tej sytuacji bez konieczności rezygnacji z ich dobrych właściwości. Poniżej opiszemy jedno z zaproponowanych rozwiązań, opracowane przez zespół kierowany przez Andersona [Anderson et al., 1992] i nazywane **aktywacją zarządcy** (ang. *scheduler activations*). Podobne prace zostały opisane w [Edlera et al., 1988] oraz [Scott et al., 1990].

Celem działania mechanizmu aktywacji zarządcy jest naśladowanie funkcji wątków jądra, ale z zapewnieniem lepszej wydajności i elastyczności — cech, które zwykle charakteryzują pakiety zarządzania wątkami zaimplementowane w przestrzeni użytkownika. W szczególności wątki użytkownika nie powinny wykonywać specjalnych, nieblokujących wywołań systemowych lub sprawdzać wcześniej, czy wykonanie określonych wywołań systemowych jest bezpieczne. Niemniej jednak, kiedy wątek zablokuje się na wywołaniu systemowym lub sytuacji braku strony, powinien mieć możliwość uruchomienia innego wątku w ramach tego samego procesu, jeśli jakiś jest gotowy do działania.

Wydajność osiągnięto dzięki uniknięciu niepotrzebnych przejść pomiędzy przestrzenią użytkownika a przestrzenią jądra. Jeśli na przykład wątek zablokuje się w oczekiwaniu na to, aż inny wątek wykona jakieś działania, nie ma powodu

informowania o tym jądra. Dzięki temu unika się kosztów związanych z przejściami pomiędzy przestrzeniami jądra i użytkownika. Środowisko wykonawcze przestrzeni użytkownika może samodzielnie zablokować wątek synchronizujący i zainicjować nowy.

Kiedy jest wykorzystywany mechanizm aktywacji zarządcy, jądro przypisuje określoną liczbę procesorów wirtualnych do każdego procesu i umożliwia środowisku wykonawczemu (przestrzeni użytkownika) na przydzielanie wątków do procesorów. Mechanizm ten może być również wykorzystany w systemie wieloprocessorowym, w którym zamiast procesorów wirtualnych są procesory fizyczne. Liczba procesorów wirtualnych przydzielonych do procesu zazwyczaj początkowo wynosi jeden, ale proces może poprosić o więcej, a także zwrócić procesory, których już nie potrzebuje. Jądro może również zwrócić wirtualne procesory przydzielone wcześniej, w celu przypisania ich procesom bardziej potrzebującym.

Podstawowa zasada działania tego mechanizmu polega na tym, że jeśli jądro dowie się o blokadzie wątku (na przykład z powodu uruchomienia blokującego wywołania systemowego lub braku strony), to powiadamia o tym środowisko wykonawcze procesu. W tym celu przekazuje na stos w postaci parametrów numer zablokowanego wątku oraz opis zdarzenia, które wystąpiło. Powiadomienie może być zrealizowane dzięki temu, że jądro uaktywnia środowisko wykonawcze znajdujące się pod znanym adresem początkowym. Jest to mechanizm w przybliżeniu analogiczny do sygnałów w Uniksie. Mechanizm ten określa się terminem **wezwanie** (ang. *upcall*).

Po uaktywnieniu w taki sposób środowisko wykonawcze może zmienić harmonogram działania swoich wątków. Zazwyczaj odbywa się to poprzez oznaczenie bieżącego wątku jako zablokowany oraz pobranie innego wątku z listy wątków będących w gotowości, ustawienie jego rejestrów i wznowienie działania. Później, kiedy jądro dowie się, że poprzedni wątek może ponownie działać (na przykład potok, z którego czytał dane, zawiera dane lub brakująca strona została pobrana z dysku), wykonuje kolejne wezwanie do środowiska wykonawczego w celu poinformowania go o tym zdarzeniu. Środowisko wykonawcze może wówczas we własnej gestii natychmiast zrestartować zablokowany wątek lub umieścić go na liście wątków do późniejszego uruchomienia.

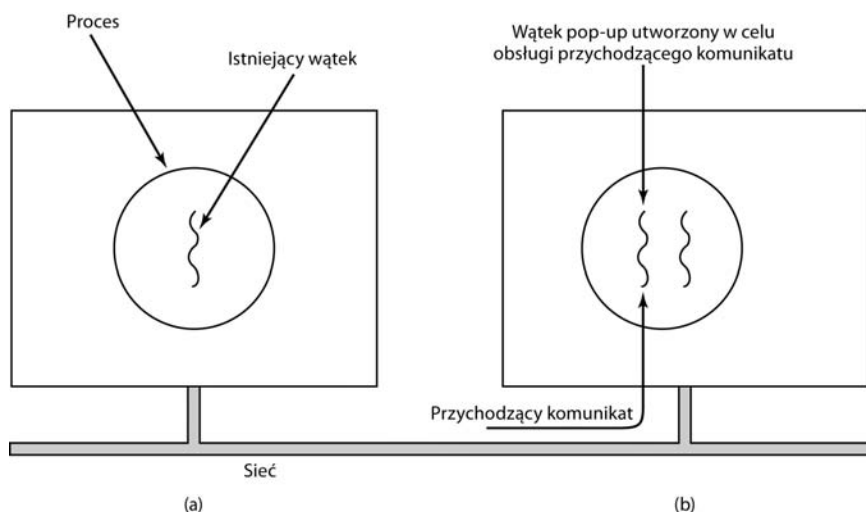
Jeżeli wystąpi przerwanie sprzętowe, gdy działa wątek użytkownika, procesor przełącza się do trybu jądra. Jeśli przerwanie jest spowodowane przez zdarzenie, którym przerwany proces nie jest zainteresowany — na przykład zakończenie operacji wejścia-wyjścia innego procesu — to kiedy procedura obsługi przerwania zakończy działanie, umieszcza przerwany wątek w tym samym stanie, w jakim znajdował się on przed wystąpieniem przerwania. Jeśli jednak proces jest zainteresowany przerwaniem — na przykład nadejście strony wymaganej przez jeden z wątków procesu — przerwany proces nie jest wznowiany. Zamiast tego jest on zawieszany, a na tym samym wirtualnym procesorze zaczyna działać środowisko wykonawcze — stan przerwonego wątku jest w tym momencie umieszczony na stosie. W tym momencie środowisko wykonawcze podejmuje decyzję o tym, jakiemu wątkowi przydzielić dany procesor: przerwanemu, nowo przygotowanemu do działania czy jakiemś innemu.

Wadą mechanizmu aktywacji zarządcy jest całkowite poleganie na wezwaniach — jest to pojęcie, które narusza wewnętrzną strukturę każdego systemu warstwowego. Zwykle warstwa n oferuje określone usługi, które może wywołać warstwa $n+1$, warstwa n nie może jednak wywoływać procedur w warstwie $n+1$. Mechanizm wezwań narusza tę podstawową zasadę.

2.2.8. Wątki pop-up

Wątki często się przydają w systemach rozproszonych. Istotnym przykładem może być sposób postępowania z nadchodzącymi komunikatami — na przykład zadaniami obsługi. Tradycyjne podejście polega na tym, że na komunikat oczekuje proces lub wątek zablokowany na wywołaniu systemowym receive. Kiedy nadejdzie komunikat, wątek ten przyjmuje go, rozpakowuje, analizuje zawartość i przetwarza.

Możliwe jest jednak całkiem inne podejście — po dotarciu komunikatu system tworzy nowy wątek do jego obsługi. Taki wątek określa się jako **wątek pop-up**. Zilustrowano go na rysunku 2.11. Zasadnicza zaleta wątków pop-up polega na tym, że ponieważ są one zupełnie nowe, nie mają żadnej historii — rejestrów, stosu, czegośkolwiek, co musiałoby być odtworzone. Każdy wątek rozpoczyna się jako nowy i wszystkie są identyczne. Dzięki temu takie wątki można tworzyć szybko. Nowy wątek otrzymuje przychodzący komunikat do przetworzenia. Dzięki zastosowaniu wątków pop-up opóźnienie pomiędzy przybyciem komunikatu a rozpoczęciem jego przetwarzania jest bardzo niewielkie.



Rysunek 2.11. Tworzenie nowego wątku po przybyciu pakietu: (a) zanim nadejdzie komunikat; (b) po nadejściu komunikatu

W przypadku wykorzystania wątków pop-up potrzebne jest pewne zaawansowane planowanie. Na przykład w którym procesie działa wątek? Jeśli system obsługuje wątki

działające w kontekście jądra, to wątek może tam działać (dlatego właśnie nie pokazaliśmy jądra na rysunku 2.11). Umieszczenie wątku pop-up w przestrzeni jądra zazwyczaj jest łatwiejsze i szybsze niż umieszczenie go w przestrzeni użytkownika. Ponadto wątek pop-up w przestrzeni jądra może łatwo uzyskać dostęp do wszystkich tabel i urządzeń wejścia-wyjścia jądra, co może być potrzebne do przetwarzania przerw. Z drugiej strony działający błędnie wątek jądra może zrobić więcej szkód niż błędnie działający wątek przestrzeni użytkownika. Jeśli na przykład działa zbyt długo i nie ma możliwości jego wywłaszczenia, wchodzące dane mogą zostać utracone.

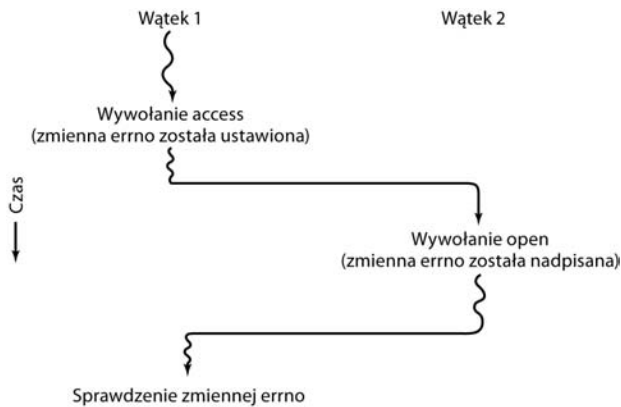
2.2.9. Przystosowywanie kodu jednowątkowego do obsługi wielu wątków

Dla procesów jednowątkowych napisano wiele programów. Ich konwersja na postać wielowątkową jest znacznie trudniejsza, niż mogłoby się wydawać na pierwszy rzut oka. Poniżej zaprezentujemy kilka problemów, które mogą wystąpić podczas takiej konwersji.

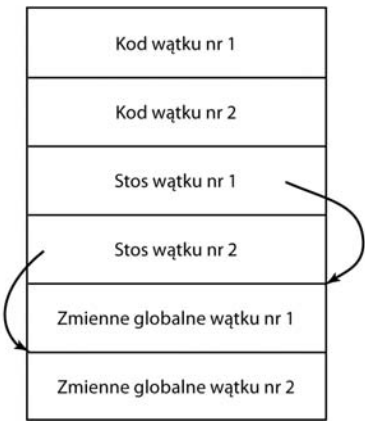
Na początek należy sobie uświadomić, że wątek, tak jak proces, zwykle składa się z wielu procedur. Mogą one mieć zmienne lokalne, zmienne globalne i parametry. Zmienne lokalne i parametry nie powodują żadnych problemów, tymczasem zmienne, które są globalne dla wątku, ale nie są globalne dla całego programu, sprawiają problem. Są to zmienne, które są globalne w tym sensie, że używa ich wiele procedur w obrębie wątku (ponieważ mogą one wykorzystywać dowolne zmienne globalne), ale inne wątki nie powinny z nich korzystać.

Dla przykładu przeanalizujemy zmienną `errno` występującą w systemie UNIX: kiedy proces (lub wątek) wykonuje wywołanie systemowe, które kończy się niepowodzeniem, do zmiennej `errno` jest zapisywany kod błędu. Na rysunku 2.12 wątek nr 1 wykonuje wywołanie systemowe `access` po to, aby się dowiedzieć, czy ma uprawnienia dostępu do określonego pliku. System operacyjny zwraca odpowiedź w zmiennej globalnej `errno`. Po zwróceniu sterowania do wątku 1., ale jeszcze przed przeczytaniem przez niego zmiennej `errno`, program szeregujący zdecydował, że wątek nr 1 miał przydzielony procesor wystarczająco długo i zdecydował przełączyć go do wątku 2. Wątek 2. uruchomił wywołanie systemowe `open`, które się nie powiodło. Spowodowało to nadpisanie zmiennej `errno`, a kod dostępu wątku 1. został utracony na zawsze. Kiedy wątek 1. później się uruchomi, przeczyta nieprawidłową wartość i będzie działał nieprawidłowo.

Możliwych jest wiele rozwiązań tego problemu. Jedno z nich polega na całkowitym wyłączeniu zmiennych globalnych. Choć mogłoby się wydawać, że jest to rozwiązanie idealne, koliduje ono z większością istniejących programów. Inne rozwiązanie to przypisanie każdemu wątkowi własnych, prywatnych zmiennych globalnych, tak jak pokazano na rysunku 2.13. W ten sposób każdy wątek będzie miał własną, prywatną kopię zmiennej `errno` i innych zmiennych globalnych, co pozwoli na uniknięcie konfliktów. Przyjęcie tego rozwiązania tworzy nowy poziom zasięgu:



Rysunek 2.12. Konflikty pomiędzy wątkami spowodowany użyciem zmiennej globalnej



Rysunek 2.13. Wątki mogą mieć prywatne zmienne globalne

zmienne widoczne dla wszystkich procedur wątku. Poziom ten występuje obok istniejących poziomów: zmienne widoczne tylko dla jednej procedury oraz zmienne widoczne w każdym punkcie programu.

Dostęp do prywatnych zmiennych globalnych jest jednak nieco utrudniony, ponieważ większość języków programowania zapewnia sposób wyrażania zmiennych lokalnych i zmiennych globalnych, ale nie ma form pośrednich. Można zaalokować fragment pamięci na zmienne globalne i przekazać go do każdej procedury w wątku w postaci dodatkowego parametru. Choć nie jest to zbyt eleganckie rozwiązanie, okazuje się jednak skuteczne.

Alternatywnie można stworzyć nowe procedury biblioteczne do tworzenia, ustawiania i czytania tych zmiennych globalnych na poziomie wątku. Pierwsze wywołanie może mieć następującą postać:

```
create_global("bufptr");
```

Wywołanie to alokuje pamięć dla wskaźnika o nazwie `bufptr` na sterpie lub w specjalnym obszarze pamięci zarezerwowanym dla wywołującego wątku. Niezależnie od tego, gdzie jest zaalokowana pamięć, tylko wywołujący wątek ma dostęp do zmiennej globalnej. Jeśli inny wątek utworzy zmienną globalną o tej samej nazwie, otrzyma inną lokalizację w pamięci — taką, która nie koliduje z istniejącą.

Do dostępu do zmiennych globalnych potrzebne są dwa wywołania: jedno do ich zapisywania i drugie do odczytu. Do zapisywania potrzebne jest wywołanie postaci:

```
set_global("bufptr", &buf);
```

Wywołanie to zapisuje wartość wskaźnika w lokalizacji pamięci utworzonej wcześniej przez wywołanie do procedury `create_global`. Wywołanie do przeczytania zmiennej globalnej może mieć następującą postać:

```
bufptr = read_global("bufptr");
```

Zwraca ono adres zapisany w zmiennej globalnej. Dzięki temu można uzyskać dostęp do jej danych.

Następny problem podczas przekształcania programu jednowątkowego na wielowątkowy polega na tym, że wiele procedur bibliotecznych nie pozwala na tak zwaną wielobieżność. Oznacza to, że nie ma możliwości wywołania innej procedury, jeśli poprzednie wywołanie się nie zakończyło. Na przykład wysyłanie wiadomości w sieci można by z powodzeniem zaprogramować w taki sposób, aby wiadomość była tworzona w ustalonym buforze w obrębie biblioteki, a następnie był wykonywany rozkaz pułapki do jądra w celu jej wysłania. Co się stanie, jeśli jeden wątek utworzył swoją wiadomość w buforze, a następnie przerwanie zegara wymusiło przełączenie do drugiego wątku, który natychmiast nadpisze bufor własną wiadomością.

Podobnie procedury alokacji pamięci, na przykład `malloc` w Uniksie, utrzymują kluczowe tabele dotyczące wykorzystania pamięci — na przykład powiązaną listę dostępnych fragmentów pamięci. Podczas gdy procedura `malloc` jest zajęta aktualizacją tych list, mogą one czasowo być w niespójnym stanie — zawierać wskaźniki donikąd. Jeśli nastąpi przełączenie wątku w chwili, gdy tabele będą niespójne i nadejdzie nowe wywołanie z innego wątku, może dojść do użycia nieprawidłowego wskaźnika, co w efekcie może doprowadzić do awarii programu. Skuteczne wyeliminowanie wszystkich tych problemów oznacza konieczność przepisania od nowa całej biblioteki. Wykonanie takiego przedsięwzięcia nie jest proste.

Innym rozwiązaniem jest wyposażenie każdej procedury w kod opakowujący, który ustawia bit do oznaczenia biblioteki tak, jakby była używana. Każda próba innego wątku skorzystania z procedury bibliotecznej, podczas gdy poprzednie wywołanie nie zostało zakończone, jest blokowana. Chociaż takie rozwiązanie jest wykonalne, w dużym stopniu eliminuje ono możliwość wykorzystania współbieżności.

Inną opcją jest wykorzystanie sygnałów. Niektóre sygnały z logicznego punktu widzenia są specyficzne dla wątku, a inne nie. Jeśli na przykład wątek wykonuje wywołanie `alarm`, logiczne jest, aby wynikowy sygnał został przesłany do wątku, który wykonał wywołanie. Jeśli jednak wątki są zaimplementowane w całości w prze-

strzeni użytkownika, jądro nie wie nawet o istnieniu wątków, a zatem trudno mu skierować sygnał do właściwego wątku. Dodatkowe komplikacje występują w przypadku, gdy proces pozwala na występowanie tylko jednego nieobsłużonego alarmu w danym momencie, a kilka wątków niezależnie wykonuje wywołanie alarmu.

Inne sygnały, na przykład przerwanie klawiatury, nie są specyficzne dla wątku. Co powinno je przechwycić? Wyznaczony wątek? Wszystkie wątki? Nowo utworzony wątek pop-up? Ponadto co się stanie, jeśli jeden wątek zmieni procedury obsługi sygnałów bez informowania pozostałych wątków? A co się wydarzy, kiedy jeden wątek będzie chciał przechwycić określony sygnał (na przykład wciśnięcie przez użytkownika kombinacji *Ctrl+C*), a inny wątek będzie potrzebował tego sygnału do zakończenia procesu? Taka sytuacja może wystąpić, jeśli jeden wątek lub kilka wątków korzysta ze standardowych procedur bibliotecznych, a inne są napisane przez użytkownika. Jest oczywiste, że życzenia tych wątków kolidują ze sobą. Ogólnie rzecz biorąc, sygnały są trudne do zarządzania w środowisku jednowątkowym. Przejście do środowiska wielowątkowego w żaden sposób nie ułatwia zarządzania nimi.

Ostatnim problemem związanym z wątkami jest zarządzanie stosem. W wielu systemach, w przypadku wystąpienia przepełnienia stosu, jądro automatycznie dostarcza takiemu procesowi więcej miejsca na stosie. Jeśli proces ma wiele wątków, musi również mieć wiele stosów. Jeśli jądro nie posiada informacji o wszystkich tych stosach, nie może ich automatycznie rozszerzać, gdy wyczerpie się na nich miejsce. W rzeczywistości jądro może nawet nie wiedzieć, że brak strony w pamięci jest związany z rozszerzeniem się stosu jakiegoś wątku.

Problemy te nie są oczywiście nie do rozwiązania, ale pokazują, że wprowadzenie wątków do istniejącego systemu bez znaczącej jego przebudowy nie zadziała. Trzeba co najmniej zmodyfikować definicję semantyki wywołań systemowych oraz biblioteki. Wszystkie te czynności trzeba dodatkowo wykonać tak, aby zachować wsteczną zgodność z istniejącymi programami, przy założeniu, że wykorzystują one procesy zawierające po jednym wątku. Więcej informacji na temat wątków można znaleźć w następujących pozycjach [Hauser et al., 1993] oraz [Marsh et al., 1991].

2.3. KOMUNIKACJA MIĘDZY PROCESAMI

Procesy często muszą się komunikować z innymi procesami. Na przykład w przypadku potoku w powłoce wyjście pierwszego procesu musi być przekazane do drugiego procesu, i tak dalej, do niższych warstw. Tak więc występuje potrzeba komunikacji między procesami. Najlepiej, gdyby miała ona czytelną strukturę i gdyby nie korzystano w niej z przerwań. W poniższych punktach przyjrzymy się niektórym problemom związanym z komunikacją międzyprocesową (ang. *InterProcess Communication — IPC*).

Mówiąc w skrócie: wiążą się z tym trzy problemy. O pierwszym była mowa już wcześniej: w jaki sposób jeden proces może przekazywać informacje do innego?

Drugi polega na zapobieganiu sytuacji, w której dwa procesy (lub większa liczba procesów) wchodzi sobie wzajemnie w drogę. Na przykład dwa procesy w systemie rezerwacji biletów jednocześnie próbują przydzielić ostatnie miejsce w samolocie — każdy innemu klientowi. Trzeci wiąże się z odpowiednim kolejkowaniem, w przypadku gdy występują zależności: jeśli proces *A* generuje dane, a proces *B* je drukuje, przed rozpoczęciem drukowania proces *B* musi czekać, aż proces *A* wygeneruje jakieś dane. Wszystkie trzy wymienione problemy omówimy, począwszy od następnego punktu.

Warto również wspomnieć o tym, że dwa spośród tych problemów mają w równym stopniu zastosowanie do wątków. Pierwszy z nich — przekazywanie informacji — jest łatwy w odniesieniu do wątków, ponieważ wykorzystują one wspólną przestrzeń adresową — wątki w różnych przestrzeniach adresowych, które muszą się komunikować, można zaliczyć do tej samej klasy problemów, do których należy komunikacja pomiędzy procesami. Jednak pozostałe dwa problemy — powstrzymanie od „skakania sobie do oczu” i kolejkovanie — mają zastosowanie do procesów w takim samym stopniu, jak do wątków. Występują te same problemy i można zastosować takie same rozwiązania. Poniżej omówimy te problemy w kontekście procesów. Pamiętajmy jednak o tym, że te same problemy i rozwiązania mają zastosowanie także do wątków.

SKOROWIDZ

#define, 112, 330
#include, 113, 329, 872
\$AttrDef, 1083
\$END, 39
\$Extend\$Secure, 1085
\$JOB, 39
\$LOAD, 39
\$RUN, 39
&, 870
.NET, 971
/dev, 916
/dev/hd1, 916
/dev/lp, 916
/etc/passwd, 898
/etc/rc, 947
/etc/ttys, 898
/proc, 945
|, 869
<, 869
>, 869
1003.1, 858
1394, 65
1401, 41
1BSD, 856
286, 967
2BSD, 856
32V, 856
386, 671
3BSD, 856
4.3BSD, 857
4.4BSD, 862
4BSD, 856
7094, 41
8080, 45, 46
8088, 242, 966

A

absolute paths, 927
abstrakcja, 33, 1145
access, 751
Access Control Entries, 1096
Access Control List, 733, 983
access token, 1095
access violation, 1047, 1056
access(), 158, 956, 957
accounting, 886
ACE, 1096
ACL, 733, 983, 1008
ACM, 116
ACPI, 507, 991
ActiveX, 811
ad infinitum, 813
Ada, 36
adapter graficzny, 487
adaptery, 402
ADC, 565
AddAccessAllowedAce(), 1098
AddAccessDeniedAce(), 1098
Address Windowing Extensions, 1051
adres IP, 692, 758, 759
 notacja dziesiętna z kropkami, 758
adres liniowy, 304
adres URL, 693, 1165
adres wirtualny, 243, 245, 287, 294, 301
adresowanie wielkich pamięci
 fizycznych, 1050
ADSL, 556, 557, 558, 920
Advanced Configuration and Power
 Interface, 507, 991
Advanced Local Procedure Call, 1032
Advanced LPC, 1003

Advanced Research Projects Agency,
 115, 856
adversaries, 721
adware, 813
Aero, 48
affinitize, 1023
affinity scheduling, 642, 895
Agencja Zaawansowanych Projektów
 Badawczych, 856
agent, 836
aging, 264
AGP, 1067
Aiken Howard, 37
AIX, 852
akceptowalne odtwarzanie strumieni
 multimedialnych, 560
aktywacja zarządcy, 155
 blokada wątku, 156
 wezwanie, 156
aktywne komunikaty, 657
aktywne oczekiwanie, 167, 418, 546
akumulacja nagłówków pakietów, 692
alarm czasowy, 880
alarm(), 160, 880, 883
Algol 68, 175
algorithmic paradigm, 1152
algorytm alokacji bazującej
 na jednokierunkowej liście
 z wykorzystaniem tabeli
 w pamięci, 341
algorytm alokacji procesorów, 666
algorytm bankiera
 dla pojedynczego zasobu, 537
 dla wielu zasobów, 538
algorytm binarnego wykładniczego
 cofania, 637
algorytm bliźniaków, 908, 909

- algorytm częstości błędów
 - braku stron, 274
- algorytm Dekkera, 168
- algorytm deszyfrujący, 724
- algorytm drugiej szansy, 260
- algorytm EDF, 581, 611
- algorytm FCFS, 456
- algorytm FIFO, 260
- algorytm JPEG, 568
- algorytm Lamporta, 763
- algorytm LRU, 262, 376
- algorytm MPEG, 571
- algorytm najgorszy pasujący, 240
- algorytm najlepszy pasujący, 240
- algorytm następny pasujący, 239
- algorytm NFU, 263
- algorytm NRU, 258
- algorytm odbierania ramek stron, 912
- algorytm organowy, 599
- algorytm Petersona, 168
- algorytm PFF, 274
- algorytm PFRA, 912, 913
- algorytm podejmowania decyzji
 - w zakresie przydziału i
 - zwalniania zasobów, 525
- algorytm postarzania, 264
- algorytm RMS, 611
- algorytm scan-EDF, 609
- algorytm SSF, 456
- algorytm sterowania
 - przyjmowaniem zgłoszeń, 560
- algorytm strusia, 526
- algorytm szeregowania, 193
 - bez wywłaszczania, 197
 - cele, 198
 - cykliczne szeregowanie, 204
 - gwarantowane szeregowanie, 208
 - kategorie, 197
 - loteryjne szeregowanie, 209
 - najpierw najkrótsze zadanie, 202
 - następny najkrótszy proces, 208
 - następny proces o najkrótszym
 - pozostałym czasie
 - działania, 203
 - oddzielanie strategii
 - od mechanizmu, 212
 - pierwszy zgłoszony, pierwszy
 - obsłużony, 201
 - priority, 205
 - sprawiedliwe szeregowanie, 210
 - starzenie, 208
 - system czasu rzeczywistego, 210
 - szeregowanie dwupoziomowe, 642
 - wielokrotne kolejki, 206
 - z wywłaszczaniem, 197
- algorytm szeregowania żądań dostępu
 - do dysku, 456
 - algorytm windy, 457
 - FCFS, 456
 - SSF, 457
- algorytm szybkiego dopasowania, 241
- algorytm windy, 457, 923
- algorytm WSClock, 269
- algorytm współdzielenia
 - przestrzeni, 643
- algorytm wykrywania zakleszczeń,
 - 530, 531
- algorytm wymiany stron, 913
- algorytm XOR, 765
- algorytm zarządzania ramieniem
 - dysku, 509
- algorytm zastępowania stron, 257,
 - 271, 1058
 - bazowanie na zbiorze
 - roboczym, 265
 - bieżący czas wirtualny, 268
 - druga szansa, 260
 - FIFO, 260, 275
 - globalny algorytm, 273
 - lokalność odwołań, 265
 - LRU, 262
 - model zbioru roboczego, 266
 - NFU, 263
 - NRU, 258
 - optymalny algorytm, 258
 - PFF, 274, 275
 - postarzanie, 264
 - programowa symulacja
 - algorytmu LRU, 263
 - przeladowanie, 266
 - rozmiar strony, 277
 - stronicowanie na żądanie, 265
 - stronicowanie wstępne, 266, 267
 - WSClock, 269
 - zarządzanie obciążeniem, 276
 - zegarowy algorytm, 261
- algorytm zegarowy, 261
- alokacja
 - pamięć, 160
 - procesory, 666
 - przestrzeń dyskowa, 1086
- ALPC, 1003, 1022, 1032
- alto, 118
- AMD, 629
- AMD Pacific, 672
- AMD64, 1100
- Amoeba, 739
- amplituda, 565
- analiza podpisów, 771
- Analog Digital Converter, 565
- Andreesen Marc, 115
- ANSI, 442, 480, 858
- ANSI C, 330
- APC, 989, 995, 996
- API, 675
- aplety, 71, 836, 840
- aplikacja, 31
 - internetowe, 684
 - system Windows, 489
- append message, 734
- append(), 328
- Apple, 47
- Apple FAT, 387
- Apple Lisa, 486
- Apple Macintosh, 47
- application engines, 1109
- Application Programming Interface,
 - 97, 675
- application verifier, 1015
- ar, 872
- architectural coherence, 1151
- architektura, 32
- architektura 386, 671
- architektura oznaczona, 737
- architektura pamięci
 - z podwójną magistralą, 406
- architektura pamięci
 - z pojedynczą magistralą, 406
- architektura publikuj-subskrybuj, 705
- architektura systemu NFS, 946
- archiwum, 324
- argc, 881
- argv, 881
- ARM, 1125
- ARPA, 115, 856
- ARPANET, 115, 687, 919
- ASCII, 95, 111
- Asymmetric Digital Subscriber Line,
 - 556, 920
- asynchroniczne wywołania
 - procedur, 996
- Asynchronous Procedure Calls,
 - 989, 996
- AT&T, 44, 854, 856, 863
- ataki, 722
 - blokada usługi, 720
 - DoS, 720, 1100
 - na warunek braku
 - wywłaszczania, 541
 - na warunek cyklicznego
 - oczekiwania, 542
 - na warunek wstrzymania
 - i oczekiwania, 541
 - na warunek wzajemnego
 - wykluczania, 540
 - poprzez upodobnienie, 835
 - powrót do biblioteki libc, 782
 - przyjmowanie przeglądarki, 813
 - przepelnienie bufora, 777
 - przepelnienie liczb
 - całkowitych, 783
 - rozszerzanie uprawnień, 786
 - wstrzykiwanie kodu, 784
 - z wewnątrz, 772
 - z wykorzystaniem łańcucha
 - formatującego, 780, 781
 - z wykorzystaniem
 - oprogramowania
 - szpiegującego, 812
- Atanasoff John, 37
- ATM, 766
- ATM OC-3, 559
- atrybuty nierezydentne, 1084
- atrybuty plików, 325, 326
- audio, 555, 565
 - ADC, 565
 - amplituda, 565
 - decybele, 565
 - digitalizacja, 566
 - efekt maskowania, 576
 - fala sinusoidalna, 566
 - kodowanie, 565
 - kompresja, 574
 - konwersja na postać cyfrową, 565
 - kwantyzacja próbek, 566
 - modulacja PCM, 566
 - MP3, 574

płyty CD, 566
 próbkowanie, 566
 próg słyszalności, 576
 rejestrowanie, 611
 szum kwantyzacji, 566
 twierdzenie Nyquista, 565
 zakres częstotliwości ludzkiego ucha, 565
 zakres dynamiczny ucha, 566
 audio CD, 566
 audyty bezpieczeństwa, 1094
 automat o skończonej liczbie stanów, 142
 Automated Teller Machine, 766
 automatyczne montowanie, 947
 autorun.inf, 818
 AV, 462
 AWE, 1051

B

B, 571, 854
 B+-drzewa, 1085, 1089
 B2B, 67
 Babbage Charles, 36
 backbone, 688
 bad-news diode, 1190
 balance set manager, 1060
 bandwidth reservation, 1069
 bank switching, 1050
 bankomat, 766
 bariery, 191, 192
 problem relaksacji, 192
 Base Class Library, 974
 bash, 80, 867
 BASIC, 46, 966
 Basic Input Output System, 66
 batch systems, 852
 baterie, 500
 zarządzanie, 506
 baza danych numerów ramek stron, 1060
 baza danych terminali, 480
 BCL, 974
 BCPL, 854
 bdfush(), 915
 begin transaction, 353
 behavioral checking, 828
 Bell Labs, 43, 853, 854
 Berkeley Fast File System, 938, 1185
 Berkeley Software Distribution, 44
 Berkeley UNIX, 856, 857
 Berners-Lee Tim, 115
 bezpieczeństwo, 717, 719, 720
 atak powrotu do biblioteki libc, 782
 atak przejmowania przeglądarki, 813
 atak rozszerzania uprawnień, 786
 atak z wewnątrz, 772
 atak z wykorzystaniem łańcuchów formatujących, 780
 atak z wykorzystaniem przepelnienia liczb, 783
 ataki, 722
 badania, 844

błędy w kodzie, 776
 bomba logiczna, 773
 C2, 1093
 deskryptor bezpieczeństwa, 1096
 domeny ochrony, 730
 dostępność systemu, 720
 efektywny identyfikator UID, 955
 firewall, 820
 hasła, 755, 761, 829, 957
 hasła jednorazowe, 763
 integralność danych, 720
 internet, 718
 intruzi, 721
 izolowanie, 837
 izolowanie kodu mobilnego, 835
 Java, 841
 karty chipowe, 766
 karty z paskiem magnetycznym, 766
 koń trojański, 790
 kopie zapasowe, 830
 kryptografia, 723
 Linux, 953
 lista uprawnień, 737
 listy kontroli dostępu, 733
 logowanie, 755, 957
 macierz ochrony, 733
 model formalny bezpiecznych systemów, 743
 moduł TPM, 729
 obowiązkowa kontrola dostępu, 745
 obrona wielostrefowa, 820
 oponenci, 721
 oprogramowanie szpiegujące, 809
 plik hasel, 758
 podglądanie przez osoby z wnętrza organizacji, 722
 podpisywanie kodu, 831
 podszywanie się pod ekran logowania, 774
 Pomarańczowa księga, 1093
 poufność danych, 720
 prywatność, 721
 przechwytywanie pakietów, 760
 przepełnienie bufora, 777
 przez ukrywanie, 724
 przypadkowa utrata danych, 723
 relacje zaufania, 1135
 robaki, 806
 rootkit, 813
 SETGID, 956
 SETUID, 742, 955
 skanowanie portów, 759
 skrypciarze, 761
 steganografia, 751
 swobodna kontrola dostępu, 745
 Symbian OS, 1132, 1133
 system wykrywania włamań, 822
 system zaufany, 740
 szpiegostwo gospodarcze, 722
 szyfrowanie, 724
 środowisko bezpieczeństwa, 719
 token dostępu, 1095
 tylne drzwi, 773
 ukryte kanały, 748

unikanie wirusów, 829
 uprawnienia, 718, 736
 uwierzytelnianie, 753
 uznaniowa kontrola dostępu, 745
 weryfikator integralności, 828
 Win32 API, 983
 Windows Vista, 1093
 wirusy, 793
 włamanie do systemu, 756
 wstrzykiwanie kodu, 784
 wtrącanie do więzienia, 832
 wykrywanie włamań z użyciem modeli, 833
 wymuszona kontrola dostępu, 745
 wywołania systemowe, 956
 zagrożenia, 720
 złośliwe oprogramowanie, 786
 zombie, 721
 bezpieczeństwo wielopoziomowe, 745
 model Bella-La Paduli, 745, 747
 model Biby, 747
 prosta reguła bezpieczeństwa, 746
 reguła *, 746
 bezpieczne logowanie, 1094
 bezpieczny system operacyjny, 740
 bezpośredni dostęp do pamięci, 63, 407
 adresy wirtualne, 410
 algorytm cykliczny, 409
 burst mode, 409
 cycle stealing, 409
 fizyczne adresy pamięci, 409
 fly-by mode, 409
 inicjowanie transferu, 408
 kontroler DMA, 407
 Symbian OS, 1128
 transfer, 408
 tryb przelotu, 409
 tryb wiązki, 409
 zabieranie cykli, 409
 biblioteki DLL, 281, 282, 1019, 1020
 cykle w grafie zależności, 1020
 dynamiczne ładowanie kodu, 1021
 IAT, 1020
 kontrolki ActiveX, 1021
 model goszczenia, 1021
 piekło DLL, 1020
 tablica importowanych adresów, 1020
 wykonywanie obok siebie, 1021
 zarządzanie wersjami, 1020
 biblioteki dołączane dynamicznie, 281
 biblioteki klas bazowych, 974
 biblioteki współdzielone, 281
 Bidirectional, 571
 bieżący czas wirtualny, 268
 bieżący katalog roboczy, 96
 Big Kernel Lock, 895
 bilety, 209
 binarne wykładnicze cofanie, 637
 binary exponential backoff, 637, 687
 binary translation, 674
 BIO, 1117
 BIOS, 66, 229, 896, 1006
 bit brudnej strony, 1055
 bit NX, 1100
 bit Obecna/nieobecna, 245, 255

bit oczekiwania na sygnał wakeup, 173
 bit SETGID, 956
 bit SETUID, 374, 742, 776, 779, 782, 792, 955
 bit wykorzystania strony, 1055
 bit zabrudzenia, 248
 BitBlt(), 494, 495
 BitLocker, 730, 1093
 Bitmap, 498
 bity rwx, 79
 BKL, 895
 black-hat hackers, 755
 bloatware, 241
 Block Started by Symbol, 899
 blok BSS, 899
 blok dwupośredni, 392, 943
 blok jednapośredni, 392, 942
 blok PEB, 1023
 blok podpisu, 728
 blok startowy, 337
 blok środowiska wątku, 1037
 blok TEB, 1023
 blok trójpśredni, 392, 943
 blokada usług, 720
 blokady, 548
 pętlowa, 167, 636, 895, 991
 współdzielona, 930
 wyłączna, 930
 bloki zarządzania procesami, 133
 blokowa pamięć podręczna, 375
 blokowanie, 930
 blokowanie dwufazowe, 543
 blokowanie magistrali pamięci, 169
 blokowanie przerwai, 166
 blokowanie przewijania zawartości ekranu, 478
 blokowanie stron w pamięci, 289
 blokowanie zmiennych, 166
 blokowe pliki specjalne, 78, 323, 928
 blokujące operacje wejścia-wyjścia, 1129
 blokujące wywołania systemowe, 142, 152
 Blue Screen Of Death, 1001
 Bluetooth, 479, 1139
 Blu-ray, 452, 556
 błędy braku stron, 287, 472, 1047
 Windows Vista, 1054
 błędy programistyczne, 845
 błędy w kodzie, 776
 błędy w oprogramowaniu, 776
 błędy wejścia-wyjścia, 431
 bomba czasowa, 773
 bomba logiczna, 773
 boot, 896
 boot block, 337
 boot sector viruses, 799
 BootMgr, 1006
 botnet, 787
 bounded-buffer, 172
 Bourne Again Shell, 867
 Bourne shell, 867
 brak strony, 245
 brama wywołania, 306
 branch prediction tables, 675
 break, 331
 breakpoint, 1050

bridges, 687
 brightness, 564
 brk(), 903, 904
 broadcast quality, 571
 Brooks Fred, 41
 browser hijacking, 813
 BSD, 44, 48, 307, 858
 BSDI, 863
 BSOD, 1001
 BSS, 899, 900
 BT.PRT, 1139
 buddy algorithm, 908
 buffer underrun, 416
 bufor cykliczny, 429
 bufor TLB, 250, 286, 1052
 miękkie chybienie, 253
 MULTICS, 301
 obsługa chybień, 252
 programowe zarządzanie, 251
 twarde chybienie, 253
 buforowa pamięć podręczna, 375
 buforowanie, 55, 375, 416, 428, 430, 1183
 bloki, 603
 błędy zbyt wczesnego opróżniania bufora, 416
 bufor cykliczny, 429
 multimedia, 603
 pliki, 605
 pliki multimedialne, 595
 ścieżki do plików, 1184
 wyjście, 430
 buforowanie w przestrzeni jądra z kopiowaniem do przestrzeni użytkownika, 429
 buforowanie w przestrzeni użytkownika, 429
 burst mode, 409
 Business-To-Business, 67
 busy waiting, 61

C

C, 104, 111, 854, 1144
 #define, 112
 #include, 113
 kompilacja, 114
 konsolidacja, 114
 liczby, 111
 linker, 114
 Makefile, 113
 makra, 112
 model fazy działania, 114
 pliki nagłówkowe, 112
 pliki obiektowe, 113
 preprocesor, 113
 projekt programistyczny, 113
 typy danych, 111
 wskaźnik, 111
 C#, 971
 C2, 1093
 CA, 729
 cache, 53, 140, 375
 cache hit, 55
 cache line, 55, 621
 Cache-Coherent NUMA, 626
 caching, 1183
 CALL, 838
 CALLBACK, 491
 capability list, 737
 case, 870
 case-insensitive, 978
 case-preserving, 978
 cat, 80, 131, 871, 873
 cavity viruses, 798
 cc, 129
 CCD, 561, 562
 CC-NUMA, 626
 katalogowy system wieloprocesorowy, 626
 CD, 382, 442, 611
 Czerwona księga, 442
 płyty dźwiękowe, 442
 CD audio, 559
 CDC 6600, 84
 CDE, 483
 CDMA, 1117
 CD-R, 447
 CD-ReWritable, 449
 CD-ROM, 77, 354, 381, 444, 556
 CDROM XA, 448
 CD-RW, 449
 Central Processing Unit, 50
 centrum certyfikacji, 729
 CERT, 808
 Certification Authority, 729
 certyfikaty, 729
 cfgetospeed(), 920
 cfgetispeed(), 920
 cfsetispeed(), 920
 cfsetospeed(), 920
 challenge-response, 765
 char, 111
 charakterystyki czasowe systemów, 816
 charakterystyki obciążenia dysku, 509
 chdir(), 89, 96, 935
 checkerboarding, 298
 checkpoint, 533
 checksum, 828
 chief programmer team, 1189
 child process, 877
 chip, 766
 Chip-level MultiProcessors, 629
 chmod, 871, 873
 chmod(), 89, 96, 956
 Chorus, 859
 chown(), 956
 chrominacja, 564
 chronione procesy, 1101
 chwila, 893
 ciągła alokacja, 338, 339
 cienki klient, 496, 497, 509
 Bitmap, 498
 Copy, 498
 Pfill, 498
 Raw, 498
 Sfill, 498
 terminal X, 497
 THINC, 498
 ciphertext, 724
 CISC, 438
 CL, 386

cleaner, 351
 client stub, 658
 clipping region, 492
 C-list, 737
 clock ticks, 467
 clocks, 466
 clone(), 889
 CLONE_FILES, 890, 891
 CLONE_FS, 890, 891
 CLONE_PARENT, 890, 891
 CLONE_PID, 890, 891
 CLONE_SIGHAND, 890
 CLONE_VM, 890
 close(), 89, 93, 327, 584, 919, 932
 Close(), 1012
 closedir(), 336, 935
 CloseHandle(), 99
 CLR, 974
 Clusters of Workstations, 647
 CM-2, 440
 CMOS, 57, 66
 CMP, 230, 234, 629
 CMS, 107
 COBOL, 82, 1144
 code injection, 785
 code reviews, 774
 color, 564
 Colossus, 37
 COM, 985, 1021
 Common Criteria, 1003
 Common Desktop Environment, 483
 Common Language Run-time, 974
 Common Object Request Broker
 Architecture, 700
 Compact Disc, 442
 Compact Disc — Read Only
 Memory, 444
 companion virus, 794
 compare&swap, 991
 Compatible Time Sharing System, 43
 Component Object Model, 985, 1021
 composite, 563
 Computer Emergency Response
 Team, 808
 computer utility, 43
 confinement problem, 749
 connect(), 1116
 connectionless service, 689
 connection-oriented service, 689
 constant data length, 596
 constant time length, 596
 CONTEXT, 1028
 Control Program for
 Microcomputers, 46
 Conversational Monitor System, 107
 cooked mode, 475
 coordination-based middleware, 701
 Copy, 498
 copy object, 734
 copy on write, 887, 1050
 CopyFile(), 981
 CORBA, 700
 IDL, 700
 IIOP, 701
 namiastka procedury, 700
 obiekty, 700
 ORB, 700
 tworzenie obiektu, 700

Core 2 Duo, 48
 core dump, 779
 core image, 91
 core memory, 57
 Core Wars, 829
 co-scheduling, 645
 covert channel, 749
 COWS, 647
 cp, 91, 333, 864, 871, 873, 881
 CP/M, 46, 966
 CPL, 854
 CPU, 50
 CPU-bound, 894
 crackers, 755
 creat(), 330, 931, 932
 create domain, 743
 create object, 743
 create(), 327, 335
 create_global(), 159
 CreateDirectory(), 98, 99
 CreateFiber(), 1035
 CreateFile(), 99, 981, 1017
 CreateFileMapping(), 1051, 1052
 CreateMutex(), 1035
 CreateProcess(), 98, 99, 128, 974,
 975, 1029, 1035, 1036, 1038, 1156
 CreateProcessA(), 981
 CreateProcessW(), 981
 CreateSemaphore(), 1010, 1033, 1035
 CreateThread(), 1035
 CreateWindow(), 490
 critical sections, 1034
 cron, 876
 crosspoint, 622
 CRT, 402, 562
 cryptographic hash function, 727
 csh, 80, 857, 867
 csrss.exe, 975, 1004
 CSY, 1139
 CTSS, 43, 206, 853
 cut, 871, 873
 Cutler Dave, 48, 968
 cycle stealing, 409
 cyfrowe biblioteki, 116
 cyfrowy film wideo, 555
 cyfrowy sygnał wideo, 564
 cykl procesora, 50
 cylinder, 58
 cylinder skew, 453
 czas cyklu przetwarzania, 200
 czas kojarzenia nazw, 1167
 czas odpowiedzi, 200
 czas rzeczywisty, 468
 czasowy przydział zasobu, 35
 czcionki, 495
 TrueType, 495
 Czerwona księga, 442
 czytanie bloków zawczasu, 378

D

D, 279
 DAC, 745
 DACL, 1095, 1099
 daemons, 876

DAG, 347
 dane, 84
 nieinicjalizowane, 900
 wejsciowe, 473
 współdzielone użytkownika, 1024
 data caging, 1136
 data confidentiality, 720
 data integrity, 720
 datagramy, 690, 691, 692
 datagramy z potwierdzeniami, 690
 dB, 565
 DCT, 569, 1181
 deadlock, 182, 517
 DebugPortHandle(), 977
 DEC, 48, 968
 DEC Alpha, 968
 DEC PDP-1, 44
 decybele, 565
 defense in depth, 820
 Deferred Procedure Call, 995
 defrag, 380
 defragmentacja dysków, 380
 plik hibernacyjny, 381
 plik stronicowania, 381
 rejestr księgowania, 381
 degradowanie procesów, 894
 Dekker Thomas, 168
 dekodowanie, 567
 delete domain, 743
 delete object, 743
 delete(), 327, 335
 Delete(), 1012
 DeleteAce(), 1098
 DeleteFile(), 99
 demand paging, 837
 demon stron, 912
 demony, 127, 433, 876
 cron, 876
 drukarka, 162
 finger, 807
 kswapd, 913
 stronicowanie, 284
 Denial of Service, 720, 1100
 dentry, 936, 937
 deskryptor adresu wirtualnego, 1053
 deskryptor bezpieczeństwa, 976, 1096
 deskryptor pliku, 77, 330, 584, 932
 deskryptor procesu, 885
 obraz pamięci, 885
 parametry szeregowania, 885
 rejestry maszynowe, 885
 rozliczenia, 886
 stan wywołania systemowego, 885
 zasoby jądra, 886
 sygnały, 885
 tablica deskryptorów plików, 885
 deskryptor segmentu kodu, 303
 deskryptor strefy, 906
 deskryptor węzłów, 907
 deskryptor woluminu, 383
 destroy object, 734
 deszyfrowanie, 724
 dev, 78
 device driver virus, 801
 device drivers, 1004
 Device Independent Bitmap, 495

device stack, 1004
 DFC, 1118
 DIB, 495
 dig, 759
 Digital Research, 46, 966
 Digital Rights Management, 989, 1101
 Digital Versatile Disk, 450, 556
 Digital Video, 571
 Digital Video Disk, 450
 digitalizacja dźwięku, 566
 Dijkstra E.W., 101, 174, 1158
 DIN, 442
 Direct Memory Access, 63, 407
 direct model, 1126
 Directed Acyclic Graph, 347
 DISCO, 673
 Discrete Cosine Transformation, 569, 1181
 Discretionary Access Control, 745
 Discretionary ACL, 1095
 disk farm, 600
 Disk Operating System, 46
 dispatcher, 872
 dispatcher objects, 994
 dispatcher_header, 998, 999
 Distributed Shared Memory, 660
 DLL, 281, 972, 973, 1020
 DLL hell, 1020
 DMA, 63, 407, 420
 transfer, 408
 Windows Vista, 993
 DNS, 693, 1166
 dnsquery, 759
 dokument 1003.1, 858
 dokument SVID, 857
 domain, 731
 Domain Name System, 693
 domena 0, 681
 domeny ochrony, 730, 731
 DoS, 720, 783
 DOS, 46
 doskonale tasowanie, 624
 dostawca usług internetowych, 688
 dostęp do plików, 325
 losowy, 325
 pliki specjalne, 916
 sekwencyjny, 325
 dostęp do sprzętu urządzenia, 423
 dostęp do zasobów w trybie klient-serwer, 1115
 dostęp do zdalnych komputerów, 684
 dostępność systemu, 720
 double indirect block, 392
 dowiązania, 347, 928
 kopie zapasowe, 371
 symboliczne, 336, 348, 1090
 twarde, 336, 1090
 tworzenie, 348
 down, 174
 DPC, 995
 drive-by download, 810
 Driver-Kernel Interface, 922
 DRM, 989, 1022, 1101
 dropper, 793
 drukowanie ciągu znaków, 417
 DMA, 420

drzewa czerwono-czarne, 911
 drzewo katalogów, 335
 drzewo procesów, 73
 DSL, 545
 DSM, 660
 dump, 369
 DuplicateHandle(), 1033
 duże projekty programistyczne, 113
 DV, 571
 DVD, 354, 449, 450, 556
 dwupoziomowe szeregowanie, 642, 643
 dwustronny dysk DVD, 451
 dwuwarstwowy dysk DVD, 451
 Dynamic Link Libraries, 281, 972, 1020
 dynamiczna relokacja, 233
 dynamiczna wymiana stron, 291
 dynamiczne ładowanie sterowników, 897
 dynamiczne skalowanie napięcia, 499
 dynamiczne szeregowanie operacji dyskowych, 608
 dyski, 84, 435, 510
 AV, 462
 dynamiczne, 1067
 elastyczne, 32
 magnetyczne, 58, 318, 435
 optyczne, 442
 dyski twarde, 58, 84
 algorytm szeregowania żądań dostępu, 456
 algorytm windy, 457
 błędne sektory, 461
 cylinder, 58
 czas przesyłania danych, 456
 czas wyszukiwania, 456
 defekty produkcyjne, 460
 FCFS, 456
 formatowanie, 452
 geometria fizyczna, 437
 geometria wirtualna, 437
 gęstość liniowa bitów, 459
 główny rekord startowy, 455
 IDE, 435
 kod ECC, 461
 logiczne adresowanie bloków, 437
 macierze RAID, 438
 obsługa błędów, 459, 461
 operacje seek zachodzące na siebie, 435
 opóźnienie związane z obrotem, 456
 pamięć podręczna kontrolera, 459
 parametry, 436
 partycje, 337
 podwójny przeplot, 455
 pojedynczy przeplot, 454
 pojemność, 454
 postępowanie z błędnymi blokami, 460
 przekos cylindrów, 453
 przekos głowic, 454
 RAID, 437
 recalibrate, 462
 SATA, 435
 seek, 435
 sektory, 452

SLED, 438
 SSF, 456, 457
 struktura, 58
 szeregowanie żądań dostępu, 456
 ścieżki, 58
 talerze, 58
 zarządzanie energią, 503
 dyskretna transformacja kosinusowa, 569
 dyspozytor, 140, 872
 dyspozytor plastrowy, 909
 dyspozytor płytowy, 909
 dyspozytor stron, 908
 dystrybucje systemu Linux, 866
 dziennik, 944
 dźwięk, 565
 dźwiękowa płyta CD, 442

E

Earliest Deadline First, 581
 early binding, 1167
 ECC, 402, 453, 463
 echo, 475
 Eckert J. Presper, 37
 e-commerce, 67
 e-cos, 71, 232
 EDF, 581, 611
 EEPROM, 57
 efekt drugiego systemu, 1192
 efekt lokalności, 1185
 efektywny identyfikator UID, 955
 EFS, 1093
 egzod, 110, 1159
 EIDE, 559, 1067
 ekonomiczność wykorzystania miejsca na dysku, 360
 ekran logowania, 775
 Electrically Erasable PROM, 57
 emacs, 474, 874
 Encryption File System, 1093
 end transaction, 353
 end-to-end argument, 1160
 energia, 499
 Engelbart Doug, 47, 116
 ENIAC, 37, 496, 499, 617, 753
 enter_region(), 169, 170, 178
 EnterCriticalSection(), 1034, 1035, 1186
 EOF, 478
 EPOC, 1108, 1109
 ERASE, 477
 errno, 86, 158
 Error-Correcting Code, 402
 ESC, 480
 escape character, 477
 escape sequences, 480
 Ethernet, 686
 klasyczny Ethernet, 686
 kolizja, 686
 most, 687
 przełączany Ethernet, 686
 przełączniki, 687
 przesyłanie pakietów, 686
 przesyłanie ruchu między sieciami, 687

przyłączanie komputera
 do sieci, 686
 unikanie problemu kolizji, 687
 event-driven paradigm, 1152
 events, 1034
 Excel, 801
 ExceptPortHandle(), 977
 exclusive locks, 930
 EXE, 972
 exec(), 91, 881, 1155, 1165
 execl(), 91, 881, 1155
 execlx(), 91, 881, 1155
 execution in-place, 1123
 execv(), 91, 881
 execve(), 89, 91, 98, 128, 880, 881
 exit(), 89, 880
 ExitFiber(), 1035
 ExitProcess(), 98, 99, 1035
 ExitThread(), 1035
 Exokernel, 1160
 explorer.exe, 1099
 ext2, 353, 926, 936, 937
 blok dwupośredni, 943
 blok jednopośredni, 942
 blok trójpśredni, 943
 bloki danych, 938
 dostęp do pliku, 939
 i-węzły, 937, 938
 katalogi, 939
 mapy bitowe, 938
 partycja dyskowa, 937
 pola struktury i-węzła, 941
 przeszukiwanie katalogów, 940
 struktura i-węzłów, 941
 superblok, 937
 śledzenie wolnych bloków
 i wolnych i-węzłów, 938
 tabela i-węzłów, 940
 tablica opisów otwartych
 plików, 942
 układ dysku, 938
 wpis katalogu, 939
 wstępna alokacja, 938
 wyszukiwanie pliku, 940
 ext3, 352, 936, 943
 dziennik, 944
 JBD, 944
 ksiega, 944
 obsługa atomowych operacji, 944
 rejestr zapisów dziennika, 944
 sekwencyjne zapisywanie zmian,
 944
 transakcje, 944
 extended file system, 936
 extent, 340

F

fala dźwiękowa, 565
 fala prostokątna, 467
 fałszywe współdzielenie, 663
 farma dysków, 600
 Fast Ethernet, 559
 FAT, 342, 387, 389
 FAT-12, 389
 FAT-16, 320, 389, 390, 1079, 1132
 FAT-32, 320, 389, 390, 391, 1079
 FCFS, 456
 fcntl(), 932
 fdatsync(), 915
 femto, 118
 FFS2, 1131
 FIFO, 260, 272, 275, 376, 892
 file, 937
 File Allocation Table, 342
 file handle, 947
 file mapping, 982
 file system filter drivers, 1005
 File Transfer Protocol, 684
 filmy cyfrowe, 555
 filmy wideo, 561
 filtrowanie operacji
 wejścia-wyjścia, 1004
 filtry, 869, 870, 871
 finger, 807
 firewall, 820
 bezzastanowy, 821
 osobisty, 823
 programowy, 821
 reguły, 822
 sprzętowy, 821
 zastanowy, 822
 system wykrywania włamań, 822
 FireWire, 65, 402
 firmware, 814
 First Berkeley Software
 Distribution, 856
 first fit, 239
 First-Come, First-Served, 201, 456
 First-In, First-Out, 260
 fizyczna geometria dysku, 437
 fizyczny sterownik urządzenia, 1127
 flagi, 868
 flash, 1131
 Flash RAM, 66
 Flash Translation Layer, 1131
 float, 111
 FlushFileBuffers, 377
 fly-by mode, 409
 FMS, 39, 40, 41
 foldery, *Patrz* katalogi
 for, 111, 870
 fork(), 89, 90, 98, 128, 148, 154, 877,
 879, 880, 881, 886, 888, 1165
 format string attacks, 781
 formatowanie dysków, 452
 główny rekord startowy, 455
 niskopoziomowe, 452
 pojedynczy przeplot, 454
 przekos cylindrów, 453
 przekos głowic, 454
 sektory, 452
 wydajność, 454
 wysokopoziomowe, 455
 FORTRAN, 39, 82, 1144, 1153, 1154
 Fortran Monitor System, 40
 fragmentacja, 339
 fragmentacja pamięci, 298
 framework UMDf, 1073
 Free Software Foundation, 862
 free(), 112, 1177

FreeBSD, 29, 31, 48, 69, 742, 852
 fsck, 372, 929
 FSCTL_QUERY_USN_JOURNAL, 1092
 FSF, 118
 fstat(), 93, 932
 fsync(), 915
 FTP, 684, 693, 822
 full I/O Path, 509
 funkcje jednokierunkowe, 727
 funkcje skrótu, 727
 MD5, 727
 SHA-1, 727
 funkcje sterujące VCR, 585
 funkcje uprzywilejowane, 30
 Furlong-Stone-Fortnight, 118

G

Gabor wavelet transformation, 771
 galezie, 985
 gang scheduling, 645
 garbage collection, 112
 Gates Bill, 46
 GB, 118
 gcc, 861, 872
 GDI, 492
 czcionki, 495
 DIB, 495
 kontekst urządzenia, 492
 mapy bitowe, 493
 metaplik Windows, 493
 region obcinania, 492
 rysowanie, 493
 tekst, 494
 GDT, 302
 GE-645, 43
 generacje systemów, 37
 czwarta generacja, 45
 druga generacja, 37
 pierwsza generacja, 37
 trzecia generacja, 40
 General Electric, 43, 853
 generic block layer, 922
 generic rights, 739
 generowanie list plików, 871
 generowanie przerwania, 62
 generowanie wyjścia, 479
 Get attributes, 328
 GetDC(), 492
 getegid(), 956
 geteuid(), 956
 GetFileAttributesEx(), 98, 99
 getgid(), 956
 GetLocalTime(), 98, 99
 getpid(), 878
 gets(), 778
 GetTokenInformation(), 1095
 getty, 898
 getuid(), 956
 GetWindowDC(), 492
 GID, 74, 446, 732, 735, 736, 953
 giga, 454
 Gigabit Ethernet, 472, 559
 gigantyczne ramki, 910
 glass master disk, 442

Global Descriptor Table, 302
 globalna tablica deskryptorów, 302
 globalne strategie alokacji pamięci, 273
 Globus Toolkit, 708
 główna tablica plików, 1081
 główny deskryptor woluminu, 382
 główny numer urządzenia, 428, 917
 główny rekord startowy, 337, 455, 799, 896
 główny system plików, 77
 GMT, 467
 gniazda, 918, 924, 1032, 1121
 adres, 919
 komunikacja sieciowa, 918, 919
 niezawodny połączeniowy
 strumień bajtów, 918
 niezawodny połączeniowy
 strumień pakietów, 918
 protokół, 919
 TCP, 919
 tworzenie, 918
 UDP, 919
 zawodna transmisja pakietów, 918
 GNOME, 34, 49, 81, 483, 862, 866
 GNU Network Object Model
 Environment, 866
 GNU Public License, 862
 GOAL, 24
 goat file, 823
 Google, 859
 goszczenie kodu, 1021
 GPL, 862
 Gradience Online Accelerated
 Learning, 24
 graficzny interfejs użytkownika, 29, 47, 486, 859
 grafika wektorowa, 493
 graf alokacji zasobów, 523
 graf skierowany, 523
 GRand Unified Bootloader, 896
 Graphical User Interface, 29, 47, 486, 859
 Graphics Device Interface, 492
 grep, 131, 864, 871, 873
 grid, 648
 Group Identification, 74
 GRUB, 896
 grupowanie plików, 75
 grupowanie procesów, 1025
 grupy, 74, 735, 953
 grupy procesów, 878
 gry komputerowe, 556
 GUI, 29, 30, 31, 47, 81, 482, 486, 511, 859, 866, 970, 973, 1030
 adapter graficzny, 487
 czcionki, 495
 Icons, 486
 ikony, 486
 karta graficzna, 487
 mapy bitowe, 493
 menu, 486
 okna, 486, 487
 oprogramowanie, 487
 Pointing device, 486
 rozdzielczość ekranu, 487
 rysowanie, 492

urządzenie wskazujące, 486
 video RAM, 487
 WIMP, 486
 Windows, 486
 wprowadzanie danych, 487
 gzip, 927

H

haker, 754
 w białym kapeluszu, 755
 w czarnym kapeluszu, 755
 HAL, 989, 1158
 HAL Development Kit, 992
 hal.dll, 1006
 Hamming distance, 771
 handheld, 228, 1108
 handheld computers, 69
 handle, 130, 976
 Hansen Brinch, 184, 185
 hard page fault, 1057
 Hardware Abstraction Layer, 989
 hash table, 256, 1183
 hasła, 31, 755, 829
 ataki, 762
 bezpieczeństwo, 761
 jednorazowe, 763
 Linux, 957
 łamanie, 757, 762
 metoda wyzwanie-odpowiedź, 765
 odgadywanie, 756
 słownik prawdopodobnych
 hasel, 762
 szyfrowanie, 761
 UNIX, 761
 HD DVD, 452, 556
 HDTV, 559
 head, 869, 871, 873, 878
 head skew, 454
 header files, 872
 Hewlett-Packard, 49
 hibernacja, 500, 1078
 hierarchia pamięci, 55, 227
 hierarchia procesów, 130
 hierarchiczny system plików, 1080
 High Sierra, 446
 hints, 1185
 hiperłącza, 693
 hipernadzorca, 669
 typ 1, 108, 671, 672
 typ 2, 109, 671, 673
 hipersześcián, 649
 hiperwatkowość, 53
 historia systemów operacyjnych, 36
 hives, 985
 holding register, 466
 honeypots, 835
 Honeywell, 43
 host, 545, 687
 hosting, 1021
 hosting WWW, 108
 HP PA, 252
 HP-UX, 852
 HTTP, 693, 1157
 hue, 564

I

I, 279, 571
 I/O Base, 61
 I/O completion ports, 1072
 I/O Limit, 61
 I/O Request Packet, 1016, 1071
 I/O scheduler, 874, 923
 IA64, 971
 IAT, 1020
 IBM, 40
 IBM 1401, 38
 IBM 360, 83
 IBM 7090, 83
 IBM 7094, 38, 43, 83, 206
 IBM AS/400, 737
 IBM PC, 46
 IBM PC/AT, 46
 IBM/370, 671
 IBSYS, 40
 IDE, 60, 64, 402, 435
 idealny procesor, 1041
 idempotentna operacja, 353
 identyfikacja głosu, 771
 identyfikatory
 GID, 74, 736, 953
 PID, 90, 877, 892, 1169
 SID, 1095
 SID poziomu integralności, 1099
 TID, 892
 UID, 74, 736, 953
 wątek, 148
 IDL, 700
 IDS, 822, 833
 IEEE, 44, 402
 IEEE 1394, 65, 402, 1067
 IEEE 802.3, 686
 IEEE Computer Society, 116
 IEEE Standards Board, 857
 if, 111, 870
 IIOP, 701
 ikony, 486
 immediate files, 1082
 impersonation, 1096
 implementacja
 katalogi, 344
 klasyczna segmentacja, 298
 pliki, 338
 procesy, 133
 RPC, 659
 system NFS, 949
 system operacyjny, 1158
 system plików, 337
 system plików Linuksa, 936
 system plików NTFS, 1081
 wątki w jądrze, 153
 wątki w przestrzeni
 użytkownika, 150

- wejście-wyjście w systemie
 Linux, 921
 z dołu do góry, 1170
 z góry na dół, 1170
 Import Address Table, 1020
 IN, 61, 403, 404
 Industry Standard Architecture, 63
 informatyka, 115
 infrastruktura klucza publicznego, 729
 init, 130, 897, 913
 InitializeAcl(), 1098
 InitializeSecurityDescriptor(), 1098
 i-node, 936, 937
 insert right, 743
 inside jobs, 772
 instrukcje uprzywilejowane, 671
 integer, 111
 integralność danych, 720
 Integrated Drive Electronics, 60, 435
 Integrity, 103
 integrity checking, 828
 integrity level, 1097
 integrity level SIDs, 1099
 Intel, 53, 629
 Intel 386, 967
 Intel 8088, 234
 Intel Core 2, 672
 Intel Itanium, 971
 inteligentne szeregowanie, 642
 inteligentny telefon, 1107
 interakcja z urządzeniami
 sieciowymi, 924
 interakcja z urządzeniami
 znakowymi, 924
 Interdata, 855
 Interdata 8/32, 945
 Interface Definition Language, 700
 interfejs pamięci wirtualnej, 284
 interfejs sterownika urządzenia, 427
 interfejs sterownika zarządzania
 energiją, 507
 interfejs sterownik-jądro, 922
 interfejs urządzenia graficznego, 492
 interfejs użytkownika, 473
 interfejs VFS, 354
 interfejs Win32 API, 97, 973
 interfejs wywołań systemowych, 1155
 Interix, 974
 interlacing, 563
 International Organization for
 Standardization, 442
 internet, 115, 545, 618, 687, 718
 dostawca usług, 688
 host, 687
 ISP, 688
 router, 688
 sieć szkieletowa, 688
 Internet Explorer, 776, 1099
 Internet Protocol, 691, 919
 Internet Service Providers, 688
 interpreter poleceń, 73
 InterProcess Communication, 161
 Interrupt Service Routines, 995
 INTR, 478
 Intracoded, 571
 Intrinsics, 483
 intruders, 721
 Intrusion Detection System, 822, 833
 intruzi, 721
 i-numer, 94
 invalid page, 1047
 inwersja priorytetów, 171
 IoCallDriver, 1073, 1074
 IoCompleteRequest, 1089, 1090
 ioctl(), 920
 IopParseDevice(), 1016, 1017
 IP, 691, 874, 919
 IPC, 161, 978
 IPv4, 691
 IPv6, 691
 IRET, 421
 IRP, 1016, 1071, 1074
 IRQ, 65
 IS 10149, 442
 ISA, 63
 ISO, 858
 ISO 11172, 571
 ISO 13818, 571
 ISO 9660, 382
 ciągłość plików, 385
 extents, 385
 flagi, 384
 główny deskryptor woluminu, 382
 identyfikator mechanizmu
 przygotowującego dane, 382
 identyfikator systemu, 382
 identyfikator woluminu, 382
 identyfikator wydawcy, 382
 informacje końcowe ścieżki, 382
 informacje początkowe ścieżki, 382
 Joliet, 386
 katalog główny, 383
 leadin, 382
 leadout, 382
 nadmiarowe kodowanie, 383
 ograniczenie głębokości
 zagnieżdżania katalogów, 385
 partycje, 382
 pliki, 384
 pola binarne, 383
 pomocniczy deskryptor
 woluminu, 383
 poziomy, 385
 Rock Ridge, 385
 rozmiar nazwy pliku, 384
 woluminy logiczne, 382
 wpisy w katalogu, 383, 384
 ISP, 688
 ISR, 995
 i-węzły, 95, 343
 izolowanie, 837
 izolowanie błędów, 1146
 izolowanie kodu mobilnego, 835, 837
 interpretacja, 840
 monitor odwołań, 839
 piaskownica, 837
 izolowanie programów, 741
- J**
- jailing, 832
 jakość usług, 560, 689
 jasność, 564
 Java, 71, 185
 aplety, 837
 bezpieczeństwo, 841
 dostęp do plików, 843
 dostęp sieciowy, 843
 kod bajtowy, 841
 maszyna wirtualna, 110
 obsługa wywołań
 systemowych, 842
 problem producent –
 konsument, 186
 programy, 841
 reguły ochrony, 843
 synchronized, 185
 Java Development Kit, 842
 Java Virtual Machine, 71, 110, 706
 JavaScript, 840
 JavaSpace, 706
 jądro obiektowe, 1112
 jądro systemu Linux, 861, 872
 dyspozytor, 872
 komponent wejścia-wyjścia, 874
 koordynator wejścia-wyjścia, 874
 mechanizm szeregujący, 875
 operacje wejścia-wyjścia, 874
 oprogramowanie sieciowe, 874
 procesy, 884
 sterowniki urządzeń, 874
 struktura jądra, 873
 system plików, 874
 task_struct, 884
 urządzenia blokowe, 874
 urządzenia sieciowe, 874
 urządzenia znakowe, 874
 wirtualny system plików, 874
 zadania, 884
 zarządzanie pamięcią, 875
 zarządzanie procesami, 875
 jądro systemu NT, 975
 JBD, 944
 JDK 1.0, 842
 jednokierunkowy łańcuch skrótu, 763
 jednolite nazewnictwo urządzeń, 415
 jednolity interfejs sterowników
 urządzenia, 427
 jednopoziomowy system katalogów, 331
 jednostka zarządzania pamięcią, 243
 jednostki miar, 118
 język
 B, 854
 BASIC, 966
 BCPL, 854
 C, 104, 111, 854, 1144
 C#, 971
 FORTRAN, 39, 1153
 IDL, 700
 Java, 71
 maszynowy, 32
 Pidgin Pascal, 184
 PL/I, 43, 1144
 PostScript, 1182

JFFS, 1131
 jiffy, 893
 Jini, 705
 klient, 706
 komunikacja, 706
 notify, 707
 protokół wykrywania, 705
 read, 706, 707
 synchronizacja, 706
 take, 707
 urządzenia, 705
 usługa wyszukiwania, 705
 usługi, 706
 write, 706, 707
 wyłączanie urządzenia, 706
 zdobywanie dzierżawy, 706
 jitter, 560, 591
 JMP, 234, 838
 jobs, 1025
 Jobs Steve, 47
 Johnson Steve, 855
 Joint Photographic Experts Group, 568
 Joliet, 386
 journal, 944
 Journaling Block Device, 944
 journaling file systems, 352, 943
 Journaling Flash File System, 1131
 JPEG, 568, 611
 kolejność transmisji
 kwantyzowanych
 wartości, 570
 kwantyzacja, 569
 przekształcenia DCT, 569
 skanowanie bloku, 570
 just-in-time, 1048
 JVM, 71, 110, 706, 840, 841
 JVM byte code, 841

K

K Desktop Environment, 866
 K42, 103
 kabel Ethernet, 686
 kanał uboczny, 769
 karta graficzna, 487
 pamięć, 487
 rozdzielczość ekranu, 487
 karta z zakodowaną wartością, 766
 karty chipowe, 71, 766
 karty elektroniczne, 71
 karty inteligentne, 766, 767
 kanał uboczny, 769
 schematy uwierzytelniania, 768
 zalety, 767
 karty pamięci, 1130
 karty rozszerzeń, 402
 karty z paskiem magnetycznym, 766
 katalog spoolera, 162
 katalog stron, 305
 katalog stron samoodzworowania, 1037
 katalogi, 75, 89, 94, 98, 323, 331, 927
 /dev, 916
 bieżący katalog, 333
 dowiązania, 94

dowiązania twarde, 336
 drzewo, 335
 ext2, 939
 główny katalog, 76, 331
 hierarchiczne systemy, 332
 implementacja, 344
 i-numer, 94
 jednopoziomowe systemy, 331
 lista plików, 871
 MS-DOS, 388
 nazwy ścieżek, 333
 operacje, 335
 roboczy katalog, 77, 333, 927
 spooler, 433
 ścieżki bezwzględne, 333
 tworzenie, 94, 335, 871
 UNIX, 391
 unlink(), 336
 usuwanie, 94, 335, 871
 wywołania systemowe, 94
 katalogowy system
 wieloprocesorowy, 626
 kB, 118
 KDE, 34, 49, 81, 483, 862, 866
 Kerckoffs Auguste, 724
 kernel32.dll, 994, 1036
 Kernel-Mode Driver Framework, 1073
 Kernighan Brian, 853
 keylogger, 787
 keys, 724
 Kildall Gary, 45, 966
 KILL, 477
 kill(), 89, 96, 130, 880, 883
 kilo, 118
 klasy użytkowników, 718
 klasyczny Ethernet, 686
 klasyczny model wątków, 143
 klawiatura, 473, 474
 blokowanie przewijania
 zawartości ekranu, 478
 echo, 475
 EOF, 478
 ERASE, 477
 INTR, 478
 KILL, 477
 kod skanowania, 474
 POSIX, 475
 QUIT, 478
 SIGINT, 478
 SIGQUIT, 478
 START, 478
 STOP, 478
 tabulacja, 476
 tryb kanoniczny, 475
 tryb niekanoniczny, 475
 tryb surowy, 475
 tryb ugotowany, 475
 znak ucieczki, 477
 znak wysuwu wiersza, 476
 znaki obsługiwane specjalnie
 w trybie kanonicznym, 477
 klient, 105
 klient X, 481
 klient-serwer, 105, 693, 947
 klipy wideo, 555
 klucz, 724

publiczny, 726
 tajny, 725
 kmalloc, 910
 KMDF, 1073
 kod algorytmiczny, 1153
 kod ASCII, 111, 474
 kod bajtowy wirtualnej maszyny
 Javy, 841
 kod ECC, 453, 463
 kod Hamminga, 440
 kod korekcji błędów, 402
 kod mobilny, 836
 kod niezależny od pozycji, 283
 kod PIN, 766
 kod powłoki, 782
 kod programu, 92, 115
 kod skanowania, 474
 kod zdarzeniowy, 1153
 Kodak PhotoCD, 448
 kodowanie, 567, 610
 audio, 565
 JPEG, 568
 kształt przebiegu, 574
 percepcyjne, 574
 RLE, 570
 video, 562
 koherencja pamięci podręcznych, 622
 kojarzenie nazw, 1167
 kolejka komunikatów, 488
 kolizja, 686
 kolor, 564
 kombinacje dwuliterowe, 725
 kompaktowanie pamięci, 236
 kompensacja ruchu, 568
 kompilacja, 114
 kompilacja warunkowa, 1172
 kompilator gcc, 861
 kompilator języka C, 855
 komponenty komputera osobistego, 50
 kompresja, 586
 kompresja audio, 574
 kodowanie kształtu przebiegu, 574
 kodowanie percepcyjne, 574
 maskowanie czasowe, 574
 maskowanie częstotliwości, 574
 MP3, 574
 próbkiwanie częstotliwości, 576
 próg słyszalności, 576
 psychoakustyka, 574
 RMS, 575
 szybkość przesyłania bitów, 577
 transformata Fouriera, 574, 576
 kompresja międzyramkowa, 568
 kompresja plików, 1090
 kompresja wideo, 567
 algorytm, 567
 dekodowanie, 567
 JPEG, 568
 kodowanie, 567
 kompensacja ruchu, 568
 kompresja międzyramkowa, 568
 kwantyzacja, 569
 MPEG, 568, 571
 MPEG-1, 571
 MPEG-2, 571

- komputer, 36
 - Apple, 47
 - Apple Macintosh, 47
 - ENIAC, 617
 - IBM PC, 46
 - Interdata, 855
 - klastrowy, 647
 - Lisa, 47
 - mainframe, 67
 - NUMA, 626
 - ogólnego przeznaczenia
 - z podziałem czasu, 43
 - osobisty, 45, 50, 68, 966
 - PC, 499
 - PDP, 44
 - PDP-11, 853, 855
 - podręczny, 69, 1108
 - równoległy, 618
 - VAX, 856
 - Z3, 37
 - zasilanie bateryjne, 1197
- komunikacja bezprzewodowa, 505
- komunikacja międzyprocesowa, 73, 161, 978
 - ALPC, 1032
 - bariery, 191
 - bit oczekiwania na sygnał
 - wakeup, 173
 - gniazda, 1032
 - monitory, 182, 183
 - muteksy, 177
 - problem czytelników i pisarzy, 218
 - problem inwersji priorytetów, 171
 - problem pięciu filozofów, 214
 - problem producent – konsument, 172
 - przekazywanie komunikatów, 188
 - regiony krytyczne, 163
 - RPC, 1032
 - sekcja krytyczna, 164
 - semaforey, 174
 - skrytki pocztowe, 1031
 - sleep(), 171
 - Symbian OS, 1121
 - synchronizacja, 176
 - wakeup(), 171
 - Windows Vista, 1031
 - wspólna pamięć, 162
 - wyścig, 162, 163
 - wzajemne wykluczanie, 164
 - wzajemne wykluczanie
 - z wykorzystaniem aktywnego oczekiwania, 165
 - zakleszczenie, 182
 - zmienne warunkowe, 179
- komunikacja pomiędzy komponentami elektronicznymi, 618
- komunikacja sieciowa, 917, 1195
- komunikacyjne zakleszczenie, 544
- komunikaty SMS, 1117
- konflikty pomiędzy wątkami, 159
- konsola odzyskiwania, 1007
- konsolidacja, 114
- kontekst urządzenia, 492
- kontrola konta użytkownika, 1100
- kontroler DMA, 407, 420
- kontroler dysku, 60
- kontroler RAID, 438
- kontrolery urządzeń, 402
 - ECC, 402
 - kod korekcyj błędów, 402
 - monitor, 402
 - preambuła, 402
- kontrolki ActiveX, 811, 832, 1021
- koń trojański, 790
- infekcja, 791
- kończenie działania procesów, 129
- koordynator wejścia-wyjścia, 874, 923
- kopie woluminów sporządzane w tle, 1067
- kopie zapasowe, 365, 830
 - aktywny system plików, 367
 - algorytm wykonywania kopii zapasowej, 370
 - dowiązania, 371
 - dump, 369
 - kosz, 366
 - logiczne kopie, 368
 - luki, 371
 - mapy bitowe, 370
 - odtworzenie danych, 366
 - pliki specjalne, 371
 - pojemność dysków, 371
 - potoki, 371
 - przyrostowe kopie, 366
 - restore, 369
 - taśmy, 365
 - wykonywanie, 366
 - zrzucanie niezmodyfikowanych katalogów, 369
 - zrzucanie uszkodzonych bloków, 368
 - zrzut fizyczny, 367
- kopiowanie plików, 329, 871
- kopiowanie przy zapisie, 280, 656, 887, 1050
- kostka, 629
- kosz, 366
- kradzież tożsamości, 788
- kraker, 755
- krotka, 702
- kryptografia, 723
 - algorytm deszyfrujący, 724
 - algorytmy, 724
 - bezpieczeństwo przez ukrywanie, 724
 - blok podpisu, 728
 - certyfikaty, 729
 - funkcje jednokierunkowe, 727
 - klucz, 724
 - klucz publiczny, 831
 - klucz tajny, 725
 - kombinacje dwuliterowe, 725
 - kryptograficzna funkcja skrótu, 727
 - moduł TPM, 729
 - podpisy cyfrowe, 727
 - podstawianie
 - monoalfabetyczne, 725
 - RSA, 726
 - SHA-1, 727
 - SHA-256, 727
 - SHA-512, 727
 - szyfrogram, 724
 - tekst jawny, 724
 - tekst zaszyfrowany, 724
 - wyznaczanie bloku podpisu, 728
 - z kluczem publicznym, 726
 - z kluczem symetrycznym, 725
 - z kluczem tajnym, 725
 - zasada Kerckhoffs, 724
- kryptograficzna funkcja skrótu, 727
- kryptogram, 723
- ksh, 80, 867
- księga, 944
- księgowanie, 1092
- księgujące systemy plików, 352, 943
- kswapd, 913
- kwant, 204, 894
- kwantyzacja, 569
- kwantyzacja próbek do 4 bitów, 566

L

- L1, 56
- L2, 56
- L4, 103
- lampka złych wiadomości, 1190
- Lamport Leslie, 763
- lampy elektronowe, 37
- LAN, 545, 685, 718, 748, 946
- lands, 443
- Large Scale Integration, 45
- LaserVision, 442
- late binding, 1167
- LBL, 760
- LCP, 1003
- ld, 281
- LDD, 1127
- LDT, 302
- leadin, 382
- leadout, 382
- Least Recently Used, 262, 1055
- leave_region(), 169, 171
- LeaveCriticalSection(), 1034, 1035
- legacy devices, 66
- lekke procesy, 144
- LFS, 350
 - wątek sprzątacza, 351
- libc, 782, 814
- licencja publiczna GNU, 862
- licencje, 682
- liczba magiczna, 323
- liczby zmiennooprzecinkowe, 111
- licznik programu, 51, 125, 131, 143
- liczniki dozorujące, 468, 470
- LIFO, 1009
- LILO, 896
- limity czasu, 545
- limity dyskowe, 349, 364
- Linda, 702
 - blokowanie się procesów, 703
 - in, 702
 - krotka, 702
 - operacje na krotkach, 702
 - out, 702
 - przestrzeń krotek, 702
 - publikuj-subskrybuj, 704
 - szablony, 703

- line disciplines, 924
- line printer, 78
- linie pamięci podręcznej, 55
- liniowa gęstość bitów, 459
- LINK, 348
- link(), 89, 94, 336, 935
- linker, 114
- links, 928
- Linus Elevator, 923
- Linux, 29, 32, 45, 48, 69, 742, 851, 852, 860, 863, 958
 - /etc/rc, 947
 - /proc, 945
 - algorytm szeregujący, 894
 - automatyczne montowanie katalogów, 948
 - bezpieczeństwo, 953
 - blokada współdzielone, 930
 - blokada wyłączne, 930
 - blokowanie, 930
 - cele, 863
 - chwila, 893
 - cron, 876
 - demon stron, 897, 912
 - demony, 876
 - deskryptor pliku, 932
 - dostęp do plików specjalnych, 916
 - dostęp do urządzeń
 - wejścia-wyjścia, 959
 - dowiązania, 928
 - dyspozytor stron, 908
 - dyskrybucje, 866
 - efektywny identyfikator UID, 955
 - ext2, 926, 937
 - ext3, 943
 - filtry, 870
 - getty, 898
 - GID, 953
 - gigantyczne ramki, 910
 - główny numer urządzenia, 917
 - GNOME, 866
 - graficzny interfejs użytkownika, 862, 866
 - grupy, 953
 - GUI, 866
 - hasła, 957
 - implementacja
 - bezpieczeństwa, 957
 - implementacja procesów, 884
 - implementacja systemu plików, 936
 - implementacja wejścia-wyjścia, 921
 - implementacja zarządzania pamięcią, 904
 - init, 897
 - interakcja z urządzeniami sieciowymi, 924
 - interakcja z urządzeniami znakowymi, 924
 - interfejs sterownik-jądro, 922
 - interfejsy, 865
 - jądro, 861, 872
 - katalogi, 927, 955
 - KDE, 866
 - komunikacja sieciowa, 917
 - koordynator wejścia-wyjścia, 923
 - licencja publiczna GNU, 862
 - login, 898, 957
 - logowanie, 957
 - ładowne moduły, 925
 - ładowanie sterowników, 897
 - mechanizm bezpieczeństwa, 953
 - mechanizm przydzielania pamięci, 908
 - model biznesowy, 862
 - model pamięci, 899, 958
 - model wątków, 891
 - model wejścia-wyjścia, 922
 - moduły, 925
 - montowanie dysku, 929
 - napędy, 929
 - NFS, 945
 - numery wersji jądra, 861
 - obsługa sieci, 917
 - ochrona zasobów, 959
 - ogólna warstwa blokowa, 922
 - określanie nazw plików, 927
 - operacje na plikach, 921
 - operacje wejścia-wyjścia, 921
 - pamięć podręczna, 922, 923
 - pamięć podręczna stron, 923
 - pamięć wirtualna, 861
 - pliki, 926
 - pliki specjalne, 916, 928
 - polecenia, 870
 - pomocniczy numer urządzenia, 917
 - POSIX, 865
 - powłoka, 866, 867
 - procesor poleceń, 866
 - procesy, 876, 884, 958
 - programy użytkowe, 870
 - protokoły obsługi, 924
 - przydzielanie pamięci, 909
 - read, 865
 - reprezentacja pamięci głównej, 906
 - runqueue, 893
 - SETGID, 956
 - specjalne pliki blokowe, 916
 - specjalne pliki znakowe, 917
 - społeczność użytkowników, 953
 - sterowniki urządzeń, 921
 - strefy pamięci, 905
 - stronicowanie, 907, 911
 - superużytkownik, 955
 - surowe pliki blokowe, 923
 - surowe urządzenia blokowe, 929
 - sygnały, 878
 - system plików, 925, 959
 - system wejścia-wyjścia, 916
 - szeregowanie, 892
 - szeregowanie cykliczne, 892
 - szeregowanie z uwzględnieniem powinowactwa, 895
 - tablica priorytetów, 893
 - tablice stron, 908
 - tryb ochrony pliku, 954
 - tworzenie procesu, 886
 - UID, 953
 - uprawnienia, 953
 - uruchamianie systemu, 896
 - urządzenia sieciowe, 924
 - urządzenia znakowe, 921
 - usuwanie usterek z systemów plików, 929
 - użytkownicy, 953
 - warstwy systemu, 865
 - wątki, 884, 887
 - wątki czasu rzeczywistego, 892
 - wątki jądra, 889
 - wątki z podziałem czasu, 892
 - wejście-wyjście, 916
 - wersja 1.0, 861
 - wersja 2.0, 861
 - wersje, 860
 - wielka blokada jądra, 895
 - winda Linusa, 923
 - wirtualna przestrzeń adresowa, 910
 - wirtualna przestrzeń adresowa procesu, 900
 - wirtualny system plików, 936
 - wywołania systemowe, 865
 - wywołania systemowe wejścia-wyjścia, 919
 - X Window System, 862, 866
 - zadania, 884
 - zarządzanie fragmentem pamięci, 908
 - zarządzanie pamięcią, 899
 - zarządzanie pamięcią fizyczną, 905
 - zarządzanie procesami, 879
 - zmiennie synchronizujące, 895
 - Lions John, 854
 - Lisa, 47
 - lista ACL, 734
 - lista DACL, 1095
 - lista jednokierunkowa, 238
 - lista kontroli dostępu, 983
 - lista uprawnień, 737
 - ochrona, 737
 - lista wolnych bloków, 362
 - listen(), 919
 - listy kontroli dostępu, 733
 - append message, 734
 - copy object, 734
 - destroy object, 734
 - GID, 736
 - grupy, 735
 - obiekty, 734
 - podmioty, 734
 - role, 735
 - sort alphabetically, 734
 - sprawcy, 734
 - symbole wieloznaczne, 735
 - UID, 736
 - little-endian, 383, 991
 - livelock, 546
 - LOAD, 619, 626
 - loadable modules, 925
 - Local Area Networks, 685
 - Local Descriptor Table, 302
 - Local Procedure Call, 975
 - locking, 930
 - logic bomb, 773
 - Logical Device Driver, 1127
 - logiczne adresowanie bloków, 437
 - logiczne kopie zapasowe, 368
 - logiczne rozmieszczenie sterowników urządzeń, 424

logiczny sterownik urządzenia, 1127
 login, 898, 957
 login spoofing, 774
 logowanie, 755, 957, 1099
 Log-structured File System, 350
 lokalna tablica deskryptorów, 302
 lokalne strategie alokacji pamięci, 273
 lokalne wywołania procedur, 975
 lokalność odwołań, 265
 lokalny algorytm zastępowania
 stron, 273
 long, 111
 LookupAccountSid(), 1098
 LPC, 975
 lpfnWndProc, 490
 LRU, 262, 272, 275, 376, 1055
 symulacja programowa, 263
 ls, 864, 868, 871, 873, 888
 lsass, 975, 1099
 lseek(), 89, 93, 889, 932, 933
 LSI, 45
 luki, 371
 luminancja, 564
 luźno związane systemy, 620

L

ładowalne moduły, 925
 ładowanie bibliotek DLL, 1123
 ładowanie sterownika, 60, 897
 łamanie hasel, 757, 762
 łańcuchy formatujące, 780
 łączy
 ADSL, 558
 DSL, 545
 Łukaszewicz Jan, 490

M

M.I.T., 853
 MAC, 745
 Mac OS, 69
 Mac OS X, 29
 Mach, 292, 859
 macierz bieżącej alokacji, 529
 macierz ochrony, 733, 743, 744
 macierz zadań, 529
 macierze dyskowe, 509
 macierze RAID, 371, 438
 pliki multimedialne, 600
 poziomy, 438
 macro virus, 801
 magazyn stron, 290
 magic number, 323
 magistrała, 63
 FireWire, 65
 IBM PC, 63
 IDE, 64
 IEEE 1394, 65
 ISA, 63
 lokalna, 64
 PCI, 63
 PCI Express, 64
 SCSI, 64, 65
 USB, 64, 65
 magistrała pamięci, 406
 magistrale niskoczęstotliwościowe, 499
 mailslots, 1031
 main(), 92, 330
 mainframe, 38, 67, 83
 major device number, 917
 make, 113, 324, 872, 873
 Makefile, 113
 makra, 112, 801, 1181
 makrobloki, 572
 maksymalizacja czasu bezczynności
 dysku, 509
 maksymalna liczba otwartych
 plików, 547
 malloc(), 92, 112, 160, 784, 901, 903,
 1009, 1167, 1177
 malware, 786
 Mandatory Access Control, 745
 mandatory access restrictions, 1071
 mapped page writer, 1062
 MapViewOfFile(), 1051
 mapy bitowe, 238, 362, 493, 494
 Mark I, 37
 marshaling, 658
 maskowanie czasowe, 574
 maskowanie częstotliwości, 574
 maskowanie dźwięków, 574
 master, 632
 Master Boot Record, 337, 799, 896
 Master File Table, 1081
 master-slave, 631
 maszyna wirtualna, 106, 108, 670,
 671, 709, 710
 CMS, 107
 hipernadzorca typu 1, 108
 hipernadzorca typu 2, 109
 Java, 110
 JVM, 110
 monitor, 107
 parawirtualizacja, 110
 procesor wielordzeniowy, 681
 rootkit, 814
 system operacyjny gospodarza, 109
 system operacyjny gościa, 109
 VM/370, 107
 VMware Workstation, 109
 Mauchley William, 37
 MBR, 337, 799, 827, 896
 MD5, 727, 728, 831
 MDL, 1075
 mechanizm aktywacji zarządcy, 155
 mechanizm FCFS, 456
 mechanizm obsługi liczników
 dozorujących, 471
 mechanizm ochrony, 720, 730
 mechanizm pamięci trwalej, 318
 mechanizm przydzielania pamięci, 908
 mechanizm publikuj-subskrybuj, 704
 mechanizm stronicowania na
 zadanie, 837
 mechanizm VT, 672
 mechanizm wywołań systemowych, 86
 mem_map, 905, 906
 memory compaction, 236
 Memory Descriptor Lists, 1075

Memory Management Unit, 59, 243, 799
 memory-mapped files, 902
 memory-resident viruses, 798
 MEMS, 509
 menedżer konfiguracji, 1003
 menedżer nakładek, 242
 menedżer obiektów, 978, 1000, 1007
 katalogi przestrzeni nazw
 obiektów, 1014
 obiekty plików reprezentujące
 egzemplarze otwartych
 plików, 1014
 obiekty urządzeń, 1015
 otwieranie pliku, 1016
 procesy, 1017
 tworzenie pliku, 1016
 wątki, 1017
 weryfikator aplikacji, 1015
 zarządzane obiekty warstwy
 wykonawczej, 1018
 zarządzanie czasem życia
 obiektów, 1014
 menedżer okien, 483
 menedżer pamięci, 228, 1002
 menedżer pamięci podręcznej, 1002
 menedżer pamięci podręcznej
 systemu Windows, 1064
 menedżer procesów, 1002
 menedżer równoważenia zbiorów, 1060
 menedżer transakcji jądra z obsługą
 transakcji koordynowanych, 986
 menedżer wejścia-wyjścia, 1001, 1067
 menedżer zadań, 1022
 menedżer zasilania, 1077
 menedżer zasobów, 34
 menu, 486
 mesh, 648
 Message Type Modules, 1139
 Message-Passing Interface, 191
 message-passing multicomputer, 619
 metaplik Windows, 493
 metody, 490, 700
 metody izolowania programów, 741
 MFT, 1081, 1082
 MICA, 968
 Micro-Electrical-Mechanical
 Systems, 509
 Microsoft, 46, 966
 Microsoft Disk Operating System,
 46, 967
 Microsoft Office, 486, 776
 middleware, 685
 miękki system czasu rzeczywistego,
 71, 211
 miękkie błędy braku stron, 1047, 1057
 miękkie chybiecie, 253
 migracja maszyn wirtualnych, 670
 miki, 479
 mikro, 118
 mikrofon, 565
 mikrojądro, 102, 859, 1113
 mikrokomputer, 45
 mili, 118
 Mills Harlan, 1189
 mimicry attack, 835
 minimalizacja czasu odpowiedzi, 200

- minimalizacja ruchu ramienia dysku, 379
- miniporty, 1005
- MINIX, 45, 858, 1150
 - graficzny interfejs użytkownika, 859
 - GUI, 859
 - jądro, 859
 - menedżer pamięci, 859
 - rozwój systemu, 859
 - wydajność, 859
 - zarządzanie pamięcią, 859
- MINIX 1, 926
- MINIX 3, 45, 103, 104, 742, 860
 - klient-serwer, 105
 - mikrojądro, 104
 - struktura procesów, 104
- minor device number, 917
- MIPS, 252
- mkdir, 871, 873
- mkdir(), 89, 94, 935
- mkfs, 929
- mm_struct, 911
- mmap(), 903, 904
- MMC, 1130
- MMU, 59, 243, 799
- mobile code, 836
- model Bella-La Paduli, 745
 - prosta reguła bezpieczeństwa, 746
 - reguła *, 746
- model bezpośredni, 1126
- model Biby, 747
 - prosta reguła integralności, 748
 - reguła integralności *, 748
- model emulatora, 1126
- model fazy działania, 114
- model formalny bezpiecznych systemów, 743
 - macierz ochrony, 744
 - operacje na macierzy ochrony, 743
 - polecenia ochrony, 743
 - stan autoryzowany, 744
 - stan nieautoryzowany, 744
- model goszczenia, 1021
- model klient-serwer, 105
- model Lampsona, 748
- model potoku, 413
- model procesów, 124
- model procesów sekwencyjnych, 142
- model programowania COM, 1021
- model ruchomy, 1126
- model SMP, 632
- model struktury systemu monolitycznego, 101
- model wątków, 143
 - Linux, 891
- model WDM, 1072
- model wgrywanie-ściąganie, 695
- model wielostronny, 1126
- model zbioru roboczego, 266
- model zdalnego dostępu, 695
- modelowanie wieloprogramowości, 135
- modelowanie zakleszczeń, 523
- ModifiedPageWriter, 1060
- modulacja PCM, 566
- modulowanie użycia procesora, 750
- moduł MMU, 59
- moduł TPM, 729
 - BitLocker, 730
 - NGSCB, 730
 - Palladium, 730
 - wykorzystanie, 730
- moduł zarządzania pamięcią, 59
- moduły, 925
- moduły CSY, 1139
- moduły MTM, 1139
- moduły PRT, 1139
- moduły TSY, 1139
- monitor, 402, 473, 475
 - CRT, 402
- monitor kontroli bezpieczeństwa odwołań, 1003
- monitor maszyn wirtualnych, 107, 671, 1193
- monitor odwołań, 742, 743, 839
- monitor użycia klawiatury, 787
- monitory, 182, 183
 - problem producent – konsument, 184
- monoalphabetic substitution, 725
- montowanie
 - dysk, 929
 - pendrive USB, 416
 - system plików, 77
- Moore Gordon, 53, 628
- Morris Robert Tappan, 806
- most, 687
- motherboard, 66
- Motif, 483
- Motion Picture Experts Group, 571
- mount(), 89, 95
- MOV, 230, 231
- moving model, 1126
- MP3, 559, 574
 - kodowanie percepcyjne, 574
 - szybkość przesyłania bitów, 577
 - transformata Fouriera, 576
- MPEG, 568, 571, 586, 611
 - dynamiczna akcja, 573
 - gwałtowne cięcia, 573
 - JPEG, 573
 - macierz różnic, 573
 - ramki, 571
 - redundancja chwilowa, 571
- MPEG audio warstwa 3, 574
- MPEG-1, 571
- MPEG-2, 559, 560, 571
 - makrobloki, 572
 - ramki B, 571
 - ramki I, 571, 573
 - ramki P, 571, 572
- MPI, 191
- MS-DOS, 46, 387, 965, 966
 - adresy dyskowe, 390
 - atrybuty, 388
 - bit ukrycia, 388
 - bloki dyskowe, 389
 - FAT, 389
 - FAT-12, 389
 - FAT-16, 390
 - FAT-32, 390, 391
 - katalogi, 388
 - maksymalny rozmiar partycji, 390
 - odczyt pliku, 388
 - partycje, 390
 - pliki tylko do odczytu, 388
 - rozmiar partycji, 390
 - rozmiar tablicy FAT, 389
 - śledzenie wolnych bloków, 391
 - tablica FAT, 389
 - wpis w katalogu, 388
- MS-DOS 1.0, 966, 967
- MS-DOS 2.0, 966
- MS-DOS 3.0, 966
- MS-DOS 5.0, 966
- MS-DOS 7.0, 966
- MS-DOS 8.0, 966
- MTM, 1139
- MULTICS, 43, 44, 84, 102, 298, 299, 308, 853
 - adres, 299
 - adres wirtualny, 301
 - bufor TLB, 301
 - konwersja dwuczłonowego adresu na adres w pamięci głównej, 301
 - pamięć wirtualna, 300
 - segmentacja ze stronicowaniem, 298
- multilevel security system, 746
- multimedia, 555, 556, 558, 1196
 - algorytm sterowania przyjmowaniem zgłoszeń, 560
 - badania, 610
 - buforowanie, 603
 - buforowanie bloków, 603
 - buforowanie plików, 605
 - dostarczanie danych w czasie rzeczywistym, 559
 - filmy wideo, 556
 - jakość usług, 560
 - jitter, 560
 - kodowanie audio, 565
 - kodowanie wideo, 562
 - kompresja audio, 574
 - kompresja wideo, 567
 - MPEG-2, 560
 - NTSC, 559
 - odtworzenie w czasie rzeczywistym, 559
 - PAL, 560
 - pliki, 561
 - przechowywanie filmów w nieciągłych blokach, 593
 - przeplatanie wideo, audio i tekstu w pojedynczym ciągłym pliku z filmem, 592
 - rozmişczenie plików, 591
 - SECAM, 560
 - serwer wideo, 557
 - synchronizacja plików pomocniczych, 561
 - system plików, 561
 - szeregowanie operacji dyskowych, 606
 - szeregowanie procesów, 577
 - szybkości przesyłania danych, 559

- video na żądanie, 557
 - video niemal na żądanie, 587
 - zapewnianie gwarancji jakości usług, 560
 - MultiMediaCard, 1130
 - multimedialny system operacyjny, 555
 - multimedialny system plików, 584
 - deskryptor pliku, 584
 - dostarczenie danych przez sieć, 586
 - funkcje sterujące VCR, 585
 - kompresja, 586
 - MPEG, 586
 - NVOD, 587
 - punkt odtwarzania, 589
 - serwer ściągania, 584
 - serwer wypychania, 584
 - uchwyt, 584
 - video niemal na życzenie, 587
 - video niemal na życzenie z funkcjami magnetowidu, 589
 - wydajność, 585
 - multiple model, 1126
 - MULTiplexed Information and Computing Service, 43, 853
 - multiuser environments, 748
 - mutation engine, 826
 - muteksy, 177
 - problem producent – konsument, 181
 - Pthreads, 179
 - Windows Vista, 1033
 - mutex, 177
 - mutex_lock(), 177, 178
 - mutex_trylock(), 178
 - mutex_unlock(), 177, 178
 - muzyka, 555
 - mv, 871
 - mysz, 473, 478
 - Bluetooth, 479
 - budowa, 478
 - mechaniczna, 478
 - miki, 479
 - optyczna, 479
- N**
- nadzorca, 988
 - nagrywarki CD-R, 446
 - zapis, 447
 - najgorszy pasujący, 240
 - najlepszy pasujący, 240
 - najpierw najkrótsze wyszukiwanie, 457
 - najpierw najkrótsze zadanie, 202
 - najpierw wcześniejszy termin zakończenia, 581
 - nakładanie blokady, 930
 - nakładki, 242
 - namiastka procedury, 700
 - nano, 118
 - nanojadro, 1114
 - nanokernel, 1114
 - nanothreads, 1118
 - nanowątki, 1118
 - napeły, 929
 - napeły CD-ROM, 442
 - Czerwona księga, 442
 - prędkość, 445
 - napeły CD-RW, 449
 - moc lasera, 449
 - napeły DVD, 450
 - naruszenie dostępu, 1056
 - narzędzia komputerowe, 43
 - następny najkrótszy proces, 208
 - następny pasujący, 239
 - następny proces o najkrótszym pozostałym czasie działania, 203
 - nasycenie, 564
 - naśladowanie, 1096
 - NAT, 821
 - nazwane obiekty, 979
 - nazwy, 1165
 - nazwy DNS, 693
 - nazwy plików, 319, 926
 - rozszerzenie, 320
 - nazwy ścieżek, 76, 333
 - względne, 333
 - nazwy trwałe, 979
 - NC-NUMA, 626
 - Near Video On Demand, 587
 - net start, 1022
 - NetWare, 969
 - network address translation, 821
 - Network File System, 355, 945
 - Never Twice the Same Color, 560
 - New Technology, 47
 - next fit, 239
 - NFS, 355, 945
 - architektura systemu, 946
 - automatyczne montowanie, 947
 - buforowanie danych, 952
 - dostęp do plików, 948
 - eksportowane katalogi, 946
 - implementacja, 949
 - klient-serwer, 947
 - mechanizm ochrony, 949
 - montowanie zdalnego systemu plików, 950
 - odczyt z wyprzedzeniem, 951
 - otwieranie zdalnego pliku, 951
 - pamięć podręczna, 952
 - podnoszenie wydajności, 952
 - protokoły, 947
 - r-węzły, 950
 - serwer, 946
 - serwer bezstanowy, 949
 - stanowy system plików, 952
 - struktura warstwy systemu plików, 950
 - uchwyt pliku, 947
 - v-węzeł, 949
 - wirtualny i-węzeł, 949
 - wywołania systemowe, 948
 - zapis, 952
 - zdalne i-węzły, 950
 - NFSv4, 952, 953
 - NFU, 263, 272
 - NGSCB, 730
 - nice(), 893
 - niebieska pigułka, 814
 - niebieski ekran śmierci, 1001
 - niebuforowane operacje wejścia-wyjścia, 1065
 - nieobowiązkowa kontrola dostępu, 1094
 - niepodzielna akcja, 174
 - niepodzielna transakcja, 353
 - nieprawidłowa strona, 1047
 - nieprecyzyjne przerwania, 414
 - nieulotna pamięć RAM, 465
 - niezależność od lokalizacji, 697
 - niezawodna usługa połączeniowa, 689
 - sekwencje komunikatów, 690
 - strumienie bajtów, 690
 - niezawodność systemu plików, 943
 - niezawodny połączeniowy strumień bajtów, 918
 - niezawodny połączeniowy strumień pakietów, 918
 - niezdefiniowane wywołania zewnętrzne, 281
 - niskopoziomowe formatowanie dysku, 452
 - niszczenie procesu, 130
 - NM, 386
 - No Cache NUMA, 626
 - No eXecute, 1100
 - nonpreemptive, 197
 - nonresident attributes, 1084
 - NonUniform Memory Access, 621, 1042
 - nośnik CD-R, 447
 - nośniki wymienne, 1130
 - Not Frequently Used, 263
 - Not Recently Used, 259
 - notacja dziesiętna z kropkami, 758
 - notacja węgierska, 490
 - notebook, 511
 - Novell, 856
 - NRU, 258, 271, 272
 - NSFv3, 952
 - nslookup, 759
 - NT, 968
 - NT API, 975
 - NT File System, 1079
 - NT OS/2, 968
 - NtAllocateVirtualMemory(), 977, 978
 - NtCancelIoFile(), 1070, 1072
 - NtClose(), 1013, 1015
 - NtCreateFile(), 977, 978, 1013, 1015, 1069, 1070
 - NtCreateProcess(), 975, 977, 1013, 1030
 - NtCreateThread(), 977, 1030
 - NtCreateUserProcess(), 1036, 1037, 1038
 - NtDeviceIoControlFile(), 1070, 1072
 - ntdll.dll, 988, 989, 1020
 - NtDuplicateObject(), 977, 978
 - NtFlushBuffersFile(), 1070, 1072
 - NTFS, 320, 352, 353, 983, 1017, 1079, 1103
 - \$AttrDef, 1083
 - \$Extend\$Secure, 1085
 - ::\$DATA, 1089
 - alokacja przestrzeni dyskowej, 1086
 - atrybuty nierezydentne, 1084

NTFS

atrybuty umieszczane w tablicy MFT, 1084
 bloki strumienia, 1086
 dodatkowy strumień danych, 1085
 domyślny strumień danych, 1085
 dostęp do skompresowanych plików, 1092
 dowiązania symboliczne, 1080, 1090
 dowiązania twarde, 1090
 EFS, 1093
 główna tablica plików, 1081, 1082
 identyfikatory plików, 1085
 implementacja systemu plików, 1081
 katalogi, 1089
 kompresja plików, 1090
 krótkie nazwy, 1085
 księgowanie, 1092
 MFT, 1081, 1082
 nazwy plików, 1079
 nazwy strumieni, 1085
 pliki, 1080
 pliki bezpośrednie, 1082
 pliki rozproszone, 1086
 punkty przyłączania, 1081, 1085, 1090
 rekord bazowy, 1081
 rekord tablicy MFT, 1087
 rekordy zmian, 1092
 reparse points, 1081
 struktura systemu plików, 1081
 strumienie, 1080
 szyfrowanie plików, 1092
 ścieżki, 1080
 śledzenie wolnej przestrzeni, 1082, 1083
 wielkość liter nazw plików, 1079
 woluminy, 1081
 wykrywanie zmian, 1092
 wyszukiwanie nazw, 1089
 NtFsControlFile(), 1070, 1071, 1092
 NtLockFile(), 1070, 1071
 NtMapViewOfSection(), 977, 978
 NtNotifyChangeDirectoryFile(), 1070, 1092
 NTOS, 973, 975, 988
 ntoskrnl.exe, 973, 999, 1006
 NtQueryDirectoryFile(), 1070
 NtQueryInformationFile(), 1070, 1071
 NtQueryVolumeInformationFile(), 1070
 NtReadFile(), 1069, 1070
 NtReadVirtualMemory(), 977, 978
 NtResumeThread(), 1031, 1038
 NTSC, 559, 560, 563
 NtSetInformationFile(), 1070, 1071
 NtSetVolumeInformationFile(), 1070
 NtUnlockFile(), 1070, 1071
 NtWriteFile(), 1069, 1070
 NtWriteVirtualMemory(), 977, 978
 NUMA, 620, 626, 907, 1042
 cechy, 626
 NUMBER_OF_THREADS, 149
 numery portów, 821
 numery wersji jądra systemu Linux, 861

NVD, 587
 NX, 1100
 Nyquist Harry, 565

O

O_CREAT, 93
 O_RDONLY, 93
 O_RDWR, 93
 O_WRONLY, 93
 ObCreateObjectType(), 1017
 obiekt sterownika, 979, 1067
 obiektowość, 1112
 obiektowy system operacyjny, 1112
 obiekty, 490, 700, 976, 1112
 COM, 985
 JavaSpace, 706
 listy kontroli dostępu, 734
 metody, 700
 ORB, 700
 obiekty aktywne, 1120
 obiekty dyspozytora, 994, 998
 obiekty kontrolne, 994
 obiekty powiadomienia, 999
 obiekty synchronizacji, 999
 obiekty urządzeń, 979, 1004
 object caches, 909
 object orientation, 1112
 Object Request Brokers, 700
 ObOpenObjectByName(), 1015
 obowiązkowa kontrola dostępu, 745
 obraz pamięci procesu, 91
 obraz rdzenia, 73
 obrona wielostrefowa, 820
 obsługa błędów, 416
 obsługa błędów braku stron, 287, 293, 1054
 obsługa długich nazw plików, 346
 obsługa liczników dozoruujących, 471
 obsługa plików, 319
 obsługa przerwania, 62, 411
 obsługa sieci, 917
 obsługa sygnałów, 889
 obsługa wątków, 147
 obsługa wielowątkowości, 146
 obsługa wirtualizacji, 109
 obsługa zegara, 466
 obszar wymiany, 912
 obszary, 340, 1082
 obszary krytyczne, 930
 ochrona, 719, 730, 820
 bezpieczeństwo
 wielopoziomowe, 745
 domeny ochrony, 730, 731
 firewall, 820
 izolowanie kodu mobilnego, 835
 listy kontroli dostępu, 733
 macierz ochrony, 733
 model formalny bezpiecznych systemów, 743
 obrona wielostrefowa, 820
 podpisywanie kodu, 831
 system zaufany, 740
 ukryte kanały, 748
 uprawnienia, 731, 736
 weryfikatory integralności, 828
 weryfikatory zachowań, 828
 wtrącanie do więzienia, 832
 wykrywanie włamań z użyciem modeli, 833
 zasada POLA, 731
 zaufana baza obliczeniowa, 742
 ochrona pamięci, 75
 oczekiwanie aktywne, 61
 od, 872, 873
 odcień, 564
 odczyt, 32
 odczyt z wyprzedzeniem, 951
 odgadywanie hasel, 756, 757
 odległość Hamminga, 771
 odpytywanie, 418
 odpytywanie o oczekiwane zdarzenie, 472
 odtwarzanie danych, 366
 odtwarzanie po awarii, 464
 odtwarzanie systemu plików, 370
 odtwarzanie w czasie rzeczywistym, 559
 odwracanie priorytetów, 1044
 odwrócone tabele stron, 255
 odwzorowania plików, 982
 odwzorowanie adresu liniowego
 na fizyczny, 305
 odwzorowanie adresu wirtualnego
 na fizyczny, 249
 odzyskiwanie pamięci, 112
 offline, 39
 ogólna warstwa blokowa, 922
 ograniczony bufor, 172
 okna, 486, 487
 parametry, 487
 pasek menu, 487
 pasek narzędzi, 487
 pasek tytułu, 487
 paski przewijania, 487
 tworzenie, 487
 okna tekstowe, 480
 sekwencje sterujące, 480
 termcap, 480
 Olsen Ken, 968
 omiatanie obrazu, 571
 one-shot mode, 467
 one-time passwords, 763
 one-way hash chain, 763
 ontogeneza, 81
 opakowanie, 152
 open, 388
 open source, 45
 open(), 89, 93, 327, 916, 932, 1116
 Open(), 1012
 opendir(), 336, 935
 OpenFileMapping(), 1051
 OpenMutex(), 1035
 OpenSemaphore(), 1011, 1035
 operacje na katalogach, 335
 operacje na macierzy ochrony, 743
 operacje na plikach, 98, 327, 921
 operacje seek zachodzące na siebie, 435
 operacje wejścia-wyjścia, 63
 priorytety, 1068
 oponenci, 721
 opóźnione wywołania procedur, 995

- oprogramowanie, 30
 - do generowania wyjścia, 479
 - do wprowadzania danych, 473
 - działające w trybie jądra, 32
 - inwigilujące, 809
 - klawiatura, 474
 - middleware, 685
 - mysz, 478
 - obsługa zegara, 468
 - pośredniczące, 685
 - reklamowe, 813
 - sieciowe, 874
 - oprogramowanie szpiegujące, 809
 - ataki, 812
 - cele marketingowe, 809
 - inwigilacja, 809
 - kategorie, 809
 - kontrolki ActiveX, 811
 - pobieranie plików bez wiedzy użytkownika, 810
 - przejmowanie przeglądarki, 813
 - rozprzestrzenianie się, 810
 - zombie, 809
 - oprogramowanie wejścia-wyjścia, 415
 - aktywne oczekiwanie, 418
 - bufor cykliczny, 429
 - buforowanie, 416, 428
 - buforowanie w przestrzeni jądra
 - z kopiowaniem do przestrzeni użytkownika, 429
 - buforowanie w przestrzeni użytkownika, 429
 - buforowanie wyjścia, 430
 - cele, 415
 - demony, 433
 - główny numer urządzenia, 428
 - jednolity interfejs sterowników urządzenia, 427
 - katalog spoolera, 433
 - niezależność od urządzeń, 415, 426
 - obsługa błędów, 416
 - odpytywanie, 418
 - podwójne buforowanie
 - w przestrzeni jądra, 429
 - pomocniczy numer urządzenia, 428
 - problem jednolitego nazewnictwa, 415
 - procedura obsługi przerwania, 421
 - programowane wejście-wyjście, 417
 - przestrzeń użytkownika, 432
 - przydzielanie dedykowanych urządzeń, 432
 - raportowanie błędów, 431
 - realizacja wejścia-wyjścia, 417
 - rozmiar bloku niezależny od urządzenia, 432
 - spooling, 433
 - sterowniki urządzeń, 422
 - transfery asynchroniczne, 416
 - transfery synchroniczne, 416
 - warstwy, 420
 - wejście-wyjście sterowane
 - przerwaniami, 419
 - wejście-wyjście
 - z wykorzystaniem DMA, 420
 - write, 430
 - zbyt wczesne opróżnianie bufora, 416
 - zwalnianie dedykowanych urządzeń, 432
 - opóżnianie bufora, 416
 - optymalizacja wydajności oprogramowania, 1179
 - optymalizacja z myślą o typowych przypadkach, 1186
 - optymalny algorytm zastępowania stron, 258
 - opuszczanie regionu krytycznego, 170
 - OR, 495
 - Orange Book, 1003
 - ORB, 700
 - organizacja pamięci, 32, 229
 - organ-pipe algorithm, 599
 - ortogonalność, 1164
 - OS/2, 968, 1020
 - OS/360, 41, 106, 1143
 - OS/390, 68
 - osłona, 152
 - osobowości, 973, 974
 - ostrich algorithm, 526
 - ośrodek, 729
 - otwarty punkt krzyżowy, 623
 - OUT, 61, 403, 404
 - overlapped seeks, 435
 - overwriting viruses, 795
- P**
- P, 174, 571
 - P1003.4, 892
 - packet sniffer, 760
 - PAE, 910, 1051, 1056
 - page allocator, 908
 - page daemon, 912
 - page fault, 152, 245
 - Page Fault Frequency, 274
 - Page Frame Number Database, 1060
 - Page Frame Reclaiming Algorithm, 912
 - pagefile, 1048
 - pakiet potwierdzający, 689
 - pakiety, 649
 - pakiety IP, 691
 - pakiety żądań wejścia-wyjścia, 1016, 1071, 1074
 - PAL, 559, 560, 563
 - Palladium, 730
 - Palm OS, 69
 - pamięć, 54, 82
 - abstrakcja, 228, 232
 - adresowanie, 228
 - asocjacyjna, 250
 - CMOS, 57
 - dynamiczna relokacja, 233
 - EEPROM, 57
 - fizyczna, 228
 - flash, 57, 1131
 - główna, 227
 - hierarchia, 55, 227
 - kompaktowanie, 236
 - menedżer pamięci, 228
 - o dostępie losowym, 57
 - organizacja, 229
 - problem relokacji, 231
 - programy o rozmiarach przekraczających objętość pamięci, 242
 - przestrzeń adresowa, 232
 - przydzielanie dynamiczne, 237
 - RAM, 57, 227, 228
 - ramka stron, 244
 - rdzeniowa, 57
 - ROM, 57, 228
 - segment danych, 92
 - segment stosu, 92
 - segment tekstu, 92
 - segmenty, 295
 - statyczna relokacja, 230
 - stronicowanie, 243
 - strony, 242, 244
 - system bez abstrakcji pamięci, 229
 - transakcyjna, 1025
 - tylko do odczytu, 57
 - wirtualizacja, 678
 - wymiana pamięci, 229, 235
 - zarządzanie, 227, 237
 - zarządzanie energią, 505
 - zewnętrzna fragmentacja, 298
 - pamięć lokalna wątku, 1024
 - pamięć podręczna, 53, 55, 140, 375, 922
 - cache hit, 55
 - L1, 56
 - L2, 56
 - linie, 55
 - poziomy, 55
 - trafienie, 55
 - wiersz, 621
 - z natychmiastowym zapisem, 377
 - pamięć podręczna obiektów, 909
 - pamięć podręczna systemu Windows Vista, 1063
 - menedżer pamięci podręcznej systemu Windows, 1064
 - niebuforowane operacje wejścia-wyjścia, 1065
 - perspektywy, 1064
 - ReadyBoost, 1065
 - ValidDataLength, 1064
 - zarządzanie adresami wirtualnymi jądra, 1064
 - zarządzanie przesunięciami ostatnich bajtów plików, 1064
 - pamięć trwała, 318
 - operacje, 318
 - pamięć wirtualna, 59, 85, 235, 241, 678, 861
 - 8088, 242
 - adres wirtualny, 243, 245
 - algorytm zastępowania stron, 257, 271
 - biblioteki współdzielone, 281
 - bit Obecna/nieobecna, 245, 255
 - bit zabrudzenia, 248
 - blokowanie stron w pamięci, 289
 - brak strony, 245
 - buforowanie wyłączone, 248
 - bufory TLB, 250

- pamięć wirtualna
 - demon stronicowania, 284
 - interfejs, 284
 - kod niezależny od pozycji, 283
 - Mach, 292
 - magazyn stron, 290
 - miękkie chybienie, 253
 - MMU, 243
 - model zbioru roboczego, 266
 - MULTICS, 300
 - obsługa błędów braku strony, 287
 - oddzielanie strategii
 - od mechanizmu, 292
 - odwrócone tabele stron, 255
 - odzworowanie adresów
 - wirtualnych na adresy fizyczne, 249
 - osobne przestrzenie instrukcji i danych, 278
 - page fault, 245
 - PFF, 274
 - pliki odwzorowane w pamięci, 283
 - problem zastępowania stron, 257
 - problemy implementacji, 285
 - programowe zarządzanie buforem TLB, 251
 - przeładowanie, 266
 - przyspieszanie stronicowania, 249
 - ramka stron, 244
 - relacja pomiędzy adresami wirtualnymi a adresami pamięci fizycznej, 244
 - rozmiar strony, 277
 - rozproszona pamięć współdzielona, 285
 - segmentacja, 294
 - strategia alokacji, 273
 - strategia czyszczenia, 284
 - stronicowanie, 243
 - stronicowanie na żądanie, 265
 - stronicowanie wstępne, 266
 - strony, 242, 244
 - strony współdzielone, 279
 - tabela stron, 246, 247
 - tabela stron dla pamięci o dużej objętości, 253
 - twarde chybienie, 253
 - wewnętrzna fragmentacja, 277
 - wielopoziomowe tabele stron, 253
 - wirtualna przestrzeń adresowa, 243
 - wpisy w tabeli stron, 247
 - wznawianie instrukcji, 288
 - zadania systemu operacyjnego w zakresie stronicowania, 286
 - zarządzanie obciążeniem, 276
 - zastępowanie stron, 257
 - zmniejszanie liczby procesów rywalizujących o pamięć, 276
- panning, 571
- paradygmat algorytmiczny, 1152
- paradygmat sterowany zdarzeniami, 1152
- paradygmaty danych, 1153
- paradygmaty interfejsu użytkownika, 1151
- paradygmaty wykonywania, 1152
- Paramecium, 1162
- parasitic viruses, 797
- paravirt ops, 677
- parawirtualizacja, 110, 675, 676
- PARC, 47, 486
- parent process, 877
- Parse(), 1012, 1015, 1016
- partycje, 96, 337, 382
- pasek menu, 487
- pasek narzędzi, 487
- pasek tytułu, 487
- paski, 438
- paski przewijania, 487
- paskowanie, 439, 595
 - losowe, 601
 - naprzemienne, 601
 - szerokie, 602
 - wąskie, 603
- passwd, 758, 898, 957
- paste, 871, 873
- Paterson Tim, 46
- PATH, 792, 1030
- pause, 132
- pause(), 880, 884
- PC, 413
- PC Card, 1067
- PCI, 63, 402, 1067
- PCI Express, 64
- PCI-x, 1067
- PCM, 559, 566
- PDA, 69, 307, 1108, 1109
- PDD, 1127
- pdflush, 915
- PDP, 44
- PDP-1, 82, 83, 754
- PDP-11, 44, 45, 83, 754, 854, 855
- PDP-11/20, 853
- PDP-11/45, 853
- PDP-11/70, 853, 945
- PDP-7, 44, 853
- PDP-8, 83, 754
- PEB, 1023, 1037
- peer-to-peer, 394
- pełna ścieżka wejścia-wyjścia, 509
- Pentium, 48, 308
 - segmentacja, 302
 - zabezpieczenia, 307
- Pentium 4, 53
- Peripheral Component Interconnect, 63
- Personal Digital Assistant, 69, 1108
- personal firewalls, 823
- personalities, 973
- perspektywy, 982, 1064
- Peterson G.L., 168
- PFF, 274
- Pfill, 498
- PFN database, 1060
- PFRA, 912, 913
 - wybór starych stron, 914
 - wymienność, 914
- PG_active, 914
- PG_referenced, 914
- Phase Alternating Line, 560
- Philips, 442
- PhotoCD, 448
- Physical Address Extension, 910, 1051, 1056
- Physical Device Driver, 1127
- piaskownica, 837
- PID, 90, 148, 877, 884, 892, 945, 1168, 1169
- Pidgin Pascal, 182, 184
- piekło DLL, 1020
- pierścienie zabezpieczeń, 306
- pierwszy pasujący, 239
- pierwszy wchodzi,
 - pierwszy wychodzi, 260
- pierwszy zgłoszony,
 - pierwszy obsługowany, 201
- PikeOS, 103
- piko, 118
- piksel, 494, 564
- PIN, 766, 772
- ping, 758
- pipe symbol, 869
- pipe(), 932, 934
- pipeline, 869
- pipes, 878
- pits, 443
- PKI, 729
- PL, 386
- PL/I, 43, 854, 1144
- plaintext, 724
- platforma .NET, 971
- platforma smartfonów, 1107
- plik hasel, 758
- plik koźła ofiarnego, 823
- pliki, 75, 89, 318, 319, 695, 926
 - alokacja bazująca na liście jednokierunkowej, 340
- archiwum, 324
- ASCII, 323, 926
- atrybuty, 325
- bezpośrednie, 1082
- binarne, 323, 926
- bity rwx, 79
- blokady współdzielone, 930
- blokady wyłączone, 930
- blokowanie, 930
- blokowe pliki specjalne, 78
- bmp, 495
- ciągła alokacja, 338
- deskryptor, 77, 330, 932
- dostęp losowy, 325
- dostęp sekwencyjny, 325
- dowiązania, 928
- drzewo, 322
- flagi, 326
- fragmentacja, 339
- implementacja, 338
- ini, 985
- i-numer, 94
- i-węzły, 343
- kompresja, 1090
- kopiowanie, 329, 871
- liczba magiczna, 323
- LINK, 348
- luki, 371
- Makefile, 113
- multimedialne, 561, 591
- nagłówkowe, 112, 872

- nazwy, 319, 926
- objektowe, 113
- obsługa długich nazw, 346
- odczyt, 93
- operacje, 327
- pola czasowe, 327
- pole klucza, 322
- przenoszenie, 871
- rozproszone, 1086
- rozszerzenie, 320
- sekwencja bajtów, 322
- sekwencja rekordów, 322
- SETGID, 956
- SETUID, 374, 955
- składowanie informacji, 319
- specjalne, 78, 323, 371, 916, 928
- specjalne blokowe, 916, 928
- specjalne znakowe, 917
- struktura, 321
- szyfrowanie, 1092
- ścieżki, 76
- ścieżki bezwzględne, 333
- tryb otwarcia, 93
- tryby ochrony, 954
- tylko do odczytu, 388
- tymczasowe, 327
- typy, 323
- usuwanie, 871
- współdzielone, 347
- wykonywalne, 324
- wyświetlanie zawartości, 871
- wywołania systemowe, 93
- zabezpieczenia, 79, 326
- zapis, 93
- zarządzanie, 93, 319
- zmiana bitów uprawnień, 871
- zmiana trybu, 96
- znakowe pliki specjalne, 78
- zwykle, 323
- pliki odwzorowane w pamięci, 283, 902
- pliki stron, 1048
- pliki stronicowania, 912
- plug and play, 65, 66, 1001
- plugboard, 37
- płynność obrazu, 564
- plyta główna, 66
- plyta macierzysta, 66
- plyty Blu-ray, 452
- plyty CD, 442, 444
 - Czerwona księga, 442
 - formatowanie danych komputerowych, 444
 - High Sierra, 446
 - Kodak PhotoCD, 448
 - lands, 443
 - logiczne rozmieszczenie danych, 445
 - odczyt, 443
 - odtwierzanie, 443
 - pits, 443
 - pola, 443
 - Pomarańczowa księga, 448
 - preambuła, 444
 - proces przygotowania, 442
 - ramka, 444
 - sektor CD-ROM, 444
 - sesje, 448
 - spis treści, 448
 - struktura, 443
 - system plików, 446
 - szklany dysk-matryca, 442
 - ścieżki CD-ROM, 448
 - tryb 1, 445
 - tryb 2, 445
 - TOC, 448
 - wgłębienia, 443
 - wielokrotny zapis, 449
 - wielosesyjne plyty CD-ROM, 448
 - Zielona księga, 446
 - Żółta księga, 444
- plyty CD-R, 447
- plyty CD-ROM, 381, 444
- plyty CD-RW, 449
- plyty DVD, 449, 556
 - dwustronne plyty, 451
 - dwuwarstwowe plyty, 451
 - formaty, 450
 - laser, 450
 - pojemność, 450
 - spiralą, 450
 - wgłębienia, 450
- plyty HD DVD, 452
- PN, 385
- pobieranie plików bez wiedzy użytkownika, 810
- poczta elektroniczna, 804
- podmioty, 734
- podpisy cyfrowe, 727, 831
 - blok podpisu, 728
 - CA, 729
 - centrum certyfikacji, 729
 - certyfikaty, 729
 - funkcje skrótu, 727
 - infrastruktura klucza publicznego, 729
 - ośrodek, 729
 - PKI, 729
 - wyznaczanie bloku podpisu, 728
- podpisywanie kodu, 831, 1134
 - kryptografia z kluczem publicznym, 831
- podstawianie monoalfabetyczne, 725
- podszycanie się pod ekran logowania, 774
- podwójne buforowanie, 429
- podwójne buforowanie w przestrzeni jądra, 429
- podwójny przeplot, 455
- podwójny torus, 648
- podział czasu, 42
- pojedynczy przeplot, 454
- pojemność dysków, 454
- pola, 443, 563
- POLA, 731
- pole klucza, 322
- polecenia, 867, 870
 - ar, 872
 - cat, 871, 873
 - chmod, 871, 873
 - cp, 871, 873, 881
 - cut, 871, 873
 - flagi, 868
 - gcc, 872
 - grep, 871, 873
 - head, 871, 873
 - ls, 871, 873, 888
 - make, 872, 873
 - mkdir, 871, 873
 - mv, 871
 - od, 872, 873
 - paste, 871, 873
 - potok, 869
 - pr, 872, 873
 - ps, 873
 - rm, 871, 873
 - rmdir, 871, 873
 - sort, 871, 873
 - tail, 871, 873
 - tr, 872, 873
- polecenia ochrony, 743
- polymorphic viruses, 826
- Pomarańczowa księga, 448, 1003, 1093, 1094
- pomocniczy deskryptor woluminu, 383
- pomocniczy numer urzędzenia, 428, 917
- POPF, 672
- popularność treści, 598
- pop-up, 157
- pop-up thread, 657
- porównaj i wymień, 991
- port wejścia-wyjścia, 403, 1072
- Portable Operating System, 857
- porty, 759, 821
- POSIX, 44, 86, 88, 90, 97, 857, 865
- POSIX 1003.2, 866, 871
- POST, 896
- postarzanie, 264, 272
- PostScript, 836, 1182
- pośrednictwo, 1173
- potok, 51, 78, 412, 869, 878, 932
 - trójfazowy, 52
 - tworzenie, 934
- potomny proces, 73
- poufność danych, 720
- Power-On-Self-Test, 896
- PowerScope, 508
- PowerShell, 986
- powłoka, 29, 73, 80, 857, 867
 - bash, 867
 - Bourne Again Shell, 867
 - Bourne'a, 867
 - csh, 867
 - definiowanie nazw wielu plików, 868
 - filtry, 869
 - flagi, 868
 - head, 869
 - ksh, 867
 - ls, 868
 - polecenia, 867
 - potok, 869
 - proces potomny, 880
 - procesy uruchamiane w tle, 870
 - przekierowanie wejścia-wyjścia, 80, 869
 - rm, 869
 - skrypty powłoki, 870
 - sort, 868

- powłoka
 - standardowe wejście, 80, 868
 - standardowe wyjście, 80, 868
 - standardowe wyjście błędów, 868
 - symbol zachęty, 80
 - symbole wieloznaczne, 868
 - wc, 870
 - znak zachęty, 867
 - znaki magiczne, 868
- poziom integralności, 1097
- późne kojarzenie, 1167
- pr, 872, 873
- prawa dostępu, 845
- prawa ogólne, 739
- prawo Moore'a, 53, 628
- prawo Murphy'ego, 163
- prawo Zipfa, 598
- preambula, 402
- Predictive, 571
- preemptive, 197
- prefiksy metryczne, 118
- preprocesor C, 113
- presja pamięci, 1059
- principals, 734
- Principle of Least Authority, 731
- Principle of Least Surprise, 864
- printf(), 87, 433, 780
- priority inversion, 1044
- priorytety, 205
- priorytety dynamiczne, 894
- priorytety wątków Win32, 1041
- PRISM/Mica, 968
- privacy, 721
- privilege escalation attack, 786
- problem braku stron w pamięci, 152
- problem czytelników i pisarzy, 218
- problem inwersji priorytetów, 171
- problem jednolitego nazewnictwa, 415
- problem pięciu filozofów, 214
- zagłodzenie, 216
- problem producent – konsument, 172
- Java, 186
- monitory, 184
- muteksy, 181
- przekazywanie komunikatów, 190
- semafony, 175
- problem programów o rozmiarach przekraczających objętość pamięci, 242
- problem relaksacji, 192
- problem relokacji pamięci, 231
- problem zamknięcia, 749
- problem zastępowania stron, 257
- problemy komunikacji
 - między procesami, 214
- problemy projektowe systemów stronicowania, 273
- procedura obsługi przerwania, 411, 421, 995
- procedura pośrednicząca klienta, 658
- procedura pośrednicząca serwera, 658
- proces, 72, 123, 124, 126, 1024
 - bariery, 191
 - blok PEB, 1023
 - blokada, 131
 - bloki zarządzania procesami, 133
 - blokowanie, 131
 - błąd krytyczny, 129
 - demon stron, 912
 - demony, 127, 876
 - deskryptor, 885
 - diagram stanów, 131
 - drzewo procesów, 73
 - działanie, 131
 - dziecko, 90, 128, 877
 - execve(), 128
 - fork(), 128
 - getty, 898
 - gotowość, 131, 132
 - grupy procesów, 878
 - hierarchia, 130
 - identyfikator PID, 877, 878, 884
 - implementacja, 133
 - init, 130, 897
 - interpreter poleceń, 73
 - jednowątkowy, 158
 - kanały komunikacyjne, 878
 - kill(), 130
 - kommunikacja międzyprocesowa, 73, 161
 - kończenie działania, 129
 - kopiowanie przy zapisie, 887
 - licznik programu, 131
 - Linux, 876, 884
 - macierzysty, 877
 - model, 124
 - modelowanie
 - wieloprogramowości, 135
 - monitory, 182
 - muteksy, 177
 - niszczenie, 130
 - obraz pamięci, 877, 885
 - obraz rdzenia, 73
 - pamięć, 92
 - parametry szeregowania, 885
 - PID, 90
 - potok, 78, 878
 - potomny, 73, 877
 - powłoka, 73
 - prace badawcze, 219
 - priorytet, 205
 - problem inwersji priorytetów, 171
 - program szeregujący, 132
 - przekazywanie komunikatów, 188
 - przełączanie procesora pomiędzy procesami, 125
 - przestrzeń adresowa, 72, 74
 - regiony krytyczne, 163
 - rejstry maszynowe, 885
 - rodzic, 90, 877
 - rozliczenia, 886
 - rozwidlanie, 880
 - sekcja krytyczna, 164
 - sekwencyjny, 124
 - semafony, 174
 - stan wywołania systemowego, 885
 - stany, 131
 - stopień wieloprogramowości, 135
 - stos jądra, 886
 - sygnał alarmowy, 74
 - sygnały, 878, 882, 885
 - Symbian OS, 1119
 - synchronizacja, 176, 1033
 - systemowy, 1028
 - szeregowanie, 193
 - tabela procesów, 73, 134
 - tablica deskryptorów plików, 885
 - task_struct, 884, 886
 - thread_info, 886
 - tworzenie, 98, 126, 877, 886, 1036
 - uchwyt, 130
 - UNIX, 128
 - uruchamianie w tle, 870
 - wątki, 137, 144
 - wektor przerwań, 133
 - wieloprogramowość, 124
 - Win32, 128
 - Windows Vista, 1023
 - wstrzymanie, 72
 - wykonywanie w tle, 876
 - wykorzystanie procesora, 135
 - wyścig, 162
 - wywołania systemowe, 90
 - zadania, 1025
 - zadania wsadowe, 128
 - zakleszczenie, 182, 517
 - zarządzanie procesami, 90, 879
 - zarządzanie zadaniami
 - systemowymi, 128
 - zdarzenia, 127
 - zombie, 882
 - zorientowany na obliczenia, 195
 - zorientowany na wejście-
wyjście, 195
- proces kolaboranta, 749
- proces rozruchu komputera, 66
- proces wymiany, 911
- procesor, 48, 50
 - hiperwątkowość, 53
 - licznik programu, 51
 - pamięć podręczna, 53
 - Pentium, 51
 - Pentium 4, 53
 - potoki, 51
 - rdzenie, 54, 629
 - rejstry, 51
 - słowo stanu programu, 51
 - SPARC, 51
 - stan uśpienia, 504
 - superskalarny, 52
 - układy wielowątkowe, 53
 - VAX, 988
 - wielordzeniowy, 53
 - wielowątkowość, 53
 - wskaźnik stosu, 51
 - zarządzanie energią, 504
 - zestaw instrukcji, 51
- procesor sieci, 651
- Process Environment Block, 1023, 1037
- Process Identifier, 877, 884
- process scheduler, 132
- procesy obliczeniowe, 894
- ProcHandle, 977
- procmon, 986
- producent – konsument, 172
- profilowanie, 471
- program, 31, 126
- program agentów, 836

- program antywirusowy, 823
- program aplikacyjny, 34
- program boot, 896
- Program Counter, 413
- program Javy, 841
- program o olbrzymich rozmiarach, 241
- program o rozmiarach
 - przekraczających objętość pamięci, 242
- program obiektowy, 100
- program przechwytyjący pakiety, 760
- program startowy, 896
- Program Status Word, 51, 230
- program szeregujący, 132, 193
- program użytkownika, 31
- program z interfejsem użytkownika, 30
- program zarządzający, 793
- program zorientowany na komunikaty, 488
- programowa symulacja algorytmu LRU, 263
- programowalne zegary, 467
- programowane wejście-wyjście, 417
- programowanie, 37
- programowanie napędów dysków elastycznych, 33
- programowanie obiektowe, 976
- programowe zarządzanie buforem TLB, 251
- projekt BSD, 48
- projekt POSIX, 858
- projekt programistyczny, 113
- projektowanie systemu operacyjnego, 1143, 1146, 1198
 - abstrakcja, 1145
 - buforowanie, 1183
 - cele, 1144
 - czas kojarzenia nazw, 1167
 - dylemat przestrzeni – czas, 1180
 - efekt lokalności, 1185
 - egzjozdro, 1159
 - graficzny interfejs użytkownika, 1151
 - GUI, 1151
 - implementacja systemu, 1158
 - implementacja z dołu do góry, 1170
 - implementacja z góry na dół, 1170
 - interfejs wywołań systemowych, 1155
 - interfejsy, 1148
 - izolowanie błędów, 1146
 - kompletność, 1149
 - komputer zasilany bateriami, 1197
 - komunikacja sieciowa, 1195
 - mechanizm, 1163
 - multimedia, 1196
 - nazewnictwo, 1165
 - optymalizacja, 1179
 - optymalizacja z myślą o typowych przypadkach, 1186
 - ortogonalność, 1164
 - paradygmat sterowany zdarzeniami, 1152
 - paradygmaty, 1150
 - paradygmaty danych, 1153
 - paradygmaty interfejsu użytkownika, 1151
- paradygmaty wykonania, 1152
 - pośrednictwo, 1173
 - programiści, 1151
 - prosty interfejs, 1149
 - przetwarzanie klient-serwer, 1161
 - rozszerzalność, 1162
 - rozwiązania siłowe, 1175
 - scenariusze wykorzystania, 1147
 - spójność architekuralna, 1150
 - sprawdzanie błędów, 1176
 - sterowniki urządzeń, 1161
 - stos jądra, 1169
 - strategia, 1163
 - stronicowanie, 1164
 - struktura systemu, 1158
 - struktury dynamiczne, 1168
 - struktury statyczne, 1168
 - system klient-serwer z mikrojądrem, 1160
 - system rozproszony, 1196
 - system równoległy, 1196
 - system wbudowany, 1197
 - system wielowarstwowy, 1158
 - szeregowanie wątków, 1163
 - trendy, 1192
 - trudności, 1146
 - układy wielordzeniowe, 1193
 - ukrywanie sprzętu, 1171
 - użytkownicy, 1147, 1151
 - wątki jądra, 1162
 - węzły czujników, 1198
 - wielka przestrzeń adresowa, 1194
 - wielobieżność, 1175
 - wielokrotne użycie kodu, 1174
 - WIMP, 1151
 - wirtualizacja, 1193
 - wskazówki, 1185
 - wydajność, 1177
 - wywołania systemowe, 1145, 1155
 - zarządzanie projektem, 1186
- projektowanie
 - systemu stronicowania, 273
- promowanie wątków interaktywnych, 894
- prompt, 80, 867
- proporcjonalność, 200
- prosta reguła bezpieczeństwa, 746
- prosta reguła integralności, 748
- protection, 719
- protection commands, 743
- protokoły, 545, 689, 691
 - DNS, 693
 - FTP, 693
 - host-host, 691
 - HTTP, 693, 1157
 - IIO, 701
 - IP, 691, 692, 919
 - koherencja pamięci podręcznych, 622
 - NFS, 355
 - router-router, 691
 - stos protokołów, 691
 - TCP, 691, 692, 919, 1157
 - TCP/IP, 857
 - THINC, 498
 - UDP, 919
- protokoły obsługi, 924
- próbkowanie częstotliwości, 576
- próbkowanie fali sinusoidalnej, 566
- PRT, 1139
- prywatność, 721
- przechowywanie aktualnej godziny, 468
- przechowywanie filmów
 - w nieciągłych blokach, 593
- przechowywanie informacji, 317
- przechwyć i emuluj, 672, 675
- przechwytywanie pakietów sieciowych, 760
- przeciwdziałanie zakleszczeniom, 540
 - atak na warunek braku wywołania, 541
 - atak na warunek cyklicznego oczekiwania, 542
 - atak na warunek wstrzymania i oczekiwania, 541
 - atak na warunek wzajemnego wykluczania, 540
- przeciwdziałanie zbyt długotrwałemu działaniu procesów, 469
- przeglądy kodu, 774
- przejmowanie przeglądarki, 813
- przejście procesora w stan bezczynności, 472
- przekazywanie komunikatów, 188
 - komunikat potwierdzający, 189
 - MPI, 191
 - problem producent – konsument, 190
 - problemy projektowe, 189
 - rendez-vous, 191
 - skrzynka pocztowa, 191
 - uwierzytelnianie, 189
- przekazywanie uprawnień, 1096
- przekierowanie wejścia-wyjścia, 80, 869
- przekos cylindrów, 453
- przekos głowic, 454
- przekształcenia DCT, 569
- przeladowanie, 266
- przełączanie banków, 1050
- przełączanie kontekstu, 59, 204
- przełączanie obwodów, 650
- przełączanie pakietów typu przechowaj i prześlij, 649
- przełączanie procesora pomiędzy procesami, 125
- przełączanie procesów, 204
- przełączany Ethernet, 686
- przełączniki krzyżowe, 622
- przenoszenie plików, 871
- przenośny kompilator języka C, 855
- przenośny system operacyjny, 857
- przenośny UNIX, 855
- przepelnienie bufora, 777
 - eliminowanie usterek, 779
 - wykrywanie narażonych programów, 779
- przepelnienie liczb całkowitych, 783
- przeplatanie, 563
- przeplatanie wideo, audio i tekstu w pojedynczym ciągłym pliku z filmem, 592
- przepustowość, 199

przerwania, 62, 410, 471
 kopia-cień rejestru, 414
 nieprecyzyjne, 412, 414
 obsługa, 62, 411
 PC, 413
 potoki, 412
 precyzyjne, 412, 413, 414
 procedura obsługi, 411
 rejestr licznika programu, 413
 sposób powstawania, 411
 stan wykonania instrukcji, 413
 układy superskalarne, 412
 wejście-wyjście, 472
 wektor przerwania, 62, 411
 przerwania na żądanie, 467
 przestrzeń adresowa, 72, 74, 232,
 278, 295, 1194
 D, 279
 I, 279
 osobne przestrzenie instrukcji
 i danych, 278
 rejestr bazowy, 233
 rejestr limitu, 233
 urządzenia wejścia-wyjścia
 odwzorowane w pamięci, 406
 wirtualna, 243
 wymiana pamięci, 235
 przestrzeń krotek, 702
 przestrzeń nazw NT, 978
 przestrzeń nazw obiektów, 1011
 przestrzeń portów wejścia-wyjścia,
 61, 403
 przesyłanie komunikatów, 619
 przetwarzanie, 658
 przetwarzanie
 duże ilości danych, 142
 klient-serwer, 1161
 rozproszone, 709
 równoległe, 618
 przetwornik ADC, 565
 przezroczystość lokalizacji, 697
 przezroczystość nazewnictwa, 697
 przyciski sterowania zasilaniem, 500
 przydzielanie
 dedykowane urządzenia, 432
 pamięć, 909
 zasoby, 35
 przyjazny użytkownikom, 47
 przynęty, 835
 przypadkowa utrata danych, 723
 przypinanie stron, 290
 przyrostowa kopia zapasowa, 366
 przyspieszenie stronicowania, 249
 przystawka STB, 558
 przystosowywanie kodu
 jednowątkowego
 do obsługi wielu wątków, 158
 ps, 873
 pseudopliki, 934
 pseudowspółbieżność, 124
 Psion, 1108
 Psion Computers, 1109
 PSW, 51, 109, 230
 psychoakustyka, 574
 pthread_attr_destroy(), 148
 pthread_attr_init(), 148

pthread_cond_broadcast(), 180
 pthread_cond_signal(), 180
 pthread_cond_wait(), 180, 181
 pthread_create(), 148, 150
 pthread_exit(), 148, 150
 pthread_join(), 148, 150
 pthread_mutex_destroy(), 179, 180
 pthread_mutex_init(), 179, 180
 pthread_mutex_lock(), 179, 180
 pthread_mutex_trylock(), 179, 180
 pthread_mutex_unlock(), 179, 180
 pthread_yield(), 148, 150
 Pthreads, 147, 148, 149
 muteksy, 179
 synchronizacja, 179
 zmienne warunkowe, 179
 Public Key Infrastructure, 729
 publikowanie, 704
 publikuj-subskrybuj, 704
 pull servers, 584
 pulpit graficzny, 81
 Pulse Code Modulation, 566
 PulseEvent(), 1034, 1035
 punkt krzyżowy, 622
 punkt odtwarzania, 589
 punkty kontrolne, 533
 punkty przyłączania, 983, 1081, 1090
 push server, 584
 PX, 385

Q

QNX, 69, 103
 QueryName(), 1012, 1013
 QUIT, 478
 quotas, 349

R

race condition, 163
 radiator, 618
 RAID, 371, 437, 438
 paski, 438
 RAID 0, 438, 440
 RAID 1, 439, 440
 RAID 2, 439, 440
 RAID 3, 439, 441
 RAID 4, 439, 441
 RAID 5, 439, 441
 RAM, 57, 228
 RAMAC, 84
 ramka, 444
 ramka stosu, 51
 ramka stron, 244
 ramka wideo, 562
 ramki B, 571
 ramki I, 571
 ramki P, 571, 572
 Random Access Memory, 57, 227
 raportowanie błędów, 431
 Rate Monotonic Scheduling, 580
 Raw, 498
 raw block files, 923
 raw mode, 475
 rdzenie, 54, 629

rdzenie wywołania systemowe NT, 975
 rdzenny interfejs programowania
 aplikacji systemu NT, 975
 READ, 624
 read ahead, 951
 Read Only Memory, 57
 read(), 86, 89, 93, 328, 330, 584, 916,
 932, 933, 941
 read_global(), 160
 READ_PORT_UCHAR(), 992
 READ_PORT_ULONG(), 992
 READ_PORT_USHORT(), 992
 readdir(), 336, 935
 ReadFile(), 99, 1090
 Read-Only Memory, 228
 ReadyBoost, 1065, 1066
 recalibrate, 462
 receive(), 157, 188, 189, 654
 Rectangle(), 493
 Red Hat, 863
 red-black tree, 911
 redundancja chwilowa, 571
 Redundant Array of Inexpensive
 Disks, 438
 Redundantna Macierz Tanich
 Dysków, 438
 reenrancy, 1175
 reference monitor, 742, 839
 referenced pointer, 1009
 REG, 243
 RegCreateKeyEx(), 987
 RegDeleteKey(), 987
 regedit, 986
 RegEnumKeyEx(), 987
 region obcinania, 492
 regiony krytyczne, 163
 RegisterClass(), 490
 registry, 985
 RegOpenKeyEx(), 987
 RegQueryValueEx(), 987
 regular files, 323
 reguła *, 746
 reguła integralności *, 748
 ReiserFS, 352, 353
 rejestr podtrzymujący, 466
 rejestr systemu Windows, 984
 BCD, 986
 COMPONENTS, 986
 DEFAULT, 986
 edycja, 986
 gałęzie, 985, 986
 HARDWARE, 986
 klucze, 986
 NTUSER.DAT, 986
 obiekty COM, 985
 przeglądanie, 986
 regedit, 986
 SAM, 986
 SECURITY, 985, 986
 SOFTWARE, 986
 SYSTEM, 985, 986
 wartości, 986
 Win32 API, 987
 rejestr zapisów dziennika, 944
 rejestrowanie dźwięku, 611

rejestry, 51, 885
 bazowy, 233
 limitu, 233
 rekord MBR, 337
 relacja pomiędzy adresami
 wirtualnymi a adresami
 pamięci fizycznej, 244
 relacje zaufania, 1135
 relative path, 927
 ReleaseDC(), 492
 ReleaseMutex(), 1033, 1035
 ReleaseSemaphore(), 1033, 1035
 relokacja „w locie”, 282
 Remote Procedure Calls, 658, 973,
 1003, 1032
 remove right, 743
 RemoveDirectory(), 98, 99
 rename(), 328, 336
 rendez-vous, 191
 reparse points, 983, 1081
 replikacja, 662
 ResetEvent(), 1034
 restore, 369
 return to libc attacks, 782
 rewinddir(), 935
 rezerwowanie przepustowości, 1069
 RGB, 563
 right, 731
 RISC, 49, 252, 438, 968
 Ritchie Dennis, 853, 1144
 RLE, 570
 rlogin, 684
 rm, 374, 864, 869, 871, 873
 rmdir, 871, 873
 rmdir(), 89, 94, 935
 RMS, 575, 580, 581, 611
 robak Morrisa, 806
 robaki, 806
 robot network, 787
 robota od wewnątrz, 772
 Rock Ridge, 385
 role, 735
 ROM, 57, 228, 229
 root, 955
 rootkit, 813
 aplikacje, 815
 biblioteki, 814
 charakterystyki czasowe
 systemów, 816
 firmware, 814
 jądro, 814
 maszyna wirtualna, 814
 niebieska pigułka, 814
 Sony, 817
 wykrywanie, 815
 round robin, 892, 1041
 router, 545, 688
 routing kanalikowy, 650
 routing tables, 806
 rozdzielczość ekranu, 487
 rozgałęźnik-wampir, 686
 rozkład organowy plików
 na serwerze wideo, 600
 rozliczanie czasu procesora, 469
 rozliczenia, 886
 rozmiar bloku, 358

niezależność od urządzenia, 432
 rozmiar kłastrów, 389
 rozmiar strony, 277
 rozmiar tablicy FAT, 389
 rozmieszczenie plików
 multimedialnych, 591
 algorytm organowy, 599
 buforowanie, 595
 duże bloki, 595
 farma dysków, 600
 jitter, 591
 macierze RAID, 600
 paskowanie losowe, 601
 paskowanie naprzemienne, 601
 paskowanie według bloku, 602
 pliki na wielu dyskach, 600
 popularność treści, 598
 przechowywanie filmów
 w nieciągłych blokach, 593
 przeplatanie wideo, audio
 i tekstu w pojedynczym
 ciągłym pliku z filmem, 592
 read, 591
 seek, 591
 skorowidz bloków, 594
 skorowidz ramek, 594
 stały blok danych, 596
 stały odcinek czasu, 596
 strategię organizacji plików, 592
 szerokie paskowanie, 602
 ten sam wzorzec paskowania dla
 wszystkich plików, 601
 umieszczanie pliku
 na pojedynczym dysku, 591
 wąskie paskowanie, 603
 wideo niemal na życzenie, 596
 wiele plików na jednym dysku, 598
 wybór liczby dysków
 do paskowania, 602
 zarządzanie pamięcią masową, 594
 rozpoznawanie tęczówki, 770
 rozproszona współdzielona pamięć,
 285, 660
 fałszywe współdzielenie, 663
 replikacja, 662
 spójność sekwencyjna, 664
 rozproszony algorytm heurystyczny
 inicjowany przez nadawcę, 667
 analityczny model
 kolejkowania, 667
 rozproszony algorytm heurystyczny
 inicjowany przez odbiorcę, 668
 rozproszony system operacyjny, 49
 rozproszony system w sieci
 rozległej, 619
 rozszerzanie uprawnień, 786
 rozszerzenia nazwy pliku, 320
 rozszerzony system plików, 936
 rozwidlanie powłoki, 899
 rozwidlanie procesu, 880
 równoważenie obciążenia, 666
 algorytm alokacji procesorów, 666
 algorytm deterministyczny
 według teorii grafów, 666

rozproszony algorytm
 heurystyczny inicjowany
 przez nadawcę, 667
 rozproszony algorytm
 heurystyczny inicjowany
 przez odbiorcę, 668
 RPC, 658, 973, 1003, 1032
 etapy wykonywania zdalnego
 wywołania, 659
 problemy implementacyjne, 659
 procedura pośrednicząca
 klienta, 658
 procedura pośrednicząca
 serwera, 658
 przekazywanie wskaźników, 659
 przetaczanie, 658
 słaba kontrola typów, 660
 typy parametrów, 660
 RSA, 726
 RSX-11M, 968
 rundy, 606
 Run-Length Encoding, 570
 runqueue, 893, 895
 Russinovich Mark, 817
 r-węzły, 950
 rysowanie, 492, 493

S

SACL, 1097
 Safe-TCL, 840
 Saltzer Jerry, 1190
 sandboxes, 837
 sandboxing, 741, 837, 1162
 Santa Cruz Operation, 856
 SATA, 402, 435, 559, 1067
 saturation, 564
 scandisk, 372
 scan-EDF, 609
 scheduler, 193, 875
 scheduler activations, 155
 schematy ukrywania wirusów, 825
 SCO UNIX, 852
 script kiddies, 761
 SCSI, 64, 65, 402
 SCSI Ultra -640, 559
 SCSI-2, 445
 Seattle Computer Products, 46
 SECAM, 559, 560, 563
 second system effect, 1192
 secret-key cryptography, 725
 SectionHandle(), 977
 Secure Digital, 1130
 Secure Hash Algorithm, 727
 Secure Virtual Machine, 672
 SecureOS, 740
 security, 719
 security by obscurity, 724
 Security ID, 1095
 security reference monitor, 1003
 Security(), 1012
 seek, 435
 operacje zachodzące na sobie, 435
 seek(), 325, 328, 401

- segmentacja, 294, 297
 - adres liniowy, 304
 - brama wywołania, 306
 - deskryptor segmentu kodu, 303
 - GDT, 302
 - globalna tablica deskryptorów, 302
 - implementacja klasycznej
 - segmentacji, 298
 - katalog stron, 305
 - klasyczna, 298
 - LDT, 302
 - lokalna tablica deskryptorów, 302
 - odzworowanie adresu liniowego
 - na fizyczny, 305
 - Pentium, 302
 - pierścienie zabezpieczeń, 306
 - segmenty, 295
 - selektor, 303
 - stronicowanie, 298
 - szachownicowanie, 298
 - współdzielenie procedur lub
 - danych, 296
 - zewnętrzna fragmentacja, 298
- segmenty, 295
 - dane, 92, 115, 236, 899
 - stos, 92
 - tekst, 92, 115, 899
- sekcje, 977, 982
- sekcje krytyczne, 164, 1034
- sektor CD-ROM, 444
- sektory dysku, 452
- sekwencje sterujące, 480
- self-map page directory, 1037
- semafony, 174, 548
 - binarne, 176
 - down, 174
 - problem producent –
 - konsument, 175
 - synchronizacja, 176
 - up, 174
 - Windows Vista, 1033
 - zabezpieczenia zasobów, 520
- semantyka sesji, 699
- semantyka współdzielenia plików, 698
- send(), 188, 189, 654
- sensory, 70
- SEquentiel Couleur Avec Memoire, 560
- Serial ATA, 435
- Series 3, 1109
- Series 3c, 1109
- server stub, 658
- Services For UNIX, 974
- serwer, 68, 105, 142
- serwer bezstanowy, 949
- serwer NFS, 946
- serwer plików, 696
- serwer reinkarnacji, 105
- serwer ściągania, 584
- serwer wideo, 557
- serwer WWW, 108, 123, 140, 473
 - obsługa wątków, 140
- serwer wypychania, 584
- serwer X, 481
- servis WWW, 68
- sesje, 448
- Set attributes, 328
- Set Top Box, 557
- set_global(), 160
- SetCurrentDirectory(), 98, 99
- SetEvent(), 1034
- SetFilePointer(), 98, 99
- SETGID, 386, 732, 956
- setgid(), 956, 957
- SetPriorityClass(), 1035, 1040
- SetSecurityDescriptorDacl(), 1098
- SetSecurityDescriptorGroup(), 1098
- SetSecurityDescriptorOwner(), 1098
- SetThreadPriority(), 1035, 1040
- SETUID, 374, 386, 732, 742, 776, 779, 782, 792, 955, 957, 1030
- setuid(), 956, 957
- Sfill, 498
- SFU, 974
- sh, 80
- SHA, 728
- SHA-1, 727, 831
- SHA-256, 727, 831
- SHA-512, 727
- shadow copy, 414
- shared locks, 930
- shared text segments, 902
- sharing_flags, 890
- shell, 867
- shell scripts, 870
- shellcode, 782
- shims, 1038
- short, 111
- Shortest Seek First, 457
- Shugart Associates, 45
- siatka, 648, 707
 - Globus Toolkit, 708
 - współdzielenie cykli procesora, 708
- SID, 1095, 1099
- side-by-side, 1021
- side-channel, 769
- sieciowy system operacyjny, 49
- sieciowy system plików, 945
- sieć, 685
 - adres IP, 692
 - akumulacja nagłówków
 - pakietów, 692
 - ARPANET, 115, 687, 919
 - botnet, 787
 - Ethernet, 686
 - firewall, 820
 - gniazda, 918
 - Internet, 115, 687
 - IP, 691
 - jakość usług, 689
 - komunikacja, 692
 - LAN, 545, 685
 - Linux, 917
 - lokalna, 685
 - most, 687
 - NAT, 821
 - nazwy DNS, 693
 - pakiet potwierdzający, 689
 - połączenie ze zdalnym hostem, 693
 - porty, 759
 - protokoły, 689, 691
 - przełączany Ethernet, 686
 - przełączniki, 687
 - przesyłanie pakietów, 686
 - router, 688
 - rozległa, 685
 - stos protokołów, 691
 - strona WWW, 693
 - skeletonowa, 688
 - TCP, 691, 692
 - tłumaczenie nazw, 693
 - transfer plików, 689
 - usługi, 689
 - usługi bezpołączeniowe, 689
 - usługi datagramów, 690
 - usługi datagramów
 - z potwierdzeniami, 690
 - usługi połączeniowe, 689
 - usługi żądanie-odpowiedź, 690
 - WAN, 685
 - WWW, 115, 693
 - zakleszczenie zasobów, 546
- sieć blokująca, 625
- sieć nieblokująca, 622
- sieć omega, 624, 625
- sieć sensorowa, 70
- SIGABRT, 879
- sigaction(), 880, 883
- SIGALRM, 879, 883
- SIGFPE, 879, 889
- SIGHUP, 879
- SIGILL, 879
- SIGINT, 478
- SIGKILL, 879
- signature block, 728
- signing, 1134
- sigpending(), 880
- SIGPIPE, 879
- sigprocmask(), 880
- SIGQUIT, 478, 879
- sigreturn(), 880
- SIGSEGV, 879
- sigsuspend(), 880
- SIGTERM, 879
- SIGUSR1, 879
- SIGUSR2, 879
- silnik analityczny, 36
- silnik aplikacji, 1109
- silnik mutacji, 826
- Simony Charles, 490
- single indirect block, 392, 942
- Single Large Expensive Disk, 438
- skaner antywirusowy, 823
 - plik kozła ofiarnego, 823
 - schematy ukrywania wirusów, 825
- silnik mutacji, 826
- skanowanie, 823
- weryfikatory integralności, 828
- weryfikatory zachowań, 828
- wykrywanie infekcji, 824
- skanowanie portów, 759, 776
- skbuff, 924
- skeleton, 700
- skierowany graf acykliczny, 347
- skrypciarze, 761
- skrypty powłoki, 870
- skryty pocztowe, 1031

- skrzynka pocztowa, 191
- skwantyzowane współczynniki
 - DCT, 570
- SL, 386
- slab allocator, 909
- slave, 632
- SLED, 438
- sleep(), 171, 172
- sleep_avg, 894
- słowo stanu programu, 51
- Small Computer System Interface, 65
- smart cards, 766
- smartfony, 1107
- smartphone, 1107
- SMP, 632
- SMS, 1117
- smss.exe, 974, 975, 1048
- snooping, 629
- socket, 918, 1121
- sockets, 1032
- soft page fault, 1047, 1057
- Solaris, 31
- Solid State Disk, 77
- sort, 80, 81, 128, 868, 871, 873, 878, 934
- source code viruses, 802
- sól, 762
- SPARC, 51, 110, 252, 990
- sparse files, 1086
- specjalne pliki blokowe, 916
- specjalne pliki znakowe, 917
- spin lock, 167, 636, 895, 991
- spooling, 42, 433
- sposoby obliczania aktualnej
 - godziny, 469
- sposób powstawania przerwań, 411
- spójność architekturealna, 1150
- spójność sekwencyjna, 664, 698
- spójność systemu plików, 371
- sprawcy, 734
- sprawdzanie błędów, 1176
- sprawdzanie spójności systemu
 - plików, 372
- sprawdziwe szeregowanie, 210
- sprzęt, 29, 33, 34, 50
 - DMA, 63
 - dyski magnetyczne, 58
 - dyski twarde, 58
 - obsługa zegara, 466
 - pamięć, 54
 - plyta główna, 66
 - plyta macierzysta, 66
 - procesor, 50
 - uruchamianie komputera, 66
 - urządzenia wejścia-wyjścia, 59
 - wejście-wyjście, 400
 - wielokomputery, 647
 - wieloprocessorowy, 620
- spyware, 809
- square-wave mode, 467
- SSD, 77
- SSF, 456, 457
- ssh, 684
- stabilna pamięć masowa, 462
 - analiza wpływu awarii na operacje
 - stabilnego zapisu, 464
- ECC, 463
- nieulotna pamięć RAM, 465
- odtworzenie po awarii, 464
- stabilny odczyt, 464
- stabilny zapis, 463
- stabilny zapis, 463
- Stallman Richard, 862
- stały blok danych, 596
- stały odcinek czasu, 596
- stan uśpienia, 504
- stan wywołania systemowego, 885
- standard error, 868
- standard input, 868
- standard JPEG, 568
- standard MPEG, 571
- standard output, 868
- standard POSIX, 866
- standard UNIX, 857
- standardowe wejście, 868
- standardowe wyjście, 868
- standardowe wyjście błędów, 868
- standby list, 1049
- standby mode, 1078
- stanowy system plików, 952
- stany niebezpieczne, 535
- stany procesów, 131
- START, 478
- starvation, 543, 923
- starzenie, 208
- stat(), 89, 93, 932, 934
- stateful firewalls, 822
- stateless firewall, 821
- static model-based intrusion
 - detection, 834
- statyczna relokacja, 230
- statyczne szeregowanie operacji
 - dyskowych, 606
- STB, 557, 558
- steganografia, 751, 752
- ukryte znaki wodne, 753
- steganography, 752
- sterowanie urządzeniem, 425
- sterowniki filtrów, 1077
- sterowniki filtrów systemu plików, 1005
- sterowniki klasy, 1005
- sterowniki startowe, 1006
- sterowniki urządzeń, 60, 422
 - dostęp do sprzętu urządzenia, 423
 - filtrowanie operacji wejścia-
 - wyjścia, 1004
 - funkcje, 424
 - główny numer urządzenia, 917
 - jednolity interfejs, 427
 - Linux, 921
 - logiczne rozmieszczenie
 - sterowników, 424
 - ładowanie do jądra, 60
 - miniporty, 1005
 - obiekty urządzeń, 1004
 - pomocniczy numer urządzenia, 917
 - protokoły sieciowe, 1005
 - read, 424
 - reentrant, 426
 - sterowanie urządzeniem, 425
 - sterowniki klasy, 1005
 - stos urządzeń, 1004
- struktura, 425
- Symbian OS, 1127
- system plików, 1005
- urządzenia blokowe, 423
- urządzenia znakowe, 423
- WDM, 1072
- weryfikator sterowników, 1073
- Windows Vista, 989, 1004, 1072
- wirusy, 801
- write, 424
- współużywalność, 426
- sterta, 901
- STOP, 478
- stopień wieloprogramowości, 135
- STORE, 619, 626
- store-and-forward packet switching, 649
- stored value cards, 766
- stos, 92, 901
- Stos, 115
- stos jądra, 886
- stos protokołów, 691
- stos urządzeń, 1004, 1073, 1076
- strategia alokacji pamięci, 273
- strategia czyszczenia, 284
- strategia organizacji plików
 - multimedialnych, 592
- strpcpy(), 782, 783
- striping, 439, 595
- strips, 438
- strona WWW, 693
 - hiperłącza, 693
- strona zerowa, 901
- stronicowanie, 243, 297, 1164
 - algorytm odbierania ramek
 - stron, 912
 - algorytm PFRA, 913
 - algorytm wymiany stron, 913
 - błędy braku stron, 287
 - demon stron, 912
 - dokładnie na czas, 1048
 - kswapd, 913
 - Linux, 907, 911
 - LRU, 262, 1055
 - mechanizm zastępowania stron,
 - 913
 - na żądanie, 265, 837, 1002, 1048
 - obszar wymiany, 912
 - pdflush, 915
 - PG_active, 914
 - PG_referenced, 914
 - pliki stronicowania, 912
 - proces init, 913
 - przyspieszenie, 249
 - segmentacja, 298
 - statyczny obszar wymiany, 291
 - strony na liście stron
 - nieaktywnych, 914
 - strony nie do odzyskania, 913
 - strony synchronizowane, 913
 - strony usuwalne, 913
 - strony wymienialne, 913
 - tryb laptopa, 915
 - Windows Vista, 1002
 - wstępne, 266, 267
 - wymienność, 914
 - z wyprzedzeniem, 1054

- strony, 242, 244
 - blokowanie w pamięci, 289
 - discardable, 913
 - magazyn stron, 290
 - nie do odzyskania, 913
 - obsługa błędów braku strony, 287
 - przypinanie, 290
 - rozmiar, 277
 - swappable, 913
 - syncable, 913
 - synchronizowane, 913
 - unreclaimable, 913
 - usuwalne, 913
 - wewnętrzna fragmentacja, 277
 - współdzielone, 279
 - wymienialne, 913
- struktura jądra systemu Linux, 873
- struktura magistral, 32
- struktura napędu dyskowego, 58
- struktura pliku, 321
- struktura systemów operacyjnych, 99
- struktury danych podręcznej pamięci
 - buforowej, 376
- struktury dynamiczne, 1168
- struktury statyczne, 1168
- subjects, 734
- subskrypcja, 704
- suma kontrolna Hamminga, 441
- Sun Microsystems, 49
- superblock, 355, 936, 937
- superblok, 337, 937
- SuperFetch, 1054
- superskalarny procesor, 52
- superuser, 955
- superużytkownik, 74, 760, 955, 957
- supervisor, 988
- supervisor mode, 30
- surowe pliki blokowe, 923
- surowe urządzenia blokowe, 929
- svchost.exe, 1022
- SVID, 857
- SVM, 672
- swap area, 912
- swapper process, 911
- swapping, 229, 235
- switch, 111
- SwitchToFiber(), 1026, 1035
- swobodna kontrola dostępu, 745
- swobodny dostęp do kodu
 - źródłowego, 45
- sygnał alarmowy, 74
- sygnał chrominancji, 564
- sygnał luminancji, 564
- sygnał przerwania, 411
- sygnał zespolony, 563
- sygnały, 74, 878, 882, 885
 - SIGABRT, 879
 - SIGALRM, 879, 883
 - SIGFPE, 879
 - SIGHUP, 879
 - SIGILL, 879
 - SIGKILL, 879
 - SIGPIPE, 879
 - SIGQUIT, 879
 - SIGSEGV, 879
 - SIGTERM, 879
- SIGUSR1, 879
- SIGUSR2, 879
- wątki, 889
- wysyłanie, 883
- Symbian, 103
- Symbian OS, 69, 1107, 1111, 1141
 - adresowanie pamięci, 1124
 - bezpieczeństwo, 1132, 1133, 1135
 - bezpośredni dostęp do
 - pamięci, 1128
 - blokujące operacje wejścia-
 - wyjścia, 1129
 - Bluetooth, 1139
 - brak strony, 1125
 - bufor TLB, 1125
 - DFC, 1118
 - diagram struktury jądra, 1115
 - DMA, 1128
 - dostęp do zasobów w trybie
 - klient-serwer, 1115
 - fizyczny sterownik
 - urządzenia, 1127
 - gniazda, 1121
 - historia systemu, 1108
 - infrastruktura komunikacji,
 - 1136, 1137
 - jądro, 1112
 - katalog stron, 1124
 - kommunikacja, 1117, 1136
 - kommunikacja
 - międzyprocesowa, 1121
 - kommunikacja sieciowa, 1116
 - LDD, 1127
 - logiczny sterownik
 - urządzenia, 1127
 - ładowanie bibliotek DLL, 1123
 - model bezpośredni, 1126
 - model emulatora, 1126
 - model ruchomy, 1126
 - model wielostronny, 1126
 - modułowość infrastruktury
 - kommunikacji, 1140
 - moduły CSY, 1139
 - moduły MTM, 1139
 - moduły PRT, 1139
 - moduły TSY, 1139
 - moduły typów komunikatów, 1139
 - multimedia, 1117
 - nanojądro, 1114
 - nanowątki, 1118
 - nośniki pamięci, 1129
 - nośniki wymienne, 1130
 - obiektość, 1112
 - obiekty aktywne, 1120
 - obsługa pamięci, 1126
 - obsługa systemu plików, 1116
 - ochrona systemu plików, 1132
 - ochrona zasobów, 1116
 - PDD, 1127
 - podpisywanie, 1134
 - połączenie PPP, 1140
 - procesy, 1116, 1119
 - przechowywanie danych, 1130
 - przestrzeń adresowa, 1122
 - relacje zaufania, 1135
 - rozszerzenia jądra, 1128
 - stany nanowątków, 1118
 - sterowniki urządzeń, 1127
 - system plików, 1116, 1131, 1132
 - szybki semafor, 1118
 - TTBR, 1124
 - uchwyt, 1112
 - urządzenia fizyczne, 1138
 - usługi GSM, 1140
 - warstwa DMA, 1128
 - warstwa implementacji
 - protokołów, 1137, 1139
 - warstwa jądra, 1115
 - warstwa sterowników
 - urządzeń, 1138
 - wątki, 1116, 1118
 - wejście-wyjście, 1127
 - wiadomości SMS, 1140
 - wykonywanie na miejscu, 1123
 - wymiana pamięć, 1125
 - zabezpieczenia plików, 1133
 - zarządzanie dostępem
 - do zasobów, 1115
 - zarządzanie pamięcią, 1116, 1121
 - zarządzanie rozmiarem
 - aplikacji, 1123
 - zarządzanie stertą, 1123
- Symbian OS 6, 1110
- Symbian OS 7, 1111
- Symbian OS 8, 1111
- Symbian OS 9, 1111
- symbol zachęty, 80
- symbole wieloznaczne, 735, 868
- symetryczne układy
 - wielordzeniowe, 629
- symetryczny system
 - wieloprocessorowy, 632
- Symmetric MultiProcessor, 632
- symmetric-key cryptography, 725
- symulacja wielu liczników czasu, 470
- sync, 377, 378
- sync(), 915
- synchronizacja, 176
- synchronizacja procesów, 220
- synchronizacja w systemach
 - wieloprocessorowych, 634
 - blokada pętlowa, 636
 - buforowanie, 636
 - muteksy, 634
 - ograniczanie ruchu na
 - magistrali, 637
 - problem rywalizacji
 - o magistralę, 636
 - przełączanie, 638
 - TSL, 634, 635
 - unikanie zatłoczenia pamięci
 - podręcznej, 638
 - wzajemne wykluczanie, 636
 - zapętlanie, 638
- synchronized, 185
- synchronizowane strony, 913
- SYSTEM, 985
- System Access Control List, 1097
- system availability, 720
- system biometryczny, 769
 - analiza podpisów, 771
 - cechy biometryczne, 769
 - długość palców, 770

- identyfikacja, 769
- identyfikacja głosu, 771
- obrazy, 771
- odległość Hamminga, 771
- rejestracja, 769
- rozpoznawanie tęczy, 770
- transformacja falkowa Gabora, 771
- system buforowania, 56
- system czasu rzeczywistego, 70
- dotrzymywanie terminów, 71
- e-cos, 71
- miękki, 71, 211
- szeregowalny system, 211
- szeregowanie, 211
- twardy, 70, 211
- zdarzenia nieokresowe, 211
- zdarzenia okresowe, 211
- system DV, 571
- system GOAL, 24
- System III, 856
- system interaktywny, 863
- system monolityczny, 100, 859
- system multimedialny, 71, 1196
- system operacyjny, 29, 32, 67
 - bezpieczeństwo, 718
 - Chorus, 859
 - CP/M, 46, 966
 - CTSS, 43
 - DOS, 46
 - e-cos, 232
 - egzodajdro, 110
 - EPOC, 1109
 - FMS, 40
 - FreeBSD, 48, 69
 - historia, 36
 - Hydra, 737
 - karty elektroniczne, 71
 - klient, 34
 - komputer mainframe, 67
 - komputer osobisty, 68
 - komputer podręczny, 69
 - Linux, 48, 69, 851, 860
 - Mac OS, 69
 - Mach, 859
 - maszyna wirtualna, 106
 - menedżer zasobów, 34
 - MICA, 968
 - mikrojądro, 102
 - MINIX, 45, 858
 - MINIX 3, 104, 742
 - model klient-serwer, 105
 - MS-DOS, 46, 966
 - MULTICS, 43, 102, 853
 - NT OS/2, 968
 - obiektywny, 1112
 - OS/2, 968
 - OS/360, 41, 106
 - OS/390, 68
 - Palm OS, 69
 - Paramecium, 1162
 - pliki, 75
 - powłoka, 80
 - PRISM/Mica, 968
 - proces, 72
 - programy aplikacyjne, 34
 - przestrzeń adresowa, 72, 74
- QNX, 69
- rozproszony, 49
- RSX-11M, 968
- serwer, 68
- sieciowy, 49
- struktura, 99
- Symbian OS, 69, 1107
- THE, 101, 1158
- TinyOS, 70
- TSS/360, 106
- UNIX, 44, 48, 852, 853, 1108
- VAX/VMS, 48, 968
- VAXELAN, 968
- VM/370, 107
- VMS, 968, 988
- VxWorks, 69
- warstwa abstrakcji, 33
- wbudowany, 69
- wejście-wyście, 79
- węzły sensorowe, 70
- wieloprosesorowość, 68
- Windows, 47
- Windows 2000, 48, 970
- Windows 2003, 971
- Windows 3.0, 967, 969
- Windows 95, 47, 967
- Windows 98, 47, 48, 967
- Windows Me, 48, 967
- Windows NT, 47, 968
- Windows NT 4.0, 969
- Windows Server 2008, 965
- Windows Vista, 48, 69, 965, 971
- Windows XP, 48, 970
- wirtualizacja, 108
- XENIX, 47
- z/VM, 106
- zabezpieczenia, 79
- zarządzanie energią, 501
- zasoby, 35
- system operacyjny-gospodarz, 671
- system operacyjny-gość, 671
- parawirtualizacja, 677
- system plików, 31, 76, 317, 319, 394
 - /proc, 945
 - algorytm alokacji bazującej
 - na jednokierunkowej liście
 - z wykorzystaniem tabeli
 - w pamięci, 341
 - alokacja bazująca na liście
 - jednokierunkowej, 340
 - alokacja bazująca na liście
 - jednokierunkowej
 - z wykorzystaniem tabeli
 - alokacji plików, 342
 - Apple FAT, 387
 - atrybuty plików, 325
 - badania, 394
 - Berkeley Fast File System, 1185
 - bit SETUID, 374
 - blok identyfikacyjny, 355
 - blok startowy, 337
 - blokowanie, 930
 - blokowe pliki specjalne, 323
 - brakujący blok, 373
 - buforowanie, 375
 - ciągła alokacja, 338
 - czytanie bloków zawczasu, 378
 - DAG, 347
 - defragmentacja dysków, 380
 - deskryptor woluminu, 383
 - deskryptory plików, 330
 - dostęp do plików, 325
 - dowiązania, 347, 928
 - dowiązania symboliczne, 336, 348
 - dowiązania twarde, 336
 - dziennik, 944
 - EFS, 1093
 - ekonomiczność wykorzystania
 - miejsca na dysku, 360
 - ext2, 353, 926, 936, 937
 - ext3, 352, 936, 943
 - FAT, 342, 387, 389
 - FAT-12, 389
 - FAT-16, 320, 389, 1079
 - FAT-32, 320, 389, 390, 1079
 - FFS2, 1131
 - flagi, 326
 - foldery, 331
 - fragmentacja, 339
 - główny rekord startowy, 337
 - główny system plików, 77
 - High Sierra, 446
 - idempotentna operacja, 353
 - implementacja, 337
 - implementacja katalogów, 344
 - implementacja plików, 338
 - interfejs VFS, 354
 - i-numer, 94
 - ISO 9660, 382
 - i-węzły, 343, 374
 - JFFS, 1131
 - Joliet, 386
 - katalogi, 323, 331, 927
 - kontrola wersji, 394
 - kopie zapasowe, 365
 - kopiowanie plików, 329
 - księga, 944
 - księgujący, 352, 943
 - LFS, 350
 - liczba magiczna, 323
 - limity dyskowe, 349, 364
 - Linux, 874, 925
 - lista wolnych bloków, 362
 - luki, 371
 - mapy bitowe, 362, 363
 - MBR, 337
 - mechanizm księgowania, 353
 - minimalizacja ruchu ramienia
 - dysku, 379
 - montowanie, 77, 95
 - MS-DOS, 387
 - multimedia, 561
 - multimedialny, 584
 - naprawa błędów, 374
 - nazwy plików, 319, 926
 - NFS, 945
 - niepoddzielna transakcja, 353
 - NTFS, 320, 352, 353, 1079
 - obsługa długich nazw plików, 346
 - obszary, 340
 - operacje na katalogach, 335
 - operacje na plikach, 327

- system plików
 - pamięć podręczna, 375
 - pamięć podręczna
 - z natychmiastowym zapisem, 377
 - partycje, 337
 - peer-to-peer, 394
 - pliki, 926
 - pliki ASCII, 323
 - pliki binarne, 323
 - pliki specjalne, 78
 - pliki współdzielone, 347
 - plyty CD-ROM, 381
 - pola czasowe, 327
 - potok, 78
 - przechowywanie informacji
 - o wolnych blokach, 361
 - ReiserFS, 352, 353
 - Rock Ridge, 385
 - rozmiar bloku, 358
 - rozmiar klastrów, 389
 - rozmiar pliku, 327
 - spójność, 371, 373
 - sprawdzanie bloków, 374
 - sprawdzanie katalogów, 373, 374
 - sprawdzanie spójności, 372
 - stanowy, 952
 - stany, 373
 - sterowniki urządzeń, 1005
 - struktura dziennika, 349
 - struktura pliku, 321
 - superblok, 355
 - superblok, 337
 - ścieżki bezwzględne, 333
 - śledzenie historii, 394
 - śledzenie wolnych bloków, 361
 - technika zarządzania wolnym miejscem na dysku, 362
 - tworzenie dowiązania, 348
 - typy plików, 323
 - UDF, 340
 - układ, 337
 - urządzenia mobilne, 1131
 - usuwanie usterek, 929
 - V7, 391
 - VFS, 354, 936
 - v-węzeł, 355
 - wątek sprzątacza, 351
 - węzeł indeksowy, 343
 - Windows Vista, 1079
 - wirtualny, 354, 926, 936
 - własny plik wykonywalny, 324
 - wolne bloki, 362
 - wydajność, 375
 - wyszukiwanie informacji, 394
 - XFS, 945
 - YAFFS, 1131
 - zabezpieczenia, 79, 326, 374
 - zapis małych fragmentów, 350
 - zarządzanie miejscem na dysku, 357
 - zarządzanie pamięcią
 - podręczną, 375
 - zdalny, 355
 - zdublowany blok danych, 373
 - zdublowany blok na liście wolnych bloków, 373
 - znakowe pliki specjalne, 323
 - zwykle pliki, 323
- system przekazywania komunikatów, 189
- system rozproszony, 619, 683, 710
 - CORBA, 700
 - Ethernet, 686
 - hierarchia katalogów, 695
 - internet, 687
 - jakość usług, 689
 - Jini, 705
 - katalogi, 695
 - Linda, 702
 - metody, 700
 - middleware, 685
 - model transferu, 695
 - model wgrywanie-ściąganie, 695
 - model zdalnego dostępu, 695
 - namieszka procedury, 700
 - niezależność od lokalizacji, 697
 - obiekty, 700
 - obiekty ORB, 700
 - oprogramowanie
 - pośredniczące, 685
 - pakiet potwierdzający, 689
 - protokoły, 689, 691
 - przezroczystość nazewnictwa, 697
 - semantyka sesji, 699
 - semantyka współdzielenia plików, 698
 - serwer plików, 696
 - siatka, 707
 - sieć komputerowa, 685
 - spójność sekwencyjna, 698
 - sprzęt sieciowy, 685
 - strona WWW, 693
 - system katalogów, 695
 - transfer plików, 689
 - usługi, 689
 - usługi bezpołączeniowe, 689
 - warstwa middleware bazująca na dokumentach, 693
 - warstwa middleware bazująca na koordynacji, 701
 - warstwa middleware bazująca na obiektach, 700
 - warstwa middleware bazująca na systemie plików, 694
 - współdzielenie cykli procesora, 708
 - WWW, 693
 - wydajność, 699
- system RSA, 726
- system scentralizowany, 497
- System V, 31, 44, 852, 856
- System V Interface Definition, 857
- System V Release 3, 857
- system w układach scalonych, 629
- system warstwowy, 101
- system wbudowany, 30, 69, 1107, 1197
- system wejścia-wyjścia, 510
- system wielokomputerowy, 683, 709, 710, 1196
- system wieloprocesorowy, 124, 617, 683, 710
- adres fizyczny, 627
- architektura magistrali, 621
- badania, 708
- bez pamięci podręcznej, 621
- binarne wykładnicze cofanie, 637
- buforowanie, 636
- CC-NUMA, 626
- CMP, 629
- doskonałe tasowanie, 624
- katalogowy system
 - wieloprocesorowy, 626
- każdy procesor ma własny
 - system operacyjny, 630
- master-slave, 631
- MMU, 627
- NC-NUMA, 626
- NUMA, 620, 626
- ograniczanie ruchu na
 - magistrali, 637
- otwarty punkt krzyżowy, 623
- pamięci z przeplotem, 625
- problem rywalizacji
 - o magistralę, 636
- protokół koherencji pamięci
 - podręcznych, 622
- przelaczanie, 638
- przełączniki krzyżowe, 622
- punkt krzyżowy, 622
- rdzenie, 629
- sieć blokująca, 625
- sieć nieblokująca, 622
- sieć omega, 624
- SMP, 632
- sprzęt wieloprocesorowy, 620
- symetryczny system, 632
- synchronizacja, 634
- system w układach scalonych, 629
- system z przeplotem, 625
- szeregowanie, 640
- szeregowanie zespołów, 645
- szpiegowanie, 629
- TSL, 634
- typy systemów, 630
- układy wielordzeniowe, 628
- UMA, 620
- UMA z architekturą magistrali, 621
- UMA z przełącznikami
 - krzyżowymi, 622
- UMA z wykorzystaniem
 - wielostopniowych przełączników krzyżowych, 623
- unikanie zatłoczenia pamięci
 - podręcznej, 638
- wąskie gardła procesora, 632
- wielokomputer, 646
- wielostopniowa sieć
 - przełączania, 624
- wielostopniowe przełączniki
 - krzyżowe, 623
- wirtualizacja, 669
- współdzielona pamięć, 619, 620
- z pamięcią podręczną, 621
- z pamięcią podręczną
 - i pamięciami prywatnymi, 621

zakleszczenie, 634
 zamknięty punkt krzyżowy, 623
 zapętlanie, 638
 system wieloprogramowy, 42, 870
 system wielowarstwowy, 1158
 system wsadowy, 37, 38, 201, 852
 system wykrywania włamań, 822, 833
 atak poprzez upodobnienie, 835
 przynęty, 835
 wykrywanie włamań z użyciem
 modeli statycznych, 834
 system X Window, 481
 system X11, 49
 system z podziałem czasu, 748
 system zarządzania pamięcią, 292
 system zaufany, 740
 izolowanie programów, 741
 zaufana baza obliczeniowa, 742
 system(), 784
 System/360, 40
 systemowa lista kontroli dostępu, 1097
 sytuacja wyścigu, 163, 172
 szablony, 703
 szachownicowanie, 298
 szeregowny system czasu
 rzeczywistego, 211
 szeregowanie, 193, 220, 709
 bez wywłaszczania, 197
 bilety, 209
 cele algorytmów, 198
 chwila, 893
 cykliczne, 204, 892
 czas cyklu przetwarzania, 200
 czas odpowiedzi, 200
 deficytowy zasób, 194
 dwupoziomowe, 642
 gwarantowane, 208
 interaktywni użytkownicy, 198
 kategorie algorytmów, 197
 kwant, 204
 Linux, 892
 loteryjne, 209
 minimalizacja czasu
 odpowiedzi, 200
 najpierw najkrótsze zadanie, 202
 następny najkrótszy proces, 208
 następny proces o najkrótszym
 pozostałym czasie
 działania, 203
 nonpreemptive, 197
 oddzielanie strategii
 od mechanizmu, 212
 pierwszy zgłoszony, pierwszy
 obsłużony, 201
 preemptive, 197
 priorytety, 205
 priorytety dynamiczne, 894
 procesy homogeniczne, 577
 procesy obliczeniowe, 894
 procesy zorientowane
 na obliczenia, 195
 procesy zorientowane na
 wejście-wyjście, 195
 proporcjonalność, 200
 przelaczanie kontekstu, 204
 przelaczanie procesów, 204

przepustowość, 199
 równoległe, 645
 runqueue, 893
 sprawiedliwe, 210
 starzenie, 208
 stosowanie, 196
 system czasu rzeczywistego,
 198, 210
 system interaktywny, 203
 system wielokomputerowy, 665
 system wsadowy, 198, 201
 w czasie rzeczywistym, 578
 wątki, 212
 wątki czasu rzeczywistego, 892
 według powinowactwa, 642
 według priorytetów, 205
 wielokrotne kolejki, 206
 Windows Vista, 994, 1038
 współdzielenie przestrzeni, 643
 z uwzględnieniem
 powinowactwa, 895
 z wywłaszczaniem, 197
 zachowanie procesów, 195
 zadania wsadowe, 644
 zadania z różnymi
 priorytetami, 894
 zespoły, 645
 szeregowanie monotoniczne
 w częstotliwości, 580
 szeregowanie operacji dyskowych
 w systemach multimedialnych, 606
 algorytm scan-EDF, 609
 dynamiczne szeregowanie, 608
 płynny przepływ danych
 do klientów, 607
 rundy, 606
 statyczne szeregowanie, 606
 szeregowanie procesów
 multimedialnych, 577
 algorytm czasu rzeczywistego, 580
 EDF, 581
 procesy homogeniczne, 577
 RMS, 580, 581
 szeregowanie monotoniczne
 w częstotliwości, 580
 szeregowanie w czasie
 rzeczywistym, 578
 szeregowanie w systemach
 wieloprocessorowych, 640
 algorytm najpierw najkrótsze
 zadanie, 644
 algorytm szeregowania
 dwupoziomowego, 642
 inteligentne szeregowanie, 642
 komunikacja pomiędzy dwoma
 wątkami, 645
 podział czasu, 641
 szeregowanie równoległe, 645
 szeregowanie według
 powinowactwa, 642
 szeregowanie zespołów, 645
 wątki jądra, 640
 wątki użytkownika, 640
 współdzielenie miejsca, 643
 współdzielenie przestrzeni, 643

szeregowanie żądań dostępu
 do dysku, 456
 szerokie paskowanie, 602
 sześcienn, 648
 szpiegowstwo gospodarcze, 722
 szpiegowanie, 629
 szum kwantyzacji, 566
 szybki semafor, 1118
 szybkie dopasowanie, 241
 szybkości przesyłania danych, 559
 szyfrogram, 724
 szyfrowanie, 724
 szyfrowanie plików, 1092

Ś

ściągnięcie, 584, 611
 ścieżki, 58, 76
 bezwzględne, 333, 927
 względne, 927
 ścieżki CD-ROM, 448
 ścisła naprzemienność, 166
 ściśle związane systemy, 620
 śledzenie pamięci, 238
 śledzenie wolnych bloków, 361
 środowisko bezpieczeństwa, 719
 środowisko uruchomieniowe
 wspólnego języka, 974
 środowisko wielodostępne, 748
 środowisko X, 484

T

tabela i-węzłów, 94, 940
 tabela procesów, 73, 133
 tabela stron, 246, 247
 odwrócona, 255
 pamięć o dużej objętości, 253
 wielopoziomowa, 253
 wpisy, 247
 tabela wątków, 150, 153
 tablica deskryptorów plików, 885
 tablica deskryptorów stron, 905
 tablica FAT, 389
 tablica importowanych adresów, 1020
 tablica mieszająca, 1183
 tablica opisów otwartych plików, 942
 tablica partycji, 337
 tablica przewidywania skoków, 675
 tablica routingu, 806
 tablica skrótów, 256
 tablica stron-cień, 678
 tablice programowe, 37
 tabulacja, 476
 tagged architecture, 737
 tail, 871, 873
 takty zegara, 467
 task_struct, 884, 886, 909
 taskmgr.exe, 1022
 taśma DV, 586
 taśmy magnetyczne, 59
 TCB, 742
 tgetattr(), 920
 TCP, 691, 692, 758, 874, 919, 1157
 TCP/IP, 857, 1116

TCPIP.PRT, 1139
 tcsetattr(), 920
 TEB, 1023, 1037
 technika plug and play, 65
 technika spoolingu, 433
 technika stałego bloku danych, 596
 technika stałego odcinka czasu, 596
 technika zarządzania wolnym miejscem na dysku, 362
 techniki antyantywirusowe, 823
 techniki antywirusowe, 823
 techniki biometryczne, 769, 770, 844
 Technische Hogeschool Eindhoven, 101
 technologia DVD, 451
 technologia SuperFetch, 1054
 tekst jawny, 724
 tekst zaszyfrowany, 724
 telefony cyfrowe, 71
 telefony komórkowe, 1107
 telewizja kablowa, 556, 557
 telnet, 684, 758, 759
 temperatura, 506
 termcap, 480
 terminal, 473, 920
 baza danych, 480
 TTY, 874
 X, 497
 zarządzanie, 920
 TerminateProcess(), 130
 TEST, 405
 Test and Set Lock, 169, 635
 test POST, 896
 TextOut(), 492
 teza koniec-koniec, 1160
 TF, 386
 THE, 101, 1158
 THINC, 498
 polecenia, 498
 Thompson Ken, 44, 853
 thrashing, 266
 Thread Environment Block, 1023, 1037
 thread_info, 886
 thread_yield(), 151, 178
 TID, 892
 time bomb, 773
 time critical, 1040
 time(), 96
 timeouts, 545
 timers, 426, 466
 timesharing, 42
 timesharing system, 748
 tint, 564
 TinyOS, 70
 TLB, 250, 286, 306, 675, 1052, 1125
 tłumaczenie binarne, 674
 tłumaczenie nazw, 693
 token, 983
 token dostępu, 1095
 TOPS-20, 324
 Torvalds Linus, 45, 860
 TPM, 729
 tr, 872, 873
 trackball, 478
 trafienie pamięci podręcznej, 55
 trajektorie zasobów, 534
 transakcje, 944
 transfer plików, 689

transfery
 asynchroniczne, 416
 DMA, 408
 synchroniczne, 416
 transformacja falkowa Gabora, 771
 transformata Fouriera, 574
 Translation Lookaside Buffer, 250, 1053
 Translation Table Base Register, 1124
 Transmission Control Protocol, 692, 919
 tranzystor, 37
 TRAP, 52, 88, 102
 trap and emulate, 672
 trap door, 773
 triple indirect block, 392
 TrueType, 495
 Trusted Computing, 730
 Trusted Computing Base, 742
 Trusted Platform Module, 729
 trusted systems, 742
 trwałe obiekty, 979
 trwałe pliki, 84
 tryb fali prostokątnej, 467
 tryb jądra, 29, 32
 tryb kanoniczny, 475
 tryb nadzorcy, 30
 tryb niekanoniczny, 475
 tryb offline, 39
 tryb przelotu, 409
 tryb surowy, 475
 tryb ugotowany, 475
 tryb użytkownika, 29, 30
 tryb wiązki, 409
 tryb wirtualnego jądra, 672
 tryb wstrzymania, 1078
 TSL, 169, 175, 634, 635
 TSS/360, 106
 TSY, 1139
 TTBR, 1124
 TTY, 874
 tty_struct, 924
 twarde błędy braku stron, 1057
 twarde chybienie, 253
 twardy system czasu rzeczywistego, 70, 211
 twierdzenie Nyquista, 565
 two-phase locking, 543
 tworzenie
 dowiązanie, 348
 gniazdo, 918
 katalog, 94
 pliki Makefile, 114
 potok, 932, 934
 proces, 98, 126, 877, 886, 1036
 semafor, 1033
 wątek, 146, 889
 włókno, 1026
 tyłne drzwi, 773, 787
 typy danych, 111
 typy plików, 323

U

UAC, 1100
 uchwyt, 130, 584, 976, 1010, 1112
 kontekst urzędzenia, 492
 plik, 947
 UDF, 340

UDP, 919
 UID, 74, 446, 732, 735, 736, 836, 953
 układ czasowy, 426
 układ DMA, 63
 układ mostka PCI, 64
 układ systemu plików, 337
 układy scalone, 40
 układy scalone wielkiej skali integracji, 45
 układy superskalarne, 412
 układy wieloprocessorowe, 708
 układy wielordzeniowe, 628, 1193
 ukryte kanały, 748, 749
 blokowanie pliku, 750
 model Lampsona, 748
 modulowanie użycia procesora, 750
 odblokowanie pliku, 750
 pozyskiwanie dedykowanych zasobów, 751
 problem zamknięcia, 749
 proces kolaboranta, 749
 steganografia, 751
 zwalnianie dedykowanych zasobów, 751
 ukrywanie informacji, 752
 ukrywanie sprzętu, 1171
 UltraSPARC, 1172
 UMA, 620
 architektura magistrali, 621
 przełączniki krzyżowe, 622
 wielostopniowe przełączniki krzyżowe, 623
 umask, 891
 UMDf, 1073
 umount(), 89, 96
 unbuffered I/O, 1065
 Unicode, 978
 UNICS, 852, 853
 Uniform Memory Access, 620
 Uniform Resource Locator, 693
 unikanie wirusów, 829
 unikanie wykrycia wirusa, 805
 unikanie zakleszczeń, 534
 algorytm bankiera dla pojedynczego zasobu, 537
 algorytm bankiera dla wielu zasobów, 538
 stany bezpieczne i niebezpieczne, 535
 trajektorie zasobów, 534
 Universal Coordinated Time, 467
 Universal Disk Format, 340
 Universal Serial Bus, 65
 UNIX, 31, 44, 48, 206, 852, 853, 854, 855, 1108
 hasła, 761
 UNIX to UNIX Copy Program, 760
 UNIX Version 1, 860
 UNIX Version 6, 856
 UNIX Version 7, 856
 unlink(), 89, 336, 935
 unmap(), 903, 904
 UnmapViewOfFile(), 1051
 up, 174
 upcall, 156

- uprawnienia, 717, 718, 731, 736
 - architektura oznaczona, 737
 - Kopiowanie obiektu, 739
 - Kopiowanie uprawnień, 739
 - lista uprawnień, 737
 - ochrona listy uprawnień, 737
 - prawa ogólne, 739
 - przekazywanie, 738
 - Usunięcie uprawnień, 739
 - utrzymywanie listy uprawnień
 - w przestrzeni
 - użytkownika, 738
 - utrzymywanie listy uprawnień
 - w ramach systemu
 - operacyjnego, 737
 - wycofywanie uprawnień, 739
 - Zniszczenie obiektu, 739
- uprzywilejowana kontrola dostępu, 1094
- URL, 693, 694, 1165
- uruchamianie
 - komputer, 66
 - proces w tle, 870
 - starsze aplikacje, 670
 - urządzenie wejścia-wyjścia, 62
 - wiele programów w systemach
 - bez abstrakcji pamięci, 229
- uruchamianie systemu Linux, 896
- automatyczna konfiguracja
 - systemu, 897
- BIOS, 896
- boot, 896
- demon stron, 897
- dynamiczne ładowanie
 - sterowników, 897
- getty, 898
- główny rekord startowy, 896
- GRUB, 896
- init, 897
- kod startowy jądra, 896
- LILO, 896
- login, 898
- MBR, 896
- program startowy, 896
- sekwencja procesów, 898
- test POST, 896
- ustawienia terminali, 898
- uruchamianie systemu Windows
 - Vista, 1006
 - bezpieczne uruchamianie, 1007
 - BIOS, 1006
 - BootMgr, 1006
 - konsola odzyskiwania, 1007
 - ostatnia znana dobra
 - konfiguracja, 1007
 - programy startowe, 1006
 - sterowniki startowe, 1006
 - WinLoad.exe, 1006
 - WinResume.exe, 1006
- urządzenia, 60
 - blokowe, 400, 423, 874
 - flash, 1025
 - JBD, 944
 - Jini, 705
 - klasyczne, 66
 - mobilne, 1108
 - PDA, 1109
 - plug and play, 66
 - podręczne, 1108
 - sieciowe, 874, 924
 - sterowniki, 60, 423
 - wirtualne, 681
 - wskazujące, 486
 - znakowe, 400, 401, 423, 874, 924
- urządzenia wejścia-wyjścia, 59, 399,
 - 400, 416
 - asynchroniczność, 416
 - buforowanie, 416
 - dyski, 435
 - klasyfikacja, 401
 - kontroler dysku, 60
 - sterownik, 60
 - szybkości przesyłania danych, 401
 - transfery asynchroniczne, 416
 - transfery blokujące, 416
 - transfery sterowane
 - przerwaniami, 416
 - transfery synchroniczne, 416
 - urządzenia blokowe, 400
 - urządzenia znakowe, 400, 401
 - zakresy szybkości, 401
- urządzenia wejścia-wyjścia
 - odwzorowane w pamięci, 403
 - magistrala pamięci, 406
 - port wejścia-wyjścia, 403
 - przestrzeń adresowa, 406
 - przestrzeń portów wejścia-wyjścia, 403
 - selektywne wyłączanie
 - buforowania, 406
- USB, 64, 65, 402, 1067
- USENET, 434, 859
- USENIX, 116
- User Account Control, 1100
- User Datagram Protocol, 919
- User ID, 953
- User Identification, 74
- user shared data, 1024
- User-Mode Driver Framework, 1073
- usługi bezpołączeniowe, 689
- usługi datagramów, 690
- usługi datagramów
 - z potwierdzeniami, 690
- usługi połączeniowe, 689
- usługi żądanie-odpowiedź, 690
- usuwanie
 - katalogi, 871
 - pliki, 871
- usuwanie zakleszczenia, 526, 532
- wycofywanie operacji, 533
- wyłączanie, 532
- zabijanie procesów, 533
- uśpione wątki, 1118
- UTC, 467
- utrwalanie danych, 509
- utrzymanie integralności systemu
 - plików, 377
- uucp, 760
- uwierzytelnianie, 189, 753, 844
 - hasła, 755
 - hasła jednorazowe, 763
 - informacje identyfikujące, 754
- karty chipowe, 766
- karty inteligentne, 766
- karty z paskiem
 - magnetycznym, 766
- logowanie, 755
- metoda wyzwanie-odpowiedź, 764
- obiekt fizyczny, 765
- techniki biometryczne, 769
- uwięzienie, 546
- UXGA, 564
- uznaniowa kontrola dostępu, 745
- użytkownicy, 718

V

- V, 174
- V7, 391
 - atrybuty, 392
 - blok dwupośredni, 392
 - blok jednapośredni, 392
 - blok trójpśredni, 392
 - bloki dyskowe, 392
 - i-węzły, 392, 393
 - katalog główny, 392
 - katalogi, 391
 - otwieranie plików, 392
 - ścieżki względne, 393
 - śledzenie bloków dyskowych, 392
 - wyszukiwanie pliku, 393
- VAD, 1053
- ValidDataLength, 1064
- vampire tap, 686
- VAX, 241, 856, 988
- VAX/VMS, 48, 968
- VAXELAN, 968
- VCR, 585
- verifier.exe, 1073
- VFS, 354, 874, 926, 936
 - blok identyfikacyjny, 355
 - dodawanie systemów plików, 356
 - implementacja, 355
 - katalog, 355
 - v-węzeł, 355, 356
- VGA, 564, 565
- vi, 857, 874
- Videotex, 920
- views, 982, 1064
- Virtual Address Descriptor, 1053
- Virtual File System, 354, 874, 926
- Virtual Machine Interface, 677
- VirtualAlloc(), 1051
- VirtualFree(), 1051
- Virtualization Technology, 672
- VirtualLock(), 1051
- VirtualProtect(), 1051
- VirtualQuery(), 1051
- VirtualUnlock(), 1051
- VM/370, 107
- vm_area_struct, 910, 911
- vmalloc, 910
- VMI, 677
- VMI Linux, 677
- VMIL, 677
- VMS, 968, 969, 988

VMware, 673, 677
 VMware Workstation, 109
 VoIP, 510
 Volume Shadow Copies, 1067
 Volume Table of Contents, 448
 VT, 672
 VT102, 486
 VTOC, 448
 v-węzeł, 355, 949
 VxWorks, 69

W

wait3(), 998
 WaitForMultipleObjects(), 998,
 1033, 1035
 WaitForSingleObject(), 98, 99, 1033,
 1034, 1035
 waitpid(), 89, 90, 880, 881, 882
 waitqueue, 895
 wakeup(), 171, 172
 WAN, 685, 946
 war dialer, 757
 warstwa abstrakcji, 33
 warstwa abstrakcji sprzętowej, 989, 990
 warstwa jądra systemu
 Symbian OS, 1115
 warstwa middleware
 bazująca na dokumentach, 693
 bazująca na koordynacji, 701
 bazująca na obiektach, 700
 bazująca na systemie plików, 694
 warstwy, 101
 warstwy systemu wejścia-wyjścia,
 420, 434
 wartość nice, 893
 warunek braku wyłączenia, 541
 warunek cyklicznego oczekiwania, 542
 warunek wstrzymania i oczekiwania, 541
 warunek wzajemnego wykluczania, 540
 warunki powstawania zakleszczeń
 zasobów, 522
 watchdog timers, 468, 470
 wąskie paskowanie, 603
 wątek sterowania, 137
 wątek stron zerowych, 1062
 wątek wyskakujący, 657
 wątek zapisujący strony
 odzwzorowane, 1062
 wątek zapisujący strony
 zmodyfikowane, 1062
 wątki, 137, 143
 aktywacja zarządcy, 155
 algorytm szeregujący, 151, 894
 alokacja pamięci, 160
 blok TEB, 1023
 blokowanie, 153
 chwila, 893
 clone(), 889
 CONTEXT, 1028
 degradowanie, 894
 dyspozytor, 140
 działanie bez przerwy, 152
 flagi uprawnień, 891
 identyfikator, 148, 1027

implementacja hybrydowa, 154
 implementacja wątków
 w jądrze, 153
 implementacja wątków
 w przestrzeni
 użytkownika, 150
 klasyczny model wątków, 143
 konflikty pomiędzy wątkami, 159
 kończenie działania, 146
 licznik programu, 143
 Linux, 884, 887
 lokalne zablokowanie, 151
 mechanizm aktywacji zarządcy, 155
 nanowątki, 1118
 naśladowanie funkcji wątków
 jądra, 155
 niszczenie na poziomie jądra, 153
 obsługa, 147
 obsługa sygnałów, 889
 opakowanie, 152
 operacje wejścia-wyjścia
 na plikach, 889
 osłona, 152
 pamięć lokalna, 1024
 POSIX, 147
 prace badawcze, 219
 pracownik, 140
 priorytety dynamiczne, 894
 problem braku stron
 w pamięci, 152
 promowanie, 894
 prywatne zmienne globalne, 159
 przetwarzanie dużych ilości
 danych, 142
 przydatność, 138
 przystosowywanie kodu
 jednowątkowego
 do obsługi wielu wątków, 158
 Pthreads, 147, 148, 149
 rezygnacja z procesora, 147
 runqueue, 893
 serwer WWW, 140
 sharing_flags, 890
 stos, 146
 sygnały, 160, 889
 Symbian OS, 1118
 szeregowanie, 212
 szeregowanie cykliczne, 892
 szeregowanie z uwzględnieniem
 powinowactwa, 895
 szeregowanie zadań, 1027
 tabela wątków, 150, 153
 tablica priorytetów, 893
 tworzenie, 146, 889, 1027
 tworzenie na poziomie jądra, 153
 wartość nice, 893
 wezwanie, 156
 wielowątkowość, 144
 Windows Vista, 1023, 1027
 włókna, 1026
 współdzielenie zasobów, 891
 wydajność, 137, 151
 wywołania systemowe, 151
 zarządzanie stosem, 161

zarządzanie w przestrzeni
 użytkownika, 150
 zastosowanie, 142
 zmienne globalne, 159
 zwielokrotnianie wątków
 użytkownika na bazie
 wątków jądra, 155
 wątki jądra, 154, 889, 1162
 pamięć wirtualna, 890
 rodzic, 891
 sharing_flags, 890
 współdzieleniu katalogu
 głównego, 891
 wątki pop-up, 157
 tworzenie, 157
 wbudowane systemy operacyjne, 69
 wc, 870
 wchodzenie do regionu krytycznego, 170
 wcześnie kojarzenie, 1167
 WDF, 1073, 1077
 WDM, 1072, 1077, 1102
 wejście niebuforowane, 429
 wejście-wyjście, 32, 79, 399, 510
 badania, 509
 błędy, 431
 DMA, 420
 Linux, 916, 921
 raportowanie błędów, 431
 realizacja, 417
 Symbian OS, 1127
 Windows Vista, 1066
 wirtualizacja, 679
 wywołania systemowe, 919
 wejście-wyjście odzwzorowane
 w pamięci, 404
 wejście-wyjście sterowane
 przerwaniami, 419
 wektor dostępnych zasobów, 529
 wektor istniejącego zasobu, 529
 wektor przerwań, 62, 133, 411
 weryfikacja integralności, 828
 weryfikacja zachowań, 828
 weryfikator aplikacji, 1015
 weryfikator integralności, 828
 weryfikator sterowników, 1073
 weryfikator zachowań, 828
 wewnętrzna fragmentacja, 277
 wezwanie, 156
 węzeł indeksowy, 343
 węzły czujników, 1198
 węzły sensorowe, 70
 wgłębienia, 443
 while, 111, 870
 white-hat hackers, 755
 wiadomości SMS, 1140
 wiązanie wątków, 1023
 Wide Area Networks, 685
 wideo, 555
 algorytmy kompresji, 567
 dekodowanie, 567
 jasność, 564
 kodowanie, 562, 567
 kolory, 563, 564
 kompresja, 567
 MPEG, 568, 571

- nasycenie, 564
- odcień, 564
- omiatanie obrazu, 571
- piksele, 564
- płynność obrazu, 564
- pola, 563
- powiększenie, 571
- progresywne, 563
- przeplatanie, 563
- ramka, 562
- RGB, 563
- sygnał chrominancji, 564
- sygnał luminancji, 564
- sygnał zespolony, 563
- sygnały cyfrowe, 564
- system DV, 571
- współczynnik kształtu, 564
- wzorzec skanowania, 563
- wideo na żądanie, 557, 588
 - ADSL, 558
 - gwarantowane pasmo, 558
 - przystawka STB, 558
 - serwer wideo, 557
 - sieć dystrybucji, 558
 - STB, 557
 - telewizja kablowa, 557
- wideo niemal na życzenie, 587
 - buforowanie, 597
 - funkcje magnetowidu, 589
 - optymalne rozmieszczenie ramek, 597
 - podwójny bufor, 597
 - rozmieszczenie plików, 596
 - wielkość bufora, 597
- wideo RAM, 487
- widzety, 483
- wielka blokada jądra, 895
- wielka przestrzeń adresowa, 1194
- wielkie projekty, 1186
- wielobieżność, 1175
- wielokomputer, 647, 683, 709, 1196
 - aktywne komunikaty, 657
 - algorytm alokacji procesorów, 666
 - blokująca operacja send, 656
 - COWS, 647
 - czterowymiarowy
 - hipersześcian, 648
 - DSM, 660
 - hipersześcian, 649
 - implementacja RPC, 659
 - interfejsy sieciowe, 650
 - kanały DMA, 651
 - karty interfejsów sieciowych, 651
 - komputer klastrowy, 647
 - komunikacja o wysokiej wydajności, 651
 - komunikacja węzła z interfejsem sieciowym, 653
 - kopiowanie bufora przy okazji zapisu, 656
 - kopiowanie przy zapisie, 656
 - nadmierne kopiowanie pakietów, 651
 - nieblokująca operacja send
 - połączona z kopiowaniem, 656
 - nieblokująca operacja send z przerwaniem, 656
 - niskopoziomowe
 - oprogramowanie komunikacyjne, 651
 - odbieranie, 654
 - oprogramowanie komunikacyjne
 - poziomu użytkownika, 654
 - pakiety, 649
 - pięściń, 648
 - podwójny torus, 648
 - pojedynczy przełącznik, 648
 - połączenia, 647
 - procesory sieci, 651
 - przekazywanie komunikatów, 619
 - przekazywanie pakietów do karty interfejsu, 653
 - przełączanie obwodów, 650
 - przełączanie pakietów typu przechowaj i prześlij, 649
 - przesyłanie komunikatów, 619
 - przetaczanie, 658
 - routing kanalikowy, 650
 - rozproszona współdzielona pamięć, 660
 - równoważenie obciążenia, 666
 - RPC, 658
 - siatka, 648
 - sprzęt, 647
 - szeregowanie, 665
 - sześcian, 648
 - średnica, 648
 - technologia wewnętrznych połączeń, 647
 - topologia gwiazdy, 648
 - umiejscowienie kart interfejsów sieciowych, 651
 - wątek wyskakujący, 657
 - węzły, 647
 - wysyłanie, 654
 - wywołania asynchroniczne, 655
 - wywołania blokujące, 654
 - wywołania nieblokujące, 655
 - wywołania synchroniczne, 654
 - zdalne wywołania procedur, 657
- wielokrotne kolejki, 206
- wielokrotne użycie kodu, 1174
- wielopoziomowe tabele stron, 253
- wielopoziomowy system
 - bezpieczeństwa, 746
- wieloprocessorowe systemy
 - operacyjne, 68
- wieloprocessorowość, 124, 617
- wieloprogramowość, 40, 41, 83, 124, 135, 144, 193
- wielosesyjne płyty CD-ROM, 448
- wielostopniowa sieć przełączania, 624
- wielostopniowe przełączniki
 - krzyżowe, 623
- wielowarstwowy system
 - operacyjny, 1158
- wielowątkowość, 53, 144, 146
- wielowątkowy serwer WWW, 140
- wiersz pamięci podręcznej, 621
- wiersz poleceń, 34, 867
- wildcard, 735, 868
- WIMP, 486, 511, 1151
- Win32, 969, 973
- Win32 API, 97, 970, 973, 980
 - bezpieczeństwo, 983
 - dostęp do plików, 984
 - dziennik, 983
 - GUI, 984
 - listy kontroli dostępu, 983
 - littery napędów, 984
 - niskopoziomowy model wejścia-wyjścia, 983
 - odzworowania plików, 982
 - operacje na graficznym interfejsie użytkownika, 984
 - perspektywy, 982
 - pliki, 983
 - przestrzeń nazw, 984
 - punkty przyłączania, 983
 - rejestr systemu Windows, 987
 - ścieżki do plików, 981
 - środowisko wykonawcze, 981
 - token, 983
 - uchwyty, 981
 - WOW32, 981
 - WOW64, 981
 - wywołania, 980
 - zarządzanie danymi, 983
- win32k.sys, 984, 1004, 1006
- WINAPI, 489, 491
- winda, 457
- winda Linusa, 923
- Windows, 29, 34, 47, 967
- Windows 2000, 31, 48, 966, 970
- Windows 2003, 971
- Windows 3.0, 966, 967, 969
- Windows 3.1, 966
- Windows 95, 31, 47, 966, 967
- Windows 98, 31, 47, 48, 966, 967
- Windows Driver Foundation, 1073, 1077
- Windows Driver Kit, 988
- Windows Driver Model, 1072, 1077
- Windows Explorer, 81
- Windows Me, 31, 48, 966, 967
- Windows na bazie MS-DOS-a, 965, 967
- Windows na bazie NT, 965, 967
- Windows NT, 31, 47, 968
- Windows NT 3.1, 966
- Windows NT 4.0, 48, 966, 969
- Windows Server 2008, 965, 971
- Windows Update, 282
- Windows Vista, 31, 48, 69, 965, 966, 971, 1102
 - abstrakcje HAL, 991
 - ACE, 1096
 - ACL, 1008
 - ACPI, 991
 - algorytm szeregowania, 1040
 - alokacja przestrzeni dyskowej, 1086
 - ALPC, 1003, 1022
 - APC, 989, 996
 - aplikacje multimedialne, 1069
 - asynchroniczne wywołania procedur, 996
 - audyty bezpieczeństwa, 1094
 - bezpieczeństwo, 1093

Windows Vista

- bezpieczne logowanie, 1094
- biblioteki DLL, 1019, 1020
- biblioteki klas bazowych, 974
- bit NX, 1100
- BitLocker, 1093
- blok PEB, 1023
- blok środowiska wątku, 1037
- blok TEB, 1023
- blokada pętlowe, 991
- blokowanie dostępu do pliku, 1071
- BSOD, 1001
- chronione procesy, 1101
- Common Criteria, 1003
- CONTEXT, 1028
- csrss.exe, 975
- cykl życia obiektu, 1012
- DACL, 1095, 1099
- dane współdzielone
 - użytkownika, 1024
- definiowanie obiektów, 1012
- deskryptor bezpieczeństwa, 976, 1096
- dispatcher_header, 998
- DLL, 973
- dostęp do urządzeń wejścia-wyjścia, 992
- dowiązania symboliczne, 1019
- DRM, 989, 1022, 1101
- dyski dynamiczne, 1067
- EFS, 1093
- filtrowanie operacji
 - wejścia-wyjścia, 1004
- framework KMDF, 1073
- framework UMDF, 1073
- grupowanie procesów, 1025
- grupowanie zasobów, 1023
- GUI, 973
- HAL, 989, 1102
- hibernacja, 1078
- idealny procesor, 1041
- identyfikacja urządzeń, 992, 1067
- identyfikator SID poziomu
 - integralności, 1099
- identyfikator wątku, 1027
- implementacja
 - bezpieczeństwa, 1098
- implementacja menedżera
 - obiektów, 1007
- implementacja procesów, 1036
- implementacja sterowników
 - urządzeń
 - wejścia-wyjścia, 1001
- implementacja systemu wejścia-wyjścia, 1072
- implementacja wątków, 1036
- implementacja zarządzania
 - pamięcią, 1052
- informacje o katalogach, 1071
- interfejsy programowania, 973
- interfejsy trybu użytkownika, 973
- Internet Explorer, 1099
- IPC, 978
- ISR, 995
- jądro, 975, 988, 990, 993
- katalog stron
 - samoodwzorowania, 1037
- katalogi obiektów, 1019
- katalogi przestrzeni nazw
 - obiektów, 1014
- kategorie typów obiektów trybu
 - jądra, 976
- klasy priorytetów, 1040
- klawiatura, 996
- kod graficznego interfejsu
 - użytkownika, 1004
- kolejki, 1018
- komponenty wejścia-wyjścia, 990
- kompresja plików, 1090
- komunikacja międzyprocesowa, 978, 1031
- kontrola konta użytkownika, 1100
- kopie woluminów sporządzane
 - w tle, 1067
- LCP, 1003
- liczniki, 993
- liczniki czasowe, 1018
- lista uprawnień, 1095
- logowanie, 1096, 1099
- lokalne wywołania procedur, 975
- LPC, 975
- magistrale, 1067
- MDL, 1075
- mechanizm dostosowawczy, 1038
- mechanizm przekazywania
 - uprawnień, 1096
- mechanizm szeregujący, 1038, 1039
- mechanizmy synchronizujące, 991
- menedżer konfiguracji, 1003
- menedżer obiektów, 978, 1000, 1007
- menedżer pamięci, 1002
- menedżer pamięci
 - podręcznej, 1002
- menedżer procesów, 1002
- menedżer transakcji jądra
 - z obsługą transakcji
 - koordynowanych, 986
- menedżer wejścia-wyjścia, 990, 1001, 1067
- menedżer zadań, 1022
- menedżer zasilania, 1077
- migawki, 1068
- monitor kontroli bezpieczeństwa
 - odwołań, 1003
- MS-DOS, 966
- muteksy, 1033
- nadzorca, 988
- nagłówki obiektu, 1009
- naśladowanie, 1096
- nazwane obiekty, 979
- nazwy trwale, 979
- niebieski ekran śmierci, 1001
- nieobowiązkowa kontrola
 - dostępu, 1094
- NTFS, 1017, 1079, 1103
- NTOS, 975, 988, 990
- obiekt sterownika, 979, 1067
- obiekt urządzenia systemu
 - plików, 1017
- obiekty, 976, 1008
- obiekty dyspozytora, 994, 998
- obiekty kontrolne, 994
- obiekty powiadomienia, 999
- obiekty synchronizacji, 999
- obiekty trybu jądra, 976, 978
- obiekty urządzeń, 979, 1004, 1015
- obiekty warstwy wykonawczej
 - zarządzane przez menedżer
 - obiektów, 1018
- ochrona przestrzeni
 - adresowej, 1094
- odwracanie priorytetów, 1044
- odwzorowania plików, 982
- ograniczenia w dostępie
 - do przedziałów bajtów
 - w plikach na poziomie
 - systemu plików, 1071
- operacje wejścia-wyjścia, 1066, 1102
- operacje wejścia-wyjścia
 - z priorytetami, 1068
- opóźnione wywołania
 - procedur, 995
- osobowości, 973, 974
- pakiet IRP, 1016
- pakiet żądania wejścia-wyjścia, 1016, 1071, 1074
- pamięć fizyczna, 1009
- pamięć lokalna wątku, 1024
- pamięć podręczna, 1002, 1063
- pamięć przydzielana
 - obiektom, 1009
- pamięć transakcyjna, 1025
- pamięć wirtualna, 990, 1002, 1102
- platformy sprzętowe, 999
- plug and play, 1001
- podnoszenie priorytetów, 1043
- podsystemy, 973, 1019
- porty wykonania operacji
 - wejścia-wyjścia, 1072
- porządkowanie listy wolnych
 - uchwyty, 1015
- PowerShell, 986
- poziom integralności, 1097
- poziomy stosu urządzeń, 1074
- priorytet bieżący wątku, 1040
- priorytety bazowe wątków, 1040
- priorytety przerwań, 995
- priorytety wątków, 1042
- priorytety wątków Win32, 1041
- procedury obsługi przerwań, 995
- proces systemowy, 1028
- procesy, 1023, 1024, 1102
- procesy chronione, 1101
- program trybu jądra, 973
- protokoły sieciowe, 1005
- przekazywanie uprawnień, 1096
- przenoszenie systemu na
 - platformy sprzętowe, 991
- przerwania, 992
- przestrzeń nazw NT, 978
- przestrzeń nazw obiektów, 1011
- rdzenny interfejs programowania
 - aplikacji, 975
- ReadyBoost, 1065
- rejestr, 984, 1003

- rezerwowanie
 - przepustowości, 1069
- RPC, 973
- sekcje, 977, 1102
- sekcje krytyczne, 1034
- semafory, 1033
- SFU, 974
- SID, 1095
- skalowalność algorytmów
 - szeregujących, 1041
- skrytki pocztowe, 1031
- sterowniki filtrów, 1077
- sterowniki urządzeń, 989, 1001, 1004, 1072
- sterowniki urządzeń trybu jądra, 1077
- stos urządzeń, 1004, 1073, 1076
- stronicowanie na żądanie, 1002
- struktura obiektów, 1008
- struktura systemu, 987, 988
- struktura wirtualnej przestrzeni adresowej, 1046
- synchronizacja, 1033
- system plików, 1005, 1017, 1079, 1103
- systemowa lista kontroli dostępu, 1097
- szeregowanie, 1038
- szeregowanie wątków jądra, 994
- szeregowanie zadań procesora, 1027
- szyfrowanie plików, 1092
- środowisko wykonawcze, 981, 988
- token dostępu, 1018, 1095
- trwale obiekty, 979
- tryb DMA, 993
- tryb jądra, 989, 1102
- tryb wstrzymania, 1078
- tworzenie migawki, 1068
- tworzenie procesów, 1029, 1036
- tworzenie wątków, 1027
- UAC, 1100
- uchwyt, 976, 1010
- uprawnienia, 1095
- uprawnienia przeglądarki IE, 1099
- uprzywilejowana kontrola dostępu, 1094
- uruchamianie systemu, 1006
- usługi, 974
- usługi trybu użytkownika, 1019
- uwierzytelnianie użytkowników, 1099
- warstwa abstrakcji sprzętowej, 989, 990
- warstwa jądra, 993, 994
- warstwa programowania, 973
- warstwa sprzętowa, 995
- warstwa wykonawcza, 999, 1102
- warstwy trybu jądra, 988
- wątki, 1023, 1027, 1102
- wątki czasu rzeczywistego, 996
- wątki systemowe stron zerowych, 1043
- WDF, 1073
- WDM, 1072, 1102
- weryfikator aplikacji, 1015
- weryfikator sterowników, 1073
- wiązanie wątków, 1023
- wielkość liter, 978
- Win32 API, 973, 974, 980
- winlogon, 1099
- wirtualna przestrzeń adresowa, 1045
- wirtualne adresy jądra, 1009
- włókna, 1025, 1026
- WOW32, 981
- WOW64, 981
- wskaźnik będący przedmiotem odwołań, 1009
- wykrywanie problemów w aplikacjach, 1015
- wywołania APC trybu jądra, 997
- wywołania APC trybu użytkownika, 997
- wywołania API operacji wejścia-wyjścia, 1069
- wywołania API związane z bezpieczeństwem, 1097
- wywołania API związane z zarządzaniem zadaniami, 1029
- wywołania DPC, 996
- względny priorytet wątku, 1040
- zadania, 1025
- zarządzanie limitami, 1000
- zarządzanie pamięcią, 1000, 1045
- zarządzanie pamięcią fizyczną, 1060
- zarządzanie pamięcią podręczną, 1002
- zarządzanie procesami, 1029
- zarządzanie procesorem, 1023
- zarządzanie wątkami, 1029
- zarządzanie włóknami, 1029
- zarządzanie zadaniami, 1029
- zarządzanie zasilaniem, 1002, 1078
- zastrzeżony token, 1025
- zdarzenia, 1034
- zegary, 993
- zgodność z podsystemem POSIX, 1031
- Windows XP, 31, 48, 966, 970
- windows.h, 489
- Windows-on-Windows, 981
- WinFS, 972
- WinLoad.exe, 1006
- winlogon, 1099
- WinResume.exe, 1006
- wirtualizacja, 108, 669, 1193
 - API, 675
 - DMA, 680
 - domena 0, 681
 - dyski, 680
 - hipernadzorca, 669
 - hipernadzorca typu 1, 671, 672
 - hipernadzorca typu 2, 671, 673
 - host, 669
 - instrukcje uprzywilejowane, 671
 - licencje, 682
 - maszyna wirtualna, 671
 - maszyna wirtualna na procesorach wielordzeniowych, 681
 - migracja maszyn wirtualnych, 670
 - monitor maszyn wirtualnych, 671
 - obsługa w procesorach, 672
 - pamięć, 678
 - paravirt ops, 677
 - parawirtualizacja, 675
 - podstawowy blok, 674
 - polecenia dyskowe IDE, 680
 - problemy licencji, 682
 - procesor wielordzeniowy, 681
 - przechwyc i emuluj, 672, 675
 - pułapki, 675
 - sprawdzanie maszyn wirtualnych, 670
 - SVM, 672
 - system operacyjny-gospodarz, 671
 - system operacyjny-gość, 671
 - tablica przewidywania skoków, 675
 - tablica stron-cień, 678
 - tłumaczenie binarne, 674
 - tryb wirtualnego jądra, 672
 - uruchamianie programu binarnego, 674
 - uruchamianie starszych aplikacji, 670
 - urządzenia wirtualne, 681
 - VMI, 677
 - VMIL, 677
 - VMware, 673
 - VT, 672, 674
 - wejście-wyjście, 679
 - wirtualny dysk, 674
 - wrażliwe instrukcje, 671
 - wymagania, 671
 - wywołanie funkcji hipernadzorczy, 675
 - zarządzanie pamięcią, 678
 - wirtualna geometria dysku, 437
 - wirtualna maszyna Javy, 71, 840
 - wirtualna przestrzeń adresowa, 243, 910
 - wirtualny dysk, 674
 - wirtualny i-węzeł, 949
 - wirtualny system plików, 354, 874, 926, 936
 - abstrakcje systemu plików, 937
 - dentry, 936, 937
 - file, 937
 - i-node, 936, 937
 - superblock, 936, 937
- wirusy, 793, 844
 - formaty programów wykonywalnych, 798
 - kod źródłowy, 802
 - makra, 801
 - MBR, 799
 - nadpisujące wirusy, 795
 - ograniczanie ryzyka identyfikacji, 805
 - pasożytnicze wirusy, 797
 - poczta elektroniczna, 804
 - polimorficzne wirusy, 826
 - program zarażający, 793
 - programy wykonywalne, 794
 - rezydujące w pamięci, 798
 - rozprzestrzenianie się, 803
 - schematy ukrywania, 825

- wirusy
 - sektor startowy, 799
 - skaner antywirusowy, 823
 - sterowniki urządzeń, 801
 - towarzyszące wirusy, 794
 - unikanie wykrycia, 805
 - właściwy kod, 793
 - wnękowe wirusy, 798
 - zasada działania, 793
- włamanie do systemu, 756
- włókna, 1025, 1026
 - szeregowanie, 1026
- WM_CREATE, 491
- WM_DESTROY, 491
- WM_PAINT, 491
- WNDCLASS, 490
- WndProc(), 491
- wojny rdzeniowe, 829
- woluminy logiczne, 382
- woluminy NTFS, 1081
- Word, 801
- working directory, 927
- World Wide Web, 693
- worm, 806
- wormhole routing, 650
- WOW, 981
- WOW16, 1036
- WOW32, 981
- WOW64, 981, 1036
- wprowadzanie danych, 473
 - GUI, 487
- wrapper, 152
- wrażliwe instrukcje, 671
- WRITE, 624
- write(), 89, 93, 328, 331, 433, 916, 932, 933
- WRITE_PORT_LONG(), 992
- WRITE_PORT_UCHAR(), 992
- WRITE_PORT_USHORT(), 992
- write-through caches, 377
- WSClock, 269, 272, 308
- wskazówki, 1185
- wskaźnik, 111
- wskaźnik będący przedmiotem odwołań, 1009
- wskaźnik stosu, 51
- wspólna pamięć, 162
- współczynnik błędów braku strony, 275
- współczynnik kształtu, 564
- współdzielenie cykli procesora, 708
- współdzielenie plików, 698
- współdzielenie procesora, 224
- współdzielenie przestrzeni, 643
- współdzielenie zasobów, 35, 497
- współdzielona pamięć, 708
- współdzielone biblioteki, 281
- współdzielone segmenty tekstu, 902
- współdzielony hosting, 108
- wstępna alokacja, 938
- wstrzykiwanie kodu, 784, 785
- wtrącanie do więzienia, 832
- WWW, 115, 693
 - mechanizm działania, 693
- wycyfywanie operacji, 533
- wycyfywanie uprawnień, 739
- wydajność, 1177
- wydajność systemu plików, 375
 - algorytm LRU, 376
 - buforowanie, 375
 - czytanie bloków zawczasu, 378
 - minimalizacja ruchu ramienia dysku, 379
 - pamięć podręczna
 - z natychmiastowym zapisem, 377
 - rozmieszczenie i-węzłów, 379
 - strategia buforowania, 378
 - struktury danych podręcznej pamięci buforowej, 376
 - utrzymanie integralności systemu plików, 377
- wyjście, 61, 479
- wykładnicze cofanie binarne, 687
- wykonywanie na miejscu, 1123
- wykonywanie obok siebie, 1021
- wykorzystanie kanału ubocznego, 769
- wykorzystanie procesora, 135
- wykrywanie programów narażonych na błędy przepełnienia bufora, 779
- wykrywanie rootkitów, 815
- wykrywanie włamań
 - modele, 833
 - modele statyczne, 834
- wykrywanie zakleszczenia, 526
 - przetwarzanie rozproszone, 549
 - zakleszczenie dla przypadku wielu zasobów każdego typu, 529
 - zakleszczenie dynamiczne, 549
 - zakleszczenie z jednym zasobem każdego typu, 527
- wyłączanie przerw, 165
- wymiana pamięci, 229, 235
- wymienialne strony, 913
- wymuszona kontrola dostępu, 745
- wypychanie, 584, 611
- wysokopoziomowe formatowanie dysku, 455
- wysokowydajne dyski flash, 509
- wysyłanie sygnałów, 883
- wyszukiwanie pliku, 940
- wyścig, 162, 163, 930
- wyświetlacz, 502
- wywłaszczanie, 532
- wywołania
 - ALPC, 1032
 - asynchroniczne, 655
 - blokujące, 654
 - GDI, 492
 - nieblokujące, 655
 - obsługa systemu plików, 328
 - synchroniczne, 654
- wywołania systemowe, 52, 85, 96, 472
 - access(), 956
 - alarm(), 880, 883
 - bezpieczeństwo, 956
 - cfgetospeed(), 920
 - cfgetispeed(), 920
 - cfsetispeed(), 920
 - cfsetospeed(), 920
 - chdir(), 935
 - chmod(), 956
 - chown(), 956
 - clone(), 889
 - close(), 932
 - closedir(), 935
 - creat(), 931, 932
 - CreateFileMapping(), 1051
 - execve(), 880
 - exit(), 880
 - fcntl(), 932
 - fork(), 128, 880, 886
 - fstat(), 932
 - getegid(), 956
 - geteuid(), 956
 - getgid(), 956
 - getuid(), 956
 - ioctl(), 920
 - kill(), 880, 883
 - link(), 935
 - Linux, 879
 - lseek(), 932, 933
 - MapViewOfFile(), 1051
 - mkdir(), 935
 - mmap(), 903
 - nice(), 893
 - NtCreateProcess(), 975
 - open(), 932
 - opendir(), 935
 - OpenFileMapping(), 1051
 - pause(), 880, 884
 - pipe(), 932, 934
 - POSIX, 88, 90
 - read(), 86, 932, 933
 - readdir(), 935
 - rewinddir(), 935
 - rmdir(), 935
 - setgid(), 956
 - setuid(), 956
 - sigaction(), 880, 883
 - sigpending(), 880
 - sigprocmask(), 880
 - sigreturn(), 880
 - sigsuspend(), 880
 - stat(), 932, 934
 - system plików, 931
 - tcgetattr(), 920
 - tcsetattr(), 920
 - unlink(), 935
 - unmap(), 903
 - UnmapViewOfFile(), 1051
 - VirtualAlloc(), 1051
 - VirtualFree(), 1051
 - VirtualLock(), 1051
 - VirtualProtect(), 1051
 - VirtualQuery(), 1051
 - VirtualUnlock(), 1051
 - waitpid(), 880
 - wejście-wyjście, 919
 - Win32 API, 97
 - write(), 932, 933
 - wykonanie, 86
 - zarządzanie katalogami, 89, 94
 - zarządzanie pamięcią, 903, 1051
 - zarządzanie plikami, 89, 93
 - zarządzanie procesami, 89, 90, 879
 - zarządzanie terminalem, 920
 - wywołanie funkcji hipernadzorcy, 675

wyzwanie-odpowiedź, 764
 wzajemne wykluczanie, 164, 168
 wzajemne wykluczanie z
 wykorzystaniem aktywnego
 oczekiwania, 165
 aktywne oczekiwanie, 167
 blokada pętlowa, 167
 blokowanie zmiennych, 166
 opuszczanie regionu
 krytycznego, 170
 rozwiązanie Petersona, 168
 ściśła naprężenie, 166
 TSL, 169
 wchodzenie do regionu
 krytycznego, 170
 wyłączanie przerwań, 165
 XCHG, 171
 względne nazwy ścieżek, 333
 wznawianie instrukcji, 288
 wzorce multimedialnych systemów
 plików, 584
 wzorzec skanowania wideo, 563

X

X, 481, 860, 866
 X Window, 34, 481, 511, 859
 GUI, 482
 Intrinsics, 483
 klient X, 481
 komunikacja pomiędzy klientem
 a serwerem, 483
 komunikaty, 483
 menedżer okien, 483
 Motif, 483
 obsługa zdarzeń, 485
 serwer, 481, 866
 szkielet aplikacji, 485
 warstwy, 483
 widzety, 483
 XCreateGC(), 485
 Xlib, 482, 483
 XOpenDisplay(), 484
 XSelectInput(), 485
 zarządzanie oknami, 483
 zasoby, 484
 X Window System, 49, 860, 866
 X11, 49, 81, 866
 x64, 1055
 x86, 48, 1055
 XCHG, 171, 175
 XCreateGC(), 485
 XDS 940, 207
 Xen, 677
 XENIX, 47, 855
 Xerox, 47
 Xerox PARC, 47, 486
 XFS, 945
 XGA, 488, 564
 Xlib, 482, 483
 XOpenDisplay(), 484
 XOR, 441, 495, 765
 XSelectInput(), 485
 xterm, 876

Y

YAFFS, 1131
 Yet Another Flash Filing System,
 1131
 yocto, 118
 YouTube, 556

Z

z/VM, 106
 Z3, 37
 zabezpieczenia, 79
 zabezpieczenia sprzętowe, 83
 zabezpieczony system
 komputerowy, 753
 zabieranie cykli, 409
 zabijanie procesów, 478, 533
 zadania, 38, 884, 1025
 FMS, 39
 wsadowe, 128
 zadania systemu operacyjnego
 w zakresie stronicowania, 286
 zagłodzenie, 216, 543, 548, 923
 zagrożenia, 720
 atak powrotu do biblioteki libc, 782
 atak przejmowania
 przeglądarki, 813
 atak rozszerzania uprawnień, 786
 atak wstrzykiwania kodu, 784
 atak z wykorzystaniem
 łańcuchów formatujących, 780
 atak z wykorzystaniem
 przepelnień liczb, 783
 błędy w kodzie, 776
 bomba logiczna, 773
 intruzi, 721
 koń trojański, 790
 oprogramowanie szpiegujące, 809
 podszycanie się pod ekran
 logowania, 774
 przepelnienie bufora, 777
 przypadkowa utrata danych, 723
 robaki, 806
 rootkit, 813
 tylne drzwi, 773
 wirusy, 793
 złośliwe oprogramowanie, 786
 zakleszczenie, 182, 517, 521, 549
 algorytm bankiera dla
 pojedynczego zasobu, 537
 algorytm bankiera dla wielu
 zasobów, 538
 algorytm podejmowania decyzji
 w zakresie przydziału
 i zwalniania zasobów, 525
 algorytm strusia, 526
 algorytm wykrywania
 zakleszczeń, 530
 badania, 548
 blokowanie dwufazowe, 543
 dynamiczne, 549
 grafy skierowane, 523
 komunikacyjne, 544
 macierz bieżącej alokacji, 529
 macierz żądań, 529
 modelowanie, 523
 pomiędzy komputerami, 517
 procesy, 517
 przeciwdziałanie
 zakleszczeniom, 540
 przetwarzanie rozproszone, 549
 punkty kontrolne, 533
 stany bezpieczne
 i niebezpieczne, 535
 strategie postępowania, 525
 system wieloprotokółowy, 634
 trajektorie zasobów, 534
 unikanie zakleszczeń, 534
 urządzenia wejścia-wyjścia, 518
 usuwanie, 526, 532
 usuwanie poprzez
 wywłaszczanie, 532
 usuwanie poprzez zabijanie
 procesów, 533
 usuwanie przez wycofywanie
 operacji, 533
 uwięzienie, 546
 warunek braku wywłaszczania,
 522, 541
 warunek cyklicznego
 oczekiwania, 522, 542
 warunek wstrzymania
 i oczekiwania, 522, 541
 warunek wzajemnego
 wykluczania, 522, 540
 warunki powstawania
 zakleszczeń zasobów, 522
 wektor dostępnych zasobów, 529
 wektor istniejącego zasobu, 529
 wycofywanie operacji, 533
 wykrywanie, 526
 wykrywanie zakleszczeń dla
 przypadku wielu zasobów
 każdego typu, 529
 wykrywanie zakleszczeń
 z jednym zasobem każdego
 typu, 527
 wywłaszczanie, 532
 zabijanie procesów, 533
 zagłodzenie, 543, 548
 zasoby, 518, 522
 zasoby bez możliwości
 wywłaszczania, 519
 zasoby w sieci, 546
 zasoby z wywłaszczaniem, 518
 zdobywanie zasobu, 520
 zamknięty punkt krzyżowy, 623
 zamykanie danych w klatce, 1136
 zapis, 32
 pliki, 340
 płyty CD-R, 447
 zapobieganie zakleszczeniom, 548
 zarządzanie bateriami, 506
 zarządzanie energią, 499, 708, 1002
 ACPI, 507
 baterie, 500
 dyski twarde, 503
 hibernacja, 500
 interfejs sterownika zarządzania
 energiją, 507

- zarządzanie energią
 - komunikacja bezprzewodowa, 505
 - notebook, 501
 - odtwarzacz wideo, 508
 - pamięć, 505
 - PowerScope, 508
- problemy po stronie systemu
 - operacyjnego, 501
- problemy sprzętowe, 500
- problemy w programach
 - aplikacyjnych, 507
- procesor, 504
- przeglądarka map, 508
- przyciski sterowania
 - zasilaniem, 500
- rozpoznawanie mowy, 508
- stan uśpienia, 504
- system operacyjny, 501
- Windows Vista, 1078
- wyświetlacz, 502
- zarządzanie bateriami, 506
- zarządzanie temperaturą, 506
- zarządzanie katalogami, 89, 94
- zarządzanie miejscem na dysku, 357
- zarządzanie obciążeniem, 276
- zarządzanie oknami, 483
- zarządzanie operacjami
 - wejścia-wyjścia, 471
- zarządzanie pamięcią, 75, 227, 285
 - algorytm najgorszy pasujący, 240
 - algorytm najlepszy pasujący, 240
 - algorytm następny pasujący, 239
 - algorytm pierwszy pasujący, 239
 - algorytm szybkie dopasowanie, 241
 - badania, 307
 - listy jednokierunkowe, 238
 - mapy bitowe, 237
 - MMU, 243
 - Symbian OS, 1121
 - wolna pamięć, 237
- zarządzanie pamięcią podręczną, 375
- zarządzanie pamięcią w systemie
 - Linux, 899
 - algorytm bliźniaków, 908, 909
 - algorytm odbierania ramek
 - stron, 912
 - algorytm wymiany stron, 913
 - blok BSS, 899
 - brk(), 903
 - dane nieinicjalizowane, 900
 - demon stron, 912
 - deskryptor pamięci wysokiego
 - poziomu, 911
 - deskryptor strefy, 906
 - deskryptor stron, 905
 - deskryptor węzłów, 907
 - dynamiczne przydzielanie
 - dotkowanej pamięci, 902
 - dyspozytor plastrowy, 909
 - dyspozytor płytowy, 909
 - dyspozytor stron, 908
 - gigantyczne ramki, 910
 - implementacja, 904
 - kmalloc, 910
 - malloc(), 903
- mechanizm zastępowania
 - stron, 913
- mechanizmy przydzielania
 - pamięci, 908
- mem_map, 905, 906
- mm_struct, 911
- mmap(), 903
- obsługa ramek stron, 910
- obszar wymiany, 912
- PAE, 910
- page, 905
- pamięć fizyczna, 900, 906
- pamięć podręczna obiektów, 909
- pamięć podręczna
 - stronicowania, 907
- pliki odwzorowywane
 - w pamięci, 902
- pliki stronicowania, 912
- POSIX, 903
- proces wymiany, 911
- przydzielanie pamięci, 909
- przypięte strefy pamięci, 905
- reprezentacja pamięci
 - głównej, 906
- rozmiar pamięci podręcznej
 - stron, 907
- segment danych, 899, 901
- segment tekstu, 899
- sterta, 901
- stos, 901
- strefy pamięci, 905
- strona zerowa, 901
- stronicowanie, 907, 911
- tablica deskryptorów stron, 905
- tablice stron, 908
- task_struct, 909
- unmap(), 903
- vm_area_struct, 910, 911
- vmalloc, 910
- wirtualna przestrzeń adresowa,
 - 904, 910
- wirtualna przestrzeń adresowa
 - procesu, 900
- współdzielenie pamięci
 - fizycznej, 904
- współdzielone segmenty
 - tekstu, 902
- wywołania systemowe, 903
- zarządzanie fragmentem
 - pamięci, 908
- ZONE_DMA, 905
- ZONE_HIGHMEM, 905
- ZONE_NORMAL, 905
- zarządzanie pamięcią w systemie
 - Windows Vista, 1045
- adresowanie wielkich pamięci
 - fizycznych, 1050
- algorytm zastępowania stron,
 - 1058
- alokowanie pamięci, 1052
- AWE, 1051
- baza danych numerów ramek
 - stron, 1060
- bit brudnej strony, 1055
- bit wykorzystania strony, 1055
- błędy braku strony, 1047, 1054
- bufor TLB, 1052
- deskryptor adresu
 - wirtualnego, 1053
- implementacja, 1052
- lista gotowości, 1049
- LRU, 1055
- menedżer równoważenia
 - zbiorów, 1060
- miękkie błędy braku stron,
 - 1047, 1057
- naruszenie dostępu, 1047, 1056
- niedwzorowane strony, 1054
- nieprawidłowa strona, 1047
- obsługa błędów braku stron, 1054
- PAE, 1056
- pagefile, 1048
- PFN database, 1060
- pliki stron, 1048
- presja pamięci, 1059
- przydzielanie adresów
 - wirtualnych, 1047
- rozmiar alokowanego
 - obszaru, 1052
- rozszerzanie stosu, 1062
- samoodwzorowanie, 1058
- stronicowanie dokładnie
 - na czas, 1048
- stronicowanie
 - z wyprzedzeniem, 1054
- strony kopiowane
 - przy zapisie, 1054
- struktura wirtualnej przestrzeni
 - adresowej, 1046
- SuperFetch, 1054
- VAD, 1053
- wątek stron zerowych, 1062
- wątek zapisujący strony
 - odwzorowane, 1062
- wątek zapisujący strony
 - zmodyfikowane, 1062
- wirtualna przestrzeń
 - adresowa, 1045
- wstępne alokowanie całej
 - pamięci wirtualnej, 1048
- wywołania systemowe, 1051
- x86, 1056
- zarządzanie pamięcią fizyczną, 1060
- zastreżona strona wirtualna, 1047
- zatwierdzona strona, 1047
- zbiór roboczy, 1059
- zwalnianie pamięci, 1052
- zarządzanie plikami, 89, 93, 319
- zarządzanie podręczną pamięcią
 - buforową, 394
- zarządzanie procesami, 89, 90, 879
- zarządzanie programami o olbrzymich
 - rozmiarach, 241
- zarządzanie projektem, 1186
 - doświadczenie, 1191
 - efekt drugiego systemu, 1192
 - lampka złych wiadomości, 1190
 - osobomiesiąc, 1187
 - spójność architekuralna, 1189
 - struktura zespołu, 1189
 - zespół, 1189
- zarządzanie ramieniem dysku, 509

- zarządzanie stosem, 161
- zarządzanie systemem plików, 357
- zarządzanie temperaturą, 506
- zarządzanie terminalem, 920
- zarządzanie wytwarzaniem
 - oprogramowania, 640
- zarządzanie zasobami, 35
- zasada Kerckhoffs, 724
- zasada lokalności, 1185
- zasada najmniejszego zaskoczenia, 864
- zasada POLA, 731
- zasilanie, 499
- zasoby, 35, 484, 518
 - bez możliwości wywłaszczania, 519
 - semafory, 520
 - z wywłaszczaniem, 518
 - zakłeszczenie, 519, 522
 - zdobywanie, 520
- zastępowanie stron, 257
- zastrzeżona strona wirtualna, 1047
- zastrzeżony token, 1025
- zaufana baza obliczeniowa, 742
 - monitor odwołań, 742
- zawieszony proces, 73
- zawodna transmisja pakietów, 918
- zbiór roboczy, 265, 267, 272
- zbyt wczesne opróżnianie bufora, 416
- zdalne i-węzły, 950
- zdalne wywołania procedur, 657
- zdalny system plików, 355
- zdarzenia, 1034
 - powiadomienia, 1034
 - synchronizacja, 1034
- zdarzenia chybionych odwołań
 - do bufora TLB, 472
- zdobywanie dzierzawy, 706
- zdobywanie zasobu, 520
- zegary, 466, 510
 - czas rzeczywisty, 468
 - GMT, 467
 - liczniki dozorujące, 468, 470
 - obsługa, 466
 - oprogramowanie obsługi, 468
 - profilowanie, 471
 - przechowywanie aktualnej godziny, 468
- przeciwdziałanie zbyt
 - długotrwałemu działaniu procesów, 469
- przepełnienie licznika, 468
- rejestr podtrzymujący, 466
- rozliczanie czasu procesora, 469
- sposoby obliczania aktualnej godziny, 469
- sprzęt obsługi, 466
- sterowanie częstotliwością przerwań, 467
- symulacja wielu liczników czasu, 470
- takty, 467
- tryb działania, 467
- tryb fali prostokątnej, 467
- tryb przerwań na żądanie, 467
- UTC, 467
- zegary programowe, 471, 472
 - błędy braku strony, 472
- przejścia procesora w stan bezczynności, 472
- przerwania wejścia-wyjścia, 472
- wywołania systemowe, 472
- zdarzenia chybionych odwołań
 - do bufora TLB, 472
- zepto, 118
- zero page, 1043
- ZeroPage thread, 1062
- zespół CERT, 808
- zespół z głównym programistą, 1189
- zestaw instrukcji, 32, 51
- zewnętrzna fragmentacja, 298
- Zielona księga, 446
- Zilog Z80, 46
- Zipf George, 598
- złośliwe oprogramowanie, 786, 844
 - botnet, 787
 - kradzież tożsamości, 788
 - monitor użycia klawiatury, 787
 - zabezpieczenia, 788
 - zastosowanie, 789
 - zombie, 787
- zmiana katalogu roboczego, 935
- zmienne, 92
 - globalne, 900
 - środowiskowe, 92
 - warunkowe, 179, 183, 184
 - współdzielone, 169
- znak pionowej linii, 869
- znak ucieczki, 477
- znak wysuwu wiersza, 476
- znak zachęty, 867
- znaki magiczne, 868
- znaki wodne, 753
- znakowe pliki specjalne, 78, 323
- zniszczenie procesu, 96
- zombie, 721, 787, 809, 882
- ZONE_DMA, 905
- ZONE_HIGHMEM, 905
- ZONE_NORMAL, 905
- zooming, 571
- zrzut fizyczny, 367
- zrzut pamięci, 779
- Zuse Konrad, 37
- zużycie energii w notebookach, 501
- zwalnianie dedykowanych urządzeń, 432
- zwielokrotnianie wątków
 - użytkownika na bazie wątków jądra, 155
- zwielokrotnianie zasobów, 35
- zwykle pliki, 323

Ż

- żądanie-odpowiedź, 690
- Żółta księga, 444

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>

GRUPA WYDAWNICZA

 **Helion SA**



SYSTEMY OPERACYJNE

ANDREW S. TANENBAUM

Ta książka to aktualne wydanie światowego bestsellera, będącego kompletnym źródłem wiedzy na temat współczesnych systemów operacyjnych. Autor tego podręcznika – Andrew S. Tanenbaum – przez wiele lat projektował trzy systemy operacyjne lub współuczestniczył w ich projektowaniu, dzięki czemu może dzielić się swą ogromną wiedzą i doświadczeniem praktyka. W tej publikacji szczególnie nacisk kładzie on na możliwie szczegółową prezentację takich aspektów systemów, jak procesy, wątki, zarządzanie pamięcią, systemy plików, operacje wejścia-wyjścia, zakleszczenia, projektowanie interfejsu, multimedia, dylematy związane z wydajnością czy najnowsze trendy w projektach systemów operacyjnych. Wśród jej współautorów znajdziesz takich specjalistów, jak Ada Gavrilovska-Habl, Michael Jipping oraz David Probert. Dzięki nim książka została wzbogacona między innymi o rozdziały poświęcone Symbianowi i Linuksowi. W pierwszej kolejności zapoznasz się z ich rysem historycznym. W końcu ich dzisiejsze możliwości to zasługa wielu dekad badań i eksperymentów. Dowiesz się także, na czym polega wirtualizacja, jak działają systemy rozproszone oraz jak chronić je przed atakami. W trakcie lektury poznasz dokładnie systemy operacyjne Windows, Linux oraz Symbian. Pozycja ta stanowi niezastąpione kompendium wiedzy na temat systemów operacyjnych, zarówno dla studentów informatyki, jak i wszystkich pasjonatów komputera.

- Rys historyczny systemów operacyjnych
- Sprzęt komputerowy – przegląd komponentów
- Rodzaje systemów operacyjnych
- Struktura systemów operacyjnych
- Obsługa wywołań systemowych
- Zarządzanie procesami i wątkami oraz komunikacja między nimi
- Szeregowanie zadań
- Zarządzanie pamięcią
- Obsługa systemów plików
- Urządzenia wejścia-wyjścia
- Metody zarządzania energią
- Rozwiązywanie problemu zakleszczeń
- Charakterystyka multimedialnych systemów operacyjnych
- Obsługa systemów wieloprocesorowych
- Wirtualizacja
- Bezpieczeństwo danych na poziomie systemu operacyjnego
- Zagrożenia ze strony złośliwego oprogramowania
- System Linux – historia, budowa, działanie
- System Windows Vista – studium przypadku
- System Symbian – przeznaczenie, działanie, obsługa

Kompendium wiedzy o współczesnych systemach operacyjnych!

Andrew S. Tanenbaum — profesor informatyki, członek organizacji IEEE oraz ACM. Prowadzi zaawansowane badania nad systemami operacyjnymi oraz ich bezpieczeństwem. Autor licznych publikacji poświęconych systemom operacyjnym, w 2007 roku uhonorowany medalem edukacji Jamesa H. Mulligana jr.

helion.pl
księgarnia
internetowa

Nr katalogowy: 14028

Księgarnia internetowa:
<http://helion.pl>

Zamówienia telefoniczne:
0 801 339900
0 601 339900



Helion

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

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

sięgnij po **WIECEJ**



KOD KORZYŚCI

Cena: 129,00 zł

ISBN 978-83-246-7115-1



9 788324 671151