

---

# Index

10BaseT Ethernet, 748  
50-percent rule, 363  
100BaseT Ethernet, 748

## A

absolute code, 354  
absolute path names, 522  
abstract data type, 506  
access:  
    anonymous, 529  
    controlled, 534  
    file, *see* file access  
access control, in Linux, 822–824  
access-control lists (ACLs), 534, 832  
access matrix, 632–636  
    and access control, 639–640  
    defined, 632  
    implementation of, 636–639  
    and revocation of access rights, 640–641  
access rights, 628, 640–641  
accounting (operating system service), 57  
accreditation, 699  
ACLs (access-control lists), 534, 832  
ACPI (advanced configuration and power interface), 862  
Active Directory (Windows 7), 875  
acyclic graph, 523  
acyclic-graph directories, 523–525  
adaptive mutex, 235  
additional-reference-bits algorithm, 418  
additional sense code, 608  
additional sense-code qualifier, 608  
address(es):  
    defined, 593  
    Internet, 752  
    linear, 385  
    logical, 355  
    physical, 355  
    virtual, 356  
address binding, 354–355  
address resolution protocol (ARP), 760  
address space:  
    logical vs. physical, 356  
    virtual, 398–399, 805–806  
address-space identifiers (ASIDs), 374  
address space layout randomization (ASLR), 832  
administrative complexity, 769  
admission-control algorithms, 286  
advanced configuration and power interface (ACPI), 862  
advanced encryption standard (AES), 677  
advanced local procedure call (ALPC), 135, 854  
advanced technology attachment (ATA) buses, 469  
advisory file-locking mechanisms, 509  
AES (advanced encryption standard), 677  
affinity, processor, 280  
aging, 271, 760–761  
allocation:  
    buddy-system, 436–437  
    of disk space, 553–561  
        contiguous allocation, 553–555  
        indexed allocation, 557–559  
        linked allocation, 555–557  
        and performance, 560–561  
    equal, 423  
    as problem, 514  
    proportional, 423  
    slab, 437–439  
ALPC (advanced local procedure call), 135, 854  
AMD64 architecture, 387  
Amdahl's Law, 167  
AMD virtualization technology (AMD-V), 720  
analytic evaluation, 300  
Android operating system, 85–86  
anomaly detection, 692

- anonymous access, 529
  - anonymous memory, 484
  - anonymous pipes, 143–145
  - APCs, *see* asynchronous procedure calls
  - API (application program interface), 63–64
  - Apple Computers, 59
  - Apple iPad, 60, 84
  - Apple Macintosh computer, 901–902
  - application containment, 713, 727–728
  - application interface (I/O systems), 597–603
    - block and character devices, 600
    - blocking and nonblocking I/O, 602–603
    - clocks and timers, 601–602
    - network devices, 600–601
  - application layer, 757
  - application programs, 4, 6, 75
    - disinfection of, 694–696
    - multistep processing of, 354–355
    - processes vs., 24–25
    - system utilities, 74–75
  - application program interface (API), 63–64
  - application proxy firewalls, 697–698
  - Aqua interface, 59, 84
  - arbitrated loop (FC-AL), 471
  - architecture(s), 12–18
    - clustered systems, 17–18
    - multiprocessor systems, 14–16
    - single-processor systems, 13–14
    - of Windows 7, 838
  - argument vector, 793
  - ARM architecture, 388
  - armored viruses, 669
  - ARP (address resolution protocol), 760
  - arrays, 31, 398
  - ASIDs (address-space identifiers), 374
  - ASLR (address space layout randomization), 832
  - assembly language, 77
  - assignment edge, 319
  - asymmetric clustering, 17
  - asymmetric encryption, 678
  - asymmetric multiprocessing, 15, 278
  - asynchronous devices, 598, 599
  - asynchronous (nonblocking) message passing, 129
  - asynchronous procedure calls (APCs), 185, 841–842
  - asynchronous thread cancellation, 185
  - asynchronous threading, 172
  - asynchronous writes, 567
  - ATA buses, 469
  - Atlas operating system, 894–895
  - atomic transactions, 210
  - attacks, 658–659. *See also* denial-of-service attacks
    - man-in-the-middle, 659
    - replay, 658
    - zero-day, 693
  - attack surface, 669
  - attributes, 865
  - augmented-reality applications, 36
  - authentication:
    - breaching of, 658
    - and communication protocols, 758–759
    - and encryption, 678–681
    - in Linux, 822
    - multifactor, 689
    - two-factor, 688
  - automatic job sequencing, 890
  - automatic variables, 664
  - automatic working-set trimming, 446
  - automount feature, 769
  - autoprobes, 791
  - auxiliary rights (Hydra), 641–642
- ## B
- back door, 599
  - background processes, 74–75, 115, 274, 296
  - backing store, 358
  - backups, 570–571
  - bad blocks, 480–482
  - balanced binary search trees, 33
  - banker's algorithm, 330–331
  - base file record, 865
  - base register, 352
  - basic file systems, 544
  - batch files, 510
  - batch interface, 56
  - Bayes' theorem, 693–694
  - Belady's anomaly, 414
  - Beowulf clusters, 18
  - best-fit strategy, 363
  - binary search trees, 33
  - binary semaphore, 214
  - binary translation, 718–720
  - binary trees, 33
  - binding, 354

- biometrics, 689
  - bit(s):
    - defined, 9
    - mode, 22
    - modify (dirty), 411
    - reference, 418
    - valid-invalid, 375
  - bit-interleaved parity organization, 488
  - bit-level striping, 486
  - bitmaps, 34
  - bit vector (bit map), 561
  - black-box transformations, 676
  - blade servers, 16
  - block(s), 65, 362, 512–513
    - bad, 480–482
    - boot, 93, 480
    - boot control, 546
    - defined, 815–816
    - direct, 559
    - file-control, 545
    - index, 557
    - index to, 514–515
    - indirect, 559
    - logical, 470
    - volume control, 546
  - block ciphers, 676
  - block devices, 598, 600, 815–816
  - block groups, 812
  - blocking, indefinite, 271
  - blocking I/O, 602–603
  - blocking (synchronous) message
    - passing, 129
  - block-interleaved distributed parity, 489
  - block-interleaved parity organization, 489
  - block-level striping, 486
  - block number, relative, 514
  - boot block, 93, 480, 546
  - boot control block, 546
  - boot disk (system disk), 93, 480
  - booting, 92–93, 862–863
  - boot partition, 480
  - boot sector, 480
  - bootstrap programs (bootstrap loaders),
    - 7, 92–93, 480, 670
  - boot viruses, 667
  - bottlenecks, 86
  - bottom half interrupt service routines, 799
  - bounded-buffer problem, 219
  - bounded capacity (of queue), 130
  - bourne-Again shell (bash), 789
  - Bourne shell command interpreter, 59
  - breach of availability, 658
  - breach of confidentiality, 658
  - breach of integrity, 658
  - bridging, 732
  - broadcasting, 760
  - BSD UNIX, 46
  - B+ tree (NTFS), 865
  - buddy heap (Linux), 801–802
  - buddy system (Linux), 801
  - buddy-system allocation, 436–437
  - buffer, 816
    - circular, 569
    - defined, 605
  - buffer cache, 565
  - buffering, 129–130, 605–606
  - buffer-overflow attacks, 663–666
  - bugs, 66
  - bus, 469
    - defined, 588
    - expansion, 588
    - PCI, 588
  - bus architecture, 12
  - bus-mastering I/O boards, 595
  - busy waiting, 591
  - byte, 9
- C**
- cache, 565–566
    - buffer, 565
    - defined, 606–607
    - in Linux, 802–803
    - as memory buffer, 352
    - nonvolatile RAM, 486
    - page, 565
    - and performance improvement, 565–568
    - slabs in, 437
    - unified buffer, 565–566
    - in Windows 7, 856–858
  - cache coherency, 29
  - cache management, 28
  - caching, 27–29, 606–607
    - client-side, 874
    - double, 566
  - Caldera, 785
  - Cambridge CAP system, 643–644
  - cancellation, thread, 185–186
  - cancellation points, 186

- capability(-ies), 637, 643
- capability-based protection systems, 641–644
  - Cambridge CAP system, 643–644
  - Hydra, 641–643
- capability lists, 637
- cascading termination, 121
- CAV (constant angular velocity), 471
- central processing unit, *see under* CPU
- certificate authorities, 681
- certification, 699
- CFQ (Completely Fair Queueing), 817
- challenging (passwords), 688
- change journal (Windows 7), 870
- character devices (Linux), 817
- character-stream devices, 598–600
- checksums, 492–493, 761
- children, 33, 116
- chipsets, 836
- Chrome, 123
- CIFS (common internet file system), 531, 871
- cipher-block chaining, 676
- circuit switching, 755
- circular buffer, 569
- circularly linked lists, 32
- circular SCAN (C-SCAN) scheduling
  - algorithm, 476
- circular-wait condition (deadlocks), 325–327
- claim edge, 329
- classes (Java), 647
- CLI (command-line interface), 56
- C library, 69
- client(s):
  - defined, 766
  - diskless, 768
  - in SSL, 683
  - thin, 35
- client interface, 766
- client-server model, 529–530, 854–855
- client-side caching (CSC), 874
- client systems, 38
- clocks, 601–602
- clock algorithm, 418–419
- clones, 579, 715
- C-LOOK scheduling algorithm, 477
- closed-source operating systems, 44
- close() operation, 507
- cloud computing, 41–42, 716
- clusters, 17–18, 479, 764, 864
- clustered page tables, 380
- clustered systems, 17–18
- clustering, 764
  - asymmetric, 17
  - in Windows, 445–446
- cluster remapping, 869
- CLV (constant linear velocity), 470
- coarse-grained multithreading, 282
- Cocoa Touch, 84
- code:
  - absolute, 354
  - reentrant, 376
- code books, 689
- code integrity module (Windows 7), 832
- collisions (of file names), 553
- COM (component object model), 873
- combined scheme index block, 559
- command interpreter, 58–59
- command-line interface (CLI), 56
- common internet file system (CIFS), 531, 871
- communication(s):
  - direct, 127
  - in distributed operating systems, 743
  - indirect, 128
  - interprocess, *see* interprocess communication
  - systems programs for, 74–75
- communications (operating system service), 57
- communication links, 127
- communication processors, 749
- communications sessions, 755
- communication system calls, 72–73
- compaction, 364, 554–555
- compiler-based enforcement, 644–647
- compile time, 354
- Completely Fair Queueing (CFQ), 817
- complexity, administrative, 769
- component object model (COM), 873
- component units, 766–767
- compression, 869
- computational kernels, 835–836
- computation migration, 746
- computation speedup, 741
- compute clusters, 764
- computer environments, 35–43
  - client-server computing, 38
  - cloud computing, 41–42
  - distributed systems, 37–38
  - mobile computing, 36–37
  - peer-to-peer computing, 39–40

- real-time embedded systems, 43
- traditional, 35–36
- virtualization, 40–41
- computer programs, *see* application programs**
- computer system(s):**
  - architecture of:
    - clustered systems, 17–18
    - multiprocessor systems, 14–16
    - single-processor systems, 13–14
  - distributed systems, 37–38
  - file-system management in, 26–27
  - I/O structure in, 12
  - memory management in, 25–26
  - operating system viewed by, 5
  - operation of, 7–9
  - process management in, 24–25
  - protection in, 29–31
  - real-time embedded systems, 43
  - secure, 658
  - security in, 29–31
  - storage in, 9–12
  - storage management in, 26–30
    - caching, 27–29
    - I/O systems, 29–30
    - mass-storage management, 27
  - threats to, 669–674
- computing:**
  - mobile, 36–37
  - safe, 694
- concurrency, 166**
- Concurrency Runtime (ConcRT), 297, 880–881**
- conditional-wait construct, 230**
- condition variables, 879**
- confidentiality, breach of, 658**
- confinement problem, 635**
- conflict phase (of dispatch latency), 285**
- conflict resolution module (Linux), 791–792**
- connectionless messages, 755**
- connectionless (UDP) sockets, 137**
- connection-oriented (TCP) sockets, 137**
- consistency checker, 568**
- consistency checking, 568–569**
- consistency semantics, 532**
- constant angular velocity (CAV), 471**
- constant linear velocity (CLV), 470**
- consumers (DTrace), 89**
- containers, 728**
- container objects (Windows 7), 701–702**
- contention scope, 277**
- context (of process), 114**
- context switches, 114, 615–616**
- contiguous disk space allocation, 553–555**
- contiguous memory allocation, 361**
- control cards, 890**
- control-card interpreter, 891**
- controlled access, 534**
- controller(s), 469, 588–589**
  - defined, 588
  - direct-memory-access, 595
  - disk, 469
  - host, 469
- control partitions, 723**
- control programs, 5**
- control register, 590**
- convenience, 3**
- convoy effect, 267**
- cooperating processes, 122**
- cooperative scheduling, 264**
- copylefting, 45**
- copy-on-write technique, 408–409**
- copy protection, 44**
- copy semantics, 606**
- core dump, 86**
- core memory, 895**
- cores, 15–16**
- counting, 563**
- counting-based page replacement algorithm, 420**
- counting semaphore, 214**
- coupling, symmetric, 17**
- covert channels, 662**
- CP/M, 900–901**
- CPU (central processing unit), 4, 351–354**
- CPU-bound processes, 113**
- CPU burst, 262**
- CPU clock, 352**
- CPU-I/O burst cycle, 262–263**
- CPU scheduler, *see* short-term scheduler**
- CPU scheduling, 20**
  - about, 261–262
  - algorithms for, 266–277
    - criteria, 265–266
    - evaluation of, 300–304
    - first-come, first-served scheduling of, 266–267
    - implementation of, 303–304
    - multilevel feedback-queue scheduling of, 275–277

**CPU scheduling** (*contd.*)

- multilevel queue scheduling of, 273–275
  - priority scheduling of, 270–271
  - round-robin scheduling of, 271–273
  - shortest-job-first scheduling of, 267–270
  - dispatcher, role of, 265
  - and I/O-CPU burst cycle, 262–263
  - models for, 300–304
    - deterministic modeling, 300–301
    - and implementation, 303–304
    - queueing-network analysis, 302
    - simulations, 302
  - multiprocessor scheduling, 278–283
    - approaches to, 278–280
    - and load balancing, 280–281
    - and processor affinity, 280
  - preemptive scheduling, 263–264
  - real-time, 283–290
    - earliest-deadline-first scheduling, 288–289
    - and minimizing latency, 283–285
    - POSIX real-time scheduling, 290
    - priority-based scheduling, 285–287
    - proportional share scheduling, 289–290
    - rate-monotonic scheduling, 287–288
  - short-term scheduler, role of, 263
  - virtual machines, 729
- crackers**, 658
- Craftworks**, 785
- crashes**, 86
- crash dumps**, 86
- creation**:
- of files, 506
  - process, 116–119
- critical sections**, 206
- critical-section problem**, 206
- and mutex locks, 212–213
  - Peterson’s solution to, 207–209
  - and semaphores, 213–218
    - deadlocks, 217
    - implementation, 215–217
    - priority inversion, 217–218
    - starvation, 217
    - usage, 214–215
  - and synchronization hardware, 209–212
- cryptography**, 674–675
- and encryption, 674–685
  - implementation of, 681–683
  - SSL example of, 683–685
- CSC (client-side caching)**, 874

- C-SCAN scheduling algorithm**, 476
- CTSS operating system**, 898
- current directory**, 521
- current-file-position pointer**, 506
- cycle stealing**, 96
- cylinder groups**, 812

**D**

- d (page offset)**, 367–368
- daemon process**, 630
- daisy chain**, 588
- Dalvik virtual machine**, 86
- data**:
  - recovery of, 568–571
  - thread-specific, 187
- data capability**, 643
- data-encryption standard (DES)**, 676–677
- data files**, 504
- datagrams**, 755
- data-in register**, 590
- data-link layer**, 757
- data loss, mean time to**, 485
- data migration**, 745–746
- data-out register**, 590
- data parallelism**, 168–169
- data section (of process)**, 106
- data striping**, 486
- DCOM**, 873
- DDOS attacks**, 658
- deadline I/O scheduler**, 817
- deadlock(s)**, 217
  - avoidance of, 322, 327–333
    - with banker’s algorithm, 330–331
    - with resource-allocation-graph algorithm, 329–330
    - with safe-state algorithm, 328–329
  - defined, 315
  - detection of, 333–337
    - algorithm usage, 336–337
    - several instances of a resource type, 334–336
    - single instance of each resource type, 334
  - methods for handling, 322–323
  - with mutex locks, 317–318
  - necessary conditions for, 318–319
  - prevention of, 323–327
    - and circular-wait condition, 325–327



- and hold-and-wait condition, 323–324
  - and mutual-exclusion condition, 323
  - and no-preemption condition, 324
- recovery from, 337–338
  - by process termination, 337–338
  - by resource preemption, 338
- system model for, 315–317
- system resource-allocation graphs for
  - describing, 319–321
- Debian, 785**
- debuggers, 66, 86**
- debugging, 72, 86–91**
  - defined, 86
  - failure analysis, 86–87
  - and performance tuning, 87
  - using DTrace for, 87–91
- dedicated devices, 598, 599**
- default signal handlers, 184**
- defense in depth, 689**
- deferred procedure calls (DPCs), 840–842**
- deferred thread cancellation, 185**
- degree of multiprogramming, 113**
- deletion, file, 506**
- demand paging, 401–407**
  - basic mechanism, 402–405
  - defined, 401
  - with inverted page tables, 442
  - and I/O interlock, 444–445
  - and page size, 440–441
  - and performance, 405–406
  - and prepaging, 439–440
  - and program structure, 442–443
  - pure, 404
  - and restarting instructions, 404–405
  - and TLB reach, 441–442
- demand-zero memory, 805**
- demilitarized zone (DMZ), 696–697**
- denial-of-service (DOS) attacks, 658, 674**
- dentry objects, 551, 809**
- DES (data-encryption standard), 676–677**
- design of operating systems:**
  - distributed operating systems, 764–765
  - goals, 75–76
  - Linux, 786–788
  - mechanisms and policies, 76
  - Windows 7, 831–837
- desktop, 59**
- desktop window manager (DWM), 831**
- deterministic modeling, 300–301**
- development kernels (Linux), 783**
- device controllers, 7, 611–613. See also I/O**
- device directory, 516. See also directories**
- device drivers, 12, 544, 588, 611–613, 891**
- device-management system calls, 71–72**
- device objects, 855**
- device queues, 111–112**
- device reservation, 607**
- DFSs, see distributed file systems**
- digital certificates, 681**
- Digital Equipment Corporation (DEC), 379**
- Digital Rights Management (DRM), 44**
- digital signatures, 680, 832**
- digital-signature algorithm, 680**
- dining-philosophers problem, 222–223, 227–229**
- direct access (files), 513–514**
- direct blocks, 559**
- direct communication, 127**
- DirectCompute, 835**
- direct I/O, 600**
- direct memory access (DMA), 12, 595–597**
- direct-memory-access (DMA) controller, 595**
- directories, 515–526**
  - acyclic-graph, 523–525
  - general graph, 525–526
  - implementation of, 552–553
  - recovery of, 568–571
  - single-level, 518–519
  - tree-structured, 521–522
  - two-level, 519–521
- direct virtual memory access (DVMA), 596**
- dirty bits (modify bits), 411**
- discovery protocols, 39**
- disinfection, program, 694**
- disk(s), 467–469. See also mass-storage structure**
  - allocation of space on, 553–561
    - contiguous allocation, 553–555
    - indexed allocation, 557–559
    - linked allocation, 555–557
    - and performance, 560–561
  - bad blocks, 480–482
  - boot, 93, 480
  - boot block, 480
  - efficient use of, 564–565
  - electronic, 11
  - floppy, 468
  - formatting, 479–480

**disk(s)** (*contd.*)

- free-space management for, 561–564
- host-attached, 471
- low-level formatted, 470
- magnetic, 10
- mini-, 516
- network-attached, 471–472
- performance improvement for, 565–568
- raw, 421, 516, 549
- scheduling algorithms, 472–478
  - C-SCAN, 476
  - FCFS, 473–474
  - LOOK, 477
  - SCAN, 475–476
  - selecting, 477–478
  - SSTF, 474–475
- solid-state, 28, 469
- storage-area network, 472
- structure of, 470
- system, 480

**disk arm**, 468

**disk controller**, 469

**diskless clients**, 768

**disk striping**, 868

**dispatched process**, 112

**dispatcher**, 265, 294

**dispatcher objects**, 233, 841, 844

**dispatch latency**, 265

**distributed denial-of-service (DDOS) attacks**, 658

**distributed file systems (DFSs)**, 529, 765–773
 

- defined, 765–766
- naming in, 767–770
- remote file access in, 770–773
- stateless, 531–532
- Windows 7, 874

**distributed information systems (distributed naming services)**, 530

**distributed lock manager (DLM)**, 18

**distributed operating systems**, 745–747

**distributed systems**, 37–38
 

- benefits of, 741–743
- defined, 741
- distributed operating systems as, 745–747
- network operating systems as, 743–745

**distributions (GNU/Linux)**, 45

**DLLs**, *see* dynamic link libraries

**DLM (distributed lock manager)**, 18

**DMA**, *see* direct memory access

**DMA controller**, 595

**DMCA (U.S. Digital Millennium Copyright Act)**, 44

**DMZ (demilitarized zone)**, 696–697

**domains**, 531, 874

**domain-name system (DNS)**, 530, 751–752

**domain switching**, 629

**DOS attacks**, *see* denial-of-service attacks

**double buffering**, 605

**double caching**, 566

**double indirect blocks**, 559

**doubly linked lists**, 32

**downsizing**, 743

**down time**, 555

**DPCs (deferred procedure calls)**, 840–842

**DRAM (dynamic random-access memory)**, 9

**driver end (STREAM)**, 613–614

**driver objects**, 855

**driver registration module (Linux)**, 790–791

**DRM (Digital Rights Management)**, 44

**DTrace**, 87–91

**dual-booted systems**, 549

**dual-core design**, 16

**dumpster diving**, 660

**DVMA (direct virtual memory access)**, 596

**DWM (desktop window manager)**, 831

**dynamic configurations**, 837, 838

**dynamic linking**, 808–809

**dynamic link libraries (DLLs)**, 357–358, 836

**dynamic loading**, 357

**dynamic protection**, 628

**dynamic random-access memory (DRAM)**, 9

**dynamic routing**, 753–754

**dynamic storage-allocation problem**, 362, 554

**E**

**earliest-deadline-first (EDF) scheduling**, 288–289

**ease of use**, 5

**ease of use features**, 830

**EC2**, 41

**ECBs (enabling control blocks)**, 90

**ECC**, *see* error-correcting code

**EDF (earliest-deadline-first) scheduling**, 288–289

**effective access time**, 405



effective memory-access time, 374  
 effective UID, 31  
 efficiency, 3, 564–565, 837  
 EIDE buses, 469  
 electronic disk, 11  
 elevator algorithm, *see* SCAN scheduling algorithm  
 emulation, 40, 727  
 emulators, 77, 713  
 enabling control blocks (ECBs), 90  
 encapsulation (Java), 649  
 encrypted viruses, 668  
 encryption, 675–676
 

- asymmetric, 678
- authentication, 678–681
- key distribution, 681
- public-key, 678
- symmetric, 676–677

 encryption, defined, 675–676  
 energy efficiency, 837  
 enhanced integrated drive electronics (EIDE) buses, 469  
 entry section, 206  
 entry set, 232  
 environmental subsystems, 836  
 environment vector, 793  
 EPROM (erasable programmable read-only memory), 93  
 equal allocation, 423  
 erasable programmable read-only memory (EPROM), 93  
 Erlang language, 241–242  
 error(s), 607–608
 

- hard, 482
- soft, 479

 error conditions, 398  
 error-correcting code (ECC), 477–478, 488  
 error detection, 57  
 escalate privileges, 31  
 escape (operating systems), 599  
 events, 233  
 event latency, 283–284  
 event objects (Windows 7), 841  
 event-pair objects, 855  
 exception dispatcher, 842  
 exceptions (with interrupts), 593  
 exclusive locks, 508  
 exec() system call, 183  
 executable files, 106, 504  
 execution of user programs, 807

execution time, 355  
 exit section, 206  
 exit() system call, 120, 121  
 expansion bus, 588  
 exponential average, 268  
 export list, 574–575  
 ext2 (second extended file system), 811  
 ext3 (third extended file system), 811–813  
 ext4 (fourth extended file system), 811  
 extended file attributes, 505  
 extended file system (extfs), 545, 811  
 extensibility, 736  
 extent (contiguous space), 555  
 extents, 865  
 external data representation (XDR), 140  
 external fragmentation, 363–364, 554

## F

failure:
 

- detection of, 762
- mean time to, 485
- recovery from, 763
- during writing of block, 494–496

 failure analysis, 86–87  
 failure modes (directories), 531–532  
 false negatives, 693  
 false positives, 693  
 fast-user switching, 863–864  
 FAT (file-allocation table), 557  
 fault tolerance, 14, 763–764, 868–869  
 fault-tolerant systems, 763–764  
 FC (fiber channel), 471  
 FC-AL (arbitrated loop), 471  
 FCB (file-control block), 545  
 FC buses, 469  
 FCFS scheduling algorithm, *see* first-come, first-served scheduling algorithm  
 feature migration, 887–888  
 fibers, 879–880  
 fiber channel (FC), 471  
 fiber channel (FC) buses, 469  
 FIFO, 32, 147  
 FIFO page replacement algorithm, 413–414  
 50-percent rule, 363  
 file(s), 26–27, 503–504. *See also* directories
 

- accessing information on, 513–515
- direct access, 513–514
- sequential access, 513

**file(s)** (*contd.*)

- attributes of, 504–505
- batch, 510
- defined, 504
- executable, 106
- internal structure of, 512–513
- locking open, 507–510
- operations on, 506–510
- protecting, 533–538
  - via file access, 533–538
  - via passwords/permissions, 537–538
- recovery of, 568–571
- storage structure for, 517–518
- file access, 508, 533–538**
- file-allocation table (FAT), 557**
- file-control block (FCB), 545**
- file descriptor, 548**
- file extensions, 510–511**
- file handle, 548**
- file info window (Mac OS X), 505**
- FileLock (Java), 508**
- file management, 74**
- file-management system calls, 71**
- file mapping, 433**
- file migration, 767**
- file modification, 74**
- file objects, 551, 809**
- file-organization module, 545**
- file pointers, 508**
- file reference, 865**
- file replication, 767**
- file session, 532**
- file sharing, 528–533**
  - and consistency semantics, 532–533
  - with multiple users, 528–529
  - with networks, 530–532
    - and client-server model, 529–530
    - and distributed information systems, 530–531
    - and failure modes, 531–532
- file systems, 503, 543–545**
  - basic, 544
  - creation of, 518
  - design problems with, 544
  - distributed, *see* distributed file systems
  - extended, 544
  - implementation of, 546–552
    - mounting, 549–550
    - partitions, 549–550
    - virtual systems, 550–552
  - levels of, 544
  - Linux, 809–815
  - log-based transaction-oriented, 569–570
  - logical, 544
  - mounting of, 526–528
  - network, 571–577
  - remote, 529
  - WAFL, 577–580
  - Windows 7, *see* Windows 7
- File System Hierarchy Standard document, 784**
- file-system management, 26–27**
- file-system manipulation (operating system service), 56**
- file transfer, 744–745**
- file transfer protocol (FTP), 529, 744–745**
- file viruses, 667**
- filter drivers, 856**
- fine-grained multithreading, 282**
- firewalls, 35, 696–698**
- firewall chains, 821**
- firewall management, 821**
- firmware, 7, 93**
- first-come, first-served (FCFS) scheduling algorithm, 266–267, 473–474**
- first-fit strategy, 363**
- fixed-partition scheme, 362**
- fixed routing, 753**
- floppy disks, 468**
- flow control, 613**
- flushing, 374**
- folders, 59**
- foreground processes, 115, 274, 296**
- fork() and exec() process model (Linux), 792–794**
- fork-join strategy, 172**
- fork() system call, 183**
- formatting, 479–480**
- forwarding, 481**
- forward-mapped page tables, 379**
- fourth extended file system (ext4), 811**
- fragments, packet, 821**
- fragmentation, 363–364**
  - external, 363–364, 554
  - internal, 363, 513
- frame(s), 367, 755**
  - stack, 664–665
  - victim, 411

**frame allocation**, 421–425  
   equal allocation, 423  
   global vs. local, 424  
   proportional allocation, 423–424  
**frame-allocation algorithm**, 412  
**frame pointers**, 664–665  
**free-behind technique**, 567  
**free objects**, 438, 803  
**Free Software Foundation (FSF)**, 45  
**free-space list**, 561  
**free-space management (disks)**, 561–564  
   bit vector, 561–562  
   counting, 563  
   grouping, 563  
   linked list, 562–563  
   and space maps, 563–564  
**front-end processors**, 616  
**FSF (Free Software Foundation)**, 45  
**FTP**, *see* file transfer protocol  
**full backup**, 570  
**FUSE file-system**, 545

## G

**Gantt chart**, 267  
**garbage collection**, 526  
**gates**, 631  
**gateways**, 754  
**GB (gigabyte)**, 9  
**gcc (GNU C compiler)**, 784  
**GCD (Grand Central Dispatch)**, 182–183  
**GDT (global descriptor table)**, 384  
**general graph directories**, 525–526  
**general trees**, 33  
**gestures**, 60  
**gigabyte (GB)**, 9  
**global descriptor table (GDT)**, 384  
**global positioning system (GPS)**, 36  
**global replacement**, 424  
**GNOME desktop**, 60  
**GNU C compiler (gcc)**, 784  
**GNU General Public License (GPL)**, 45  
**GNU/Linux**, 45  
**GNU Portable Threads**, 169  
**GPL (GNU General Public License)**, 45  
**GPS (global positioning system)**, 36  
**graceful degradation**, 14  
**Grand Central Dispatch (GCD)**, 182–183  
**granularity, minimum**, 797

**graphs, acyclic**, 523  
**graphical user interfaces (GUIs)**, 59–62  
**graphics shaders**, 835  
**grappling hook**, 670  
**Green threads**, 169  
**group identifiers**, 31  
**grouping**, 563  
**group policies**, 875  
**group rights (Linux)**, 823  
**guard pages**, 847  
**GUIs (graphical user interfaces)**, 59–62

## H

**Hadoop**, 765  
**Hadoop distributed file system (HDFS)**, 767  
**HAL**, *see* hardware-abstraction layer  
**handheld computers**, 5  
**handles**, 844  
**handle tables**, 844  
**handshaking**, 591, 611  
**handshaking procedure**, 695  
**hands-on computer systems**, 20  
**hard affinity**, 280  
**hard-coding techniques**, 128  
**hard errors**, 482  
**hard links**, 525  
**hardware**, 4  
   I/O systems, 588–597  
     direct memory access, 595–597  
     interrupts, 592–595  
     polling, 591  
   for storing page tables, 372–374  
   synchronization, 209–212  
   virtual machines, 720–721  
**hardware-abstraction layer (HAL)**, 836  
**hardware objects**, 627  
**hash collisions**, 471  
**hashed page tables**, 380  
**hash functions**, 33–34, 680  
**hash maps**, 471  
**hash tables**, 552–553  
**hash value (message digest)**, 680  
**HDFS (Hadoop distributed file system)**, 767  
**heaps**, 106, 883  
**heavyweight processes**, 163  
**hibernation**, 860–861  
**hierarchical paging**, 378–380  
**high availability**, 17

- high-availability clusters, 764
  - high performance, 834
  - high-performance computing, 17
  - hijacking, session, 659
  - hit ratio, 374, 441
  - hive, 861
  - hold-and-wait condition (deadlocks), 323–324
  - holes, 362
  - homogeneity, 278
  - host adapter, 589
  - host-attached storage, 471
  - host controller, 469
  - host-id, 751
  - hot spare disks, 491
  - hot-standby mode, 17
  - human security, 660
  - hybrid cloud, 42
  - hybrid operating systems, 83–86
    - Android, 85–86
    - iOS, 84–85
    - Mac OS X, 84
  - Hydra, 641–643
  - hypercalls, 726
  - hyperspace, 846
  - hypervisors, 712
    - type 0, 723–724
    - type 1, 724–725
    - type 2, 725
- I**
- IA-32 architecture, 384–387
    - paging in, 385–387
    - segmentation in, 384–385
  - IA-64 architecture, 387
  - IaaS (infrastructure as a service), 42
  - IBM OS/360, 899–900
  - identifiers:
    - file, 504
    - group, 31
    - user, 31
  - idle threads, 294, 840
  - IDPSs (intrusion-prevention systems), 692
  - IDSs (intrusion-detection systems), 691–694
  - IKE protocol, 682
  - immutable shared files, 533
  - imperative languages, 241
  - impersonation, 853
  - implementation:
    - of CPU scheduling algorithms, 303–304
    - of operating systems, 76–77
    - of transparent naming techniques, 769–770
  - implicit threading, 177–183
    - Grand Central Dispatch (GCD), 182–183
    - OpenMP and, 181–182
    - thread pools and, 179–181
  - incremental backup, 571
  - indefinite blocking (starvation), 217, 271
  - independence, location, 767
  - independent disks, 485
  - independent processes, 122
  - index, 514
  - index block, 557
  - indexed disk space allocation, 557–559
  - index root, 865
  - indirect blocks, 559
  - indirect communication, 128
  - information-maintenance system calls, 72
  - infrastructure as a service (IaaS), 42
  - inode, 545
  - inode objects, 551, 809
  - input/output, *see under* I/O
  - input queue, 354
  - InServ storage array, 493
  - instruction-execution cycle, 10, 351–352
  - instruction register, 10
  - integrity, breach of, 658
  - intellimirror, 875
  - Intel processors, 383–387
    - IA-32 architecture, 384–387
    - IA-64 architecture, 387
  - interactive (hands-on) computer systems, 20
  - interface(s):
    - batch, 56
    - choice of, 61–62
    - client, 766
    - defined, 597
    - intermachine, 766
    - Windows 7 networking, 870–875
  - interlock, I/O, 444–445
  - intermachine interface, 766
  - internal fragmentation, 363, 513
  - international use, 837
  - Internet address, 752
  - Internet Key Exchange (IKE), 682
  - Internet Protocol (IP), 681–683
  - interpretation, 40
  - interpreted languages, 727

**interprocess communication (IPC), 122–130**  
 in client-server systems, 136–147  
 remote procedure calls, 138–142  
 sockets, 136–138  
 in Linux, 783, 818–819  
 Mach example of, 131–134  
 in message-passing systems, 126–130  
 POSIX shared-memory example of, 130–131  
 in shared-memory systems, 124–126  
 Windows example of, 135

**interrupt(s), 8–9, 592–595**  
 defined, 592  
 in Linux, 799–800

**interrupt chaining, 593**

**interrupt-controller hardware, 593**

**interrupt-dispatch table (Windows 7), 843–844**

**interrupt-driven data transfer, 436**

**interrupt-driven operating systems, 21–24**

**interrupt-handler routine, 592**

**interrupt latency, 284–285**

**interrupt priority levels, 593**

**interrupt-request line, 592**

**interrupt service routines (ISRs), 840**

**interrupt vector, 8–9, 360, 593**

**intruders, 658**

**intrusion detection, 691–694**

**intrusion-detection systems (IDSs), 691–694**

**intrusion-prevention systems (IDPSs), 692**

**inverted page tables, 381–383, 442**

**I/O (input/output), 4, 12**  
 memory-mapped, 435–436  
 overlapped, 892–894  
 programmed, 436  
 virtual machines, 731–732

**I/O-bound processes, 113**

**I/O burst, 262**

**I/O channel, 616**

**I/O interlock, 444–445**

**I/O manager, 855–856**

**I/O operations (operating system service), 56–57**

**I/O ports, 436**

**I/O request packet (IRP), 855**

**iOS operating system, 84–85**

**I/O subsystem(s), 29–30**  
 kernels in, 604–610  
 procedures supervised by, 610

**I/O system(s), 587–588**  
 application interface, 597–603  
 block and character devices, 600  
 clocks and timers, 601–602  
 network devices, 600–601  
 nonblocking and asynchronous I/O, 602–603  
 vectored I/O, 603–604

**hardware, 588–597**  
 direct memory access, 595–597  
 interrupts, 592–595  
 polling, 591

**kernels, 604–610**  
 buffering, 605–606  
 caching, 606–607  
 data structures, 608–609  
 error handling, 607–608  
 I/O scheduling, 604–605  
 and I/O subsystems, 610  
 protection, 608  
 spooling and device reservation, 607

**Linux, 815–817**  
 block devices, 816  
 character devices, 817  
 STREAMS mechanism, 613–615  
 and system performance, 615–618  
 transformation of requests to hardware operations, 611–612

**IP (Internet Protocol), 681–683**

**iPad, *see* Apple iPad**

**IPC, *see* interprocess communication**

**IPSec, 682**

**IRP (I/O request packet), 855**

**ISCSI, 472**

**ISO Reference Model, 682**

**ISRs (interrupt service routines), 840**

## J

**Java:**  
 file locking in, 508–509  
 language-based protection in, 647–649  
 monitors in, 232

**Java threads, 176–177**

**Java Virtual Machine (JVM), 107, 726, 736–737**

**jobs, processes vs., 106–107**

**job objects, 852**

**job pool, 20**

job queues, 111  
 job scheduler, 112  
 job scheduling, 20  
 journaling, 813–814  
 journaling file systems, 569–570  
 just-in-time (JIT) compilers, 727  
 JVM, *see* Java Virtual Machine

**K**

KB (kilobyte), 9  
 K Desktop Environment (KDE), 60  
 kernel(s), 6, 604–610  
   buffering, 605–606  
   caching, 606–607  
   computational, 835  
   data structures, 608–609  
   error handling, 607–608  
   I/O scheduling, 604–605  
   and I/O subsystems, 610  
   Linux, 787–789  
   nonpreemptive, 207  
   preemptive, 207  
   protection, 608  
   spooling and device reservation, 607  
   task synchronization (in Linux), 798–800  
   Windows 7, 839–844, 875  
 kernel code, 96  
 kernel data structures, 31–34  
   arrays, 31  
   bitmaps, 34  
   hash functions and maps, 33–34  
   lists, 31–33  
   queues, 32  
   stacks, 32  
   trees, 31–33  
 kernel environment, 84  
 kernel extensions, 84  
 kernel memory allocation, 436–439  
 kernel mode, 22, 787  
 Kernel-Mode Driver Framework (KMDF), 856  
 kernel-mode threads (KT), 844  
 kernel modules, 789–792  
   conflict resolution, 791–792  
   driver registration, 790–791  
   Linux, 96–101  
   management of, 789–790  
 kernel threads, 169  
 kernel transaction manager (KTM), 862

Kernighan’s Law, 87  
 keys, 638, 641  
   private, 678  
   public, 678  
 key distribution, 681  
 key ring, 681  
 keystreams, 677  
 keystroke logger, 669  
 kilobyte (KB), 9  
 KMDF (Kernel-Mode Driver Framework), 856  
 KT (kernel-mode threads), 844  
 KTM (kernel transaction manager), 862

**L**

language-based protection systems, 644–649  
   compiler-based enforcement, 644–647  
   Java, 647–649  
 LANs, *see* local-area networks  
 latency:  
   in real-time systems, 283–285  
   target, 797  
 layers (of network protocols), 681  
 layered approach (operating system structure), 79–81  
 lazy swapper, 401  
 LCNs (logical cluster numbers), 864  
 LDAP, *see* lightweight directory-access protocol  
 LDT (local descriptor table), 384  
 least-frequently used (LFU) page-replacement algorithm, 420  
 least privilege, principle of, 626–627  
 least-recently-used (LRU) page-replacement algorithm, 416–418  
 left child, 33  
 LFH design, 883–884  
 LFU page-replacement algorithm, 420  
 lgroups, 425  
 libraries:  
   Linux system, 787–788  
   shared, 358, 400  
 LIFO, 32  
 lightweight directory-access protocol (LDAP), 531, 875  
 limit register, 352  
 linear addresses, 385  
 linear lists (files), 552



- line discipline, 817**
- link(s):**
  - communication, 127
  - defined, 523–524
  - hard, 525
  - resolving, 524
  - symbolic, 845
- linked disk space allocation, 555–557**
- linked lists, 562–563**
- linked scheme index block, 559**
- linking, dynamic vs. static, 357–358, 808–809**
- Linux, 45, 781–824**
  - design principles for, 786–788
  - file systems, 809–815
    - ext3 file system, 811–813
    - journaling, 813–814
    - process, 814
    - virtual, 809–811
  - history of, 781–786
    - distributions, 785
    - first kernel, 782–784
    - licensing, 785–786
    - system description, 784
  - interprocess communication, 818–819
  - I/O system, 815–817
    - block devices, 816–817
    - character devices, 817
  - kernel modules, 96–101, 789–792
  - memory management, 800–809
    - execution and loading of user programs, 807
    - physical memory, 801–804
    - virtual memory, 804–807
  - network structure, 819–821
  - process management, 792–795
    - fork() and exec() process model, 792–794
    - processes and threads, 795
  - process representation in, 110
  - scheduling in, 795–800
    - example, 290–294
    - kernel synchronization, 798–800
    - process, 796–797
    - symmetric multiprocessing, 800
  - security model, 821–824
    - access control, 822–824
    - authentication, 822
  - swap-space management in, 484
  - synchronization in, 234–235
  - threads example, 189–191
- Linux distributions, 782, 785**
- Linux kernel, 782–784**
- Linux system(s):**
  - components of, 782, 787–789
  - obtaining page size on, 370
- lists, 31–32, 398**
- Little’s formula, 302**
- LiveCD, 45**
- LiveDVD, 45**
- live migration (virtual machines), 716, 733–735**
- load balancing, 280–281**
- loader, 891**
- loading:**
  - dynamic, 357
  - in Linux, 807–808
- load sharing, 278, 742**
- load time, 354**
- local-area networks (LANs), 17, 37, 747–750**
- local descriptor table (LDT), 384**
- locality model, 427**
- locality of reference, 404**
- local procedure calls (LPCs), 834**
- local replacement, 424**
- local replacement algorithm (priority replacement algorithm), 427**
- location, file, 504**
- location independence, 767**
- location-independent file identifiers, 769–770**
- location transparency, 767**
- lock(s), 209, 638**
  - acquire, 69
  - advisory, 509
  - exclusive, 508
  - in Java API, 508–509
  - mandatory, 509
  - mutex, 212–214
  - reader-writer, 220–222
  - release, 69
  - shared, 508
- lock-key scheme, 638**
- lock() operation, 508**
- log-based transaction-oriented file systems, 569–570**
- log files, 87**
- log-file service, 866**
- logging area, 867**
- logical address, 355**
- logical address space, 356**

**logical blocks**, 470  
**logical cluster numbers (LCNs)**, 864  
**logical file system**, 545  
**logical formatting**, 479  
**logical memory**, 21, 398–399. *See also* virtual memory  
**logical records**, 513  
**login, network**, 531  
**long-term scheduler (job scheduler)**, 112  
**LOOK scheduling algorithm**, 477  
**loopback**, 138  
**loosely-coupled systems**, 17  
**love bug virus**, 694  
**low-fragmentation heap (LFH) design**, 883–884  
**low-level formatted disks**, 470  
**low-level formatting (disks)**, 479  
**LPCs (local procedure calls)**, 834  
**LRU-approximation page replacement algorithm**, 418–420

## M

**MAC (message-authentication code)**, 680  
**MAC (medium access control) address**, 760  
**Mach operating system**, 81–82, 131–134, 902–903  
**Macintosh operating system**, 901–902  
**Mac OS X operating system**, 84  
**macro viruses**, 667  
**magic number (files)**, 511  
**magnetic disk(s)**, 10, 467–469. *See also* disk(s)  
**magnetic tapes**, 469–470  
**mailboxes**, 128  
**mailbox sets**, 134  
**mainframes**, 5  
**main memory**, 9–10
 

- and address binding, 354–355
- contiguous allocation of, 360–361
  - and fragmentation, 363–364
  - mapping, 361
  - methods, 362–363
  - protection, 361
- and dynamic linking, 357–358
- and dynamic loading, 357
- and hardware, 352–354
- Intel 32 and 64-bit architectures example:
  - paging, 385–387
  - segmentation, 384–385

- and logical vs. physical address space, 356
- paging for management of, 366–383
  - basic method, 367–372
  - hardware, 372–374
  - hashed page tables, 380
  - hierarchical paging, 378–380
  - Intel 32 and 64-bit architectures
    - example, 385–387
  - inverted page tables, 381–383
  - and Oracle SPARC Solaris, 383
  - protection, 375–376
    - and shared pages, 376–377
- segmentation for management of, 364–366
  - basic method, 364–366
  - hardware, 365–366
  - Intel 32 and 64-bit architectures
    - example, 384–385
  - and swapping, 358–360

**MANs (metropolitan-area networks)**, 37  
**mandatory file-locking mechanisms**, 509  
**man-in-the-middle attack**, 659  
**many-to-many multithreading model**, 170–171  
**many-to-one multithreading model**, 169  
**marshalling**, 872  
**Mars Pathfinder**, 218  
**maskable interrupts**, 593  
**masquerading**, 658  
**mass-storage management**, 27  
**mass-storage structure**, 467–470
 

- disk attachment:
  - host-attached, 471
  - network-attached, 471–472
  - storage-area network, 472
- disk management:
  - bad blocks, 480–482
  - boot block, 480
  - formatting of disks, 479–480
- disk scheduling algorithms, 472–478
  - C-SCAN, 476
  - FCFS, 473–474
  - LOOK, 477
  - SCAN, 475–476
  - selecting, 477–478
  - SSTF, 474–475
- disk structure, 470–471
- extensions, 492
- magnetic disks, 467–469
- magnetic tapes, 469–470
- RAID structure, 484–494

- performance improvement, 486
- problems with, 492–494
- RAID levels, 487–491
- reliability improvement, 485–486
- stable-storage implementation, 494–496
- swap-space management, 482–484
- master book record (MBR), 480**
- master file directory (MFD), 519**
- master file table, 546**
- master key, 641**
- master secret (SSL), 684**
- matchmakers, 141**
- MB (megabyte), 9**
- MBR (master book record), 480**
- MCP operating system, 902–903**
- mean time to data loss, 485**
- mean time to failure, 485**
- mean time to repair, 485**
- mechanisms, 76**
- medium access control (MAC) address, 760**
- medium-term scheduler, 113**
- megabyte (MB), 9**
- memory:**
  - anonymous, 484
  - core, 895
  - direct memory access, 12
  - direct virtual memory access, 596
  - logical, 21, 398–399
  - main, *see* main memory
  - over-allocation of, 409
  - physical, 21
  - secondary, 404
  - semiconductor, 11
  - shared, 122, 400
  - transactional, 239–240
  - unified virtual memory, 565
  - virtual, *see* virtual memory
- memory-address register, 355**
- memory allocation, 362–363**
- memory leaks, 101**
- memory management, 25–26**
  - in Linux, 800–809
    - execution and loading of user programs, 807–809
    - physical memory, 801–804
    - virtual memory, 804–807
  - with virtual machines, 730–731
  - in Windows 7, 882–884
    - heaps, 883
    - memory-mapping files, 882–883
    - thread-local storage, 884
    - virtual memory, 882
- memory-management unit (MMU), 356, 384, 849–850**
- memory-mapped files, 847**
- memory-mapped I/O, 435–436, 589**
- memory mapping, 361, 430–436**
  - basic mechanism, 430–432
  - defined, 430
  - I/O, memory-mapped, 435–436
  - in Linux, 807–808
  - in Win32 API, 433–435
- memory-mapping files, 882–883**
- memory protection, 361**
- memory-resident pages, 402**
- memory stall, 282**
- memory-style error-correcting organization, 488**
- messages:**
  - connectionless, 755
  - in distributed operating systems, 743
- message-authentication code (MAC), 680**
- message digest (hash value), 680**
- message modification, 658**
- message passing, 122**
- message-passing model, 72, 126–130**
- message queue, 897**
- message switching, 755**
- metadata, 531, 866**
- metaslabs, 563**
- methods (Java), 647**
- metropolitan-area networks (MANs), 37**
- MFD (master file directory), 519**
- MFU page-replacement algorithm, 420**
- microkernels, 81–82**
- Microsoft Windows, *see* Windows operating system**
- micro TLBs, 388**
- migration:**
  - computation, 746
  - data, 745–746
  - file, 767
  - process, 747
  - with virtual machines, 733–735
- minicomputers, 5**
- minidisks, 516**
- minimum granularity, 797**
- miniport driver, 856**
- mirroring, 485**
- MMU, *see* memory-management unit**

- mobile computing**, 36–37
  - mobile systems**:
    - multitasking in, 115
    - swapping on, 360, 407
  - mobility, user**, 573
  - mode bit**, 22
  - modify bits (dirty bits)**, 411
  - modules**, 82–83, 613–614
  - module entry point**, 97
  - module exit point**, 97
  - monitors**, 223–232
    - dining-philosophers solution using, 227–229
    - implementation of, using semaphores, 229–230
    - resumption of processes within, 230–232
    - usage of, 225–227
  - monitor calls**, *see* **system calls**
  - monoculture**, 669
  - Moore's Law**, 6, 835
  - Morris, Robert**, 670–672
  - most-frequently used (MFU) page-replacement algorithm**, 420
  - mounting**, 549–550
  - mount points**, 526, 869–870
  - mount protocol**, 574
  - mount table**, 546, 611
  - MS-DOS**, 900–901
  - multicore processors**, 281–283
  - multicore programming**, 166–169
  - multicore systems**, 14, 16, 166
  - MULTICS operating system**, 630–632, 887, 888, 898–899
  - multifactor authentication**, 689
  - multilevel feedback-queue scheduling algorithm**, 275–277
  - multilevel index**, 559
  - multilevel queue scheduling algorithm**, 273–275
  - multinational use**, 837
  - multipartite viruses**, 669
  - multiple-partition method**, 362
  - multiple universal-naming-convention provider (MUP)**, 873
  - multiprocessing**:
    - asymmetric, 15, 278
    - memory access model for, 15
    - symmetric, 15–16, 279, 800
  - multiprocessor scheduling**, 278–283
    - approaches to, 278–280
    - examples of:
      - Linux, 290–294
      - Solaris, 297–299
      - Windows, 294–296
    - and load balancing, 280–281
    - and multicore processors, 281–283
    - and processor affinity, 280
  - multiprocessor systems (parallel systems, tightly coupled systems)**, 14–16, 166
  - multiprogramming**, 19–20, 113
  - multitasking**, *see* **time sharing**
  - multithreading**:
    - benefits of, 165–166
    - cancellation, thread, 185–186
    - coarse-grained, 282
    - and **exec()** system call, 183
    - fine-grained, 282
    - and **fork()** system call, 183
    - models of, 169–171
    - and scheduler activations, 187–188
    - and signal handling, 183–185
    - and thread-specific data, 187
  - multi-touch hardware**, 863
  - MUP (multiple universal-naming-convention provider)**, 873
  - mutant (Windows 7)**, 841
  - mutex**:
    - adaptive, 235
    - in Windows 7, 841
  - mutex locks**, 212–214, 317–318
  - mutual exclusion**, 319
  - mutual-exclusion condition (deadlocks)**, 323
- ## N
- names**:
    - resolution of, 751–753
    - in Windows 7, 845
  - named pipes**, 872
  - name server**, 752
  - namespaces**, 793
  - naming**, 127–129, 530–531
    - defined, 767
    - domain name system, 530
    - of files, 504

- lightweight directory-access protocol, 531
    - and network communication, 751–753
  - NAT (network address translation), 732**
  - national-language-support (NLS) API, 837**
  - NDIS (network device interface specification), 870**
  - nested page tables (NPTs), 720**
  - network(s). *See also* local-area networks (LANs); wide-area networks (WANs)**
    - communication protocols in, 756–760
    - communication structure of, 751–756
      - and connection strategies, 755–756
      - and naming/name resolution, 751–753
      - and packet strategies, 755
      - and routing strategies, 753–754
    - defined, 37
    - design issues with, 764–765
    - example, 760–761
    - in Linux, 819–821
    - metropolitan-area (MANs), 37
    - robustness of, 762–764
    - security in, 660
    - small-area, 37
    - threats to, 669–674
    - types of, 747–748
    - in Windows 7, 870–875
      - Active Directory, 875
      - domains, 874
      - interfaces, 870
      - protocols, 871–874
      - redirectors and servers, 873–874
    - wireless, 35
  - network address translation (NAT), 732**
  - network-attached storage, 471–472**
  - network computers, 35**
  - network devices, 600–601, 816**
  - network device interface specification (NDIS), 870**
  - network file systems (NFS), 571–577**
    - mount protocol, 574
    - NFS protocol, 574–575
    - path-name translation, 574–575
    - remote operations, 577
  - network information service (NIS), 530**
  - network layer, 757**
  - network-layer protocol, 681**
  - network login, 531**
  - network operating systems, 38, 743–745**
    - new state, 107
  - NFS, *see* network file systems**
  - NFS protocol, 574–575**
  - NIS (network information service), 530**
  - NLS (national-language-support) API, 837**
  - nonblocking I/O, 602–603**
  - nonblocking (asynchronous) message passing, 129**
  - noncontainer objects (Windows 7), 702**
  - nonmaskable interrupt, 593**
  - nonpreemptive kernels, 207**
  - nonpreemptive scheduling, 264**
  - nonrepudiation, 680–681**
  - nonresident attributes, 865**
  - nonsignaled state, 233**
  - non-uniform memory access (NUMA), 16, 425, 834**
  - nonvolatile RAM (NVRAM), 11**
  - nonvolatile RAM (NVRAM) cache, 486**
  - nonvolatile storage, 11–12**
  - no-preemption condition (deadlocks), 324**
  - NPTs (nested page tables), 720**
  - NTFS, 864–866**
  - NUMA, *see* non-uniform memory access**
  - NVRAM (nonvolatile RAM), 11**
  - NVRAM (nonvolatile RAM) cache, 486**
- 
- objects:**
    - access lists for, 36–37
    - in cache, 437–438
    - free, 438
    - hardware vs. software, 627
    - in Linux, 803
    - used, 438
    - in Windows 7, 844–846
  - object files, 504**
  - object linking and embedding (OLE), 873**
  - object types, 551, 845**
  - off-line compaction of space, 555**
  - OLE (object linking and embedding), 873**
  - OLPC (One Laptop per Child), 902**
  - One Laptop per Child (OLPC), 902**
  - one-time pad, 689**
  - one-time passwords, 688**
  - one-to-one multithreading model, 170**
  - on-line compaction of space, 555**

**open-file table**, 507, 546–547  
**OpenMP**, 181–182, 240–241  
**open operating systems**, 669  
**open() operation**, 507  
**OpenSolaris**, 46  
**open-source operating systems**, 43–48  
**operating system(s)**:  
   closed-source, 44  
   defined, 3, 6  
   design goals for, 75–76  
   early, 888–894  
     dedicated computer systems, 888–889  
     overlapped I/O, 892–894  
     shared computer systems, 890–892  
   feature migration with, 887–888  
   features of, 3  
   functioning of, 3–6  
   hybrid systems, 83–86  
   implementation of, 76–77  
   interrupt-driven, 21–24  
   mechanisms for, 76  
   network, 38  
   open-source, 43–47  
   operations of:  
     modes, 21–23  
     and timer, 24  
   policies for, 76  
   portability of, 836–837  
   real-time, 43  
   as resource allocator, 5  
   security in, 660  
   services provided by, 55–58  
   structure of, 19–21, 78–86  
     layered approach, 79–81  
     microkernels, 81–82  
     modules, 82–83  
     simple structure, 78  
   study of, 48  
   system’s view of, 5  
   user interface with, 4–5, 53–55  
**optimal page replacement algorithm**, 414–415  
**Oracle SPARC Solaris**, 383  
**Orange Book**, 832  
**OS/2 operating system**, 829–830  
**OSI model**, 757–758  
**OSI protocol stack**, 757–758  
**OSI Reference Model**, 682  
**out-of-band key delivery**, 681  
**overallocation (of memory)**, 409

**overcommitment**, 729  
**overlapped I/O**, 892–894  
**owner rights (Linux)**, 822–823

## P

**p (page number)**, 367  
**PaaS (platform as a service)**, 42  
**packets**, 755, 821  
**packet switching**, 755–756  
**packing**, 512  
**pages**:  
   defined, 367  
   shared, 376–377  
**page address extension (PAE)**, 396  
**page allocator (Linux)**, 801  
**page-buffering algorithms**, 420–421  
**page cache**, 565, 804  
**page directory**, 847  
**page-directory entries (PDEs)**, 847  
**page directory pointer table**, 386  
**page fault**, 403  
**page-fault-frequency (PFF)**, 429–430  
**page-fault rate**, 407  
**page frames**, 847  
**page-frame number (PFN) database**, 850–851  
**page number (p)**, 367  
**page offset (d)**, 367–368  
**pageout (Solaris)**, 446–447  
**pageout policy (Linux)**, 806  
**pager (term)**, 401  
**page replacement**, 409–421. *See also* **frame allocation**  
   and application performance, 421  
   basic mechanism, 410–413  
   counting-based page replacement, 420  
   FIFO page replacement, 413–414  
   global vs. local, 424  
   LRU-approximation page replacement, 418–420  
   LRU page replacement, 416–418  
   optimal page replacement, 414–415  
   and page-buffering algorithms, 420–421  
**page replacement algorithm**, 412  
**page size**, 440–441  
**page slots**, 484  
**page table(s)**, 367–372, 404, 847  
   clustered, 380  
   forward-mapped, 379



- hardware for storing, 372–374
- hashed, 380
- inverted, 381–383, 442
- page-table base register (PTBR)**, 372
- page-table entries (PTEs)**, 847
- page-table length register (PTLR)**, 376
- page-table self-map**, 846
- paging**, 366–383
  - basic method of, 367–372
  - hardware support for, 372–374
  - hashed page tables, 380
  - hierarchical, 378–380
  - Intel 32 and 64-bit architectures example, 385–387
  - inverted, 381–383
  - in Linux, 806
  - and memory protection, 375–376
  - and Oracle SPARC Solaris, 383
  - priority, 447
  - and shared pages, 376–377
  - swapping vs., 482
- paging mechanism (Linux)**, 806
- paired passwords**, 688
- PAM (pluggable authentication modules)**, 822
- parallelism**, 166, 168–169
- parallelization**, 17
- parallel systems**, *see* multiprocessor systems
- paravirtualization**, 713, 725–726
- parent process**, 116
- partition(s)**, 362, 515, 516, 549–550
  - boot, 480
  - control, 723
  - raw, 483
  - root, 549
- partition boot sector**, 546
- partitioning, disk**, 479
- passwords**, 685–689
  - one-time, 688–689
  - securing, 687–688
  - vulnerabilities of, 685–687
- path name**, 520
- path names**:
  - absolute, 522
  - relative, 522
- path-name translation**, 574–575
- PCBs**, *see* process control blocks
- PCI bus**, 588
- PCS (process-contention scope)**, 277
- PC systems**, 3, 863
- PDA (personal digital assistants)**, 11
- PDEs (page-directory entries)**, 847
- peer-to-peer computing**, 39–40
- penetration test**, 690
- performance**:
  - and allocation of disk space, 560–561
  - and I/O system, 615–618
  - of Windows 7, 834–836
- performance improvement**, 486, 565–568
- performance tuning**, 87
- periodic processes**, 286
- periodic task rate**, 286
- permissions**, 536
- per-process open-file table**, 547
- personal computer (PC) systems**, 3, 863
- personal digital assistants (PDAs)**, 11
- personal firewalls**, 697
- personal identification number (PIN)**, 688
- personalities**, 83
- Peterson’s solution**, 207–209
- PFF (page-fault-frequency)**, 429–430
- PFN database**, 850–851
- phishing**, 660
- physical address**, 355
- physical address space**, 356
- physical formatting**, 479
- physical layer**, 757
- physical memory**, 21, 397–398, 801–804
- physical security**, 659
- PIC (position-independent code)**, 809
- pid (process identifier)**, 116
- PIN (personal identification number)**, 688
- pinning**, 857
- PIO**, *see* programmed I/O
- pipes**, 142–147
  - anonymous, 143–145
  - named, 145–147
  - ordinary, 142–145
  - use of, 148
- pipe mechanism**, 818
- platform as a service (PaaS)**, 42
- platter (disks)**, 467–468
- plug-and-play and (PnP) managers**, 860
- pluggable authentication modules (PAM)**, 822
- point-to-point tunneling protocol (PPTP)**, 871
- policy(ies)**, 76
  - group, 875
  - security, 689–690
- policy algorithm (Linux)**, 806

- polling, 591
- polymorphic viruses, 668
- pools:
  - of free pages, 408
  - of storage, 494
- pop, 32
- pop-up browser windows, 662
- ports, 436, 588
- portability, 836–837
- portals, 35
- port driver, 856
- port scanning, 673
- position-independent code (PIC), 809
- positioning time (disks), 468
- POSIX, 829–830, 833–834
  - interprocess communication example, 130–131
  - real-time scheduling, 290
- possession (of capability), 637
- POST (power-on self-test), 862
- power manager (Windows 7), 860–861
- power-of-2 allocator, 436
- power-on self-test (POST), 862
- PPTP (point-to-point tunneling protocol), 871
- P + Q redundancy scheme, 489–490
- preemptive kernels, 207
- preemptive scheduling, 263–264
- premaster secret (SSL), 684
- prepaging, 439–440
- presentation layer, 757
- primary thread, 876
- principle of least privilege, 626–627
- priority-based scheduling, 285–287
- priority-inheritance protocol, 218, 236
- priority inversion, 217–218, 236
- priority number, 230
- priority paging, 447
- priority replacement algorithm, 427
- priority scheduling algorithm, 270–271
- private cloud, 42
- private keys, 678
- privileged instructions, 22
- privileged mode, *see* kernel mode
- privilege levels, 23
- probes (DTrace), 89
- procedural languages, 241
- process(es), 20
  - background, 74–75, 115, 274, 296
  - communication between, *see* interprocess communication
  - components of, 106–107
  - context of, 114, 794
  - and context switches, 114
  - cooperating, 122
  - defined, 105
  - environment of, 793–794
  - foreground, 115, 274, 296
  - heavyweight, 163
  - independent, 122
  - I/O-bound vs. CPU-bound, 113
  - job vs., 106
  - in Linux, 795
  - multithreaded, *see* multithreading
  - operations on, 116–119
    - creation, 116–119
    - termination, 120–121
  - programs vs., 24–25, 106–107
  - scheduling of, 110–114
  - single-threaded, 163
  - state of, 107
  - system, 8
    - as term, 105–106
    - threads performed by, 109
    - in Windows 8, 876
- process-contention scope (PCS), 277
- process control blocks (PCBs, task control blocks), 107–108
- process-control system calls, 67–71
- process file systems (Linux), 814–815
- process identifier (pid), 116
- process identity (Linux), 792–793
- process management:
  - about, 24–25
  - in Linux, 792–795
    - fork() and exec() process model, 792–794
    - processes and threads, 795
- process manager (Windows 7), 852–853
- process mix, 113
- process objects (Windows 7), 841
- processor affinity, 280
- processor groups, 835
- processor sets, 280
- processor sharing, 273
- process representation (Linux), 110
- process scheduler, 111
- process scheduling:
  - in Linux, 796–797
  - thread scheduling vs., 261

**process synchronization:**

- about, 204–206
  - alternative approaches to, 238–242
    - functional programming languages, 241–242
    - OpenMP, 240–241
    - transactional memory, 239–240
  - bounded-buffer problem, 219
  - critical-section problem, 206–207
    - hardware solution to, 209–212
    - Peterson’s solution to, 207–209
    - software solution to, 212–213
  - dining-philosophers problem, 222–223, 227–229
  - examples of:
    - Java, 232
    - Linux, 234–235
    - Pthreads, 237–238
    - Solaris, 235–237
    - Windows, 233–234
  - monitors for, 223–232
    - dining-philosophers solution, 227–229
    - resumption of processes within, 230–232
    - semaphores, implementation using, 229–230
    - usage, 225–227
  - readers-writers problem, 220–222
  - semaphores for, 213–218
- process termination, deadlock recovery by, 337–338**
- production kernels (Linux), 783**
- profiling (DTrace), 88–89**
- programs, processes vs., 106–107. *See also***
- application programs
- program counters, 25, 106**
- program execution (operating system service), 56**
- program files, 504**
- program loading and execution, 74**
- programmable interval timer, 601**
- programmed I/O (PIO), 436, 595**
- programming-environment virtualization, 713, 726–727**
- programming-language support, 74**
- program threats, 661–669**
- logic bombs, 663
  - stack- or buffer overflow attacks, 663–666
  - trap doors, 662
  - Trojan horses, 661–662
  - viruses, 666–667

**projects, 299**

- proportional allocation, 423**
- proportional share scheduling, 289–290**
- protection, 73, 625**
  - access control for, 533–538
  - access matrix as model of, 632–636
    - control, access, 639–640
    - implementation, 636–639
  - capability-based systems, 641–644
    - Cambridge CAP system, 643–644
    - Hydra, 641–643
  - in computer systems, 29–31
  - domain of, 627–628
    - MULTICS example, 630–632
    - structure, 628–629
    - UNIX example, 629–630
  - error handling, 607–608
  - file, 504
  - of file systems, 533–538
  - goals of, 625–626
  - I/O, 608
  - language-based systems, 644–649
    - compiler-based enforcement, 644–647
    - Java, 647–649
  - as operating system service, 57–58
  - in paged environment, 375–376
  - permissions, 536
  - and principle of least privilege, 626–627
  - and revocation of access rights, 640–641
  - security vs., 657
  - static vs. dynamic, 628
  - from viruses, 694–696
- protection domain, 628, 721**
- protection mask (Linux), 823**
- protection subsystems (Windows 7), 838**
- protocols:**
  - discovery, 39
  - Windows 7 networking, 871–873
- providers (DTrace), 89**
- pseudo-device driver, 730–731**
- PTBR (page-table base register), 372**
- PTEs (page-table entries), 847**
- PTE tables, 847**
- Pthreads, 172–174**
  - scheduling, 277–278
  - synchronization in, 237–238
  - thread cancellation in, 186–187
- PTLR (page-table length register), 376**
- public cloud, 41**
- public domain, 785**

public keys, 678  
 public-key encryption, 678  
 pull migration, 281  
 pure code, 376  
 pure demand paging, 404  
 push, 32  
 push migration, 281, 769

## Q

quantum, 839  
 queue(s), 111–112  
   capacity of, 129–130  
   input, 354  
   message, 897  
   ready, 111–112, 359  
 queueing diagram, 112  
 queueing-network analysis, 302

## R

race condition, 205  
 RAID (redundant arrays of inexpensive disks), 484–494  
   levels of, 487–491  
   performance improvement, 486  
   problems with, 492–494  
   reliability improvement, 485–486  
   structuring, 485  
 RAID array, 485  
 RAID levels, 487–491  
 RAID sets, 868  
 RAM (random-access memory), 9  
 random-access devices, 598, 599, 893  
 random-access memory (RAM), 9  
 random-access time (disks), 468  
 rate, periodic task, 286  
 rate-monotonic scheduling, 287–288  
 rate-monotonic scheduling algorithm, 287–288  
 raw disk, 421, 516, 549  
 raw I/O, 600  
 raw partitions, 483  
 RBAC (role-based access control), 639  
 RC4, 677  
 RC 4000 operating system, 897  
 RDP, 717  
 read-ahead technique, 567  
 read-end (of pipe), 142  
 readers, 220  
 reader-writer locks, 220–222  
 readers-writers problem, 220–222  
 reading files, 506  
 read-modify-write cycle, 489  
 read only devices, 599  
 read-only memory (ROM), 93, 480  
 read queue, 817  
 read-write devices, 599  
 ready queue, 111–112, 359  
 ready state, 107  
 ready thread state (Windows 7), 839  
 real-time class, 294  
 real-time CPU scheduling, 283–290  
   earliest-deadline-first scheduling, 288–289  
   and minimizing latency, 283–285  
   POSIX real-time scheduling, 290  
   priority-based scheduling, 285–287  
   proportional share scheduling, 289–290  
   rate-monotonic scheduling, 287–288  
 real-time embedded systems, 43  
 real-time operating systems, 43  
 real-time range (Linux schedulers), 796  
 real-time systems, 43  
 reconfiguration, 762–763  
 records:  
   logical, 513  
   master boot, 480  
 recovery:  
   backup and restore, 570–571  
   and consistency checking, 568–569  
   from deadlock, 337–338  
     by process termination, 337–338  
     by resource preemption, 338  
   from failure, 763  
   of files and directories, 568–571  
   Windows 7, 866–867  
 red-black trees, 35  
 Red Hat, 785  
 redirectors, 873  
 redundancy, 485. *See also* RAID  
 redundant arrays of inexpensive disks, *see* RAID  
 Reed-Solomon codes, 489–490  
 reentrant code (pure code), 376  
 reference bits, 418  
 Reference Model, ISO, 682  
 reference string, 412

- register(s)**, 65
    - base, 352
    - limit, 352
    - memory-address, 355
    - page-table base, 372
    - page-table length, 376
    - for page tables, 372
    - relocation, 356
  - registry**, 74, 861
  - relative block number**, 514–515
  - relative path names**, 522
  - relative speed**, 207
  - release() operation**, 508
  - reliability**, 755
    - of distributed operating systems, 742–743
    - of Windows 7, 832–833
  - relocation register**, 356
  - remainder section**, 206
  - remote file systems**, 529
  - remote file transfer**, 744–745
  - remote login**, 744
  - remote operations**, 577
  - remote procedure calls (RPCs)**, 872
  - remote-service mechanism**, 770
  - removable storage media**:
    - magnetic disks, 467–469
    - magnetic tapes, 469–470
  - rendezvous**, 129
  - repair, mean time to**, 485
  - replay attacks**, 658
  - replication**, 491, 579–580
  - repositioning (in files)**, 506
  - request edge**, 319
  - request manager**, 816
  - resident attributes**, 865
  - resident monitor**, 890
  - resolution**:
    - name, 751–753
    - and page size, 441
  - resolving links**, 524
  - resource allocation (operating system service)**, 57
  - resource-allocation graph algorithm**, 329–330
  - resource allocator, operating system as**, 5
  - resource preemption, deadlock recovery by**, 338
  - resource-request algorithm**, 332
  - resource sharing**, 165, 742
  - resource utilization**, 5
  - response time**, 20, 265–266
  - restart area**, 867
  - restore**:
    - data, 570–571
    - state, 114
  - resume**, 715
  - reverse engineering**, 44
  - revocation of access rights**, 640–641
  - rich text format (RTF)**, 694
  - right child**, 33
  - rights amplification (Hydra)**, 642
  - risk assessment**, 690–691
  - roaming profiles**, 874
  - robustness**, 762–764
  - roles**, 639
  - role-based access control (RBAC)**, 639
  - roll out, roll in**, 358
  - ROM (read-only memory)**, 93, 480
  - root partitions**, 549
  - root uid (Linux)**, 823
  - rotational latency (disks)**, 468, 473
  - round-robin (RR) scheduling algorithm**, 271–273
  - routers**, 754
  - routing**:
    - and network communication, 753–754
  - routing protocols**, 754
  - routing table**, 753
  - RPCs (remote procedure calls)**, 872
  - RR scheduling algorithm**, 271–273
  - RTF (rich text format)**, 694
  - running state**, 107
  - running system**, 93
  - running thread state (Windows 7)**, 839
  - RW (read-write) format**, 27
- S**
- SaaS (software as a service)**, 42
  - safe computing**, 694
  - safe sequence**, 328
  - safety algorithm**, 331–332
  - sandbox (Tripwire file system)**, 694
  - SANs, *see* storage-area networks**
  - SATA buses**, 469
  - save, state**, 114
  - scalability**, 166, 765
  - Scala language**, 241–242
  - SCAN (elevator) scheduling algorithm**, 475–476

- scheduler(s)**, 112–113
  - long-term, 112
  - medium-term, 113
  - short-term, 113
- scheduler activation**, 187–188
- scheduling**:
  - cooperative, 264
  - CPU, *see* CPU scheduling
  - disk scheduling algorithms, 472–478
    - C-SCAN, 476
    - FCFS, 473–474
    - LOOK, 477
    - SCAN, 475–476
    - selecting, 477–478
    - SSTF, 474–475
  - earliest-deadline-first, 288–289
  - I/O, 604–605
  - job, 20
  - in Linux, 795–800
    - kernel synchronization, 798–800
    - process, 796–797
    - symmetric multiprocessing, 800
  - multiprocessor, *see* multiprocessor scheduling
  - nonpreemptive, 264
  - preemptive, 263–264
  - priority-based, 285–287
  - proportional share, 289–290
  - rate-monotonic, 287–288
  - SSDs and, 478
  - thread, 277–278
  - in Windows 7, 839–841, 877–881
- scheduling rules**, 877
- SCM (Service Control Manager)**, 860
- SCOPE operating system**, 904
- script kiddies**, 666
- SCS (system-contention scope)**, 277
- SCSI (small computer-systems interface)**, 12
- SCSI buses**, 469
- search path**, 521
- secondary memory**, 404
- secondary storage**, 10, 543. *See also* disk(s)
- second-chance page-replacement algorithm (clock algorithm)**, 418–419
- second extended file system (ext2)**, 811
- section objects**, 135
- sectors, disk**, 468
- sector slipping**, 481–482
- sector sparing**, 481, 869
- secure by default**, 669
- secure single sign-on**, 531
- secure systems**, 658
- security**. *See also* file access; program threats; protection; user authentication
  - classifications of, 698–699
  - in computer systems, 29–31
  - and firewalling, 696–698
  - implementation of, 689–696
    - and accounting, 696
    - and auditing, 696
    - and intrusion detection, 691–694
    - and logging, 696
    - and security policy, 689–690
    - and virus protection, 694–696
    - and vulnerability assessment, 690–691
  - levels of, 659–660
  - in Linux, 821–824
    - access control, 822–824
    - authentication, 822
  - as operating system service, 57–58
  - as problem, 657–661
  - protection vs., 657
  - and system/network threats, 669–674
    - denial of service, 674
    - port scanning, 673
    - worms, 670–673
  - use of cryptography for, 674–685
    - and encryption, 674–685
    - implementation, 681–683
    - SSL example, 683–685
  - via user authentication, 685–689
    - biometrics, 689
    - passwords, 685–689
  - in Windows 7, 699–702, 831–832, 867
- security access tokens (Windows 7)**, 699
- security context (Windows 7)**, 699–702
- security descriptor (Windows 7)**, 701
- security domains**, 696
- security identity (SID)**, 853
- security policy**, 689–690
- security reference monitor (SRM)**, 858–859
- security-through-obscurity approach**, 691
- security tokens**, 853
- seeds**, 688–689
- seek, file**, 506
- seek time (disks)**, 468, 473
- segmentation**, 364–366
  - basic method, 364–366
  - defined, 364
  - hardware, 365–366



- Intel 32 and 64-bit architectures example, 384–385
- segment base**, 366
- segment limit**, 366
- segment tables**, 366
- semantics**:
  - consistency, 532–533
  - copy, 606
  - immutable-shared-files, 533
  - session, 533
- semaphore(s)**, 213–218
  - binary, 214
  - counting, 214
  - and deadlocks, 217
  - defined, 213
  - implementation, 215–217
  - implementation of monitors using, 229–230
  - and priority inversion, 217–218
  - and starvation, 217
  - usage of, 214–215
- semaphore objects (Windows 7)**, 841
- semiconductor memory**, 11
- sense key**, 608
- sequential access (files)**, 513
- sequential-access devices**, 893
- sequential devices**, 598, 599
- serial ATA (SATA) buses**, 469
- server(s)**, 5
  - defined, 766
  - in SSL, 683
- server-message-block (SMB)**, 871
- server subject (Windows 7)**, 700
- Service Control Manager (SCM)**, 860
- services, operating system**, 55–58, 115
- session hijacking**, 659
- session layer**, 757
- session manager subsystem (SMSS)**, 862
- session object**, 847
- session semantics**, 533
- session space**, 846
- sharable devices**, 598, 599
- shares**, 299
- shared files, immutable**, 533
- shared libraries**, 358, 400
- shared lock**, 508
- shared memory**, 122, 400
- shared-memory model**, 73, 124–126
- sharing**:
  - load, 278, 742
  - and paging, 376–377
  - resource, 742
  - time, 20
- shells**, 58
- shell script**, 511
- shortest-job-first (SJF) scheduling**
  - algorithm, 267–270
- shortest-remaining-time-first scheduling**, 269
- shortest-seek-time (SSTF) scheduling**
  - algorithm, 474–475
- short-term scheduler (CPU scheduler)**, 113, 263
- shoulder surfing**, 686
- SID (security identity)**, 853
- signals**:
  - Linux, 818
  - UNIX, 183–185
- signaled state**, 233
- signal handlers**, 183–185
- signatures**, 692–693
- signature-based detection**, 692
- simple operating system structure**, 78
- simple subject (Windows 7)**, 700
- simulation(s)**, 302
- single indirect blocks**, 559
- single-level directories**, 518–519
- single-processor systems**, 13–14, 261
- single-threaded processes**, 163
- singly linked lists**, 32
- SJF scheduling algorithm**, 267–270
- Skype**, 40
- slab allocation**, 437–439, 802–803
- Slackware**, 785
- slices**, 516
- slim reader-writer (SRW) locks**, 879
- SLOB allocator**, 439
- SLUB allocator**, 439
- small-area networks**, 37
- small computer-systems interface**, *see under* SCSI
- SMB (server-message-block)**, 871
- SMP**, *see* symmetric multiprocessing
- SMSS (session manager subsystem)**, 862
- snapshots**, 570
- sniffing**, 686
- social engineering**, 660
- sockets**, 136–138
- socket interface**, 601
- soft affinity**, 280
- soft error**, 479
- software as a service (SaaS)**, 42

- software capability, 643
- software interrupts (traps), 593
- software objects, 627
- Solaris, 46
  - and processor affinity, 280
  - scheduling example, 297–299
  - swap-space management in, 483–484
  - synchronization in, 235–237
  - virtual memory in, 446–447
- solid-state disks (SSDs), 11, 28, 469, 478
- sorted queue, 817
- source-code viruses, 667
- source files, 504
- space maps, 563–564
- SPARC, 383
- sparseness, 381, 400
- spawn, 670
- speed:
  - of operations (I/O devices), 599
  - relative, 207
- spoofed client identification, 530
- spoofing, 697
- spool, 607
- spooling, 607, 893–894
- spyware, 662
- SRM (security reference monitor), 858–859
- SRW (slim reader-writer) locks, 879
- SSDs, *see* solid-state disks
- SSL 3.0, 683–685
- SSTF scheduling algorithm, 474–475
- stable storage, 494–496
- stack, 65, 106
- stack algorithms, 417
- stack frame, 664–665
- stack inspection, 648
- stack-overflow attacks, 663–666
- stalling, 352
- standard swapping, 358–360
- standby thread state (Windows 7), 839
- starvation, *see* indefinite blocking
- state (of process), 107
- stateless DFS, 532
- state restore, 114
- state save, 114
- static linking, 357–358, 809
- static protection, 628
- status information, 74
- status register, 590
- stealth viruses, 668
- storage, 9–12. *See also* mass-storage structure
  - definitions and notations, 9
  - nonvolatile, 11–12
  - secondary, 10, 543
  - tertiary, 27
  - thread-local, 187
  - utility, 493
  - volatile, 11
- storage-area networks (SANs), 18, 471, 472
- storage array, 485
- storage management, 26–30
  - caching, 27–29
  - I/O systems, 29–30
  - mass-storage management, 27
  - with virtual machines, 732–733
- stored program computers, 888
- stream ciphers, 677
- stream head, 613
- stream modules, 613–614
- STREAMS mechanism, 613–615
- string, reference, 412
- stripe set, 868
- stubs, 357
- stub routines, 872
- subsystems, 135
- SunOS, 46
- superblock, 546
- superblock objects, 551, 809
- superuser, 688
- supervisor mode, *see* kernel mode
- Surface Computer, 863
- SuSE, 785
- suspended state, 715, 878
- swap map, 484
- swapper (term), 401
- swapping, 20, 113, 358–360, 401
  - in Linux, 806
  - on mobile systems, 360, 407
  - paging vs., 482
  - standard, 358–360
- swap space, 404
- swap-space management, 482–484
- switch architecture, 12
- switching:
  - circuit, 755
  - domain, 629
  - fast-user, 863–864
  - message, 755
  - packet, 755–756

symbolic links, 845  
 symbolic-link objects, 845  
 symmetric coupling, 17  
 symmetric encryption, 676–677  
 symmetric encryption algorithm, 676  
 symmetric mode, 17  
 symmetric multiprocessing (SMP), 15–16, 279, 800  
 synchronization, 129. *See also* process synchronization  
 synchronous devices, 598, 599  
 synchronous message passing, 129  
 synchronous threading, 172  
 synchronous writes, 567  
 SYSGEN, 91–92  
 system boot, 92–93  
 system calls (monitor calls), 8, 62–65  
   and API, 63–64  
   for communication, 72–73  
   for device management, 71–72  
   for file management, 71  
   functioning of, 62–65  
   for information maintenance, 72  
   for process control, 67–71  
 system-call firewalls, 698  
 system-call interface, 64  
 system-contention scope (SCS), 277  
 system daemons, 8  
 system-development time, 715  
 system disk, *see* boot disk  
 system files, 520  
 system generation (SYSGEN), 91–92  
 system hive, 861  
 system libraries (Linux), 787, 788  
 system mode, *see* kernel mode  
 system processes, 8, 844–845  
 system programs, 74–75  
 systems programs, 6  
 system resource-allocation graph, 319–321  
 system restore point, 861  
 system utilities, 74–75, 787, 788  
 system-wide open-file table, 546

## T

**table(s)**, 398  
   file-allocation, 557  
   hash, 552–553  
   master file, 546  
   mount, 549, 611  
   open-file, 507  
   page, 404, 847  
   per-process open-file, 547  
   routing, 753  
   segment, 366  
   system-wide open-file, 546  
 tags, 637  
 tapes, magnetic, 469–470  
 target latency, 797  
 target thread, 185  
 task control blocks, *see* process control blocks  
 task parallelism, 168–169  
 TCB (trusted computer base), 698  
 TCP/IP, *see* Transmission Control Protocol/Internet Protocol  
 TCP sockets, 137  
 TDI (transport driver interface), 870  
 TEBs (thread environment blocks), 880  
 telnet, 616, 744  
 terminal applications, 96  
 terminal concentrators, 616  
 terminal server systems, 864  
 terminated state, 107  
 terminated thread state (Windows 7), 839  
 termination:  
   cascading, 121  
   process, 120–121, 337–338  
 tertiary storage devices, 27  
 text files, 504  
 text section (of process), 106  
 theft of service, 658  
 THE operating system, 896–897  
 thin clients, 35  
 third extended file system (ext3), 811–813  
 thrashing, 425–430  
   cause of, 426–427  
   defined, 426  
   and page-fault-frequency strategy, 429–430  
   and working-set model, 427–429  
 threads. *See also* multithreading  
   cancellation, thread, 185–186  
   components of, 163  
   functions of, 163–166  
   idle, 294  
   implicit threading, 177–183  
   kernel, 169  
   in Linux, 189–191, 795  
   and multicore programming, 166–169

- threads.** (*contd.*)
    - and process model, 109
    - scheduling of, 277–278
    - target, 185
    - user, 169
    - in Windows 7, 188–189, 839–841, 876, 878–880
  - thread attach, 853**
  - thread environment blocks (TEBs), 880**
  - thread libraries, 171–177**
    - about, 171–172
    - Java threads, 176–177
    - Pthreads, 172–174
    - Windows threads, 174–176
  - thread-local storage, 187**
  - thread pools, 179–181, 879**
  - thread scheduling, 261**
  - thread-specific data, 187**
  - threats, 658. *See also* program threats**
  - throughput, 265**
  - thunking, 834**
  - tightly coupled systems, *see* multiprocessor systems**
  - time:**
    - compile, 354
    - effective access, 405
    - effective memory-access, 374
    - execution, 355
    - of file creation/use, 504
    - load, 354
    - response, 20, 265–266
    - turnaround, 265
    - waiting, 265
  - time-out schemes, 762**
  - time profiles, 72**
  - time quantum, 271**
  - timer:**
    - programmable interval, 601
    - variable, 24
  - timers, 601–602**
  - timer objects, 841**
  - time sharing (multitasking), 20, 115**
  - time slice, 796**
  - timestamp counters (TSCs), 840–841**
  - TLB, *see* translation lookaside buffer**
  - TLB miss, 373**
  - TLB reach, 441–442**
  - top half interrupt service routines, 799**
  - TOPS-20, 900**
  - Torvalds, Linus, 781**
  - touch screen (touchscreen computing), 5, 60**
  - trace tapes, 303**
  - tracks, disk, 468**
  - traditional computing, 35–36**
  - transactions:**
    - atomic, 210
    - defined, 813
    - in Linux, 813–814
    - in log-structured file systems, 569–570
  - transactional memory, 239–240**
  - transfer rate (disks), 468, 470**
  - transition thread state (Windows 7), 839**
  - translation lookaside buffer (TLB), 373–375, 849–850**
  - transmission control protocol (TCP), 758**
  - Transmission Control Protocol/Internet Protocol (TCP/IP), 758–761, 871**
  - transparency, 764, 766, 767**
  - transport driver interface (TDI), 870**
  - transport layer, 757**
  - transport-layer protocol (TCP), 681**
  - traps, 21, 403, 594**
  - trap-and-emulate method, 717–718**
  - trap doors, 662**
  - trees, 33, 35**
  - tree-structured directories, 521–522**
  - triple DES, 677**
  - triple indirect blocks, 559**
  - Tripwire file system, 695**
  - Trojan horses, 661–662**
  - trusted computer base (TCB), 698**
  - TSCs (timestamp counters), 840–841**
  - tunneling viruses, 668**
  - turnaround time, 265**
  - turnstiles, 236**
  - two-factor authentication, 688**
  - twofish algorithm, 677**
  - two-level directories, 519–521**
  - two tuple, 365**
  - type 0 hypervisors, 712, 723–724**
  - type 1 hypervisors, 712, 724–725**
  - type 2 hypervisors, 713, 725**
  - type safety (Java), 649**
- 
- U**
- UAC (User Account Control), 701**
  - UDP (user datagram protocol), 758**
  - UDP sockets, 137**

UFD (user file directory), 519  
 UFS (UNIX file system), 545  
 UI (user interface), 52–55  
 UMA (uniform memory access), 16  
 UMDF (User-Mode Driver Framework), 856  
 UMS, *see* user-mode scheduling  
 unbounded capacity (of queue), 130  
 UNC (uniform naming convention), 872  
 UNICODE, 837  
 unified buffer cache, 565  
 unified virtual memory, 565  
 Unifix, 785  
 uniform memory access (UMA), 16  
 uniform naming convention (UNC), 872  
 universal serial buses (USBs), 469  
 UNIX file system (UFS), 545  
 UNIX operating system:  
   consistency semantics for, 532–533  
   domain switching in, 629–630  
   feature migration with, 887, 888  
   and Linux, 781  
   permissions in, 536  
   signals in, 183–185  
   swapping in, 360  
 unreliability, 755  
 upcalls, 188  
 upcall handler, 188  
 U.S. Digital Millennium Copyright Act (DMCA), 44  
 USBs (universal serial buses), 469  
 used objects, 438, 803  
 users, 4–5, 528–529  
 user accounts, 699  
 User Account Control (UAC), 701  
 user authentication, 685–689  
   with biometrics, 689  
   with passwords, 685–689  
 user datagram protocol (UDP), 758  
 user-defined signal handlers, 184  
 user file directory (UFD), 519  
 user identifiers (user IDs), 31  
   effective, 31  
   for files, 504  
 user interface (UI), 52–55  
 user mobility, 573  
 user mode, 22, 787  
 User-Mode Driver Framework (UMDF), 856

user-mode scheduling (UMS), 296–297, 835, 880–881  
 user-mode threads (UT), 844  
 user programs (user tasks), 105–106, 807  
 user rights (Linux), 822–823  
 user threads, 169  
 UT (user-mode threads), 844  
 utilities, 888  
 utility storage, 493  
 utilization, 889

## V

VACB (virtual address control block), 857  
 VADs (virtual address descriptors), 852  
 valid-invalid bit, 375  
 variables:  
   automatic, 664  
   condition, 879  
 variable class, 294  
 variable timer, 24  
 VAX minicomputer, 379–380  
 VCPU (virtual CPU), 717  
 vectored I/O, 603–604  
 vector programs, 670  
 vfork() (virtual memory fork), 409  
 VFS, *see* virtual file system  
 victim frames, 411  
 views, 847  
 virtual address, 356  
 virtual address control block (VACB), 857  
 virtual address descriptors (VADs), 852  
 virtual address space, 398–399, 805–806  
 virtual CPU (VCPU), 717  
 virtual file system (VFS), 550–552, 809–811  
 virtualization, 40–41  
   advantages and disadvantages of, 714–716  
   and application containment, 727–728  
   and emulation, 727  
   and operating-system components, 728–735  
   CPU scheduling, 729  
   I/O, 731–732  
   live migration, 733–735  
   memory management, 730–731  
   storage management, 732–733  
   para-, 725–726  
   programming-environment, 726–727

**virtual machines, 711–738. See also****virtualization**

advantages and disadvantages of, 714–716  
 and binary translation, 718–720  
 examples, 735–737  
 features of, 715–717  
 and hardware assistance, 720–721  
 history of, 713–714  
 Java Virtual Machine, 736–737  
 life cycle of, 722–723  
 trap-and-emulate systems, 717–718  
 type 0 hypervisors, 723–724  
 type 1 hypervisors, 724–725  
 type 2 hypervisors, 725  
 VMware, 735–736

**virtual machine control structures (VMCSs), 721****virtual machine manager (VMM), 22–23, 41, 712****virtual machine sprawl, 723****virtual memory, 20–21, 397–400**

and copy-on-write technique, 408–409  
 demand paging for conserving, 401–407  
   basic mechanism, 402–405  
   with inverted page tables, 442  
   and I/O interlock, 444–445  
   and page size, 440–441  
   and performance, 405–406  
   and prepaging, 439–440  
   and program structure, 442–443  
   pure demand paging, 404  
   and restarting instructions, 404–405  
   and TLB reach, 441–442

direct virtual memory access, 596

and frame allocation, 421–425

  equal allocation, 423

  global vs. local allocation, 424

  proportional allocation, 423–424

kernel, 806–807

and kernel memory allocation, 436–439

in Linux, 804–807

and memory mapping, 430–436

  basic mechanism, 430–432

  I/O, memory-mapped, 435–436

  in Win32 API, 433–435

page replacement for conserving, 409–421

  and application performance, 421

  basic mechanism, 410–413

  counting-based page replacement, 420

  FIFO page replacement, 413–414

  LRU-approximation page replacement, 418–420

  LRU page replacement, 416–418

  optimal page replacement, 414–415

  and page-buffering algorithms, 420–421

  separation of logical memory from

    physical memory by, 398–399

  size of, 398

  in Solaris, 446–447

  and thrashing, 425–430

    cause, 426–427

    page-fault-frequency strategy, 429–430

    working-set model, 427–429

  unified, 565

  in Windows, 445–446

**virtual memory fork, 409****virtual memory (VM) manager, 846–852****virtual memory regions, 805****virtual private networks (VPNs), 682, 871****virtual routing, 753–754****viruses, 666–669, 694–696****virus dropper, 667****VMCSs (virtual machine control structures), 721****VMM, see virtual machine manager****VM manager, 846–852****VMware, 714, 735–736****vnode, 550****volatile storage, 11****volumes, 516****volume control block, 546****volume management (Windows 7), 868–869****volume shadow copies, 870****volume table of contents, 516****von Neumann architecture, 10****VPNs, see virtual private networks****vulnerability scans, 690–691****W****WAFL file system, 570, 577–580****waiting state, 107****waiting thread state (Windows 7), 839****waiting time, 265****wait queue, 818****wait() system call, 120–122****WANs, see wide-area networks**



- Web distributed authoring and versioning (WebDAV), 872**
- wide-area networks (WANs), 17, 37, 749–750**
- WiFi networks, *see* wireless networks**
- Win32 API, 433–435, 830–831, 875**
- Windows operating system (generally), 360, 901–902**
  - interprocess communication example, 135
  - scheduling example, 294–296
  - threads example, 188–189
  - virtual memory in, 445–446
- Windows 7, 829–884**
  - application compatibility of, 833–834
  - design principles for, 831–837
  - desktop versions of, 830–831
  - dynamic device support, 837, 838
  - and energy efficiency, 837
  - extensibility of, 836
  - fast-user switching with, 863–864
  - file system, 864–870
    - change journal, 870
    - compression and encryption, 869
    - mount points, 869–870
    - NTFS B+ tree, 865
    - NTFS internal layout, 864–866
    - NTFS metadata, 866
    - recovery, 866–867
    - security, 867
    - volume management and fault tolerance, 868–869
    - volume shadow copies, 870
  - history of, 829–831
  - networking, 870–875
    - Active Directory, 875
    - domains, 874
    - interfaces, 870
    - protocols, 871–873
    - redirectors and servers, 873–874
  - performance of, 834
  - portability of, 836
  - programmer interface, 875–884
    - interprocess communication, 881–882
    - kernel object access, 875
    - memory management, 882–884
    - process management, 876–879
    - sharing objects between processes, 875–876
  - reliability of, 832–833
  - security in, 700–701, 831–832
  - synchronization in, 833–834, 878–879
  - system components for, 838–863
    - executive, *see* Windows executive
    - hardware-abstraction layer, 838–839
    - kernel, 839–844
    - terminal services, 863–864
    - user-mode scheduling in, 296–297
- Windows 2000, 832, 833, 836**
- Windows executive, 844–863**
  - booting, 862–863
  - cache manager, 856–858
  - client-server computing, 854–855
  - I/O manager, 855–856
  - object manager, 844–846
  - plug-and-play manager, 860
  - power manager, 860–861
  - process manager, 852–853
  - registry, 861–862
  - security reference monitor, 858–859
  - virtual memory manager, 846–852
- Windows group policy, 875**
- Windows NT, 829–830**
- Windows Task Manager, 87, 88**
- Windows thread library, 174–176**
- Windows Vista, 830**
  - security in, 700
  - symbolic links in, 869–870
- Windows XP, 830**
- Winsock, 881**
- wireless (WiFi) networks, 35, 748–749**
- Witness, 326**
- word, 9**
- WorkGroup Solutions, 785**
- working sets, 427, 431**
- working-set maximum, 446**
- working-set minimum, 446**
- working-set model, 427–429**
- Workstation (VMWare), 735–736**
- workstations, 5**
- world rights (Linux), 823**
- World Wide Web, 529**
- worms, 670–673**
- WORM (write-once, read-many) format, 27**
- worst-fit strategy, 363**
- write-end (of pipe), 142**
- write only devices, 599**
- write queue, 817**
- writers, 220**
- writing files, 506**

## X

x86-64 architecture, 387  
XDR (external data representation), 140  
XDS-940 operating system, 895–896  
Xen, 714  
Xerox, 59  
XML firewalls, 698

## Z

zero capacity (of queue), 130  
zero-day attacks, 693  
zero-fill-on-demand technique, 409  
ZFS file system, 563–564, 570  
zombie systems, 673  
zones, 728, 801