

Operating Systems and Middleware: Supporting Controlled Interaction
by Max Hailperin

The commercially published version of this work (ISBN 0-534-42369-8) was Copyright © 2007 by Thomson Course Technology, a division of Thomson Learning, Inc., pursuant to an assignment of rights from the author.

This free re-release is Copyright © 2005-2010 by Max Hailperin, pursuant to an assignment of the rights back to him by Course Technology, a division of Cengage Learning, Inc., successor-in-interest to the publisher. Rights to illustrations rendered by the publisher were also assigned by Course Technology to Max Hailperin and those illustrations are included in the license he grants for this free re-release.

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 United States License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/us/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

The free re-release was prepared from final page proofs and should be completely identical to the commercially published version. In particular, all the errata listed on the web site still apply. (The author intends to release subsequent versions that incorporate the corrections as well as updates and improvements. Subsequent versions may also be in a more easily modifiable form to encourage participation by other contributors. Please email suggestions to max@gustavus.edu.)

Credits from the commercially published version:

Senior Product Manager: Alyssa Pratt
Managing Editor: Mary Franz
Development Editor: Jill Batistick
Senior Marketing Manager: Karen Seitz
Associate Product Manager: Jennifer Smith
Editorial Assistant: Allison Murphy
Senior Manufacturing Coordinator: Justin Palmeiro
Cover Designer: Deborah VanRooyen
Compositor: Interactive Composition Corporation

Index

Note:

- Page numbers in bold type indicate term introductions/definitions.
- Page numbers in italic type indicate illustrations.
- Page numbers followed by *q* indicate quotations.
- Page numbers followed by (2) indicate two separate discussions.
- Cross-reference targets beginning with “:—” indicate the current main heading in cross-references from one subheading to another.

Symbols

& (ampersand): shell command character, 228
 *-property, **410**
 > (greater than sign): redirect standard output symbol, 276
 < (less than sign): redirect standard input symbol, 276
 + (plus sign): wildcard character, 372–373
 # (pound sign): wildcard character, 373
 / (slash character): pathname delimiter, 276

Numbers

3DES (encryption standard), 362
 32-bit systems:
 address space, 186–187
 multilevel page tables, 190
 and multiple address space systems, 234
 64-bit systems:
 address space, 187
 and multiple address space systems, 234

A

abort actions (operations), **125**, 129
 compensating for not executing in workflow systems, 134–135
 upon restart, 148–149
 and undo logs, 148
 abstract datatypes, 314
 abstraction:
 of hardware resources, 4
 of transactions, 157, 158
 access control bits:
 linear page table entry valid bits, 185–186, **186**, 237
 in protection key registers, 237
 access control lists. *See* ACLs

access matrix/matrices, 238, 239
 capabilities and, 244
 change mechanisms, 238–239
 fitting protection mechanisms into, 240
 general model, 240–241
 print resource on, 240, 267
 access permissions. *See* permissions (file access permissions)
 access points (in networks), **325**, 356
 access rights (of processes), 221
 control policy categories, 241
 directed. *See* capabilities
 fundamentals, 237–241
 mechanism for granting, 228–229
 sets (protection domains), 238; structure. *See* access matrix/matrices; switching mechanism, 228–229, 238(2)
 See also ACLs (access control lists); permissions
 access services, **269**
 forms, 270
 access time. *See* disk access time
 accountability (security objective), **16**, **398**
 ACID test of system support for transactions, 125
 print resource on, 163
 ACLs (access control lists), 245–252, 300–301
 for files, 250
 MAC systems with, 410
 messaging system use of, 387
 Microsoft Windows example, 245–246, **246–248**
 NTFS storage of, 301
 permission specification, 250; sticky bit limitation, 251
 restricted form (POSIX), 249–252
 RPC server use of, 387
 as used with capabilities, 251–252
 activation record stacks. *See* stacks

- activation records, **428**, **433**
 - local array buffer overflow/input overwrite vulnerabilities, **412–414**, **414**, **433**
 - register saves and restores as into and from, **432–433**
- activations. *See* procedure activations
- address mapping (translation), **168–169**, **180–181**, **181**, **182**
 - granularity, **180**
 - hashed table mapping, **196**, **196**
 - IA-32 paged mapping, **191–193**, **192**
 - limitation mechanism, **169**
 - linear page table mapping, **186**, **186**, **188–189**, **189**
 - process switching and, **185**
 - recent address translation storage. *See* TLBs (translation lookaside buffers)
 - tables for. *See* address-mapping table; page tables; segment table
- Address Resolution Protocol (ARP), **356**
- address space (virtual address space):
 - copying, **223**
 - identifiers (ASIDs), **185**
 - multiple address space systems, **234–235**
 - as private storage, **169**, **170–171**
 - of processes, **172**; child processes, **223**
 - protection schemes, **231–232**, **234–237**
 - regions, **171**
 - segmentation of. *See* segmentation (of address space)
 - single address space systems, **235–237**; print resources on, **267**
 - sparse address space allocation, **176–177**
 - thread groups sharing. *See* processes
- address space identifiers (ASIDs), **185**
- address translation. *See* address mapping (translation)
- address-mapping function, **168**
 - See also* address-mapping table; page tables; segment table; TLBs (translation lookaside buffers)
- address-mapping table, **168–169**
 - page type. *See* page tables
 - segment table, **198**
- addressable capabilities, **243**
 - preventing forgery of, **243–244**
 - print resources on, **267**
 - research systems, **267**
- addresses (memory addresses), **165–169**
 - mapping of. *See* address mapping (translation)
 - processor vs. memory uses of, **166–167**
 - return addresses, **431–432**
 - See also* physical addresses; virtual addresses
- Advanced Encryption Standard (AES), **362**
- adversaries (of systems), **397**, **401**
 - arbitrary code execution by, **412–413**
 - decoy systems (honeypots) for, **419**
- Adya, Atul, **164**
- AES (Advanced Encryption Standard), **362**
- affinity: processor affinity (for threads), **45**, **46**
- Aho, Alfred V., **219**
- aliases: domain names as, **336**
- allocate operation (stacks), **429–430**, **431**
- AMD64 architecture:
 - multilevel page tables, **194**
 - print resource on, **218**
- ampersand (&): shell command character, **228**
- Anderson, Dave, **322**
- anomaly detection: with IDSes, **360–361**, **418**
- antivirus scanning programs, **414**
- API services as provided by operating systems or middleware, **6**
- APIs (Application Programming Interfaces), **4–5**
 - file APIs, **13**
 - interfaces used in this book, **22**
 - portable and non-portable APIs, **22**
 - socket APIs, **340–344**
 - threads support, **10**
 - transport protocol support, **331**
 - See also* Java API; POSIX API; pthreads API
- application layer (in networking systems), **14–15**, **329**, **330**, **332–339**
 - security problems and enhancements, **358**
 - Web communication example, **332–334**
- Application Programming Interfaces. *See* APIs
- application programs (programs), **14–15**
 - address use, **166**
 - application protocol support, **331**
 - avoiding quadratic growth in code size, **162–163**
 - demand-driven loading, **178**
 - distributed applications, **15**
 - gift programs and Trojan horses, **258**
 - library support–application systems, **4**
 - limiting program actions, **411**
 - loading and running, **224–227**; programming examples, **226**, **227**
 - middleware and, **6–7**; distributed applications, **15**
 - multiple-threaded. *See* multiple-threaded programs
 - operating system–application process boundary, **12**, **232–233**, **232**, **233**
 - operating systems and, **4–5**
 - running: loading and running, **224–227**; programming examples, **226**, **227**, **277**; from shells, **225**, **227–228**, **276–278**
 - single-threaded programs, **20**
 - spatial locality, **45**
 - temporal locality, **45**
 - thread–program distinction, **19**
 - write permission vulnerability, **258**
- application protocols, **328**, **330**, **332**
 - SSL (Secure Sockets Layer), **358**
 - support sources, **331–332**

446 ► Index

- arbitrary code execution by adversaries, 412–413
- ARP (Address Resolution Protocol), 356
- arrays:
 - local array buffer overflow vulnerabilities/attacks, 412–414, 414, 433
 - storing buffers with `Lists` vs., 100
- AS/400 (IBM), 237
 - print resource on, 267
- ASIDs (address space identifiers), 185
 - segmentation-ASID comparison, 201
- associativity (of caches), 204
- assurance (product security assurance), 414–417
- assurance requirements:
 - at EALs, 415–416
 - in PPs and STs, 415
- asymmetric-key encryption (cryptography), 362
 - RSA system, 362–363
- Atlas computer, 217, 218
 - print resources on, 218
- atomic exchange operations, 83–84
- atomic transactions. *See* transactions
- atomicity (of exchange operations), 83–84
- atomicity (of transactions), 124–125, 125–126
 - aspects, 124, 137
 - failure atomicity, 124, 144–145; ad hoc programming difficulties, 145, 146, 157; and durability, 148–149; in system crashes, 164; undo logging (transaction processing system), 145–147
 - mechanisms for ensuring, 137–147
- attacks:
 - buffer overflow attacks, 412–414, 414
 - denial of service (DoS) attacks, 65–67, 157, 358, 412
 - isolation of machines exposed to, 360
 - replay attacks, 406
 - risk-management approach to, 397
 - rollback of transactions as a sign of, 157
 - spoofing attacks, 402–403, 402
 - See also* vulnerabilities
- authentication (of users), 16, 388–390, 398
 - basic authentication, 388
 - biometric authentication, 405, 406
 - mutual authentication, 389
 - password authentication: with cryptographic hash functions, 363, 364, 403–404; weaknesses, 388, 401–402
 - as the prerequisite of security, 398
 - two-factor authentication, 405–406
- availability (security objective), 16, 398
- B**
- B-trees (balanced search trees), 290, 291, 297–300, 309–310
 - vs. binary search trees, 298
 - inserting an entry into a full node, 299, 299
 - nodes, 298; root node, 298, 299
 - in persistence systems, 298
 - print resource on, 299, 323
- B⁺-trees, 290, 299–300, 300, 318
 - print resource on, 323
- Babaoglu, Ozalp, 219
- back doors, 418
- balanced search trees (B-trees), 290, 291
- Banga, Gaurav, 71
- barriers (barrier synchronization), 92–93, 93
 - monitors with condition variables and, 98
- base priorities, 58
 - in decay usage scheduling, 59–62
- basic authentication, 388
- basic spinlocks, 83–84, 84
- batch processing, 47–48
- Bayer, R., 299, 323
- Belady, L. A., 219
- Belady's anomaly, 210
- Bell-LaPadula model, 409–411
 - print resource on, 426
- Bellovin, Steven M., 426
- Bernstein, Philip A., 18, 163, 164
- best security practices, 419–421
- best-fit policies (disk space allocation), 290
- bin hopping, 205
 - print resource on, 219
- binary search trees: vs. B-trees, 298
- binding (of sockets): to their own addresses, 340
- bindings (in registries), 376
- biometric authentication, 405, 406
- biometric identification, 406
- bitmaps (of disk space allocation), 288–289, 290
- Blasgen, Mike, 122
- block groups (of disk blocks), 288–289
 - selecting for a new inode, 320
- block mapping (in file systems), 283, 291, 295
- block maps, 295
- book objectives, 7–9
- book overview, xvii–xviii, 1
- bound state (of sockets), 340–342
- boundaries: operating system–application process boundary, 12, 232, 233
- bounded buffers, 89–91
 - monitors with condition variables and, 94–98
 - semaphores and, 98, 121
- `BoundedBuffer` class, 94–98, 95, 99–100, 100
 - testing, 94
- Brinch Hansen, Per, 122
- brokers (in publish/subscribe messaging systems), 371
- browsers. *See* web browsers
- buffer overflow vulnerabilities/attacks, 412–414, 414, 433
 - print resource on, 426
- buffers:
 - cache buffers (in disk drives), 274
 - and concurrent threads, 89

- storing, 100
 - See also* bounded buffers
- bugs: in synchronization, 74, 114–115
- bursts of processor time: and scheduling priority, 48
- busy waiting, 41
- Byers, J. W., 368
- C
- C API. *See* POSIX API
- C-lists, 242–243
- cache buffers (in disk drives), 274
- cache coherence protocols, 46, 84–85
- cache memories (caches):
 - bypassing when zero filling, 179; print resource on, 218
 - full associativity, 204
 - reduced performance in, 45–46
 - set-associative caches, 204
 - and throughput-oriented scheduling, 44–46
- cache misses, 205
- placement policy and, 204–205
- cache-conscious spinlocks, 84–85
- callee saves, 432, 433
- caller saves, 433
- calls. *See* procedure activations
- canonical names (domain names), 336
- CAP protection system: print resource on, 267
- capabilities (of processes), 221, 241–245
 - access control lists as used with, 251–252
 - and the access matrix, 244
 - addressable. *See* addressable capabilities
 - C-lists, 242–243
 - of child processes, 223
 - file execution and, 228
 - forgery prevention, 243–244
 - irrevocable capabilities, 245, 252
 - Microsoft Windows approach, 242
 - POSIX approach, 242, 245
 - print resource on, 267
 - selective granting of, 244–245
 - selective revocation of, 245
 - sending and receiving, 242–243
 - sets (storage area), 242, 244; C-lists, 242–243
- capability forgery: preventing, 243–244
- capacity misses (cache misses): conflict misses vs., 205
- Cartesian product of compartment partial orders, 408, 409, 423
- CAs (Certification Authorities), 387
- certificates (of servers), 387
 - web service use of, 387–388
- Certification Authorities (CAs), 387
- Chase, Jeffery S., 267
- `chdir` procedure (POSIX API), 276
- checkpointing, 151
- checkpoints (for persistent storage): and consistent states, 151
- Chen, Peter M., 322
- Chen, Shuo, 426
- Cheswick, William R., 426
- `child` procedure, 22
- child processes:
 - address space, 223
 - creating, 222–223; to run different programs, 224–225
 - as parent process copies, 223
 - waiting for, 228
- child threads: spawning of, 22
- `childThread` object, 22
- `chooseNextThread` procedure, 32
- CIFS (Common Internet File System), 330, 338–339
- classification levels of information, 407–409
 - compartment-classification level partial orders, 408, 408; Cartesian product, 408, 409, 423
 - total order, 408, 408
- cleanup code (for failure handlers): sharing, 162–163
- client authentication. *See* authentication (of users)
- `client` class, 344
- client-side stream sockets: life cycle (through states), 341
- clients and servers:
 - in messaging systems, 371, 371–372, 372
 - in RPC systems, 374–375, 374, 375
- clock replacement, 211
- `close` procedure (POSIX API), 275
- closed state (of sockets), 342
- closing files, 275
- clustered page tables, 219
- clustered paging, 203–204
 - overriding, 217
- CNAME records, 336
- Codd, E. F., 38q, 70q
- Coffman, E. G., 122
- columns (of tables), 270
- Comer, Douglas, 323
- command-line arguments, 226
- commit actions (operations), 125, 129
 - delaying, 136
 - durability and, 150
 - redo logs and, 151
- Common Criteria (ISO 15408), 415, 416
 - print resources on, 426, 427
 - website resource on, 426
- Common Internet File System (CIFS), 338–339
- Common Object Request Broker Architecture (CORBA), 376
- communication middleware systems, 369–382

448 ► Index

- compartments of information, 407–409
 - partial orders, 408, 408; Cartesian product, 408, 409, 423
- compensating transactions, 134–135
- computations:
 - early terminology, 266
 - multiple computations on single computers, 2, 9–10. *See also* multiple threads
 - See also* threads
- computer systems:
 - Atlas computer, 217, 218
 - components, 44
 - hard-real-time systems, 54–56
 - housekeeping work, 26
 - isolation of machines exposed to attack, 360
 - multiprocessor system cache memories, 44–46
 - persistent storage integrity in system crashes
 - requirement, 13, 271
 - resource allocation, 48, 49, 62, 221
 - resource utilization, 24, 26–27, 44
 - response time, 47, 48, 67–68
 - responsiveness, 24–26, 27
 - security aspects (objectives), 16, 398
 - security issues. *See under* security
 - throughput, 44, 58; convoy phenomenon and, 109, 113
 - See also* internets; middleware; networks; operating systems; servers; storage
- concurrency:
 - in message-queuing systems, 132–133
 - multiversion concurrency control (MVCC), 153–154
 - of transactions, 126, 151; increasing, 152
 - See also* concurrent threads; concurrent transactions
- Concurrent Pascal (programming language), 122
- concurrent threads, 20–21, 21
 - buffers and, 89
 - processor-intensive vs. disk-intensive activities, 26, 27, 58
 - reasons for using, 24–27
 - switching. *See* thread switching
 - See also* thread interactions
- concurrent transactions, 11, 126, 130
- condition variables (with monitors), 73, 94–98
 - basic operations, 94
 - print resources on, 122
 - vs. semaphores, 98
- confidential data writing security issue, 212–213
- confidentiality (security objective), 16, 398
- conflict misses (cache misses): vs. capacity misses, 205
- congestion (in TCP transmission): control
 - measures, 346–347, 347–348
- connected state (of sockets), 342
- connection of sockets to their partners, 340
- consistency (of transactions), 125, 144
- consistent states, 123, 125
 - checkpoints and, 151
 - monitors as in, 123
- context switching, 33
 - See also* process switching; thread switching
- contiguous allocation problems (disk space), 285
- contiguous allocation problems (memory), 174–175
 - virtual memory solution, 175–176
- contingency planning: and security, 316–317
- Control Program. *See* CP
- convoy phenomenon, 87, 111–114
 - print resource on, 122
 - and throughput, 109, 113
- convoys (of threads in wait queues), 112–113
- cooperative multitasking (in thread switching), 33
- coordinating transaction processing system
 - subsystems, 154–156
- copy on write (COW):
 - address space copying, 223
 - message passing, 173, 174; print resources on, 218
- `copy_process` procedure (Linux kernel), 163
- copying address space, 223
- copying files, 279–281
- CORBA (Common Object Request Broker Architecture), 376
- Corbató, Fernando J., 18*q*
- Courtois, P. J., 122
- covert channels, 411
 - print resource on, 426
- COW. *See* copy on write
- CP (Control Program) (in z/VM), 267–268
- CP-67, 268
 - print resource on, 268
- `cpmm` program, 280–281
- crashes. *See* system crashes
- Creasy, R. J., 268
- `create table` command (SQL), 127
- `CreateThread` procedure (Win32 API), 37
- credentials (of processes), 221, 257
 - checking, 252
 - of child processes, 223
 - file execution impact on, 228; from `setuid` program files, 228, 238, 259
 - Microsoft Windows ACL example, 246
 - `setuid` program credential propagation, 228, 238, 259
 - Trojan horse vulnerabilities, 257–259
- cryptographic file systems, 315
- cryptographic hash functions: password
 - authentication with, 363, 364, 403–404
- cryptographic techniques, 361–365
 - categories, 362
 - for preventing capability forgery, 244
- cumulative acknowledgment of segment transmissions, 346

- current directory: leaving out of the search path, and running programs in, 258
- currentThread** method (Java API), 161
- cylinders (of disk tracks), 272
- D**
- DAC (Discretionary Access Control) systems, 241
 - Trojan horse danger in, 258–259
- Daggett, Marjorie Merwin, 18*q*
- Daley, Robert C., 18*q*, 267
- dangling links (in file systems), 308
- data access mechanisms, 303–310
 - data structures for directories or indexes, 308–310
 - directories: vs. database indexes, 303–304. *See also* directories
 - file linking, 301–303, 305–308
 - indexes, 270, 304–305; vs. file directories, 303–304
- Data Encryption Standard (DES), 362
- data link layer. *See* link layer
- data location structures (metadata), 291–300
 - B-trees, 290, 291, 297–300
 - block mapping, 283, 291, 295
 - extent maps, 291, 295–297, 298
 - inodes, 291–293, 291; with indirect blocks, 292, 293–295, 293, 294, 295
- data packets. *See* packets
- data structures (in operating systems and middleware), 13
 - for directories or indexes, 308–310
 - shared. *See* shared data structures
 - trie structure, 194
- See also* data location structures (metadata); files; persistent objects; tables (in relational databases)
- data transmissions (TCP):
 - congestion control measures, 346–347, 347–348
 - redundant formats, 348, 348
 - segment transmission and acknowledgment mechanisms, 345–346
- data warehouses, 152
- database indexes, 303
 - vs. file directories, 303–304
- database systems (relational database systems), 7
 - deadlock detection and recovery in, 106, 107, 128–130, 131
 - separate system approach, 152
 - transactions in, 127–130
- See also* relational databases
- databases. *See* relational databases
- datagram sockets: life cycle (through states), 341
- datagrams (network-layer data transmission chunks), 340
- DatagramSocket** class (Java API), 341
- deadline inheritance, 111
- deadlocks (of threads), 11, 73, 73, 101–109
 - avoidance vs. prevention, 121–122
 - detection and recovery from: ex post facto, 105–106, 107, 128–130, 131; immediate, 106–109, 108
 - prevention through resource ordering, 104–105
 - print resources on, 121
 - the problem, 101–103, 128
 - self deadlock, 80
 - between transactions, 144
- deallocate operation (stacks), 430, 431
- debugging: in single address space systems, 235
- decay usage schedulers, 59–62
- decay usage scheduling, 52, 59–62
 - fixed-priority scheduling adjustments, 110
 - proportional-share scheduling via, 61–62, 71
- declaring mutexes, 78
- decoy systems (honeypots), 419
- defense in depth principle, 261–262
- delayed allocation technique, 289–290
- deletion of files, 301–303, 306–307
 - as difficult, 323
 - as inadequate for security, 315
- demand paging, 202–203
- demand-driven file reading, 177
- demand-driven program loading, 178
- denial of service (DoS) attacks:
 - vulnerability to, 65–67, 157, 358
 - worms and, 412
- Denning, Dorothy, 426(2)
- Denning, Peter J., 218(2), 219, 426
- Dennis, Jack B., 218, 266, 267(2)
- DES (Data Encryption Standard), 362
- descriptor tables, 242
- descriptors (POSIX). *See* file descriptors
- desktop environments, 5
- destroying mutexes, 78, 79
- deterministic execution (of transactions): vs. serializable execution, 138
- DFS (Distributed File System), 338
- digital signatures, 364
 - message signature mechanism, 390
 - non-repudiation feature, 364, 390
- digital trees, 194
- Dijkstra, Edsger W., 121(2), 122(2)
- dining philosophers problem, 103, 122
 - Java simulation, 119–120, 120
 - print resource on, 122
- directories (file directories), 270, 303
 - data structures for, 308–310
 - vs. database indexes, 303–304
- disk space allocation policies, 289
 - keys, 303–304
 - leaving the current directory out of the search path, 258
 - permissions for, 250–251, 261
 - running programs in the current directory, 258

450 ► Index

- directories (*cont.*)
 - traversing, 251
 - working directory: changing, 276
 - directory trees, 309
 - dirty bits, 177–178, 211
 - print resource on, 218
 - dirty pages, 177
 - tracking of, 177–178
 - Discretionary Access Control (DAC) systems, **241**
 - Trojan horse danger in, 258–259
 - disk access time:
 - aspects, 273–274
 - locality issues, 274, 287–288
 - performance issues, 272, 274
 - variance, 272
 - disk blocks, 283
 - extents, **284**
 - groups, **288–289**
 - indirect. *See* indirect blocks
 - size issues, 285, 286–287
 - disk drives, 272–274
 - components, 272, 273
 - mechanism, 272–273
 - request queuing and handling, 274–275
 - disk operation requests, 272, 274
 - queuing and handling of, 274–275
 - disk space:
 - allocation of. *See* disk space allocation
 - fragmentation of, **284–287**
 - disk space allocation, 283–290
 - best-fit policies, 290
 - bitmap approaches, 288–289, 290
 - contiguous allocation problems, 285
 - delayed allocation technique, 289–290
 - directory and subdirectory placement
 - policies, 289
 - file placement policy, 289
 - first-fit policies, 290
 - fragmentation and, 284–287
 - locality and, 287–288
 - objectives, 284
 - observed behavior strategy, 288
 - policies and mechanisms, 288–290
 - print resource on, 323
 - write order strategy, 287–288
 - disk storage:
 - vs. memory (RAM), 179
 - paging to (substituting RAM for), 180
 - disk technology, 272–274
 - print resources on, 322
 - disk-intensive vs. processor-intensive thread
 - activities, 26, 27
 - maximizing throughput, 58
 - disks (of disk drives) (platters), 272
 - dispatching threads, 33, 43
 - distributed applications: middleware and, 15
 - Distributed File System (DFS), 338
 - distributed file systems, **331**, 337–339
 - protocols, 338–339
 - DLLs (dynamic-link libraries): process use of,
 - 171–172, 172
 - DNS. *See* Domain Name System
 - Domain Name System (DNS), 330, **331**, 334–337
 - application protocol support, 331
 - as complicated, 336–337
 - information storage modes, 337
 - domain names (for internet addresses):
 - as aliases, 336
 - as analogous to pathnames, 335–336
 - relative and absolute names, 335
 - resource records, 336
 - translating into address numbers, 336
 - See also* internet addresses
 - Dorward, Sean, 323
 - DoS attacks (denial of service attacks):
 - vulnerability to, 65–67, 157, 358
 - worms and, 412
 - dotted decimal format, **351**
 - double indirect blocks, **294**
 - down** operation, 99
 - Druschel, Peter, 71
 - durability (of transactions), 125
 - and commit operations, 150
 - failure atomicity and, 148–149
 - issues, 147–148
 - in message-queuing systems, 131–132
 - in system crashes, 164
 - update storage, 150–151
 - write-ahead logging, 149–150
 - Dykes, Jim, 322
 - dynamic-link libraries (DLLs): process use of,
 - 171–172, 172
 - dynamic-priority scheduling, 52, 56–62
 - testing adjustments, 69
- E**
- EALs (Evaluation Assurance Levels), 415–416, **416**
 - Earliest Deadline First (EDF) scheduling, 52, 56–57
 - deadline inheritance for, 111
 - ECN (Explicit Congestion Notification), 348
 - EDF scheduling. *See* Earliest Deadline First (EDF) scheduling
 - ELOOP** error code, 308
 - Elphick, M., 122
 - email delivery software: TOCTTOU race bugs in,
 - 115, 122
 - email protocols, 332, 358
 - email worms, **258**, 412
 - embedded systems: without operating systems, 3
 - encryption, 362
 - end-to-end principle (of Internet communication), **330**
 - NAT routing violation of, 355
 - endpoint addresses, **386**

- entity tags (ETags), 334
- equivalence (of serializable executions/histories), 137, 139–141
 - print resources on, 164
- equivalence-preserving swaps (of transaction actions), 139–141, 142–143
 - illegal and legal swaps, 140
- erasure coding, 348
 - print resource on, 368
- error output, 276
- Eswaran, K. P., 164
- ESX Server VMM (VMware), 254, 254
 - print resource on, 267
- ETags (entity tags), 334
- ethereal** tool, 367
- Ethernet, 356
- Evaluation Assurance Levels (EALs), 415–416, 416
- ex post facto deadlock detection and recovery, 105–106, 107, 128–130, 131
- exception potential in Java RMI, 377–378
- excess memory demand: swapping solution, 208
- exchange operations: atomicity, 83–84
- exclusive access. *See* mutual exclusion
- exec family procedures, 225–227
 - protection mechanism interaction, 228
- execer** program, 226
- exec1** procedure (POSIX API), 225–226
- exec1p** procedure (POSIX API), 226, 227
 - Trojan Horse danger, 258
- execute permission. *See* **x** permission
- execution (of threads): status information, 30
- execution (of transactions):
 - deterministic execution, 138
 - equivalence of serializable execution, 137; print resources on, 164
 - histories. *See* system histories (transaction histories)
 - serial and serializable execution, 137
- execve** procedure (POSIX API), 225
- exit** procedure (POSIX API), 230
- Explicit Congestion Notification (ECN), 348
- Explorer (Microsoft), 5
- ext3fs file system (Linux), 288–289, 291, 312
- Extensible Markup Language (XML): and web services, 383, 384
- extent maps, 291, 295–297
 - B-trees used as, 298
 - extent descriptions, 295–296
 - speed, 296
- extents (of disk blocks), 284
 - extent map descriptions of, 295–296
- external fragmentation:
 - of disk space, 285–287
 - of memory, 175, 175
 - print resource on, 322
- F**
- Fabry, R. S., 267
- failure atomicity (of transactions), 124, 144–145
 - ad hoc programming difficulties, 145, 146, 157
 - and durability, 148–149
 - in system crashes, 164
 - undo logging (transaction processing system), 145–147
- failure handlers: cleanup code sharing, 162–163
- failures (of transactions), 124, 125, 145
 - testing complications, 145, 146
- fair-share scheduling, 49–50
 - proportional-share scheduling vs., 50
- fetch policy (page frame assignment timing), 202–204
- fibers (user-level threads), 21, 232, 233, 233
 - print resource on, 71
- FIFO (first in, first out) policy:
 - page replacement, 209–210, 210, 211; print resources on, 219
 - thread scheduling, 53–54
- file access permissions. *See* permissions
- file APIs, 13
 - categories, 13
 - to hidden mechanisms in operating systems, 13
- file descriptors (descriptors), 242, 252, 275–279
 - obtaining, 275
 - standard, 276
- file linking, 301–303, 305–308
- file location structures. *See* data location structures (metadata)
- file locks, 92
- file name argument (**open** procedure), 276
- file names:
 - adding to/removing from files, 306–307
 - as metadata, 291, 301–303
 - multiple names for single files, 306, 307; as symbolic links, 307–308
- file offsets: setting, 283
- file systems, 12–13, 135
 - block mapping, 283, 291, 295
 - cryptographic systems, 315
 - data access structures, 303–310
 - data location structures (metadata), 291–300
 - disk space allocation policies and mechanisms, 288–290
 - distributed file systems, 331, 337–339
 - historical systems, 323
 - journalled file systems, 135–136, 311, 312, 313
 - links in: hard links, 307, 308; symbolic links (soft links), 307–308, 307, 336
 - log-structured file systems (LFSs), 313, 323
 - polymorphic implementations, 313–314
 - print resources on, 322
 - See also* ext3fs; HFS Plus; NTFS; XFS

452 ► Index

- file-processes** program, 277
 - files, 269–270
 - access control lists (ACLs) for, 250
 - access permissions. *See* permissions
 - adding names to, 306
 - attributes indexed, 305, 308
 - closing, 275
 - copying, 279–281
 - deletion of, 301–303, 306–307; as difficult, 323;
 - as inadequate for security, 315
 - descriptors, 242, 252, 275–279
 - disk space allocation policies, 289
 - with holes (sparse files), 295
 - locating with indexes, 304–305
 - location structures. *See* data location structures (metadata)
 - mapping of into virtual memory, 279–281
 - metadata attribute operations, 278–279
 - mode number, 301
 - names. *See* file names
 - offsets, 283
 - opening, 251–252, 275, 276–278, 282
 - POSIX API, 275–283, 306, 307
 - reading, 279–283; sequentially, 282–283;
 - at specified positions, 281–282
 - as referenced (in POSIX), 275
 - removing names from, 306–307
 - size metadata, 301
 - sparse files, 295
 - time stamps, 301
 - user groups (owners), 249–250
 - versioning of, 323; print resources on, 323;
 - snapshots, 312
 - writing, 279–283; sequentially, 282–283; at
 - specified positions, 281–282
 - firewalls, 359–360, 361
 - misconfiguration of, 360; print resource on, 368
 - first in, first out (FIFO) policy:
 - page replacement, 209–210, 210, 211; print
 - resources on, 219
 - thread scheduling, 53–54
 - first-fit policies (disk space allocation), 290
 - fixed-priority scheduling, 52, 52–56
 - avoidance of, 66, 110
 - decay usage adjustments to, 110
 - hard-real-time systems use of, 54
 - need for, 110
 - as not viable in open, general-purpose
 - environments, 54
 - numerical priorities, 52–54
 - as off-limits in typical systems, 66
 - priority inheritance and, 110–111
 - rate-monotonic scheduling, 54, 56; testing
 - real-time schedules, 54–56
 - theorems on, 54–56, 57
 - forgery: preventing capability forgery, 243–244
 - fork** procedure (POSIX API), 222–223, 224, 224–225, 227
 - if** statement with, 223
 - programming example, 265
 - forker** program, 224
 - forking off child processes, 222–223, 224–225
 - access matrix changes from, 238
 - to run different programs, 224–225
 - forward error correction, 348
 - print resource on, 368
 - forward-mapped page tables. *See* multilevel page tables
 - forwarding tables (of routers), 352
 - Fotheringham, John, 218
 - fragmentation (of disk space), 284–287
 - extent-oriented definition, 284
 - external fragmentation, 285–287
 - internal fragmentation, 284–285
 - print resource on, 322
 - fragmentation (of memory): external
 - fragmentation, 175, 175
 - frames (link-layer data transmission chunks), 355
 - free page frames:
 - inventory water marks, 205–206
 - working sets in excess of, 208
 - free page list, 206, 207
 - freeing page frames in advance of demand, 206
 - fstat** procedure (POSIX API), 278–279
 - fstater** program, 278
 - fsync** procedure (POSIX API), 310
 - ftruncate** procedure (POSIX API), 279
 - full associativity (of caches), 204
 - functional requirements (in PPs and STs), 415
- G**
- Ganger, Gregory R., 323
 - Gantt charts, 55
 - real-time schedule testing, 54–56
 - response time illustration, 67–68
 - Garfinkel, Simson, 426
 - gateway routers, 351
 - general access matrix model, 240–241
 - getpid** procedure (POSIX API), 225, 226
 - gift programs: Trojan horse danger, 258
 - global replacement (of pages), 208
 - Goodman, Nathan, 164
 - GoogleSearch security limitation, 388–389
 - Gray, Jim, 154, 163(2), 164(2)
 - greater than sign (>): redirect standard output
 - symbol, 276
 - greatest fixed-point solution theorem, 253
 - Güntsch, Fritz-Rudolf, 217, 218
 - Gutmann, Peter, 323

H

Habermann, A. N., 121–122
 hackers: terminology, 401
 Haerder, Theo, 125, 163
 handle tables, 242
 handles (Microsoft Windows), 242
 hard disks. *See* disk drives
 hard links (in file systems), 307, 308
 hard-real-time system scheduling, 54
 print resource on, 71
 testing real-time schedules, 54–56
 Härder, Theo, 125, 163
 hardware interrupts, 34
 timer interrupts, 41–42, 43
 Harrison, Michael A., 240, 267
 hash buckets, 195, 196, 197
 hash collisions, 195
 handling mechanism, 196–197
 hash function, 195
 hash tables, 309
 Hashed Message Authentication Codes (HMACs), 363, 364
 hashed page tables, 185, 194–197, 195
 address mapping in, 196, 196
 collision-handling mechanism, 196–197
 creation of, 218–219
 entry storage problem, 219
 and inverted page tables, 218
 and multilevel page tables, 197
 page number tags, 185, 196
 protection keys (in entries), 236
 Havender, J. W., 121
 Hays, Jim, 219
 head switch time, 273
 head switches, 272–273
 heads (of disk drives), 272
 Hellerstein, Joseph L., 71
 Heymans, F., 122
 HFS Plus file system (Mac OS X), 135, 312
 B-tree, 310
 extent maps, 297
 symbolic links, 308
 hierarchical page tables. *See* multilevel page tables
 high-water mark (free frame inventory), 205
 Hill, Mark D., 219
 Hilton, Paris: user authentication failure involving:
 print resources on, 427
 histories. *See* system histories (transaction histories)
 Hoare, C. A. R., 96, 122
 Hohpe, Gregor, 394
 holes: files with (sparse files), 295
 Holt, Richard C., 122
 honeypots, 419
 housekeeping work (of computers): concurrent
 threads and, 26
 HTTP (Hypertext Transfer Protocol), 327, 330,
 332, 332

 request-response format, 332
 web communication example, 332–334
 web services use of, 384–386
 Huck, Jerry, 219
 Hydra operating system: print resource on, 267
 Hypertext Transfer Protocol. *See* HTTP

I

I/O (input/output):
 POSIX procedures, 279–283
 security issues, 158–159
 standard library procedures
 recommendation, 283
 i386 architecture. *See* IA-32 architecture
 IA-32 architecture (i386/x86 architecture):
 multilevel page tables, 190–194, 191, 192
 Physical Address Extension (PAE) mode, 214
 print resource on, 218
 protection scheme, 232
 return address storage approach, 432
 segmentation support, 197; paging–segmentation
 combination, 198–199, 199
 software/hardware interface, 184
 thread-switching code, 31–32
 underutilized features, 267
 IBM DB2: read committed mode, 152
 IBM iSeries (System/38–AS/400), 237
 print resource on, 267
 IBM z/VM, 254
 idempotency (of the undo operation), 149
 IDSes (intrusion detection systems), 360–361, 418
if statement: with **fork** procedure, 223
 IIOP (Internet Inter-Orb Protocol), 376
 illegal address notion, 169
 IMAP (Internet Message Access Protocol), 330, 332
 immediate deadlock detection, 106–109, 108
 importance (of tasks/threads), 48, 49
 importance-oriented scheduling, 49, 51, 58
 impurities (of transactions), 142
 removing, 142–144
 inactivity periods (of processes), 180
 inconsistency of variable values, 75–76
 medical consequences, 76
 inconsistent states: in transactions (temporary),
 144–145
 independent allocation (of memory) goal, 170
 indexes, 270
 data structures for, 308–310
 database indexes vs. file directories, 303–304
 file attributes indexed, 305, 308
 keys, 303–304
 locating files with, 304–305
 unique indexes, 304
 indirect blocks, 293
 inodes with, 292, 293–295, 293, 294; as block
 maps, 295; tree structure, 295
 read requirements, 296

454 ► Index

- information classification levels. *See* classification levels of information
 - information compartments. *See* compartments of information
 - information-flow control, 407
 - Bell-LaPadula model, 409–410
 - lattice model: print resource on, 426
 - in MLS systems, 407–410
 - initializing mutexes, 78
 - inodes, 291–293, 291
 - with indirect blocks, 292, 293–295, 293, 294; as block maps, 295; tree structure, 295
 - metadata categories, 292, 292, 293, 301
 - numbers contained in, 301
 - selecting block groups for a new inode, 320
 - input overwrite (buffer overflow)
 - vulnerabilities/attacks, 412–414, 433
 - input/output. *See* I/O (input/output)
 - instruction pointers (IP registers), 30, 31
 - instructions (in code): in virtual addresses, 235
 - integrity (security objective), 16, 398
 - persistent storage integrity in system crashes requirement, 13, 271
 - See also* metadata integrity
 - integrity monitoring, 418
 - Intel architecture. *See* IA-32 architecture; Itanium architecture
 - interactions between computations. *See* thread interactions
 - interfaces:
 - APIs. *See* APIs (Application Programming Interfaces)
 - software/hardware interface, 182–185
 - user interface design: and security, 261
 - web service interface specification standard (WSDL), 383–384
 - See also* specific Java interfaces
 - interleaving:
 - of threads, 72, 76
 - of transaction histories, 138
 - internal fragmentation:
 - of disk space, 284–285
 - print resource on, 322
 - Internet: end-to-end principle, 330
 - internet addresses, 334
 - design problem, 350
 - format, 326, 350, 351; cost and benefit, 327
 - NAT rewriting of, 353, 354
 - prefixes, 350
 - See also* domain names; network addresses (MAC addresses)
 - Internet Inter-Orb Protocol (IIOP), 376
 - Internet Message Access Protocol (IMAP), 330, 332
 - Internet Protocol. *See* IP
 - internets, 326
 - security issues, 356–365
 - interrupt handler, 34, 41–42
 - interrupts, 231
 - hardware interrupts, 34; timer interrupts, 41–42, 43
 - MMU interrupts. *See* page faults
 - intrusion detection systems (IDSes), 360–361, 418
 - innumbers, 292
 - invariants (invariant properties), 75, 123
 - spanning multiple objects, 123–124
 - inverted page tables, 218
 - I/O. *See* I/O (input/output) (*at "i-/o", above*)
 - IP (Internet Protocol), 329, 330, 349–351
 - IPv6 vs. IPv4, 350
 - IP headers, 349–350
 - IP registers (instruction pointers), 30, 31
 - `ipconfig` command, 368
 - IPsec, 350
 - VPN support, 359
 - irrevocable capabilities, 245, 252
 - security issue, 245
 - iSeries (IBM), 237
 - capabilities storage approach, 242
 - ISO 15408 (Common Criteria), 415, 416
 - isolating threads of different processes, 11
 - isolation (of machines exposed to attack), 360
 - isolation (of transactions), 11, 124, 125, 126
 - print resources on, 164
 - reduced, 152–153
 - security issues, 158
 - snapshot isolation, 153–154
 - Itanium architecture:
 - page table support, 187, 194, 197
 - print resource on, 218, 267
 - single address space system, 235
- ## J
- J2EE (Java 2 Platform, Enterprise Edition), 7
 - JAX-RPC, 386–387, 394
 - JMS, 370, 394
 - website resource on, 163, 394
 - Java (programming language):
 - limitations, 98
 - monitor approximation, 81–82, 83
 - website resource on, 394
 - Java 2 Platform. *See* J2EE
 - Java API, 22, 122
 - website resource on, 38, 122
 - Java API for XML-Based RPC. *See* JAX-RPC
 - Java Message Service (JMS), 370
 - website resource on, 394
 - Java objects: protecting from each other within single processes, 252–253
 - Java RMI:
 - exception potential, 377–378
 - publish/subscribe messaging example, 377–382
 - website resource on, 394

- Java Virtual Machine. *See* JVM
- JAX-RPC (Java API for XML-Based RPC), 386–387
 - website resource on, 394
- Jessen, Eike, 217, 218
- JFS file system (Linux), 135
- JMS (Java Message Service), 370
 - website resource on, 394
- jobs (in batch processing), 47–48
- journaling strategy (journaled file systems), 135–136, 311, 312, 313
- journals. *See* logs
- Joy, William, 219
- JVM (Java Virtual Machine), 252–253
 - type system, 244
- JVM verifier, 253
 - dataflow analysis theorem, 253, 264
 - print resource on, 267

- K**
- Kempster, Tim, 164
- kernel (of operating systems), 5
 - print resource on, 368
- kernel mode (system mode) (of processors), 231
- kernel threads, 232–233, 232
- kernel-supported user threads (native threads), 232–233, 232
- Kessler, R. E., 219
- key pair cryptographic techniques, 362
- key security practices, 419–421
- keyloggers, 402
- keys: in directories vs. indexes, 303–304
- Khalidi, Y. A., 219
- kill** procedure (POSIX API), 230
- Knuth, Donald Ervin, 323
- Krebs, Brian, 427
- Kurose, James F., 368

- L**
- label-switching routers, 352
- Lampson, Butler W., 163, 426
- Landwehr, Carl E., 426
- LANs (local area networks), 326
- launcher** program, 227
- layers. *See* networking protocol layers
- Layland, James W., 54–56, 57, 71
- leaf/nonleaf procedure activations, 432
- Least Recently Used (LRU) replacement policy, 209–211, 210
 - print resources on, 219
- Lehoczky, John P., 122
- less than sign (<): redirect standard input symbol, 276
- Leveson, Nancy G., 121
- Levy, Henry, 219
- LFSs (log-structured file systems), 313, 323
 - print resource on, 323
- lifetimes (of threads), 20
- Linden, Theodore A., 267
- Lindholm, Tim, 267
- linear lists (unordered), 308–309
- linear page tables, 185–190
 - address mapping in, 186, 186, 188–189, 189
 - entry storage problem, 186–187, 219
 - and multilevel page tables, 188, 190
 - process switching and, 185
 - recursion problem, 188
 - valid/access control bits (in entries), 185–186, 186, 237
 - virtual memory storage solution, 188–190
- link layer (in networking systems), 14, 329, 330, 355–356
 - data transmission chunks (frames), 355
 - security issues, 359
- link** procedure (POSIX API), 306
- linked lists: hash collision-handling in, 196–197
- links (in file systems):
 - hard links, 307, 308
 - symbolic links (soft links), 307–308, 307, 336
- links (in networks), 325
 - shared links, 355–356
- Linux:
 - clustered paging (read around), 203–204; overriding, 217
 - demand paging, 203
 - file systems, 135. *See also* ext3fs; XFS
 - immediate deadlock detection in, 109
 - kernel, 5
 - runqueue structure, 104
 - scheduler, 63–65, 71
 - and segmentation, 201
 - source code, 38, 104, 122, 163
 - “swapping” misnomer, 208
 - vmstat** program, 38
- Liskov, Barbara, 164
- listening state (of sockets), 342
- Lists**: storing buffers with, 100
- Liu, C. L., 54–56, 57, 71
- ln** command, 308
- local area networks (LANs), 326
 - technology, 356
- local arrays: buffer overflow vulnerabilities/attacks, 412–414, 414, 433
- local replacement (of pages), 207, 208
- local variables: procedure activation value storage options, 432–433
- locality: of disk space allocation: access time issues, 274, 287–288
 - See also* spatial locality; temporal locality
- locating files with indexes, 304–305
- locating remote objects, 376, 377, 378–379, 379
- lock actions (operations):
 - assumptions regarding, 139
 - notation for, 137–138

456 ► Index

- lock actions (*cont.*)
 - transaction processing system rules, 141–142
 - See also* locking mutexes; two-phase locking
- locking mutexes, 77–78, 77
 - basic spinlocks, 84, 84, 85
 - cache-conscious spinlocks, 85, 85
 - queuing mutexes, 87, 88
 - recursive locking, 80
- locking pages into memory, 212–213
- locks (on shared data structures):
 - file locks, 92
 - operational and functional assumptions regarding, 139
 - predicate locks, 164
 - See also* deadlocks (of threads); monitors; mutexes; readers/writers locks; spinlocks
- log-structured file systems (LFSs), 313, 323
 - print resource on, 323
- logging:
 - redo logging, 150–151
 - undo logging, 145–147
 - write-ahead logging, 149–150
- logging devices (append only), 418
- logging in using passwords: weaknesses, 401–402
- login forms (screens): sending policy, 403
- login monitoring, 418
- login password storing, 212
- logs (journals), 135
 - combined log of undo logs, 147, 149
 - See also* redo logs; undo logs
- looped links (in file systems), 308
- lottery scheduling, 62–63, 66–67
 - print resource on, 71
- low-water mark (free frame inventory), 205
- LRU (Least Recently Used) replacement policy, 209–211, 210
 - print resources on, 219
- lseek** procedure (POSIX API), 283
- Luby, M., 368
- M**
- MAC addresses (Media Access Control addresses), 356
- Mac OS X:
 - disk space allocation strategy, 288
 - file system. *See* HFS Plus
 - immediate deadlock detection in, 109
 - scheduler, 59–60, 60, 61, 71
 - and segmentation, 201
 - Spotlight search feature, 303, 304–305
 - UNIX base, 308
- MAC systems (Mandatory Access Control systems), 241, 258–259
 - with ACLs, 410
 - Bell-LaPadula model, 409–411
 - MLS systems, 407–410
 - SELinux system, 411
- MacKinnon, R. A., 268
- MACs (Message Authentication Codes), 362
 - computing techniques, 363, 364
 - digital signatures, 364
 - Hashed Message Authentication Codes, 363, 364
- madvise** procedure (POSIX API), 203, 215, 217
- major page faults: converting to minor page faults, 203–204
- malware, 411–414
- Mandatory Access Control systems. *See* MAC systems
- mapping:
 - of addresses. *See* address mapping (translation)
 - of file blocks, 283, 291–295, 295
 - of files into virtual memory, 279–281
- mapping function, 168
 - See also* address-mapping table; page tables; segment table; TLBs (translation lookaside buffers)
- maps:
 - block maps, 295
 - extent maps, 291, 295–297
 - See also* address-mapping table; page tables
- masks (of Internet address numbers), 351
- Mattson, R., 219
- McCreight, E., 299, 323
- McWilliams, Brian, 427
- MD5 (Message Digest 5), 364
- Media Access Control (MAC) addresses, 356
- median key (in B-trees), 299
- memcpy** procedure (POSIX API), 279–280
- memory (main memory) (RAM):
 - address space. *See* address space (virtual address space)
 - address use, 166–167, 167
 - addresses. *See* addresses; physical addresses; virtual addresses
 - cache memories, 44–46
 - contiguous allocation problems, 174–175; virtual memory solution, 175–176
 - vs. disk storage, 179
 - excess demand: swapping solution, 208
 - external fragmentation of, 175, 175
 - independent allocation goal, 170
 - locking pages into, 212–213
 - protecting. *See* protection (memory protection)
 - saving and restoring registers into and from (push and pop operations), 29, 30–31, 31, 429, 432–433
 - shared. *See* shared memory
 - sparse address space allocation, 176–177
 - stacks as represented in, 430–431
 - substituting for disk storage, 180
 - virtual. *See* virtual memory
 - as volatile, 125
 - zero filling of. *See* zero filling (of memory)
 - See also* storage

- memory management unit. *See* MMU
- memory protection. *See* protection (memory protection)
- memory word tag bits: for preventing capability forgery, 243
- Message Authentication Codes. *See* MACs
- message authentication feature (TCP), 358
- Message Digest 5 (MD5), 364
- message digest functions (cryptographic hash functions), 363, 364
- message passing, 173, 174
See also messaging
- message queuing (and message-queuing systems), 130–131, 370, 371, 371
 - concurrency in multi-server systems, 132–133
 - durability in, 131–132
 - transaction processing, 15, 130–135
 - workflow systems, 133–135, 163–164
- message-oriented middleware (MOM). *See* messaging (and messaging systems)
- MessageRecipient** interface, 378, 378
- messaging (and messaging systems), 15, 369, 370–373
 - ACL use with, 387
 - message passing, 173, 174
 - message queuing. *See* message queuing
 - message selection, 372–373
 - print resource on, 394
 - publish/subscribe messaging, 370–371, 371, 372–373
 - vs. RPC, 371–372
 - as synchronization, 74
- metadata (metadata attributes), 13, 135, 290–303
 - access control metadata, 300–301
 - for data location. *See* data location structures (metadata)
 - file attributes indexed, 305, 308
 - file names as, 291, 301–303
 - file operations on, 278–279
 - file size, 301
 - inode metadata categories, 292, 292, 293, 301
 - integrity. *See* metadata integrity
 - page table–metadata analogy, 290–291, 294–295
 - time stamps (for files), 301
 - transactions and, 136
- metadata integrity:
 - maintenance strategies, 311–313
 - system crashes and, 310
 - violations of, 310–311
- methods: **synchronized** keyword for public methods, 81, 82, 94
See also specific methods
- Meyer, R. A., 268
- microshell** program, 229
- Microsoft Windows:
 - capabilities storage approach, 242
 - clustered paging (read around), 203
 - disk space allocation strategy (XP), 288
 - file system. *See* NTFS
 - handles, 242
 - page lists, 206–207, 207
 - print resource on, 218
 - replacement policy, 208
 - scheduler, 59, 60–61, 71; print resource on, 71
 - and segmentation, 201
- middleware, 6–7, 6
 - and application programs, 6–7; distributed applications, 15
 - communication middleware systems, 369–382
 - and distributed applications, 15
 - example systems, 7
 - indexes as, 304–305
 - marketing definition, 7
 - message-queuing systems, 130–135
 - operating system–middleware differences, 6
 - persistent storage forms. *See* persistent objects; tables (in relational databases)
 - print resource on, 18
 - protection job, 252
 - See also* database systems
- minor page faults: converting major page faults to, 203–204
- MIPS architecture:
 - page table research tool, 197
 - return address storage approach, 432
 - software/hardware interface, 185
- Mizenmacher, M., 368
- mlock** procedure (POSIX API), 213
- mlocka11** procedure (POSIX API), 213
- MLS (Multi-Level Security) systems, 407–410
- mmap** procedure (POSIX API), 279, 279–281, 282
 - problems using, 281–282
- MMU (memory management unit), 167, 168
 - address mapping. *See* address mapping (translation)
 - page table entry loading into TLBs, 184
 - page table walker, 184
 - recent address translation storage. *See* TLBs (translation lookaside buffers)
- MMU interrupts. *See* page faults
- modified page list, 206, 207
- modified page writer thread, 206
- modularization (modularity):
 - concurrent threads and, 24
 - transaction failure testing and, 145
- Mogul, Jeffery C., 71
- MOM (message-oriented middleware). *See* messaging (and messaging systems)
- monitor** keyword, 81

458 ► Index

- monitoring: security monitoring, 398, 417–419
 - monitors, 73, 81–82
 - with condition variables, 73, 94–98; print resources on, 122; vs. semaphores, 98
 - as in consistent states, 123
 - Java approximation of, 81–82, 83
 - tree monitors, 126
 - MPLS (Multiprotocol Label Switching), 352
 - MQSeries (IBM), 370
 - Multi-Level Security (MLS) systems, 407–410
 - multicasts, 325
 - Multics system, 267(2), 323
 - paging–segmentation combination, 199–200, 200
 - print resources on, 218, 267, 323
 - multiforker** program, 265
 - multilevel feedback queue schedulers, 61
 - print resource on, 71
 - multilevel page tables, 190–194
 - entry storage problem, 219
 - four-level tables, 194
 - and hashed page tables, 197
 - IA-32 paged address mapping, 191–193, 192
 - and linear page tables, 188, 190
 - page directory, 191, 192
 - process switching and, 185
 - two-level tables, 190–194, 191
 - multiple address space systems, 234–235
 - advantages, 169, 170–171, 234
 - disadvantages, 171, 234–235
 - multiple computations on single computers, 2, 9–10
 - See also* multiple threads
 - multiple password systems, 404–405
 - print resource on, 426
 - multiple threads, 19–21, 20, 21
 - sequential threads, 21
 - See also* concurrent threads; thread interactions
 - multiple-threaded programs, 20
 - simple examples, 22–23, 23
 - multiple-threaded systems: waiting mechanism, 41–42, 42
 - multiple-threaded web servers, 25
 - multiplexing, 328, 339
 - transport protocol for, 339
 - multiprocessor systems:
 - cache memories in, 44–46
 - page frames in, 204
 - Multiprotocol Label Switching (MPLS), 352
 - multiserver systems: for message-queuing systems, 132–133, 132
 - multithreading. *See* thread switching
 - multiversion concurrency control (MVCC), 153–154
 - mutexes, 73, 77–80
 - basic spinlocks, 83–84, 84
 - cache-conscious spinlocks, 84–85
 - contention over, 111
 - declaring and initializing, 78
 - destroying, 78, 79
 - locking, 77–78, 77; queuing mutexes, 87, 88; recursive locking, 80; spinlocks, 84, 84, 85, 85
 - more efficient versions, 82–88
 - OOP-structured models. *See* monitors
 - programming examples, 78–79, 79, 113–114, 113
 - queuing mutexes, 86–87
 - vs. readers/writers locks, 92
 - recursive mutexes, 80, 96
 - semaphores as, 99
 - spinlocks, 83–85; vs. queuing mutexes, 86
 - states, 77
 - types, 80, 83
 - unlocking, 77–78, 77; queuing mutexes, 87, 87, 88; spinlocks, 84, 84
 - wait queue dump version, 113–114, 113
 - mutual authentication, 389
 - mutual exclusion (of threads), 73, 123
 - with load and store instructions, 121
 - with mutexes and monitors, 76–88
 - need for, 74–76
 - print resource on, 121
 - mutual exclusion locks. *See* mutexes
 - MVCC (multiversion concurrency control), 153–154
- N**
- name servers, 336, 337
 - NAT (Network Address Translation) technology, 352–353
 - routers, 353, 354, 360
 - routing, 353–355
 - NAT routers, 353, 354, 360
 - NAT routing, 353–355
 - problems and benefits, 355
 - native threads (kernel-supported user threads), 232–233, 232
 - Need-To-Know security principle, 410
 - nesting of procedure activations, 432
 - Network Address Translation technology. *See* NAT
 - network addresses (MAC addresses), 356
 - See also* internet addresses
 - Network File System (NFS), 338–339, 367
 - ONC RPC and, 376
 - network layer (in networking systems), 14, 329, 330, 349–355
 - data transmission chunks (datagrams), 340
 - security problem and enhancement, 358–359
 - networking, 324–368
 - print resource on, 368
 - security issues, 356–365
 - vulnerabilities, 357
 - See also* internets; networking protocol layers; networking protocols; networks

- networking protocol layers (protocol layers), 14–15, 327–330, 330
 - application layer, 332–339
 - link layer, 355–356
 - network layer, 349–355
 - physical layer, 355–356
 - security issues, 357–359
 - transport layer, 339–349
- networking protocols, 327–330, 329
 - application protocols, 328, 331–332
 - ARP (Address Resolution Protocol), 356
 - CIFS (Common Internet File System), 330, 338–339
 - distributed file system protocols, 338–339
 - DNS (Domain Name System), 330, 331, 334–337
 - email protocols, 332, 358
 - HTTP (Hypertext Transfer Protocol), 327, 330, 332, 332–334, 384–386
 - IMAP (Internet Message Access Protocol), 330, 332
 - IP (Internet Protocol), 329, 330, 349–351
 - MPLS (Multiprotocol Label Switching), 352
 - NFS (Network File System), 330, 338–339, 367, 376
 - POP3 (Post Office Protocol–Version 3), 330, 332
 - print resource on, 368
 - SCTP (Stream Control Transmission Protocol), 330, 349
 - SMTP (Simple Mail Transfer Protocol), 330, 332
 - software layers responsible for, 331
 - SSL (Secure Sockets Layer), 358
 - TCP (Transport Control Protocol). *See* TCP
 - transport protocols, 328, 331, 349
 - UDP (User Datagram Protocol), 339
 - website resource on, 368
- networks, 325
 - addresses, 356
 - links in, 325; shared links, 355–356
 - technologies, 356
 - types, 326, 359
 - See also* internets
- Newcomer, Eric, 163
- next** pointer, 30
- NFS (Network File System), 330, 338–339, 367
 - ONC RPC and, 376
- nice** command, 70
- niceness parameter (for scheduling), 51
 - in the Linux scheduler, 63, 64
- non-portable APIs, 22
- non-repudiation feature (of digital signatures), 364, 390
- nonrepeatable reads, 153, 154
- nonserializable histories (of transaction executions), 139
- Northcutt, Stephen, 426
- notify** method (Java API), 96
 - vs. **notifyAll** method, 97
- notify operation (condition variables), 94, 96–97
 - reducing notification, 97
- notifyAll** method (Java API), 94, 95, 96, 97
 - vs. **notify** method, 97
- NTFS file system (Microsoft), 135, 312
 - ACL storage, 301
 - extent map system, 297, 323
- NULL** pointer, 226
- numerical scheduling priorities, 52–54
 - tie-breaking strategies, 53–54
- O**
- O_CREAT** argument, 276–278
- O’Neil, Patrick E., 164
- O_RDONLY** argument, 276
- O_RDWR** argument, 276
- O_TRUNC** argument, 278
- O_WRONLY** argument, 276
- OASIS web site, 394
- objectives:
 - of this book, 7–9
 - of disk space allocation, 284
 - of security, 16, 398
- objects, 237
 - persistent objects, 12, 269, 270
 - protecting from each other: in whole operating systems, 253–257; within single processes, 252–253
 - remote. *See* remote objects
 - shared object virtual address names, 168, 235
 - subjects as, 237
 - See also* stored objects/entities
- observed behavior disk space allocation
 - strategy, 288
- ONC RPC (Open Network Computing RPC), 376
- open file descriptors, 252
- Open Network Computing (ONC) RPC, 376
- open** procedure (POSIX API), 251–252, 275
 - arguments, 251, 276–278
- opening files, 251–252, 275, 276–278, 282
- operating systems, 2–5, 4
 - application process–operating system boundary, 12, 232, 233
 - and application programs, 4–5
 - common perception of, 4–5
 - embedded systems without, 3
 - file systems. *See* file systems
 - hidden mechanism APIs, 13
 - historical definition, 4
 - Hydra system: print resource on, 267
 - kernel, 5; print resource on, 368
 - memory access needed, 231
 - middleware–operating system differences, 6
 - networking and transport layer services, 14, 328–329

460 ► Index

- operating systems (*cont.*)
 - persistent storage form. *See* file systems
 - private memory, 231
 - and protection, 231
 - protection schemes, 231–232, 234–237
 - services, 2–3, 3, 4–5; networking and transport layer services, 14, 328–329
 - transfers to, 231
 - user-interface programs in, 5
 - VMS system, 208
 - vulnerabilities, 15–16
 - See also* Linux; Mac OS X; Microsoft Windows; middleware; Multics system; UNIX-family operating systems
- operations (on objects), 237
 - See also specific operations, procedures, and methods*
- OPT (optimal replacement) policy, 209, 210
 - print resources on, 219
- optimal replacement (OPT) policy, 209, 210
 - print resources on, 219
- Oracle database system:
 - deadlock detection and recovery in, 106, 107, 128–130, 131
 - multiversion concurrency control, 153
 - website resource on, 163
- Orlov, Grigoriy, 289, 323
 - website resource on, 323
- OSI (Open Systems Interconnection) reference model, 329, 330
- Ousterhout, John K., 323
- outgoing pointer, 30
- overview of this book, xvii–xviii, 1
- P**
- P (down) operation, 99
- packets (data transmission chunks):
 - communication requirements, 340
 - datagrams, 340
 - frames, 355
 - headers (IP headers), 349–350
 - incorrect source address problem, 358–359
 - IPsec encryption, 359
 - segments, 345–346
- PAE (Physical Address Extension) mode, 214
- page coloring, 205
 - print resource on, 219
- page directory (in multilevel page tables), 191, 192
- page faults (MMU interrupts), 169, 231
 - and demand paging, 203
 - file page access and, 177
 - prepaging and, 203–204
 - soft page faults, 205
- page frames (of physical addresses), 169
 - assignment (to pages) (paging), 197, 202–204, 205
 - assignment selection (placement policy), 204–205
 - assignment timing (fetch policy), 202–204
 - free frame inventory water marks, 205–206
 - freeing in advance of demand, 206
 - in multiprocessor systems, 204
 - size, 180–181
 - temporary state inventories, 206–207
 - zero filling of. *See* zero filling (of memory)
- page lists (Microsoft Windows), 206–207, 207
- page table walker, 184
 - software/hardware interfaces for architectures with/without, 184–185
- page tables, 168–169
 - clustered page tables, 219
 - data structure alternatives, 181–182
 - entry loading into TLBs, 184
 - future systems research, 197
 - granularity, 180
 - hashed. *See* hashed page tables
 - inverted page tables, 218
 - linear. *See* linear page tables
 - metadata structure–page table analogy, 290–291, 294–295
 - multilevel. *See* multilevel page tables
 - operating systems and, 169
 - performance vs. TLB performance, 184
 - process switching and, 185
 - protection settings (in entries), 232
 - reference bits (in entries), 211; print resource on, 218
 - use of, 182
- pages (of virtual addresses), 169
 - active pages, 180
 - dirty bits, 177–178, 211; print resource on, 218
 - dirty pages, 177; tracking of, 177–178
 - global replacement, 208
 - local replacement, 207
 - locking into memory, 212–213
 - page frame assignment to (paging), 197, 202–204, 205
 - sending to disk, 180. *See also* replacement policy
 - size, 180, 183, 184; print resource on, 218
- paging (page frame assignment):
 - demand paging, 202–203
 - prepaging, 202, 203–204
 - print resources on, 217, 218
 - reducing cache conflicts, 205
 - timing (fetch policy), 202–204
 - virtual memory as, 197
- paging (to disk), 180
- Pang, Ruoming, 368
- parent processes:
 - forking off child processes, 222–223, 224–225; access matrix changes from, 238; to run different programs, 224–225

- parent threads: spawning of child threads, 22
- Parmelee, R. P., 268
- Parnas, D. L., 122
- partially ordered sets (of classification and compartment labels), 408, 408
- password authentication:
 - with cryptographic hash functions, 363, 364, 403–404
 - weaknesses, 388, 401–402
- password wallet systems, 405
- password-changing program, 212, 229
- passwords:
 - authentication of. *See* password authentication
 - checking without storing, 403–404
 - multiple password systems, 404–405; print resource on, 426
 - theft, 402; via spoofing and phishing, 402–403
 - token use with, 405
- pathname delimiter (/), 276
- pathnames, 275
 - domain names as analogous to, 335–336
 - relative and absolute names, 335
- PCs (program counters), 30
- per-thread stacks, 30
 - saving registers in, 30–31, 31
- performance:
 - disk access time issues, 274
 - scheduling goals, 51
 - TLB design considerations, 183–184
 - See also* resource utilization; responsiveness
- permissions (file access permissions) (of users), 245–252
 - ACL specification of, 250
 - checking for, 251–252
 - denying vs. not allowing, 246–249
 - determining, 301, 302
 - for directories, 250–251, 261
 - for files, 250
- persistence services/systems, 269
 - B-trees in, 298
 - virtual memory and, 177–178
 - See also* persistent storage
- persistent objects, 12, 269, 270
 - identification of, 269
- persistent storage, 12–13, 269–323
 - checkpoints, 151
 - design influences on, 270
 - forms, 269–270
 - importance, 13
 - integrity in system crashes requirement, 13, 271
 - security issues, 315–316
 - security strategies, 315
 - vulnerabilities, 357
 - write operations to, 148
 - writing redo log entries to, 150–151
- PGP (Pretty Good Privacy), 358, 362
- phantoms, 164
- phase one (of transactions), 142
- phase one impurities (of transactions), 142
 - removing, 142–144
- phase two (of transactions), 142
- phase two impurities (of transactions), 142
 - removing, 142–144
- phishing, 403
- Physical Address Extension (PAE) mode, 214
- physical addresses, 165–168
 - page frames. *See* page frames (of physical addresses)
 - as shared by processes, 170–171
 - See also* addresses
- physical layer (in networking systems), 14, 329, 330, 355–356
 - security issues, 359
- pipelines (between threads), 89
- pipes (UNIX-family operating systems feature), 90–91, 119
- placement policy (page frame assignment selection), 204–205
- platters (of disk drives), 272
- plus sign (+): wildcard character, 372–373
- point-to-point architecture, 370
- pointers:
 - instruction pointers (IP registers), 30, 31
 - NULL pointer, 226
 - as problematic in shared memory, 234–235
 - to thread control blocks, 30
 - virtual file system implementations, 314
 - See also* stack pointers
- polymorphism, 271, 313
 - file system implementations, 313–314
- POP3 (Post Office Protocol–Version 3), 330, 332
- pop operation (popping values off the stack), 30–31, 429
- port numbers, 328, 339
 - NAT rewriting of, 353, 354
- portable APIs, 22
- POSIX API (standard), 22, 53
 - for access control lists, 249–252
 - for capabilities storage, 242
 - for condition variables, 98
 - for descriptors, 242
 - for file locks, 92
 - for files, 275–283, 306, 307
 - for fixed-priority scheduling, 53, 54
 - for mutexes, 78–80, 114
 - original UNIX API: print resource on, 322
 - print resource on, 266
 - for process management mechanisms, 222–230
 - for readers/writers locks, 92
 - for replacement policy exception procedures, 213
 - for threads (pthreads API), 22, 33
 - website resource on, 38, 121

462 ► Index

- Post Office Protocol–Version 3 (POP3), 330, 332
- PostgreSQL:
 - multiversion concurrency control, 153
 - website resource on, 163
- pound sign (#): wildcard character, 373
- PowerPC architecture: page table support, 194, 196
- PPs (Protection Profiles), 415
- pread** procedure (POSIX API), 279, 282(2)
 - simulating, 283
- predicate locks, 164
- preemption of threads, 33–34, 72, 111–112
- preemptive multitasking (in thread switching), 33–34
- prepaging, 202
 - clustered paging, 203–204; overriding, 217
 - payoff, 204
- presentation layer (in networking systems), 329, 330
- Pretty Good Privacy (PGP), 358, 362
- principals, 237, 238
 - vs. subjects, 257
- priority. *See* scheduling priority
- priority ceiling protocol, 122
- priority inheritance, 110–111
 - print resource on, 122
- priority inversion problem, 61, 109–111
 - solutions to, 110–111, 122
- priority queues, 53
- private storage:
 - address space as, 169, 170–171
 - virtual memory (MMU) and, 169, 170–171
- procedure activations:
 - leaf/nonleaf activations, 432
 - nesting of, 432
 - records (activation records), 428
 - using stacks for, 431–433
 - value storage options, 29, 432–433
- procedures:
 - calls to. *See* procedure activations
 - return addresses, 431–432
 - See also specific procedures*
- process ID numbers, 222, 223, 226
- process management mechanisms (POSIX), 222–230
- process switching:
 - and address translation, 185
 - context switching as, 33
 - and the page table, 185
- processes (protection environments), 9, 25, 220–268, 252
 - access rights. *See* access rights
 - address space, 172; of child processes, 223
 - capabilities. *See* capabilities
 - characteristics (definitions), 220–221
 - creating, 222–223; programming example, 224; to run different programs, 224–225
 - credentials. *See* credentials
 - DLL use, 171–172, 172
 - early terminology, 266
 - file descriptors for, 276
 - ID numbers, 222, 223, 226
 - inactivity periods, 180
 - loading and running programs, 224–227; programming examples, 226, 227
 - management mechanisms (POSIX), 222–230
 - operating system–application process boundary, 12, 232–233, 232, 233
 - POSIX API, 222–230
 - protection. *See* protection (memory protection)
 - protection mechanisms. *See* protection mechanisms
 - resource allocation for, 48, 49, 62, 221
 - running programs: loading and running, 224–227; programming examples, 226, 227; from shells, 225
 - shared memory use, 171–174, 172, 173
 - swapping of, 208
 - system context, 221
 - terminating, 230
 - thread–process distinction, 10
 - threads as in, 10, 220
 - vs. users, 257
 - working sets, 180, 208; print resource on, 218
- processor affinity (for threads), 45, 46
- processor architectures: VAX architecture, 187
 - See also* IA-32 architecture; Itanium architecture; MIPS architecture
- processor-intensive vs. disk-intensive thread
 - activities, 26, 27
 - maximizing throughput, 58
- processors:
 - address use, 166–167, 167
 - bursts of processor time: and scheduling priority, 48
 - memory locations. *See* registers
 - modes of operation, 231–234; VMM support, 255–256
 - switching from one thread to another. *See* thread switching
- product security assurance, 414–417
- program counters (PCs), 30
- proportional-share scheduling, 50, 51, 62–65
 - basic mechanisms, 62
 - via decay usage scheduling, 61–62, 71
 - Linux scheduler, 63–65
 - using resource shares/containers in, 66, 67
- protection (memory protection), 170, 230–237
 - address space schemes, 231–232, 234–237
 - context, 170, 228
 - mechanisms. *See* protection mechanisms
 - middleware’s job, 252
 - for multiple address space systems, 234–235

- operating systems and, 231
 - processor modes, 231–234
 - vs. security, 257
 - security issues, 257–262
 - for single address space systems, 235–237
 - for whole operating systems, 253–257
 - within single processes, 252–253
 - See also* access rights
 - protection domains, 238
 - creating: for whole operating systems, 253–257;
 - within single processes, 252–253
 - switching mechanism, 228–229, 238(2)
 - protection key registers, 236–237
 - protection keys (in hashed page table entries), 236
 - protection mechanisms (operations), 11–12, 239
 - access rights granting mechanism, 228–229, 238(2)
 - design options, 238
 - exec family procedure interaction with, 228
 - fitting into the access matrix model, 240
 - operating system–application process boundary, 12, 232–233, 232, 233
 - Protection Profiles (PPs), 415
 - protection systems theory, 240–241
 - protocol layers. *See* networking protocol layers
 - protocols, 327
 - cache coherence protocols, 46, 84–85
 - for networking. *See* networking protocols
 - priority ceiling protocol, 122
 - two-phase commit protocol, 154–156
 - See also specific protocols*
 - proxies (stub proxies) (in RPC), 374, 374, 375, 376
 - ps command (program), 216, 225–226, 226, 276, 277, 282
 - PSPACE-complete problems, 240, 267
 - `pthread_cond_broadcast` procedure (POSIX API), 98
 - `pthread_cond_init` procedure (POSIX API), 98
 - `pthread_cond_signal` procedure (POSIX API), 98
 - `pthread_cond_wait` procedure (POSIX API), 98
 - `pthread_create` procedure (POSIX API), 22
 - `pthread_kill` procedure (POSIX API), 36–37
 - `PTHREAD_MUTEX_DEFAULT`, 80
 - `PTHREAD_MUTEX_ERROR_CHECK`, 80
 - `pthread_mutex_init` procedure (POSIX API), 78
 - `pthread_mutex_lock` procedure (POSIX API), 78
 - `PTHREAD_MUTEX_NORMAL`, 80, 82
 - `PTHREAD_MUTEX_RECURSIVE`, 80
 - `pthread_mutex_t` type, 78
 - `pthread_mutex_timedlock` procedure (POSIX API), 79
 - `pthread_mutex_trylock` procedure (POSIX API), 79
 - `pthread_mutex_unlock` procedure (POSIX API), 78
 - pthread API, 22
 - `yield` procedure equivalent, 33
 - public methods: `synchronized` keyword for, 81, 82, 94
 - public-key cryptography, 362
 - publish/subscribe messaging, 370–371, 371, 372–373
 - Java RMI example, 377–382
 - `Publisher` class, 379
 - purity (of transactions), 142
 - push operation (pushing values onto the stack), 30–31, 429
 - `pwrite` procedure (POSIX API), 279, 282(2)
 - simulating, 283
- ## Q
- quadratic growth in code size: avoiding, 162–163
 - quality (system quality):
 - risk-management approach to, 396
 - and security, 396–397
 - quanta (thread running times), 58, 71
 - queuing mutexes, 86–87
 - vs. spinlocks, 86
 - Quinlan, Sean, 323
- ## R
- `r` permission:
 - ACL specification of, 250
 - for directories, 250
 - using with/without the `x` permission, 251
 - ractions (uncontrolled thread interactions), 72, 75–76
 - avoiding vs. analyzing, 121
 - race condition vulnerabilities, 114–115
 - TOCTTOU race bugs, 115, 122
 - radiation therapy machine (Therac-25), 76, 121
 - print resource on, 121
 - radix trees, 194
 - Rago, Stephen A., 266
 - RAIDs (Redundant Arrays of Independent Disks):
 - print resources on, 322
 - Rajkumar, Ragnathan, 71, 122
 - RAM (random access memory):
 - in disk drives, 274
 - in main memory. *See* memory (main memory)
 - Randell, B., 322
 - rate-monotonic scheduling, 54, 56
 - print resource on, 71
 - testing real-time schedules, 54–56
 - read actions (operations):
 - demand-driven reading, 177
 - nonrepeatable reads, 153, 154
 - notation for, 137
 - repeatable reads, 154
 - as reversible in conflicts, 140–141
 - read ahead (paging), 203
 - read around (clustered paging), 203–204
 - read committed mode (for transactions), 152–153

464 ► Index

- read permission. *See* **r** permission
- read** procedure (POSIX API), 279, 282–283
- read requests, 272, 274
- read-only virtual address space, 171
- read-only virtual addresses, 169
- read/write virtual address space, 171
- readers/writers locks, 91–92, 91, 122
 - monitors with condition variables and, 98
 - vs. mutexes, 92
 - print resource on, 122
 - rules for guaranteeing serializability, 137, 141–142
 - starvation of threads by, 92, 122
- reading files, 279–283
 - sequentially, 282–283
 - at specified positions, 281–282
- real-time schedules: testing, 54–56
- receive method, 380, 391
- receiving capabilities, 242–243
- records. *See* rows (of tables)
- recovery:
 - deadlock detection and recovery, 105–106, 107, 128–130, 131
 - speed after system crashes, 151
- recovery processing, 148
 - rules, 149–150
- recursion problem in linear page tables, 188
- recursive locking, 80
- recursive mutexes, 80, 96
- recvmsg** procedure (POSIX API), 243
- redirect standard input symbol (<), 276
- redirect standard output symbol (>), 276
- redo logging, 150–151
- redo logs, 150
 - and commit operations, 151
 - writing entries into persistent storage, 150–151
- Redundant Arrays of Independent Disks (RAIDs):
 - print resources on, 322
- redundant data transmission formats, 348, 348
- reference bit (in page table entries), 211
 - print resource on, 218
- Regehr, John, 71
- registers (processor registers):
 - IP registers (instruction pointers), 30, 31
 - procedure activation value storage in, 29, 432–433
 - protection key registers, 236–237
 - saving into and restoring from memory (push and pop operations), 29, 30–31, 31, 429, 432–433; as into and from activation records, 432–433
 - See also* stack pointers (SP registers)
- registries (of remote objects), 376
 - passing and binding remote objects to names in, 376–377, 379–381, 380, 381
- relational database systems. *See* database systems
- relational databases:
 - security issues, 158
 - tables, 12, 270; creating, 127; displaying, 127–128
- Remote** interface (Java RMI), 377
- Remote Method Invocation. *See* RMI
- remote objects:
 - locating, 376, 377, 378–379, 379
 - passing and binding to names in registries, 376–377, 379–381, 380, 381
 - registries of, 376
- Remote Procedure Call. *See* RPC
- RemoteException** class (Java RMI), 377
- repeatable reads, 154
- replacement policy (page eviction), 205–212
 - context, 205–208
 - creating exceptions to, 212–213
 - specific policies, 209–212; comparisons of, 210, 219
- replay attacks, 406
- resizing stacks, 429–430
- resolvers (DNS request agents), 336
- resource allocation (for processes), 48, 49, 62, 221
- resource allocation graphs, 105, 106, 108
 - print resource on, 122
- resource allocation-oriented scheduling, 49–50
 - See also* proportional-share scheduling
- resource consumption: by transactions, 158
- resource containers:
 - print resource on, 71
 - use in proportional-share scheduling, 67
- resource managers, 154, 155, 156
- resource ordering:
 - deadlock prevention through, 104–105; print resource on, 121
- resource records: for domain names, 336
- resource utilization (by computer systems), 24, 44
 - concurrent threads and, 24, 26–27
- response time (of computer systems), 47
 - Gantt chart illustration, 67–68
 - thread switching frequency and, 48
 - See also* responsiveness
- response time-oriented scheduling, 47–48, 58–59
- responsiveness (of computer systems), 24
 - concurrent threads and, 24–26, 27
 - See also* response time
- restoring registers (pop operation), 29, 30–31, 31, 429, 432–433
- ret** instruction, 34
- retrieving information about standard input, 278–279
- return addresses (of procedures), 431–432
 - storage options, 432
- return from interrupt instruction, 34
- Reuter, Andreas, 125, 163(2)

- reversion of transactions. *See* failure atomicity
 - Riedel, Erik, 322
 - Rijndael (encryption standard), 362
 - risk-management approach to attacks, 397
 - Ritchie, Dennis M., 322
 - RMI (Remote Method Invocation), 376
 - Java RMI exception potential, 377–378
 - Java RMI publish/subscribe messaging example, 377–382
 - website resource on, 394
 - rmiregistry** program (Java RMI), 381
 - Robbins, Kay A. and Steven, 266
 - rollback** command, 129
 - rollback of transactions, 106, 129, 131
 - as a sign of attack, 157
 - root** account, 260
 - Rosenblum, Mendel, 323
 - Rosencrantz, Daniel J., 164
 - Ross, Blake, 426
 - Ross, Keith W., 368
 - rotational latency (in disk access times), 274
 - round-robin (RR) policy, 53–54
 - routers, 326, 326–327
 - forwarding tables, 352
 - gateway routers, 351
 - label-switching routers, 352
 - message authentication feature, 358
 - NAT routers, 353, 354, 360
 - routing, 351–352
 - label-switching vs., 352
 - NAT routing, 353–355
 - rows (of tables), 270
 - RPC (Remote Procedure Call), 15, 369–370
 - ACL use with, 387
 - vs. messaging, 371–372
 - object-oriented and non-object-oriented standards, 376
 - principles of operation, 374–377, 374, 375
 - RR policy (round-robin policy), 53–54
 - RSA system (of asymmetric-key encryption), 362–363
 - Rubin, Aviel D., 426
 - run** method (Java API), 22, 161
 - run queue, 41–42, 42, 43, 61
 - Runnable** interface (Java API), 161
 - run** method, 22, 161
 - runnable state (of threads), 42–43
 - changes in, 43
 - running programs, 224–227
 - in the current directory, 258
 - programming examples, 226, 227, 277
 - from shells, 225, 276–278; **xclock**, 227–228
 - running state (of threads), 42–43
 - changes in, 43
 - running times (of threads) (before switching), 58, 71
 - runqueue structure (Linux), 104
 - runtime environment, 428
 - runtime stacks. *See* stacks
 - Russinovich, Mark E., 38, 71
 - Ruzzo, Walter L., 240, 267
- S**
- S/MIME (Secure/Multipurpose Internet Mail Extensions), 358
 - Saltzer, Jerome H., 267, 399, 426
 - Santry, Douglas S., 323
 - Sarbanes-Oxley Act security requirements, 316
 - Sathaye, Shirish S., 71
 - saving registers (push operation), 29, 30–31, 31, 429, 432–433
 - sched_yield** procedure (POSIX API), 33
 - schedulers (processor schedulers), 39
 - decay usage schedulers, 59–62
 - Linux scheduler, 63–65, 71
 - Mac OS X scheduler, 59–60, 60, 61, 71
 - Microsoft Windows scheduler, 59, 60–61, 71; print resource on, 71
 - multilevel feedback queue schedulers, 61; print resource on, 71
 - preempt authority, 33, 72, 111
 - scheduling (of threads), 10, 32, 39–71
 - decay usage scheduling, 52, 59–62
 - dynamic-priority scheduling, 52, 56–62; testing adjustments, 69
 - Earliest Deadline First (EDF) scheduling, 52, 56–57; deadline inheritance for, 111
 - fair-share scheduling, 49–50; proportional-share scheduling vs., 50
 - fixed-priority scheduling, 52, 52–56
 - goals. *See* scheduling goals
 - importance-oriented scheduling, 49, 51, 58
 - lottery scheduling, 62–63, 66–67; print resource on, 71
 - mechanisms, 40, 52, 52–65
 - processor affinity and, 45, 46
 - proportional-share. *See* proportional-share scheduling
 - rate-monotonic scheduling, 54, 56; print resource on, 71
 - resource allocation-oriented scheduling, 49–50. *See also* proportional-share scheduling
 - response time-oriented scheduling, 47–48, 58–59
 - security issues, 65–67
 - stride scheduling, 67; print resource on, 71
 - synchronization and, 73, 109–114
 - thread states, 42–43; changes in, 43
 - throughput-oriented scheduling, 44–46, 58
 - urgency-oriented scheduling, 48–49, 51
 - scheduling goals, 40, 44–52, 51
 - importance, 48, 49
 - resource allocation, 48, 49, 62

466 ► Index

- scheduling goals (*cont.*)
 - response time, 47
 - throughput, 44, 58
 - urgency, 48–49
- scheduling priority, 48–50
 - base priorities, 58; in decay usage scheduling, 59–62
 - bursts of processor time and, 48
 - convoy problem, 112–113
 - dynamic priority adjustments, 56, 57–61; testing, 69
 - inversion problem, 61, 109–111
 - numerical priorities, 52–54
 - operating system parameters, 51
 - reasons for adjusting, 58–59
 - synchronization and, 73
 - as unclear without further clarification, 50–51
 - user-specified priorities, 58
 - See also* scheduling goals
- Scholten, Carel S., 121
- Schroeder, Michael D., 399, 426
- Schwartz, Alan, 426
- SCTP (Stream Control Transmission Protocol), 330, 349
- search path: leaving the current directory out, 258
- Seawright, L. H., 268
- sectors (of disk tracks), 272
- Secure Hash Algorithm 1 (SHA-1), 364
- Secure Sockets Layer. *See* SSL
- Secure/Multipurpose Internet Mail Extensions (S/MIME), 358
- security (system security), 15–16, 395–427
 - authentication as the prerequisite of, 398. *See also* authentication (of users)
 - best practices, 419–421
 - confidential data writing issue, 212–213
 - contingency planning and, 316–317
 - database issues, 158
 - defense in depth principle, 261–262
 - deleting files as inadequate, 315
 - firewalls, 359–360, 361
 - guiding principles, 398–401
 - I/O issues, 158–159
 - information-flow control, 407–410
 - intrusion detection systems (IDSes), 360–361, 418
 - irrevocable capabilities issue, 245
 - isolation issues, 158
 - isolation of machines exposed to attack, 360
 - malware, 411–414
 - monitoring, 398, 417–419
 - networking issues, 356–365
 - nontechnical means, 397–398
 - objectives, 16, 398
 - overwriting files as inadequate, 315–316
 - persistent storage issues, 315–316
 - persistent storage strategies, 315
 - policy categories, 241
 - print resources on, 426
 - product security assurance, 414–417
 - vs. protection, 257
 - protection issues, 257–262
 - and quality, 396–397
 - risk-management approach to, 397
 - Sarbanes-Oxley Act requirements, 316
 - scheduling issues, 65–67
 - setuid program enhancement of, 260–261
 - setuid program issues, 229, 259–260
 - synchronization issues, 114–115
 - thread-switching (multithreading)
 - issues, 34–35
 - transaction enhancement of, 156
 - transaction issues, 157–159
 - transaction processing system enhancement of, 157, 158
 - transaction processing system issues, 156
 - user interface design and, 261
 - vandalism mitigation, 316
 - virtual memory issues, 212–213
 - VMM issue, 261–262
 - website resource on, 426
- security monitoring, 398, 417–419
- security policies: categories, 241
- security requirements (in STs), 415
- Security Targets (STs), 415
- Security-enhanced Linux (SELinux) system, 411
- seek times, 273–274
- seeks (on disk drives), 273
- segment table, 198
- segmentation (of address space), 169, 182, 197–201
 - ASID-segmentation comparison, 201
 - disincentives for, 201
 - paging-segmentation combination, 198–201, 199, 200
 - print resource on, 218
- Segmented FIFO (SFIFO) replacement policy, 211
 - print resources on, 219
- segments (of address space), 198
- segments (transport-layer data transmission chunks), 345–346
- select** command (SQL), 127–128
- selective acknowledgment of segment transmissions, 348
- SELinux system (Security-enhanced Linux system), 411
- semaphores, 73, 98–100, 99
 - and bounded buffers, 98, 121
 - vs. monitors with condition variables, 98
 - as mutexes, 99
 - operations on, 99
 - print resource on, 121
 - use for, 99–100
- sending capabilities, 242–243
- sendmsg** procedure (POSIX API), 243

- sequential I/O, 282–283
- sequential threads, 21
 - processor-intensive vs. disk-intensive activities, 26, 27
- serial execution (of transactions), 137
 - histories. *See* serial histories
- serial histories (of systems/transactions), 138, 142, 144
 - print resources on, 164
 - vs. serializable histories, 144
- serializability, 137, 142
 - print resources on, 164
 - rules for guaranteeing, 137, 141–142
 - sacrificing to increase concurrency, 152
 - snapshot isolation and, 154
- serializable execution (of transactions), 137
 - vs. deterministic execution, 138
 - equivalence, 137; print resources on, 164
 - histories. *See* serializable histories
- serializable histories (of systems/transactions), 139, 142, 144
 - equivalence, 139; print resources on, 164; verification of, 139–141, 142–143
 - print resources on, 164
 - rules for guaranteeing serializability, 137, 141–142
 - vs. serial histories, 144
- Server** class, 343
- Server Message Block (SMB) protocol, 338
- server-side stream sockets: life cycle (through states), 341
- servers:
 - certificates, 387–388
 - iSeries (System/38–AS/400) (IBM), 237, 242, 267
 - in messaging systems, 132–133, 132, 371, 371–372, 372
 - name servers, 336, 337
 - in RPC systems, 374–375, 374, 375
 - throughput, 44
 - See also* web servers
- ServerSocket** class (Java API), 341, 342
- session layer (in networking systems), 329, 330
- set group ID bit, 252
- set user ID bit, 228
- set-associative caches, 204
- setgid bit, 252
- setgid programs: programming guidelines, 260
- setuid bit, 228
- setuid programs, 228–229, 238(2)
 - programming oversights and guidelines, 259–260
 - security enhancement, 260–261
 - security issues, 229, 259–260
- SFIFO (Segmented FIFO) replacement policy, 211
 - print resources on, 219
- Sha, Lui, 71, 122
- SHA-1 (Secure Hash Algorithm 1), 364
- shadow paging, 311, 312, 313, 323
 - print resource on, 323
- Shapiro, Jonathan S., 426–427
- shared data structures:
 - locks. *See* locks (on shared data structures)
 - persistent storage forms, 12
 - race problem, 74–76
 - transaction actions on stored entities, 137–138
- shared links (in networks), 355–356
- shared memory (in multiple address space systems), 234
 - pointers as problematic in, 234–235
 - virtual memory and, 171–174
- shared objects: virtual addresses as names for, 168, 235
- shared secret cryptographic techniques, 362
 - MACs, 363, 364
- shells, 5
 - running programs from, 225, 276–278; **xclock**, 227–228
 - stripped-down shell, 228, 229
- Shortest Job First (SJF) policy, 47–48
- Shoshani, A., 122
- shoulder surfing, 402
- signal operation (notify operation), 94, 96–97
- signatures (of viruses and worms), 414
- Simple Mail Transfer Protocol (SMTP), 330, 332
- simple security property, 410
- single address space systems, 235–237
 - debugging in, 235
 - protection key protection scheme, 236–237
- single indirect blocks, 293
- single-threaded programs, 20
- single-threaded systems: waiting mechanism, 41
- single-threaded web servers, 25
- SJF (Shortest Job First) policy, 47–48
- skeletons (ties) (in RPC), 375, 375
- slash character (/): pathname delimiter, 276
- sleep** method (Java API), 23, 35, 117
- sleep** procedure (POSIX API), 23
- sleep** procedure (Win32 API), 37
- smashing the stack buffer overflow, 413, 414
- SMB (Server Message Block) protocol, 338
- SMTP (Simple Mail Transfer Protocol), 330, 332
- snapshot isolation, 153–154
 - print resources on, 164
 - and serializability, 154
- snapshots, 312
- sniffers, 402
- SOAP (Simple Object Access Protocol—formerly), 384
 - website resource on, 394
- social engineering, 401
- socket APIs, 340–344
- Socket** class (Java API), 341, 342

468 ► Index

- sockets (for transport-layer connections),
 - 14, 340
 - binding of (to their own addresses), 340
 - life cycles (through states) of all three types, 341
 - server connections and communications with, 342–344; programming examples, 343, 344
 - states, 340–342
 - types, 40
- soft links. *See* symbolic links
- soft page faults, 205
- soft updates strategy, 312, 313
 - print resource on, 323
- software TLBs, 197
- software/hardware interface, 182–185
 - for architectures with/without a page table walker, 184–185
- Solomon, David A., 38, 71
- Soltis, Frank G., 267
- Soules, Craig A. N., 323
- SP registers. *See* stack pointers
- Spafford, Gene, 426
- sparse address space allocation, 176–177
- sparse files, 295
- spatial locality:
 - of processes: and demand paging, 203
 - of programs, 45
 - of virtual addresses, 182–183
 - of virtual memory accesses, 190
- speed of recovery after system crashes, 151
- spinlocks, 121
 - basic spinlocks, 83–84, 84
 - cache-conscious spinlocks, 84–85
 - print resource on, 121
 - vs. queuing mutexes, 86
- spoofing attacks, 402–403, 402
- Spotlight search feature (Mac OS X), 303, 304–305
- spurious wakeups, 96
- SQL commands, 127–128
- SQL Server: read committed mode, 152
- SQL Slammer worm, 358
- SSL (Secure Sockets Layer), 358, 362, 403
 - limitation, 389–390
 - web services usage, 389
- stack algorithms, 210–211
- stack pointers (SP registers), 30, 31, 430–431, 431
 - saving and switching, 30–31
- stack-allocated storage, 429
- stacks (of activation records), 428–433, 431
 - allocate and deallocate operations on, 429–430, 431
 - per-thread stacks, 30–31, 31
 - pushing and popping registers onto and off, 30–31, 429
 - as represented in memory, 430–431
 - resizing, 429–430
 - smashing the stack buffer overflow, 413, 414
 - using for procedure activations, 431–433
 - virtual memory use, 430
- standard error output file descriptor, 276
- standard input, 276
 - retrieving information about, 278–279
- standard input file descriptor, 276
- standard output, 276
- standard output file descriptor, 276
- standby page list, 206, 207
- star property (*-property), 410
- starvation (of threads), 65
 - by readers/writers locks, 92, 122
- stat** procedure (POSIX API), 301
- stater** program, 301, 302
- STDERR_FILENO**, 276
- STDIN_FILENO**, 276
- STDOUT_FILENO**, 276
- Stearns, Richard E., 164
- Stevens, W. Richard, 266, 368
- sticky bit **w** permission limitation, 251
- Stirling, Colin, 164
- stop** method (Java API), 37
- storage:
 - private storage, 169, 170–171
 - procedure activation value storage options, 432–433
 - stack-allocated storage, 429
 - See also* disk storage; memory; persistent storage; virtual memory
- stored objects/entities:
 - persistent forms, 12
 - phantoms, 164
 - protection goal, 170
 - transaction actions on, 137–138
- storing:
 - buffers, 100
 - of login passwords, 212
 - undo logs, 148
- Stream Control Transmission Protocol (SCTP), 349
- stride scheduling, 67
 - print resource on, 71
- STs (Security Targets), 415
- stub proxies (in RPC), 374, 374, 375, 376
- Sturgis, Howard E., 163
- subdirectories: disk space allocation
 - policies, 289
- subjects, 237
 - as objects, 237
 - vs. principals, 257
 - See also* processes
- subscribe method, 379
- Subscriber** class, 380–381, 382
- substituting RAM for disk storage, 180
- Sun Microsystems: email delivery software
 - TOCTTOU race bugs, 115, 122

- supervisor mode (kernel mode) (of processors), 231
 - swapping (of processes), 208
 - swaps (of transaction actions), 139–141, 142–143
 - illegal and legal swaps, 140
 - switches (in networks), 325
 - switchFromTo** procedure (POSIX API), 29, 34
 - switching threads. *See* thread switching
 - SwitchToThread** procedure (Win32 API), 33–34
 - symbolic links (soft links) (in file systems), 307–308, 307, 336
 - symmetric-key encryption (cryptography), 362
 - standard techniques, 362
 - synchronization (of threads), 10–11, 25–26, 72–122
 - barriers, 92–93, 93, 98
 - bounded buffers, 89–91
 - bugs in, 74, 114–115
 - condition variables (with monitors), 73, 94–98
 - messaging as, 74
 - mutual exclusion. *See* mutual exclusion
 - patterns, 73, 89–93
 - problem from. *See* deadlocks
 - readers/writers locks, 91–92, 91, 98
 - and scheduling, 73, 109–114
 - security issues, 114–115
 - semaphores, 73, 98–100, 99
 - synchronized** keyword: for public methods, 81, 82, 94
 - synchronized** statement, 81–82, 100
 - synchronous writes strategy, 311, 312
 - system calls, 231
 - for C-list entries, 242
 - See also specific procedures*
 - system crashes:
 - from buffer overflow, 412–413
 - durability in, 164
 - failure atomicity in, 164
 - and metadata integrity, 310
 - persistent storage integrity requirement, 13, 271
 - recovery speed after, 151
 - system histories (transaction histories), 138–144
 - equivalence of serializable histories, 139; verification of, 139–141, 142–143
 - nonserializable histories, 139
 - print resources on, 164
 - rules for guaranteeing serializability, 137, 141–142
 - serial histories, 138, 142, 144; vs. serializable histories, 144
 - serializable histories, 139, 142, 144; vs. serial histories, 144
 - system mode (kernel mode) (of processors), 231
 - system security. *See* security
 - System/38 (IBM), 237
 - print resource on, 267
- T**
- tables. *See* address-mapping table; clustered page tables; hashed page tables; inverted page tables; linear page tables; multilevel page tables; page tables; segment table; tables (in relational databases)
 - tables (in relational databases), 12, 270
 - creating, 127
 - displaying, 127–128
 - tag bits (on memory words): for preventing capability forgery, 243
 - Talluri, Madhusudhan, 218
 - Tanenbaum, Andrew S., 368
 - Targets of Evaluation (TOEs), 415
 - task control blocks. *See* thread control blocks
 - tasks, 25
 - See also* processes; threads
 - TCBs. *See* thread control blocks
 - TCP (Transport Control Protocol), 328, 330, 339, 344–347
 - congestion control, 346–347, 347–348
 - evolution within and beyond, 347–349
 - message authentication feature, 358
 - segment transmission and acknowledgment mechanisms, 345–346
 - TCP sockets. *See* sockets
 - TCP Vegas, 348
 - telnet** program:
 - connecting to web servers with, 333–334, 342–343
 - TCP connection, 339
 - temporal locality:
 - of programs, 45
 - of virtual addresses, 182–183
 - of virtual memory accesses, 190
 - TENEX file system, 323
 - terminating processes, 230
 - testing:
 - the **BoundedBuffer** class, 94
 - dynamic priority adjustments, 69
 - real-time schedules, 54–56
 - Thanish, Peter, 164
 - theorems: on fixed-priority scheduling, 54–56, 57
 - theory of protection systems, 240–241
 - Therac-25 (radiation therapy machine), 76, 121
 - print resource on, 121
 - Thompson, Ken, 268, 322
 - thrashing, 208
 - Thread** class, 161
 - childThread** object, 22
 - currentThread** method, 161
 - sleep** method, 23, 35, 117
 - stop** method, 37

470 ► Index

- thread control blocks (TCBs), 29
 - pointers to, 30
 - saving registers in, 30–31, 31
- thread interactions, 10, 72
 - controlling, 2, 10–12. *See also* synchronization; virtual memory
 - supporting across space, 3, 14–15. *See also* messaging; networking; RPC (Remote Procedure Call); web services
 - supporting across time, 2, 12–13. *See also* file systems
 - uncontrolled. *See* races
- thread scheduling. *See* scheduling (of threads)
- thread states, 42–43
 - changes in, 43
- thread switching (multithreading):
 - automatic, 33
 - context switching as, 33
 - cooperative tasking approach, 33
 - determining the number of switches per second, 38
 - frequency: and response time, 48
 - overhead cost, 44–46, 71, 111, 112
 - preemptive multitasking approach, 33–34
 - programming example, 31–32
 - running times (before switching), 58, 71
 - security issues, 34–35
- threads, 9, 19–38
 - API support for, 10
 - categories, 12
 - concurrent. *See* concurrent threads
 - convoy phenomenon, 87, 109, 111–114
 - convoys, 112
 - deadlocks. *See* deadlocks
 - dispatching, 33, 43
 - DoS attack strategies and defenses, 65–67
 - execution status information, 30
 - fibers (user-level threads), 21, 232, 233, 233; print resource on, 71
 - groups sharing a virtual address space. *See* processes
 - implementation options, 232–233
 - interactions between. *See* thread interactions
 - interleaving of, 72, 76
 - isolating threads of different processes, 11
 - kernel threads, 232–233
 - lifetimes, 20
 - multiple. *See* multiple threads
 - native threads (kernel-supported user threads), 232–233, 232
 - pipelines between, 89
 - preemption of, 33–34, 72, 111–112
 - process–thread distinction, 10
 - as in processes, 10, 220
 - processor affinity for, 45, 46
 - processor-intensive vs. disk-intensive activities, 26, 27, 58
 - program–thread distinction, 19
 - referring to, 29
 - running times (before switching), 58, 71
 - scheduling. *See* scheduling (of threads)
 - sequential threads, 21
 - spawning of, 22
 - stacks for, 30
 - starvation of, 65, 92, 122
 - switching. *See* thread switching
 - synchronization of. *See* synchronization (of threads)
 - user-level threads (fibers), 21, 232, 233, 233; print resource on, 71
 - virtual memory as accessible to, 220–221
 - waiting mechanisms: busy waiting, 41; wait operation, 94, 96. *See also* wait queues
 - waking mechanisms: notify operation, 94, 96–97; timer interrupts, 41–42, 43
- throughput (of computer systems), 44, 58
 - convoy phenomenon and, 109, 113
- throughput-oriented scheduling, 44–46, 58
- ticket-sales example, 74–76, 91
- tie-breaking strategies for numerical scheduling priorities, 53–54
- ties (skeletons) (in RPC), 375, 375
- time bursts (of processor time): and scheduling priority, 48
- Time Of Check To Time Of Use (TOCTTOU) race bugs, 115, 122
- time slices (thread running times), 58, 71
 - size: and TLB performance, 184
- time stamps (for files), 301
- timer interrupts, 41–42, 43
- TLB hits, 183
- TLB misses, 183, 185
- TLBs (translation lookaside buffers), 183
 - ASID tags, 185
 - design considerations, 183–184
 - entry flushing, 185
 - page table entry loading into, 184
 - performance vs. page table performance, 184
 - software TLBs, 197
- TOCTTOU (Time Of Check To Time Of Use) race bugs, 115, 122
- TOEs (Targets of Evaluation), 415
- tokens, 405
- `top` command, 70
- Topic interface, 378, 378
- TopicServer class, 379, 380, 381, 391
- tracks (on disks), 272
- transaction actions:
 - conflicting pairs, 140–141, 143
 - equivalence-preserving swaps, 139–141, 142–143; illegal and legal swaps, 140
 - execution of. *See* execution (of transactions)

- notation for, 137–138
- in serial, serializable, and nonserializable histories, 138–139
- transaction context, 155
- transaction histories. *See* system histories
- transaction manager, 154, 155, 156
- transaction processing systems:
 - components, 154
 - coordinating subsystems, 154–156
 - print resources on, 163
 - rules and operation, 141–142
 - security enhancements, 157, 158
 - security issues, 156
 - undo logging, 145–147
 - write operation approach, 149
- transactions (atomic transactions), 11, 74, 123–164
 - abort actions, 125, 129; compensating for not executing in workflow systems, 134–135; upon restart, 148–149; and undo logs, 148
 - abstraction of, 157, 158
 - ACID test of system support for, 125; print resource on, 163
 - actions. *See* transaction actions
 - atomicity, 124–125, 125–126; aspects, 124, 137; mechanisms for ensuring, 137–147. *See also* —: failure atomicity, *below*
 - commit actions, 125, 129; delaying, 136
 - compensating transactions, 134–135
 - concurrency, 126, 151; increasing, 152
 - concurrent transactions, 11, 126, 130
 - consistency, 125, 144
 - context, 155
 - in database systems, 127–130
 - deadlocks between, 144
 - deterministic execution of, 138
 - durability, 125; and commit operations, 150; failure atomicity and, 148–149; issues, 147–148; in message-queuing systems, 131–132; in system crashes, 164; update storage, 150–151; write-ahead logging, 149–150
 - equivalence of serializable executions, 137; print resources on, 164
 - execution of, 137, 138
 - failure atomicity, 124, 144–145; ad hoc programming difficulties, 145, 146, 157; and durability, 148–149; in system crashes, 164; undo logging (transaction processing system), 145–147
 - failures, 124, 125, 145; testing complications, 145, 146
 - histories. *See* system histories
 - impurities, 142; removing, 142–144
 - inconsistent states (temporary), 144–145
 - isolation, 11, 124, 125, 126; print resources on, 164; reduced, 152–153; security issues, 158; snapshot isolation, 153–154
 - in message-queuing systems, 15, 130–135 and metadata, 136
 - phases, 142
 - print resources on, 163
 - processing systems. *See* transaction processing systems
 - purity, 142
 - read committed mode, 152–153
 - resource consumption, 158
 - reversion of. *See* —: failure atomicity, *above*
 - rollback of, 106, 129, 131; repeated rollback attacks, 157
 - rooted tree analogy, 123–124, 124, 125–126
 - security enhancement, 156
 - security issues, 157–159
 - serial and serializable execution of, 137
 - serial histories, 138, 142, 144
 - serializable histories. *See* serializable histories as storage-related, solely, 138
 - two-phase locking, 137–144; concurrency price, 151; rules, 137, 141–142; systems suitable for, 151–152; undoing write operations and, 147
 - update storage, 150–151
 - visibility to the user, 136
 - workload mix, 151–152
- translation lookaside buffers. *See* TLBs
- Transport Control Protocol. *See* TCP
- transport layer (in networking systems), 14, 328, 330, 339–349
 - data transmission chunks (segments), 345–346
 - security problems and enhancements, 358
 - services, 339
- transport protocols, 328, 330, 349
 - support sources, 331
- traps, 231
 - VMM trap simulation, 255–256, 256
- traversing directories, 251
- tree monitors, 126
- trie data structure, 194
- triple indirect blocks, 294
- Tripwire system, 418–419
- Trojan horses, 257
 - credential-related vulnerabilities, 257–259
 - print resource on, 268
 - viruses and worms vs., 411
- tuples. *See* rows (of tables)
- turnaround time (in batch processing), 47–48
- Turner, Clark S., 121
- Turner, Rollins, 219
- two-factor authentication, 405–406
- two-phase commit protocol, 154–156
 - components, 154, 155
 - process steps, 155–156, 155

472 ► Index

two-phase locking, 137–144
 concurrency price, 151
 print resource on, 164
 rules, 137, 141–142
 systems suitable for, 151–152
 undoing write operations and, 147
 type systems: for preventing capability
 forgery, 244

U

UDDI (Universal Description, Discovery, and
 Integration), 387
 UDP (User Datagram Protocol), 339
 Ullman, Jeffery D., 219, 240, 267
 unbound state (of sockets), 340
 undo logging, 145–147
 undo logs, 135, 145–147
 abort actions and, 148
 combined log of, 147, 149
 storing, 148
 write operations to, 148
 undo operation: idempotency, 149
 undoing write operations, 147
 unique indexes, 304
 unitary, 163
 United States military classification
 levels, 407–409
 Universal Description, Discovery, and Integration
 (UDDI), 387
 UNIX-family operating systems:
 file system design, 291–292
 pipes feature, 90–91, 119
 replacement policy, 208
 and segmentation, 201
See also Linux; Mac OS X; POSIX API (standard)
unlink procedure (POSIX API), 306–307
 unlock actions (operations):
 assumptions regarding, 139
 notation for, 138
 transaction processing system rules, 141–142
See also unlocking mutexes
 unlocking mutexes, 77–78, 77
 queuing mutexes, 87, 87, 88
 spinlocks, 84, 84
 unordered linear lists, 308–309
up operation, 99
 update storage (for transactions), 150–151
 urgency (of tasks/threads), 48–49, 51
 specification of, 49
 urgency-oriented scheduling, 48–49, 51
 USENIX Association: print resources on, 17, 18
 user authentication. *See* authentication (of users)
 User Datagram Protocol (UDP), 339
 user groups (file owners), 249–250
 user interface design: and security, 261
 user mode (of processors), 231
 user-interface programs, 5

user-level threads (fibers), 21, 232, 233, 233
 print resource on, 71
 user-specified priorities, 58
 users:
 authentication of. *See* authentication (of users)
 file access permissions. *See* permissions
 principals, 237, 238
 vs. processes, 257

V

v (up) operation, 99
 valid bits: linear page table entry access control
 bits, 185–186, 186, 237
 values:
 inconsistency of variable values, 75–76
 procedure activation value storage options,
 432–433
See also registers
 Van Horn, Earl C., 266
 vandalism mitigation, 316
 variables:
 condition variables (with monitors), 73, 94–98
 inconsistency of values, 75–76
 local variable value storage options, 432–433
 VAX architecture, 187
 print resource on, 218
 verifier (JVM), 253
 dataflow analysis theorem, 253, 264
 print resource on, 267
 versioning of files, 323
 print resources on, 323
 snapshots, 312
vfs_write procedure (Linux kernel), 314,
 314
 VFSs (virtual file systems), 314
 distributed file system support, 338
 virtual address space. *See* address space
 virtual addresses, 165–168
 instructions in, 235
 as names for shared objects, 168, 235
 pages. *See* pages (of virtual addresses)
 read-only addresses, 169
 space for. *See* address space (virtual
 address space)
 spatial locality, 182–183
 temporal locality, 182–183
 undefined addresses, 169
See also addresses
 virtual file systems. *See* VFSs
 Virtual Machine Monitors. *See* VMMs
 virtual memory, 11, 165–219
 as accessible to threads, 220–221
 demand-driven program loading, 178
 dirty page tracking, 177–178
 flexibility, 169
 function (essence), 165–166
 linear page table storage solution, 188–190

- mapping of files into, 279–281
 - mechanisms, 169, 180–201. *See also* MMU (memory management unit)
 - memory allocation, 174–176; of sparse address space, 176–177
 - as paging (page frame assignment), 197
 - paging to disk, 180
 - and persistence services, 177–178
 - policies for, 201–212
 - print resource on, 218
 - and private storage, 170–171
 - properties, 168–169
 - security issues, 212–213
 - and shared memory, 171–174
 - sparse address space allocation, 176–177
 - stack use of, 430
 - uses for, 170–180, 218
 - VMM handling of, 256–257
 - zero filling. *See* zero filling (of memory)
 - See also* MMU (memory management unit)
 - virtual memory accesses: temporal and spatial locality, 190
 - Virtual Private Networks (VPNs), 359
 - viruses, 412
 - antivirus scanning programs and, 414
 - signatures, 414
 - Vista (Windows), 303
 - VM/370, 268
 - print resource on, 268
 - VMMs (Virtual Machine Monitors), 254–255
 - operation of, 255–257, 256
 - print resource on, 267
 - security issue, 261–262
 - trap simulation, 255–256, 256
 - VMS operating system, 208
 - print resource on, 218
 - vmstat** program (Linux), 38
 - volatile memory, 125
 - Vossen, Gottfried, 163, 164
 - VPNs (Virtual Private Networks), 359
 - vulnerabilities:
 - buffer overflow vulnerabilities, 412–414, 414, 433
 - of credentials, 257–259
 - to denial of service (DoS) attacks, 65–67, 157, 358, 412
 - in networking, 357
 - of operating systems, 15–16
 - of persistent storage, 357
 - of race conditions, 114–115
 - See also* attacks
- W**
- w permission:
 - ACL specification of, 250
 - application program vulnerability, 258
 - for directories, 250, 251
 - WAFL file system, 312
 - wait** method (Java API), 94, 95, 96
 - wait operation (condition variables), 94, 96
 - wait queues, 41–42, 42, 43
 - convoys in, 112–113
 - dump mutex, 113–114, 113
 - waiting for child processes, 228
 - waiting mechanisms (for threads):
 - busy waiting, 41
 - wait operation, 94, 96
 - waitpid** procedure, 228
 - See also* wait queues
 - waiting state (of threads), 42–43
 - changes in, 43
 - waitpid** procedure (POSIX API), 228
 - waking mechanisms (for threads):
 - notify operation, 94, 96–97
 - spurious wakeups, 96
 - timer interrupts, 41–42, 43
 - WAL (write-ahead logging), 149–150
 - Waldspurger, Carl A., 66–67, 267
 - WANs (wide area networks), 326
 - web browsers: system responsiveness needs during download time, 25–26
 - web servers:
 - connecting to with **telnet**, 333–334, 342–343
 - responsiveness during request time, 24
 - single- and multiple-threaded servers, 25
 - web services, 15, 331, 382–387
 - certificate use, 387–388
 - client authentication, 388–390
 - GoogleSearch security limitation, 388–389
 - interface specification standard (WSDL), 383–384
 - registry standard (UDDI), 387
 - support tools, 386–387
 - transmission format (SOAP), 384
 - transport mechanism (HTTP), 384–386
 - website resource on, 394
 - XML and, 383, 384
 - Web Services Activity web site, 394
 - Web Services Description Language (WSDL), 383–384
 - website resource on, 394
 - Web Services Interoperability Organization:
 - Basic Security Profile, 389, 390
 - web site, 394
 - Web Services Security Standard: message digital signature mechanism, 390
 - WebSphere MQ (IBM), 7, 370
 - wedding analogy (two-phase commit protocol), 154, 164
 - Weikum, Gerhard, 163, 164
 - Wi-Fi, 356
 - security issues, 359
 - wide area networks (WANs), 326
 - technology, 356

474 ► Index

- wildcard characters (+, #), 372–373
 - Wilkes, John, 322
 - Win32 API: `yield` procedure equivalent, 33–34
 - Windows. *See* Microsoft Windows
 - Windows Vista, 303
 - WinFS, 303
 - Wool, Avishai, 368
 - Wolf, Bobby, 394
 - workflow systems: for message-queuing systems, 133–135, 163–164
 - working directory: changing, 276
 - working sets (of processes), 180
 - in excess of free page frames, 208
 - print resource on, 218
 - worms, 412
 - antivirus scanning programs and, 414
 - buffer overflow attacks from, 412–414, 414
 - and DoS attacks, 412
 - email worms, 258, 412
 - signatures, 414
 - SQL Slammer worm, 358
 - vs. Trojan horses, 411
 - write actions (operations):
 - notation for, 137
 - transaction processing system approach to, 149
 - to undo logs and persistent storage, 148, 150–151
 - undoing, 147
 - write order disk space allocation strategy, 287–288
 - write permission. *See* `w` permission
 - `write` procedure (POSIX API), 279, 282–283
 - write requests, 272, 274
 - handling with pointers, 314
 - write-ahead logging (WAL), 149–150
 - writing files, 279–283
 - sequentially, 282–283
 - at specified positions, 281–282
 - WSDL (Web Services Description Language), 383–384
 - website resource on, 394
- X**
- `x` permission:
 - ACL specification of, 250
 - for directories, 251, 261
 - using with/without the `r` permission, 251
 - x86 architecture. *See* IA-32 architecture
 - `xclock` program, 227–228
 - Xeon hardware, 254
 - XFS file system (Linux), 135
 - disk space allocation approach, 290
 - extent maps, 297
 - XML (Extensible Markup Language): and web services, 383, 384
- Y**
- Yellin, Frank, 267
 - `yield` procedure, 32–33
 - in preemptive systems, 33–34
- Z**
- z/VM (IBM), 254, 267–268
 - evolution of, 268
 - print resources on, 268
 - zero filling (of memory), 178–179
 - concurrent threads and, 26
 - print resources on, 38, 218
 - zero page list, 205, 207
 - zero page thread, 206