

INDEX

Conventions Used in This Index

- *System calls and library functions are indexed by their names followed by empty parentheses, as in `abort()`.*
- *Significant functions developed in the book, such as those used by more than one program or those with their own complete listing, are indexed by their names followed by the listing or prototype label, depending on whether the implementation is given or just its prototype.*
- *Demo, or example, programs with their own complete or partial listings are indexed with the program label.*
- *Commands are indexed with the command label.*
- *Italicized page numbers refer to figures.*

Numbers and Symbols

& (backgrounding operator), 497
.
(current directory entry), 19
#! (interpreter script), 561
..
(parent directory entry), 19
/
(root directory), 22
1:1 (one-to-one) threading model, 711
32-bit compatibility mode, 198–199

A

ABI (Application Binary Interface), 499
abnormal termination, 540
`abort()`, 539, 540
absolute pathname, 24–25, 281
absolute sleep, 448
access mode, 155, 159
`acct()`, 69–70
active process, 493
`addch()`, 907
`addchstr()`, 907
address space layout randomization
(ASLR), 512
`addstr()`, 903, 904
`adjust_time()` listing, 124

advisory record lock, 552
AIO API, 808–809
`aio_cancel()`, 810
aio_cp.c program, 816–819
`aio_error()`, 810, 812–813
`aio_fsync()`, 810
AIO functions, 809–813
aio.h header file, 808–809
`aio_read()`, 808, 809, 810–811, 815
`aio_return()`, 810, 811
`aio_suspend()`, 810
`aio_write()`, 808, 809, 812
aio_write_demo.c program, 813
`alarm()`
 alarm_demo1.c, 451–452
 alarm_demo2.c, 452–453
 applications, 451–453
 canceling, 451–453, 459
 as an interval timer, 459
 one-shot limitation, 453
 overview, 450–451
 progress bar based on, 454–461
 return value, 451
 and `SIGALRM`, 451
 signal handler for, 452–453
 weaknesses, 461
alarm clocks vs. timers, 436–437
alarm_demo1.c program, 451–452
alarm_demo2.c program, 452–453
alternate screen, 889–890
alternation operator, 37
alternative methods of I/O
 multiplexed I/O
 example program, 824–830
 overview, 819–820
 `select()`, 820–824
nonblocking I/O, 792–796
POSIX AIO
 AIO API, 808–809
 AIO functions, 809–813
 asynchronous version of *spL*
 cp1.c, 816–819
 diagram, 815
 with disk files, 813–815
 overview, 807–808

- ul style="list-style-type: none;">
- alternative methods of I/O (*continued*)
 - signal-driven I/O
 - diagram, 799
 - overview, 796–798
 - procedure for enabling, 798–799
 - and real-time signals, 802
 - sigio_counter.c* program, 800–801
 - sigio_demo.c* program, 802–807
 - signal generation in, 799–802
- American Standard Code for Information Interchange (ASCII), 955–956
- ampersand (&)
 - backgrounding operator, 497
- ancestor of a process, 493
- ancestors.c* program, 524–526
- ANSI escape sequences, 476–477, 886
- a.out* files, 499
- API-based reading, 213
- append indicator, 171
- Application Binary Interface (ABI), 499
- application programming interface (API), 6, 8
- application programming interfaces by category. *See also* terminal
 - AIO, 808–812
 - Curses, 901–909
 - directory, 314–328
 - filesystem, 263–264
 - kernel, 6–10
 - password database, 200–204
 - POSIX IPC, 602–607, 616–617, 628–630
 - Pthreads* IPC overview, 711, 716–717. *See also Pthreads* API
 - read-write lock, 775–776
 - System V semaphore, 616
 - utmp, 197–199
 - utmpx, 217, 220, 225
- application-provided documentation, 32–33
- apropos* command, 39–40
- ar* command, 54, 104
 - operation code, 54
- arguments of a command
 - defined, 11–12
 - diagram, 74
 - extracting from command line, 73–75
- argv[]* array, 474
- arming timers, 464–465
- Arnold, Ken, 895
- ASCII (American Standard Code for Information Interchange), 955–956
- ASCII codeset, 133
- asctime()*, 112
- ASLR (address space layout randomization), 512
- assembler output file, 499
- asynchronous cancelability of threads, 725
- asynchronous I/O, 392, 798. *See also* POSIX asynchronous I/O
- asynchronous I/O control block, 808–809
- asynchronous notification
 - and message queues, 634–636
 - program receiving, 636–642
- asynchronous signal, 384
- asynchronous waiting for signals, 582–590
- async-signal-safety, 397, 431–432
- atexit()*, 557–558
- atexit_demo.c* program, 559–560
- atof()*, 86
- atoi()*, 86
- atomic variable, 412–414
- AT&T Bell Labs, 41–42
- attributes
 - condition variable, 757
 - file, 18–19. *See also* file metadata
 - filesystem, 256, 263–266
 - mutex, 741, 749
 - process, 517–518
 - read-write lock, 776
 - terminal, 844–847, 852–856
 - thread, 712–715
- ATTRIBUTES section, man pages, 36
- attr_t* Curses data type, 900
- authentication, 13–14
- AUTHORS section, man pages, 36
- auto-append mode, 552
- ## B
- backgrounding operator (&), 497
 - background process, 497–498
 - background process group, 497–498
 - backlink in the directory hierarchy, 373
 - backspace character, 843
 - bad login attempts, 213
 - barrier synchronization
 - diagram, 764
 - overview, 762–764
 - program using, 765–773
 - Pthreads* barrier, 764–765
 - barrier synchronization point, 764

- base functions, Curses, 904–905
- basename(), 80–81
- basename_demo.c* program, 81
- bash shell, 13, 33–34
- bc command, 688–689
- Berkeley Fast File System (FFS), 252
- Berkeley Software Distribution (BSD), 42
- binary file, 18
- binary semaphore, 615
- binary utility, 55
- /bin* directory, 23
- birth time, 277, 279
- bitmask, 159, 849
- bits/utmp.h* files, 197–198
- block group, 254, 255–257
- blocking read, 652, 791, 880–881
- blocking signals
 - overview, 405–407
 - pselect(), 921–922
 - signal set, 407–408
 - sigprocmask(), 408–416
 - sigwait(), 415
- block method for passing parameters, 59
- bool Curses data type, 900
- boot block, 254
- /boot* directory, 23
- boot time, 531
- BOOT_TIME utmp entry, 231–232
- Bourne, Stephen, 13
- Bourne Again SHell, 13
- Bourne shell, 13
- brk(), 514
- broken-down time structure, 108–109
- BSD (Berkeley Software Distribution), 42
- _BSD_SOURCE macro, 70
- buffer cache, 182–185
- buffering
 - buffer size, 174–176
 - in canonical mode, 878–879
 - and child processes, 549–550
 - double, 814, 816
 - in file copying program (*spl_cpl.c*), 182–185
 - file stream, 549–550
 - kernel, 171, 797
 - line, 459
 - and running time, 182–185
 - user space buffering of input, 242–244
- BUGS section, man pages, 36
- builtins, shell, 13

C

- calendar time, 108, 109–111, 439
- canceling threads, 724–725
- cancellation points, 725
- canonical input queue, 841
- canonical mode, 837, 878–884
- Card, Rémy, 253
- Cartwright, Alexander, 26
- cat command, 186, 312, 656–660
- categorical mutual exclusion, 774
- categories, locales, 129–131, 141
- cbreak(), 905, 906
- cbreak mode, 879
- c_cc array of termios, 851, 853, 868
- c_cflag of termios, 851, 853, 868
- %c date/time format specifier, 107, 113–114
- cd command, 20–21
- cfgetispeed(), 865
- cfgetospeed(), 865
- cfmakeraw(), 879
- character echoing, 837
- character encoding, 955–960
 - form, 957
 - scheme, 957
- character maps, 133
- character representations, 956
- character sets, 133
- chdir(), 375
- child node, 22
- child process. *See also* process
 - creating
 - basics of fork(), 541–543
 - child's memory image, 543–545
 - child's process descriptor, 545
 - code executed by child after fork, 541
 - exit status and termination, 558
 - orphan process, 559
 - overview, 493, 541–542
 - reaping, 570
 - SIGCHLD signal, 393
 - zombie status, 570
- chtype Curses data type, 900
- c_iflag flagset of termios, 849, 850
- C I/O Library, line buffering in, 459
- c_ispeed field of termios, 849
- cleanup() signal handlers, 892, 913
- clear() (Curses), 905
- C library, role in I/O, 4–5
- client-server applications
 - concurrent client-server
 - diagram, 698
 - global data, 698

- client-server applications (*continued*)
 - concurrent client-server (*continued*)
 - overview, 695–696
 - upcase.c* client program, 697–701
 - upcased.c* server program, 702–706
 - daemons, 683–686
 - introduction to, 680–681
 - iterative client-server
 - defined, 686
 - design considerations, 686–687
 - design of client, 690–691
 - design of server, 692–693
 - diagram, 687
 - overview of example
 - application, 686–689
 - spl_calc.c* client program, 690–692
 - spl_calc.h* header file, 689–690
 - spl_calc_server.c* server program, 692–695
 - use of *bc* calculator in, 688–689
 - multithreaded concurrent server
 - design of, 731–732
 - threaded_upcased.c* server program, 733–735
 - overview, 667
 - system logging facilities, 681–683
- client-server-like application
 - client-server interaction, 675–676
 - ifomonitor.c* server program, 672–674
 - ifosender.c* client program, 674–675
 - overview, 667
 - use of FIFO in, 671–672
- c_line* field of *termios*, 849, 866, 869
- clobbering, 172
- clock attribute of a condition variable, 757
- clock_gettime()*, 109–110, 111, 426, 445
- CLOCK_MONOTONIC clocks, 463–464
- clock_nanosleep()*, 447–450, 468
- clock_nanosleep_demo.c* program, 449–450
- clocks, 438–440, 463
- clock tick
 - definition, 439–440
 - use in *stat* file in */proc* subdirectories, 527–530
- clone()*, 557
- close()*, 163–164
- closedir()*, 320
- close-on-exec flag, 664
- closing files
 - errors when, 164
 - kernel I/O interface, 163–165
- clusters in disk geometry, 248
- cmdline* file in */proc* subdirectories, 522, 523
- cmp* command, 176–177
- code point, 133, 957, 959
- codeset, 133–134
- codespace, 957
- c_oflag* flagset of *termios*, 849–850, 863–864
- COLS (Curses), 900
- combination settings of terminal, 844
- command code of *ioctl()*, 871
- command line
 - arguments, extracting, 73–75
 - and locales, 132–135
 - options, extracting, 81–86
 - tutorial, xxxviii
- command line interpreter, 11, 12
- comment (GCOS) field of password file entry, 202
- comm* file in */proc* subdirectories, 522, 523
- common code
 - common_hdrs.h* header file, 95–97
 - diagram, 103
 - error-handling functions, 102–103
 - file organization, 103–104
 - functions for extracting numbers, 97–102
 - header guards, 96–97
 - organization, 95
 - sys_hdrs.h* header file, 95
- common* directory, 103–104
- common_hdrs.h* header file, 95–97
- comparison function, 331
- compiler collection, 4
- concurrent server. *See* client-server applications
- condition variable
 - condition attributes, 757
 - declaring and initializing, 754
 - destroying, 756–757
 - need for, 752–753
 - overview, 716, 751–752
 - signaling, 755–756
 - typical steps for using, 753
 - use in multithreaded producer-consumer program, 757–762
 - waiting on, 754–755
- configuration functions in Curses, 905–906
- CONFORMING TO section, man pages, 36
- connection object in I/O, 155

- console, 475
- constants in Curses, 900
- contents of files, 18–19
- Control Sequence Introducer (CSI), 476
- control settings of terminal, 844
- cooked mode, 879
- coordinate system in Curses, 897–898
- copying files
 - with asynchronous I/O, 813, 816–819
 - general discussion, 171–173
 - overview, 157
 - spl_cpl.c* program
 - buffering, 182–185
 - design of, 173–174
 - implementation of, 174–176
 - overview, 261
 - testing, 176–177
 - universality of, 177–179
- copy-on-write, 543
- core dump, 500
- core file, 500
- c_ospeed field of termios, 849
- counting semaphore, 615
- C++ language, xxxii
- cpp preprocessor, 197
- C programming language, xxxii, 41, 45–46
- CPU data displayed in *top* command, 932–933
- CPU local locks, 779
- creat()*, 158–159
- created files, attributes of, 161–162
- create_windows()* function of *spl_top.c*, 920
- critical region, 406
- critical section. *See also* race conditions
 - defined, 406
 - diagram, 741
 - keeping them small, 748
 - protecting with condition variables, 753, 757–762
 - protecting with mutexes, 736, 740–741
 - protecting with semaphores, 615
- CSI (Control Sequence Introducer), 476
- C Standard Library, 45–46, 57–58
- ctime()*, 112, 118
- cultural conventions, 71–72
- current directory (*.*), 19
- current working directory, 20, 25
- curses_demo1.c* program, 902–904
- curses.h* header file, 895, 897

- Curses library and API
 - basics, 897–901
 - categories of functions, 901–902
 - compiling and building curses programs, 897
 - configuration functions, 905–906
 - coordinate system, 897–898
 - data types in, 900
 - diagram, 898
 - example programs
 - curses_demo1.c* program, 902–903
 - getstr_demo.c* program, 906
 - spl_top.c* program, 915–939
 - sprite_curses.c* program, 912–914
 - tiled_windows.c* program, 909–911
 - function naming convention, 904–905
 - history, standards, and names, 895–896
 - input functions, 906–907
 - internationalization in, 900
 - miscellaneous useful functions, 908–909
 - output functions, 907
 - overview, 894
 - predefined constants and variables in, 900
 - screen updating
 - description of, 898–899
 - diagram, 898
 - terminology, 896–897
 - tiled windows in *spl_top.c*
 - diagram, 919
 - management of, 919–920
 - window functions, 908
- Curtis, Pavel, 895
- cwd* file of */proc* subdirectories, 523
- cylinder
 - defined, 248
 - diagram, 249
- cylinder group, 254

D

- daemon, 496, 683–686, 702
- data block area, 257
- data block bitmap, 255
- data structures in C
 - aio_cb*, 809
 - dirent*, 316–318, 326
 - FTSENT, 366–368
 - FTW, 349

- data structures in C (*continued*)
 - group, 294
 - itimerspec, 465
 - lastlog, 197–199
 - list_head, 520
 - mutexattr_t, 749
 - passwd, 202–203
 - procstat, 527, 924–925
 - pthread_attr_t, 717–718
 - SCREEN, 896
 - sigaction, 417–419, 584–585
 - sigevent, 463, 467, 635
 - siginfo_t, 585
 - stat, 267–269
 - statx, 278–281
 - struct tm, 109
 - struct tty_struct, 838
 - task_struct, 518–519, 713–714
 - termios, 848, 852–853, 880–884
 - timespec, 441–442
 - timeval, 111, 218–220
 - utmp, 216, 217–219
 - utmpx, 220
 - WINDOW, 896
- data transfer methods in IPC, 599–600
- data types in Curses, 900
- date command, 104–106, 107, 118
- date/time formats and representations
 - broken-down time, 108–109
 - calendar time in Unix, 108
 - calendar time system calls, 109–111
 - format specifiers, 961–963
 - internationalization, 71–72
 - locales
 - categories, 129–131
 - command-level interface to, 132–135
 - fuzzy time, 118–119
 - internationalized version of *spl_date2.c*, 139–140
 - locale objects, 144–146
 - other ways to internationalize programs, 140–144
 - overview, 107
 - programming interface to, 138–139
 - structure of, 135–138
 - time zones, 131
 - planning, 104–107
 - spl_date1.c* program, 114
 - development, 107–113
 - diagram of control flow, 113
 - listing, 114
 - spl_date2.c* program, 114–116
 - spl_date3.c* program
 - diagram of control flow, 120
 - program logic, 119–128
 - time adjustment specifications, 117–118
 - user interface, 116–119
 - time conversion functions, 112–114
- DEAD_PROCESS, 231
- deferred cancelability of threads, 725
- delayed binding, 262
- delch() (Curses), 907
- deleteIn() (Curses), 907
- delivery of signals, 388
- delwin() (Curses), 908
- demo directories with common code, 103–104
- demo program naming convention, xi
- demo programs, xviii
- deprecated feature, 111
- dereferenced symbolic link, 24
- descendants of a process, 493
- DESCRIPTION section, man pages, 36–37
- design considerations
 - extensible design, 321
 - signal handlers, 431–432
 - spl_lastlog.c* program, 206–208
 - spl_top.c* program, 917–918
 - system programs, 121
 - with threads, 715
- destroying objects in *Pthreads*
 - condition variables, 756–757
 - mutexes, 743
 - read-write locks, 775
- destructive read, 647
- detached thread, 719, 722–724
- /dev directory, 23
- device driver, 250
- device files, 18, 23, 178
- dev_t, 267, 269, 294
- df command, 339
- diff command, 176, 211–212
- Dijkstra, Edsger, 614
- directed acyclic graph, 22
- directed edge, 22
- directory
 - contents, 19
 - defined, 19–21
 - diagram of contents, 19
 - diagram of user perception of, 19
- directory API
 - closedir(), 320
 - dirent structure, 315

- opendir(), 319
 - overview, 324–325
 - processing directories, 313–320
 - readdir(), 315
 - scandir(), 329–335
 - seekdir(), 325–328
 - telldir(), 325–328
- directory entries, 19
- directory hierarchy
 - diagram, 535
 - directory structure, 19–20, 312–313
 - implementing a simple ls program, 321–324
 - mounting filesystems, 22–23, 337–341
 - overview, 22–24
 - pwd implementation
 - constructing directory tree, 371–373
 - overview, 371
 - spl_pwd.c* program, 379–380
 - strategy for implementing, 374–381
 - spl_ls1.c* program, 321–322
 - tree walks
 - du command, writing, 354–365
 - fts tree traversal functions, 365–370
 - nftw(), 347–354
 - overview, 341
 - recursive, using readdir(), 341–345
 - recursive, using scandir(), 345–347
- directory-relative pathname, 282
- directory streams, 316, 318–319
- directory tree
 - constructing from inode numbers and links, 371–373
 - overview, 335–337
- dirent.h* header file, 317, 319
- dirent structure, 316–318, 326
- dirfd(), 324
- DIR object, 318
- dirpath argument to nftw(), 348
- dirstfirstsort() listing, 332
- disk allocation, 192–193
- disk device driver, 250
 - diagram, 250
- disk file, 813–815
- disk geometry
 - components, 248–250
 - diagram, 248

- disk head, 249
- disk partitioning
 - benefits of, 251–252
 - diagram, 251
 - layout of a partition
 - diagram of block groups in, 254
 - organization, 254
- disk usage, 336
- displayvm.c* program, 515–517
- d_name field of dirent struct, 316
- double buffering, 814, 816
- double-indirect block, 257
- drand48(), 456
- d_type field of dirent struct, 317–318
- du command
 - diagram of path through directory tree, 347
 - implementation of, 354–365
 - for obtaining file size, 192
 - as a tree walk, 336
- dumping, 222
- dup(), 658–660
- dup2(), 658–660
- duplicate inode numbers, 340–341
- duplicating file descriptors, 658–659, 664
- duplocale(), 144
- dynamically linked (shared) libraries, 52, 944–945. *See also* libraries

E

- EBCDIC (Extended Binary Coded Decimal Interchange), 955–956
- echo attribute of terminal, program for changing, 853–854
- echo command, xxxix, 35
- echoing by terminal, 837
- edata, 511–514
- edge-triggered notification, 797–798
- edition of UNIX research system, 41
- effective user ID, 153–155, 161–162
- EINVAL error code, 100
- elapsed time, 108
- ELF. *See* Executable and Linking Format
- ELF header, 500, 503–506
- emulator trap, 391
- encoding, character, 955–960
- end, 511–514
- endpwent(), 204
- endwin() (Curses), 901, 903, 905, 913, 921
- entry point, 8, 498
- environ_demo.c* program, 76
- environ* file in */proc* subdirectories, 523

- environment, 15–17
- environment list, 15
- environment string, 15–16
- environment variables, 15–16, 75–78, 130, 134, 135
- ENVIRONMENT VARIABLES section, man pages, 36
- environ variable, 76–77
 - diagram, 77
- Epoch, 108
- epoll(), 820
- ERANGE error code, 100
- erase character, 843
- errno -1 command, 62–63
- errno variable, 62, 64
- error_exits.c* program, 103
- error_exits.h* header file, 102
- error handling
 - common functions, 102–103
 - in concurrent server, 701, 704–705
 - errors from library functions, 66
 - overview, 62
 - reporting usage errors, 78–79
 - system call errors, 62–66
 - when closing files, 164–165
 - when opening files, 162–163
- error_message() prototype and listing, 102–103
- ERRORS section, man pages, 36, 62, 63
- escapeseq_demo1.c* program, 477
- escapeseq_demo2.c* program, 477
- escape sequences, 476–477, 886
- /etc* directory, 23
- /etc/locale.gen* file, 134–135
- etext, 511–514
- event notifications, 463
- EXAMPLES section, man pages, 36–37, 274
- exceptfds parameter of select(), 821
- exceptional event, 820
- exec() family of functions
 - execl(), 565–567
 - execle(), 565–569
 - execlp(), 565–567
 - execv(), 565–567
 - execvp(), 565–567
 - execvpe(), 565–567
- execl_demo.c* program, 567
- execle_demo.c* program, 568–569
- execlp_demo.c* program, 567–568
- Executable and Linking Format (ELF)
 - diagram of linking and executable views, 500
 - diagram of overlapped views, 502
 - diagram of overlapping segments, 506
- executable file format, 499
- file contents, 502–506
- file structure, 500–502
- overview, 53, 499–500, 943
- program to print program header table, 506–511
- readelf command, 55–56
- executable code (executables), 25
- executable program file. *See* Executable and Linking Format
- executing programs. *See* exec() family of functions
- execution view, 500–501
- execv(), 565–566
- execve(), 540, 560, 561–565
- execve_demo1.c* program, 562–563
- execve_demo2.c* program, 563–564
- execvp(), 565–566, 569, 591
- execvp_demo.c* program, 569
- execvpe(), 565–566
- exe* file in */proc* subdirectories, 523
- exit(), 540, 557–558, 718
- _exit(), 540, 558
- exit functions, 557–560
- exiting threads, 718
- exit status of a process, 558
- EXIT STATUS section, man pages, 36
- explicit seeking and reading, 212–213
- Extended Binary Coded Decimal Interchange (EBCDIC), 955–956
- extended filesystem (Ext), 253
- extensible design, 321
- external storage, 17
- extracting
 - accessing environment variables, 75–78
 - command line arguments, 73–75
 - command line options, 81–86
 - memory statistics from *stat* file of */proc* subdirectories, 770
 - numbers, functions for, 97–102
 - numbers from strings, 86–89
 - program name, 79–81
 - reporting usage errors, 78–79

F

- fakeinput_demo.c* program, 872–873
- fakelogin.c* program, 854–856
- fake signal functions, 396

- faking input, 872
- FAT32 filesystem, 261
- `fatal_error()` prototype and listing, 102–103
- FAT filesystem, 261
- `fchdir()`, 375
- `fcntl()`, 552, 793, 798
- feature test macros, 28–29, 37, 67–70
- `fflush()`, 550
- FFS (Berkeley Fast File System), 252
- `field_t`, 926
- FIFO (unnamed pipe)
 - in concurrent server, 695–706
 - creating, 669–670
 - creating named pipes in shell, 667–669
- FIFO-based server-like program
 - client-server interaction, 675–676
 - FIFO client program, 674–675
 - monitor program, 672–674
 - overview, 671–672
- in iterative server, 686–687, 689–695
- opening, 670–671
- overview, 646–647, 667
- private vs. public, 669
- fifomonitor.c* server program, 672–674
- fifosender.c* client program, 674–675
- file
 - appending to, 551, 552, 578
 - attributes of created files, 161–162
 - closing, 163–165
 - creating, 259–260
 - creation flags, 159, 160–161
 - creation masks, 150–153
 - creation time, 279
 - descriptor. *See* file descriptor
 - errors on
 - closing, 164–165
 - opening, 162
 - reading, 167
 - writing, 173, 175
 - holes
 - cause of, 190–193
 - diagrams, 191, 192
 - and `ls -l`, 192
 - makefilehole.c*, 191
 - read operation in, 190
 - I/O. *See* file I/O
 - metadata. *See* file metadata
 - name. *See* filename
 - offset, 155, 188–193, 547–548
 - opening, 159–161
 - permissions. *See* file permissions
 - reading from, 199
 - diagram of effect on file and memory, 169
 - diagram of kernel structures, 167
 - overview, 165–169
 - reading structures from, 199
 - seeking in, 188–190, 249
 - size
 - actual, 192
 - apparent, 192
 - sparse, 193, 206
 - status flags, 159–160, 551, 578
 - types, 18
 - Unix, 17–18
 - writing, 169–171, 260–261
- file attributes. *See* file metadata
- file creation flags, 159, 160–161
- file creation masks, 150–153
- file creation time, 279
- file descriptor
 - `close()`, 163–164
 - close-on-exec flag, 664
 - duplicating
 - diagram of, with pipe, 658
 - diagram of redirection with, 659
 - with `dup()`, 658–660
 - with `dup2()`, 660–662
 - to set up a pipe, 658–660
 - enabling signal-driven I/O, 799–800
 - lowest-numbered unused, 659
 - modifying set of flags on, 552
 - open file description (OFD), 155–156
- file hole, 190–193
- file input, 149
- file I/O. *See also* login accounting
 - controlling position of I/O operations, 188–193
- file creation mask, 150–153
- high-level vs. low-level, 150
- I/O API
 - closing files, 163–165
 - opening files, 158–163
 - reading from files, 165–169
 - seeking, 188–193
 - writing to files, 169–171
- I/O mechanics, 155–157
 - diagram, 156
- setuid bit, 154–155

- file I/O (*continued*)
 - spl_cpl1.c* program
 - design of, 173–174
 - implementation of, 174–176
 - testing, 176–177
 - universality of, 177–179
 - standard file descriptors, 157
 - timing programs, 179–185
 - universal I/O, 150
- file metadata
 - displaying with `stat` command, 264
 - in inode, 256
 - obtaining with `stat()` and `lstat()`, 266–277
 - program for displaying, 285–297
 - `stat` structure, 267–273
 - superblock, 255
 - types of, 18
- file mode
 - diagram, 270
 - flags and macros for, 270–271
 - overview, 18
- file mode bits, 270
- filename
 - checking existence of, 259
 - diagram, 22
 - in `dirent` structure, 316–317
 - `get_filename()` listing, 378–379
 - multiple names for a file, 22
 - overview, 19, 21–22
 - recording in directory, 260
- `fileno()`, 550
- file offset, 155, 188–193, 547–548
- file permissions
 - created files, 161–162
 - file creation mask, 150–153
 - and `setuid` bit, 154–155
- file pointer, 155, 188–193, 547–548
- FILES section, man pages, 36
- file status, 18. *See also* file metadata
 - file status flags, 159, 160
- file stream, 549
- file structure of ELF files, 500–502
- file structure table, 156
- filesystem
 - disks and disk partitions, 248–252
 - Ext filesystems, 253
 - filesystem API, 263–264
 - kernel interface, 259–261
 - mounting, 337–341
 - overview, 22
 - `setgid` bit, 272–273
 - spl_stat.c* program
 - auxiliary functions, 292–297
 - design of an enhanced version, 297–298
 - design of `main()`, 285–287
 - design of `print_statx()`, 287–292
 - overview, 285
 - spl_statfs.c* program,
 - hybrid solution, 301–307
 - overview, 299
 - `statfs()`, 299–300
 - `statvfs()`, 300–301
 - testing, 307–308
 - `stat()`, 266–267
 - `stat` command, 264–266
 - `statx()`
 - calling, 281–284
 - overview, 278
 - `statx` structure, 278–281
 - sticky bit, 273–274
 - structure of Ext filesystems, 253–259
 - supported by Linux, 252–253
 - virtual
 - diagram of VFS, 263
 - overview, 261–263
- file types, 18
 - file mode bits, 270
 - in `dirent` structure, 317–318
 - Unix, 18
- file utility command, 499
- fillqueue.c* program demonstrating
 - `ioctl()`, 874
- filter commands, 200
- `find` command, 336
- `findmnt` command, 340
- flags bit mask, 159
- flagsets, 849–851
- flavors of Unix, 7
- flushing, 550
- `fn()` parameter of `nftw()`, 348–349
- followed link, 24
- `fopen()`, 664
- foreground processes, 497–498
- foreground process groups, 497–498
- `fork()`, 541–543. *See also* process
- fork_demo1.c* program, 542–543
- fork_demo2.c* program, 543–545
- fork_demo3.c* program, 547–548
- fork_demo4.c* program, 548–549, 550–551
- `format_time_diff()` listing, 236
- `fpath` parameter of `nftw()`, 348

- fprintf(), 79
- free(), 127
- Free Software Foundation (FSF), 42
- fsblkcnt_t, 299–300
- fsfilcnt_t, 299
- fsid_t, 304
- fstat(), 267
- fstatat(), 267
- _fsword_t, 299–300
- fsync(), 165
- ftruncate(), 603–604, 606
- fts_children(), 366
- fts_close(), 366
- fts_demo.c* program, 368–370
- FTSENT structure, 366–368
- fts_open(), 366, 368
- fts_read(), 366, 368
- fts_set(), 366
- fts tree traversal functions, 365–370
- ftw(), 348
- FTW_ACTIONRETVAL flag, 351
- ftwbuf parameter, 349
- FTW_CHDIR flag, 350
- FTW_DEPTH flag, 350, 354
- FTW_MOUNT flag, 350, 364–365
- FTW_PHYS flag, 351
- fully buffered streams, 549
- function pointer parameters, 329–330, 333
- fuzzy time, 118–119

G

- Game of Life, 762–763
- gcc command, 203
- gdb utility, xxxiii
- gecos field of passwd struct, 202
- generation of signals, 387
- get_all_vmsizes() listing, 770
- get_boot_time() listing, 531
- getch(), 903
- getconf command, 839
- getcwd(), 380
- getdents(), 314–315
- get_dev_ino() listing, 376
- getenv(), 16–17, 75–76
- getenv_demo.c* program, 16–17, 75
- get_filename() listing, 378–379
- getgrgid(), 293–294
- get_hertz() listing, 530–531
- gethostname(), 63
- gethostname_demo.c* program, 63–64
- get_int() listing, 101

- get_int() prototype, 98
- getitimer(), 462
- getline(), 68–69, 524, 527–528
- get_loadavg() listing, 930–931
- get_long() listing, 101
- get_long() prototype, 98–100
- get_nums.h* header file, 97–98
- getopt(), 82–85
- getopt_demo.c* program, 84–85
- getopt_long(), 82–83
- getparentid(), 524–525
- getpggrp(), 495
- getpid(), 27
- getpid_demo.c* program, 27
- getpwnent(), 204, 206–207
- getpwnam(), 200–201
- getpwuid(), 201–203
- getpwuid_demo.c* program, 203
- getsid(), 496
- get_speed() listing, 865
- getstr(), 906
- getstr_demo.c* program, 906–907
- gettid(), 28, 60–61
- gettid_demo.c* program, 28, 60–61
- gettimeofday(), 111
- getting process ID of parent of a process, 524
- getutent(), 219
- getutxent(), 219
- get_vmsize() listing, 771
- getwin() (Curses), 908
- gid2name() listing, 294
- gid_t, 201, 267, 290, 294
- git command, xxxiii, 37
- glibc, 57, 139–140
- global constants, *ncurses*, 900
- globally unique identifier partition table (GPT), 251
- glyph, 476, 956
- gmtime(), 112
- GNU C Library, 57, 139–140
- GNU C Library Reference Manual, 34
- gnu_get_libc_version(), 57
- GNU/Linux, xxxiv
- GNU Manifesto, The*, 42
- GNU Project, xxxiv, 13, 42
- _GNU_SOURCE macro, 70, 81, 83
- GPT (globally unique identifier partition table), 251
- grep command, 200, 656–660
- group, Unix, 13–15
- group descriptor, 255

- group ownership of newly created file, 161–162
- groups command, 14
- Grudin, Robert, 108

H

- handle (fts), 365
- handle_sigchld() reaping handler, 586–589
- handling signals, 393–399
- hard disk, 248–249
- hard link, 24
- hardware clocks, 439
- hardware timers, 439
- header, man page, 30
- header guard, 96–97
- heap, 511, 513–514
- hello_world.c* listing, 2
- “Hello, world” program, 2–3
- hexdump command, 56
- hidden canvas, 898
- high-level vs. low-level file I/O, 150
- High Precision Event Timers (HPETs), 439
- high-resolution sleep functions.
See sleep functions
- home* directories, 23
- homing the cursor, 803
- Horton, Mary Ann, 895
- HZ macro, 530

I

- icrnl switch of stty, 845–846
- id command, 14
- id_t, 570
- idtype_t, 570
- illegal instruction, 391
- #include directive, 2
- include* directory in repository, 95, 103
- include guard, 96–97
- incomplete type, 319
- indirect blocks, 256–257
- info command, 32
- Info documentation system, 31–32
- initialized data segment, 513
- initial value of timer interval, 462
- init process, 221
- initscr() (Curses), 900–901, 903, 905, 906
- inode
 - creating, 260
 - duplicate numbers, 340–341
 - file block pointers in
 - description, 256–257
 - diagram, 257

- inode bitmap, 255–256
- inode() man page, 268–269
- inode table, 256–257
- overview, 18
- performance of file access, 257–258
- updating, 260
- ino_t, 267, 315, 362, 373
- input
 - diagram, 5
 - overview, 2–4
- input buffering in canonical mode, 878–879
- input functions of Curses API, 906–907
- input settings of terminal, 844
- insch() (Curses), 907
- insstr() (Curses), 907
- integer types, system, 280–281
- interactive programming. *See also* terminal
 - canonical mode of terminal, 878–879
 - Curses and the *ncurses* library. *See* Curses library and API;
spl_stty.c program; *spl_top.c* program; *sprite.c* program
 - definition of, 834
 - fakeinput_demo.c* program, 872
 - fakelogin.c* program, 855–856
 - fillqueue.c* program, 874
 - max_input.c* program, 839
 - mintime_test_demo.c* program, 883–884
 - and nonblocking I/O, 793–794
 - noncanonical modes of terminal, 879–884
 - cbreak mode, 879
 - cooked mode, 879
 - effects of MIN and TIME parameters, 880–894
 - polling read in, 880
 - raw mode, 879
 - sigio_demo.c* program, 802–807
 - toggle_echo.c* program, 854
 - watchtty.c* program, 873
 - interbyte timer, read with, 881–884
- internal fragmentation, 258
- internationalization and locales
 - changing the locale temporarily, 135, 144–145
 - command-level interface to, 132–135

- `duplocale()`, 144
- `freelocale()`, 145
- functions using locale information, 140–144
- internationalized version of *spl_date.c*, 139–140
- locale-aware library functions, 141
- locale categories, 129–131
- locale object, 144–146
- locales in *ncurses*, 900
- man pages, 128–129
- `newlocale()`, 144–145
- newlocale_demo.c* program, 145–146
- `nl_langinfo()`, 142–144
- nl_langinfo_demo1.c* program, 143–144
- overview, 71–72
- programming interface to, 138–139
- running programs with different locales, 135
- `setlocale()`, 138
- setting the locale, 138–139
- strcoll_demo.c* program, 141–142
- structure of a locale, 135–138
- time zones, 131
- TZ, 131
- `uselocale()`, 144–145
- interpreter script (#!), 561
- interprocess communication (IPC)
 - data transfer methods
 - compared to shared memory, 599
 - defined, 599
 - facilities based on, 601
 - diagram comparing paradigms, 599
 - IPC facility, defined, 597
 - message queue overview, 599
 - need for, 598
 - overview, 598–602
 - POSIX message queue
 - API summary, 629
 - asynchronous notification, 634–636
 - attributes, 630
 - closing, 630–631
 - comparison with System V message queues, 600–601
 - mqrvc_demo.c* program, 632
 - mqsend_demo.c* program, 633–634
 - opening and creating message queues, 630
 - overview, 628–631
 - program receiving
 - asynchronous notifications, 636–642
 - receiving messages from, 631
 - sending messages through, 631
 - sequence of steps for using, 629–630
 - setting up signal-based notification, 635–636
 - simple example of, 631–634
 - ulogger_client.c* program, 637–638
 - ulogger.c* program, 638–642
 - ulogger.h* header, 637
 - unlinking, 630–631
- POSIX semaphores
 - atomicity of operations, 614–615
 - binary vs. counting, 615
 - creating and initializing, 617–618
 - named semaphore API
 - overview, 617
 - namedsem_demo.c* program, 621–622
 - named vs. unnamed, 616
 - overview, 614–615
 - `post()` operation, 614–615
 - producer consumer programs
 - using, 623–628
 - properties, 615
 - sequence of steps for using
 - named, 617
 - steps for using unnamed, 617–618
 - typical use, 615
 - unnamed semaphore API
 - overview, 617
 - unnamedsem_demo.c* program, 618–619
 - `wait()` operation, 614–615
- POSIX shared memory
 - API summary, 603–605
 - changing size of, 606
 - diagram, 610
 - mapping into process, 606–607
 - mode of memory mapping, 607
 - offsets and shared memory, 607–608, 613
 - opening and creating, 605–606
 - overview, 603–605
 - pointer pitfalls in shared memory, 610–613
 - race conditions, 613–614

IPC (continued)

POSIX shared memory (continued)

- sequence of steps for using, 603–605
- shared memory API, 605–607
- shared memory example program, 607–609
 - shm_consumer.c* program, 626
 - shm_producer.c* program, 627
- shared memory overview, 598
- shared memory producer consumer program, 623–628
- shared memory vs. data transfer facilities, 598–599
- signals as, 598
- System V IPC overview, 600–601
- System V semaphore API, 616
- interrupt signal, 384, 385–386
- interval timer
 - history, 461–462
 - POSIX timer
 - arming and disarming, 464–465
 - counting timer overruns, 465–466
 - creating and deleting, 462–464
 - initializing `itimerspec` structure, 465
 - linking programs with `-lrt`, 462
 - overview, 462
 - progress_bar2.c* program, 466–471
 - progress bar based on, 466–471
 - setting the notification method, 463
 - using the `sigevent` structure, 463–464
 - principles, 462
 - programmable interval timer (PIT), 439
 - real-time signals and multiple timers
 - overview, 484–487
 - posix_timer_demo1.c*, 485–487
 - resource monitor using POSIX timers
 - managing and refreshing screen, 475–478
 - overview, 471–474
 - running program, 483–484
 - sequence of steps in main program, 473, 481
 - setting up file list, 474–475
 - synopsis, 473

- terminal escape sequences used in, 477
- updating file sizes, 478–479
- watchfiles.c* program, 471–486

`intrflush()`, 905, 906

`ioctl()`, 856, 870–875

I/O event notification facility, 820

io file in */proc* subdirectories, 523

I/O multiplexing

- comparison of `select()`, `pselect()`, and `poll()`, 820

diagram, 825

example program, 824–830

`ndfs` value in `select()`, 822–823

overview, 819–820

`select()`, 820–824

select_demo.c program, 826–829

select law (guidance), 823–824

`iostat`, 436

`iowait()` listing, 922–924

IPC. *See* interprocess communication

`isatty()`, 550

`isdir()`, 326

iterative server. *See* client-server applications

`itimerspec` structure, 465

J

jiffy, 439, 440

job control signals, 392–393

joinability of threads, 718, 719

joining threads, 718–720

journal file, 253

journaling, 253

K

kernel

definition, 8

diagram, 10

general discussion, 8–11

integer types in, 281

involvement in signals, 384–387, 406

process representation in, 517–521

program paths to its services

- diagram, 61
- list of, 61–62

read-write locks in, 779

roles and responsibilities, 8–10

services, 10

thread support in, 711

- kernel API
 - defined, 6
 - explained, 8
- kernel buffering, 171, 797
- kernel filesystem interface, 259–261
- kernel I/O interface
 - closing files, 163–165
 - opening files, 158–163
 - reading from files, 165–169
 - structures involved in, 167–168
 - writing to files, 169–171
- kernel mapping in process image, 512
- kernel memory, 26
- kernel mode, 58
- kernel space, 9
- kernel stack, 518
- Kernighan, Brian W., 2, 17, 43
- Kerrisk, Michael, 731
- keypad() (Curses), 905, 906, 918
- keys, System V IPC, 616
- keywords for locale categories, 136–138
- kill(), 401–404
- kill character, 845
- kill command, 400
- kill_demo.c* program, 401–402

L

- LANG environment variable, 130
- LANGUAGE environment variable, 130
- lastb command, 194, 213
- last command
 - analysis and description, 213–215
 - and login records, 215–217
 - spl_last.c* program
 - design of, 230–232
 - implementation of, 232–241
 - use of *wtmp* file in, 227–230
- lastlog command
 - analysis and description, 194–199
 - lastlog structure, 196–199
 - spl_lastlog.c* program, developing, 205–213
 - design of, 206–208
 - implementation of, 208–209
 - program logic, 208
- lastlog* file
 - diagram of out-of-order reading, 207
 - diagram of structure, 196
 - displaying contents, 206–207
 - out-of-order reads in, 206–207
 - structure of, 196
 - use of, 196
- lastlog.h* header file, 197
- latency, file access, 258
- layered view of filesystem interfaces,
 - diagram, 250
- LC_ADDRESS category, 130
- LC_ALL environment variable, 130, 133, 139
- LC_COLLATE category, 130, 133, 141
- LC_CTYPE category, 129, 130, 136, 141
- LC_IDENTIFICATION category, 130
- LC_MEASUREMENT category, 130
- LC_MESSAGES category, 130
- LC_MONETARY category, 130, 141
- LC_NAME category, 130
- LC_NUMERIC category, 130, 136, 141
- LC_PAPER category, 130
- LC_TELEPHONE category, 130
- LC_TIME category, 130, 141
- ldd command, 56, 57
- ld-linux.so linker, 53
- ld.so linker, 53
- learning system programming, 93–94
- length modifier in printf(), 275
- less command, 34
- level-triggered notification, 797
- lib* directory, 23
 - in repository, 95, 103
- libraries
 - commands to query contents of, 54–56
 - commands to show libraries linked to programs, 56–57
 - C Standard Library, 57–58
 - overview, 50–53
 - shared (dynamically linked)
 - creating, 949–951
 - linking to, 951–953
 - shared vs. static, 53–54, 944–945
 - static
 - creating, 946–947
 - identifying, 946
 - linking to, 947–949
 - overview, 52–53
 - system, 51–52
- libspl.a*, 103
- lightweight process, 711
- limits.h*, 71
- line, login, 221
- line-buffered stream, 549
- line buffering in C I/O, 459
- line discipline, 838–841
- line editing in canonical mode, 878–879

- line feed character, 458
- LINE5 (Curses), 900
- line speed, terminal, 865
- link. *See also* filename
 - in directories, 19–21, 312
 - hard link, 24
 - symbolic or soft
 - and cp command, 173
 - and cycles, 335
 - defined, 24
 - following by nftw(), 348–349
 - following by stat() and lstat(), 266–267
 - in pathnames, 25
 - readlink(), 284
 - readlink command, 171
 - in time zone directories, 131
- linkage editor, 53, 944
- linker, defined, 4
- linker names, 949
- linking view of ELF, 500–501. *See also*
 - Executable and Linking Format
- Linux
 - brief history, 43
 - filesystems supported by, 252–253
 - and GNU, xxxiv
 - kernel, 43
 - overview, xxxiv
 - vs. Unix, xxxiv
- Linux Standard Base (LSB), 300
- lio_listio(), 810
- list_head structure in kernel, 520
- live process, 493
- lnext (literal next) character of
 - termios, 845
- load average, 930–931
- locale. *See* internationalization and locales
- locale-aware functions, 64
- locale categories, 129–131, 141
- locale command, 133–134
- localeconv(), 131, 143
- localedef command, 135
- locale definition file, 135–137
- locale object, 144–146
- locale_t, 144–145
- local headers in repository, 95
- local settings of terminal, 844
- localtime(), 112, 113, 208
- locking a mutex, 742–743
- LOCPATH environment variable, 130
- loff_t, 523

- logged-in users, number of, 930
- logging in, 14
- logical (virtual) addresses, 501
- logical disks, 251
- login accounting. *See also* last command;
 - lastlog command
 - displaying last logged in users, 213–241
 - displaying last login information, 193–199
 - files associated with, 220
- long options, 11
- low-level vs. high-level file I/O, 150
- lrt linker option, 462
- LSB (Linux Standard Base), 300
- ls command, 20, 313
- lseek()
 - diagram, 189
 - examples of, 190
 - and file offset, 188–190
 - to get size of file, 207
- lstat(), 267, 274, 275–276
- lstat_manpage_example.c program, 274–277

M

- M:1 (many-to-one) threading
 - model, 711
- major(), 275
- makedev(), 294
- makefilehole.c program, 191–192
- make_me_a_daemon() listing, 685–686
- make tutorial, xxxviii
- make utility, xxxiii
- malloc(), 127, 613
- man command, 30, 34–35, 38
- man pages
 - advantages of using, 30
 - header, 30
 - manual sections, 29
 - pager for, 34
 - searching through (apropos command), 38–40
 - sections of, 36–37
 - structure of, 35–37
 - use in learning system
 - programming, 94
- many-to-one (M:1) threading model, 711
- maps file of /proc subdirectories, 523–524
- master boot record (MBR), 251
- MAX_INPUT (limit constant), 839
- max_input.c program, 839

- McKusick, Marshall Kirk, 8, 252
- /media* directory, 23
- memory allocation functions, 127
- memory image, 491, 501, 511–517, 543–545
- memory management unit of
 - processor, 501
- memory objects in POSIX shared
 - memory, 602
- memory segments in System V shared
 - memory, 602
- memset(), 352, 458
- mesg command, 273
- message queue. *See also* interprocess communication
 - descriptor, 629
- Minix File System, 252
- Minix operating system, 43, 252
- minor(), 275
- MIN parameter of *termios*, 880–884
- minitime_test_demo.c* program, 883–884
- mkfifo(), 669
- mkfifo command, 668
- mknod command, 668
- mktime(), 112, 118–119
- mmap()
 - description, 603–607
 - diagram, 604
- /mnt* directory, 23
- mode2str() listing, 292–293
- mode_t, 158–159, 267, 605, 619, 630, 669
- Molay, Bruce, 94
- Morris, Robert, 29
- mount command, 337, 339–340
- mounting filesystems, 22–23, 337–341
 - descriptor, 262
- mount points, 337, 339–340
- mq_attr structure, 630
- mq_close(), 629, 630
- mqd_t, 630–633, 637–639
- mq_getattr(), 629
- mq_notify(), 629, 630, 635–636
- mq_open(), 629, 630–631
- mqrcv_demo.c* program, 632–633
- mq_receive(), 629, 630, 631, 636
- mq_send(), 629, 630, 631
- mqsend_demo.c* program, 633–634
- mq_setattr(), 629
- mq_unlink(), 629, 630, 631
- multiplexed I/O. *See* I/O multiplexing
- multithreaded process, 28

- multithreading. *See also* threads; *Pthreads* API
 - correctness and performance considerations, 740
 - multithreaded concurrent server, 731–735
 - multithreaded multiple producer, multiple consumer program, 757–762
 - program design considerations with, 715
 - pros and cons of, 711–712
- munmap(), 603, 604, 606–607
- mutex. *See Pthreads* API
- mutexattr_t structure, 749
- mutex lock, 614
- mvwhline(), 910
- mvwin(), 908

N

- named pipe. *See* FIFO
- named semaphore. *See* interprocess communication
 - namedsem_demo.c* program, 621–622
- NAME_MAX macro, 316–317
- name of program, extracting, 79–81
- NAME section, man pages, 36–37
- nanosleep(), 441–447, 449, 455, 467
- nanosleep_demo1.c* program, 442–444
- nanosleep_demo2.c* program, 445–447
- Native POSIX Thread Library (NPTL), 711
- ncurses* library. *See* Curses library and API
- network logins, 221–222
- Network Time Protocol (NTP) server, 440
- newline character, 2, 458
- newlocale(), 144, 145–146
- newterm(), 901
- newwin() (Curses), 908
- nftw(), 347–354, 368
- nftw_demo.c* program, 353–354
- nlink_t, 267
- nl_langinfo(), 143–144
- NLS_PATH environment variable, 130
- nm command, 55
- nodelay() (Curses), 905
- noecho() (Curses), 905, 906, 918
- nonblock_demo1.c* program, 793–796
- nonblock_demo2.c* program, 796
- nonblocking I/O, 792–796

- noncanonical modes of terminal,
 - 878–884. *See also*
 - interactive programming
- cbreak mode, 879
- cooked mode, 879
- effects of MIN and TIME parameters,
 - 880–894
- polling read in, 880
- raw mode, 879
- nonvolatile storage, 17
- normal library of Curses, 901
- NOTES section, man pages, 36
- NTFS filesystem, 261
- NTP (Network Time Protocol)
 - server, 440
- NULL byte, 3
- numbered subdirectories in */proc*,
 - 521–522
- numbers, extracting from strings, 86–89,
 - 97–102

O

- O_APPEND flag, 188, 551–552, 578, 598
- O_ASYNC flag, 798
- objdump command, 54–55
- object, locale, 144–146
- object library. *See* libraries
- object module, 50
- O_CREAT flag, 160, 161
- octal dump, 56
- od command, 56, 193, 845
- O_EXCL flag, 160, 161
- OFD (open file description), 155–157,
 - 164, 168
- offsetof(), 324
- offsets in shared memory, 613
- off_t, 189–190, 207, 209–210, 269
- one-to-one (1:1) threading model, 711
- onlcr switch in *termios*, 847
- online documentation, Unix, 29–34
- online materials, xxxvii
- O_NONBLOCK flag, 792–793, 885–886
- open()
 - creation flags in, 160
 - errors, 162
 - opening FIFO with, 670
 - prototype, 159
 - status flags in, 160
 - using, 159–163, 312, 551–552
- openat(), 158–159
- openat2(), 158–159
- opendir()

- diagram showing incorrect use, 342
 - overview, 318
 - prototype and description, 319–320
 - use in recursive tree walk, 341–343
- open file, sharing of
 - diagram, 546
 - overview, 546–551
- open file description (OFD), 155–157,
 - 164, 168
- open file table, 156
- Open Group, The, 44
- openlog(), 681–682
- open message queue description, 629
- open source software, xxxi
- operating system, xxxiv, 5–6, 8
 - diagram of layers, 6
- /opt* directory, 23
- options, command line
 - defined, 11–12
 - extracting, 81–86
- OPTIONS section, man pages, 36–37
- O_RDONLY access mode, 159
- O_RDWR access mode, 159
- orphan process, 559
- Ossanna, Joseph, 29
- O_TRUNC flag, 160, 161
- output, overview, 2–4
- output functions, *ncurses*, 907
- output settings of terminal, 844
- overrun count, 466
- O_WRONLY access mode, 159

P

- pad (Curses), 897
- pager, 34
- paging, 501
- PAM (pluggable authentication
 - modules), 221
- parallel reduction
 - algorithm, 767–768
 - diagram, 768
- parent directory (..), 19, 22, 339
- parent process, 493
- parent waiting for children. *See* process
- parent window in Curses, 896
- parking cursor, 903
- partial write, 170–171
- partitioning, disk, 251–252, 254
- passing data to a thread, 720–722
- passwd command, 154
- passwd* file, 202
- passwd structure, 201–202

- password database
 - API functions, 202
 - getpwuid_demo.c* program, 203
 - overview, 202–203
 - passwd structure, 201–202
 - showallusers.c* program, 204–205
- passwords, 13–14
- pathconf(), 839
- PATH_MAX macro, 377
- pathname, 24–25, 376–380
- pathname variable, 71, 317
- paths.h* file, 194
- pause(), 451–452
- pclose(), 663–666
- _PC_MAX_INPUT, 840
- pending signal, 388
- %c date/time format specifier, 107, 113–114
- performance issues
 - analysis of, in *spl_cp1.c*, 182–185
 - comparison of time() and clock_gettime(), 111
 - design considerations, 121
 - in disk file accesses, 257–259
 - in high-level vs. low-level I/O, 150
 - and kernel buffering
 - diagram, 182
 - with reads from disk, 182
 - measuring with time command, 179–181
 - and multithreading, 712, 715, 740
 - and nopenfd parameter of nftw(), 350
 - and out-of-order reads, 207
 - overhead of system calls, 181
 - in *pthread_rwlock_demo.c*, 779–780
 - and user space buffering, 243–244
 - using hash tables in *spl_du.c*, 361
 - using select() and poll(), 820–822
- perlipc man page, 600
- permission bits, 270
- permissions, 18–19
- per-process memory data, acquiring, 934–935
- per-process metadata, 512
- perror(), 64–65
- perror_demo.c* program, 65
- PGID (process group ID), 494–495
- PGRP of process, 494–495
- physical address, 501
- physical block, 248
- physical screen, 898
- PID (process identifier), 26–27, 492
- pidof command, 526, 656
- pidstat command, 435
- pid_t, 401, 541–542, 575
- Pike, Rob, 16–17
- pinpoint access, locales, 143
- pipe
 - basic principles, 646–648
 - diagrams showing sharing after fork(), 650
 - named. *See* FIFO
 - overview, 645–646
 - unidirectional
 - diagram, 647
 - limitations, 647
 - unnamed
 - behavior of read operations on, 651–652
 - behavior of write operations on, 652–653
 - best practices regarding, 662–663
 - producer-consumer example using, 653–656
 - shell pipe simulation, 656–662
 - simple program using, 648–651
- pipe(), 646, 648, 651
- PIPE_BUF, 648, 824–825
- pipe_demo1.c* program, 648–651
- pipe_demo2.c* program, 653–656, 657
- pipefs* filesystem, 651
- PIT (programmable interval timer), 439
- pkill command, 400
- plain files, 17
- platters, disk, 248, 249
- pluggable authentication modules (PAM), 221
- pointer arithmetic, 80
- pointers, 329, 610–613
- poll(), 820
- polling I/O, 796
- polling read, 880, 885
- popen(), 663–666, 688, 689, 693–694
- popen_demo.c* program, 665–666
- P operation, 614
- portability, xxxiv, 67–71
- position-independent code, 950
- POSIX (Portable Operating System Interface), 7, 44–45
- POSIX.1.2024 standard, 45
- POSIX asynchronous I/O (POSIX AIO)
 - AIO API, 808–809

POSIX AIO (*continued*)

- AIO functions, 809–813
 - asynchronous version of *spl_cp1.c*, 816–819
 - diagram, 808
 - error handling in, 811
 - overview, 807–808
 - performance benefits with disk files, 813–815
 - read operation using
 - diagram, 810
 - steps in, 810–811
- `_POSIX_C_SOURCE` macro, 70
- `_POSIX_SOURCE` macro, 70
- posix_timer_demo1.c* program, 485–487
- `POS_ONLY` flag, 99
- `post()`, 615
- postorder traversal, 336
- `pread64()`, 578
- `pread()`, 578–579, 598
- preorder traversal, 350
- preprocessor search path, 197
- present working directory, 25
- primary group, 14
- printable characters, 476
- printargs1.c* program, 74
- printargs2.c* program, 74–75
- print_args_env.c* program, 564–565
- print_elfphdr.c* program, 506–511
- `printenv` command, 16
- `printf()`, 2–4
 - format specifications, 4, 106–107, 113
- `printw()` (Curses), 907
- private FIFO, 669, 686–687
- privileged instruction, 15
- privileged mode, 15
- probability of race condition, 551
- process. *See also* waiting for child of a
 - process
 - address space
 - heap, 513
 - initialized data segment, 513
 - program displaying virtual memory locations, 515–516
 - relationship to process abstraction, 492
 - stack segment, 514
 - structure of, 511–514
 - text segment, 511
 - uninitialized data segment, 513
 - and virtual memory, 501, 511–517

- changing what the process executes
 - `exec()` family of functions, 565–569
 - `execve()`, 561–565
 - overview, 560
- components of, 492–493
- converting into daemon, 684–686
- creating a new process
 - child's memory image, 543–545
 - child's process descriptor, 545
 - `fork()`, 541–543
 - fork_demo1.c* program, 542
 - fork_demo2.c* program, 543–544
 - fork_demo3.c* program, 547–548
 - fork_demo4.c* program, 548–550
 - fork_demo5.c* program, 550
 - other functions for, 556–557
 - parts of descriptor not copied into child, 545
 - sharing of files previously opened, 546–549
- foreground and background, 497–498
- kernel's representation of, 517–521
- lifetime of, 540
- overview, 25–27
- POSIX definition, 492
- process descriptor
 - diagram, 519
 - overview, 517–518
 - parts shared by threads, 713
- process group, 493–496
- process metadata
 - representation in kernel, 517–518
 - shared/not shared by threads, 713
- process session, 496–497
- process tree, 493
- program files
 - contents of an executable file, 498–499
 - ELF, 499–506
 - overview, 498
 - program to print ELF program header table, 506–511
- simple shell demo program, 590–593
- terminating, 557–560
- and threads, 711–715
- user IDs for, 153–154
- process and job control signals, 392–393

- process descriptor, 518–521, 545, 713–714
- process-directed signal, 728
- process group, 493–496
- process group ID (PGID), 494–495
- process group leader, 494
- process identifier (PID), 26–27, 492
- processing command line
 - accessing environment, 75–78
 - arguments, 73–75
 - get_opt_demo.c* program, 84–85
 - numbers from strings, 86–89
 - options, 81–86
 - printargs1.c* program, 74
 - printargs2.c* program, 74–75
 - print_args_env.c* program, 564–565
 - program name, 79–81
 - reporting usage errors, 78–79
- processing directories. *See* directory API
- process list in Linux, 520
- process metadata, 492, 517–518
- process resource limits, 350
- process-shared attribute, 757
- process structure, 518
- process time, 108, 179–185
- process tree, 493
- /proc* pseudofilesystem
 - ancestors.c* program, 524–526
 - developing a simple ps program, 526–534
 - magic of, 522–523
 - NULL characters instead of space
 - characters in, 522
 - numbered directories, 521–522
 - overview, 492, 521
 - parsing text files in, 526–530
 - read operation in, 523
 - as snapshot of kernel data
 - structures, 523
 - spl_ps.c* program, 526–534
 - useful per-process files, 523–524
- procstat data structure, 527, 924–925
- /proc/tty/drivers* file, 529–530
- producer-consumer problem
 - and condition variables, 752–753
 - diagram, 576
 - multithreaded multiple producer, multiple consumer
 - program, 757–762
 - pipe_demo2.c* program, 653–656
 - process synchronization with
 - signals, 553
 - program with pipes, 653–656
 - pthread_prodcons.c* program, 757–762
 - shared memory producer consumer
 - program, 623–628
 - waitpid_demo.c* program, 575–580
- progname_demo.c* program, 80
- program break, 513–514
- program file
 - contents of executable files, 498–499
 - ELF
 - file contents, 502–506
 - file structure, 500–502
 - overview, 499–500
 - program to print program header table, 506–511
 - overview, 498
- program header, 501
- program header table, 501, 502, 506–511
- program libraries, 944
- programmable interval timer (PIT), 439
- program name, extracting, 79–81
- progress bar
 - based on alarms, 454–461
 - based on POSIX timers, 466–471
 - using normal mutex, 743–748
- progress_bar1.c* program, 460–461, 743
- progress_bar2.c* program, 470–471, 743
- Provenzano, Chris, 262
- ps command, 526–534
- pselect(), 820, 921–922
- pthread_attr_init(), 723
- pthread_attr_setstacksize(), 725–726
- pthread_attr_t structure, 717–718
- pthread_barrierattr_t, 717, 764
- pthread_barrier_destroy(), 765
- pthread_barrier_t, 764
- pthread_barrier_wait(), 765
- pthread_cancel(), 716, 724–725
- pthread_cleanup_push(), 718
- pthread_condattr_t, 717, 754, 757
- pthread_cond_broadcast(), 755
- pthread_cond_destroy(), 756–757
- pthread_cond_init(), 754
- PTHREAD_COND_INITIALIZER macro, 754
- pthread_cond_signal(), 755
- pthread_cond_t, 717, 754–756, 758–789
- pthread_cond_timedwait(), 754–755, 757
- pthread_cond_wait(), 754–755, 756
- pthread_create(), 716, 717–718, 720
- pthread_create_demo.c* program, 719–720
- pthread_detach(), 723
- pthread_detach_demo.c* program, 724

- `pthread_equal()`, 721
- `pthread_exit()`, 716, 718, 719
- `pthread_join()`, 716, 718–719
- `pthread_mutexattr_t`, 717, 741, 749–750
- `PTHREAD_MUTEX_DEFAULT`, 749
- `pthread_mutex_destroy()`, 743
- `PTHREAD_MUTEX_ERRORCHECK`, 749
- `pthread_mutex_init()`, 741–742
- `pthread_mutex_lock()`, 742
- `PTHREAD_MUTEX_NORMAL`, 749
- `PTHREAD_MUTEX_RECURSIVE`, 749
- `pthread_mutex_t`, 717, 741–743, 745, 750, 752
- `pthread_mutex_trylock()`, 742–743
- `pthread_mutex_unlock()`, 743
- `pthread_prod_cons.c` program, 761–762
- `pthread_rwlockattr_setkind_np()`, 780
- `pthread_rwlock_demo.c` program, 779–787
- `pthread_rwlock_rdlock()`, 777, 778
- `pthread_rwlock_t`, 717, 775–776, 781
- `pthread_rwlock_timedrdlock()`, 777
- `pthread_rwlock_tryrdlock()`, 777
- `pthread_rwlock_unlock()`, 777, 778
- `pthread_rwlock_wrlock()`, 778
- Pthreads* API
 - barriers, 764–765
 - barrier synchronization, 762–764
 - example program, 765–773
 - condition variables
 - condition attributes, 757
 - declaring and initializing, 754
 - destroying, 756–757
 - multithreaded multiple
 - producer, multiple
 - consumer program, 757–762
 - pthread_prodcons.c* program, 757–762
 - signaling, 755–756
 - waiting on, 754–755
 - data types, 717
 - mutexes
 - declaring and initializing, 741–742
 - destroying, 743
 - locking and unlocking, 742–743
 - other types of, 749–751
 - overview, 740–741
 - program using normal, 743–748
 - recursive_mutex_demo.c* program, 750–751
 - threaded_progrbar.c* program, 743–748
- overview, 711, 716–717
- read-write locks
 - example of, 779–787
 - further details about *Pthreads*, 778–779
 - overview, 774
 - pthread_rwlock_demo.c* program, 779–784
 - read-write lock API, 775–776
 - use and semantics of, 776–777
- thread attributes
 - getting and setting stack size, 725–726
 - initializing attributes, 717, 723
- thread management
 - canceled (terminating), 724–725
 - creating, 717–718
 - detaching, 722–724
 - diagram of partitioning array among, 721
 - distributing an array to threads, 721
 - exiting, 718
 - identifying, 722
 - joining, 718–720
- `pthread_self()`, 716, 721
- `pthread_setcancelstate()`, 725
- `pthread_setcanceltype()`, 725
- `pthread_sigmask()`, 728
- pthread_signal_demo.c* program, 729–731
- `pthread_t`, 717, 719, 721–726, 729, 733, 745
- public FIFO, 669
- `putwin()` (Curses), 908
- `pwd` command
 - absolute pathnames, 25
 - diagram of prepending to string, 377
 - and directory tree, 336, 371–373
 - overview, 371
 - strategy for implementing, 374–381
- pwd.h*, 202

Q

- `qsort()`, 925
- `qsort_r()`, 925–926

R

- race conditions
 - avoiding using pipes and message queues, 599–601

- with `dup()`, 659
 - explained, 455–456, 551, 613–614
 - `O_APPEND` flag, 551–552, 578
 - potential in multithreading, 712
 - preventing, 552, 576–578, 598
 - probability of, 551–552
 - program with race conditions, 455–456
 - and semaphores, 618–619, 621–622
 - shared memory, 613–614
 - signals for preventing, 553–556
 - sync_io_demo.c* program, 554–555, 578, 598
- radix character, 72
- `raise()`, 404
- raise_demo.c* program, 404–405
- `rand()`, 181
- raw mode of terminal, 251, 838, 879
- Raymond, Eric, 895
- `read()`
 - buffering and running time, 182–183
 - demo using, 173–179
 - MIN and TIME parameters, 880–884
 - and nonblocking I/O, 792–796
 - overview, 4
 - and pipes, 663
 - position of I/O operations, 188
 - reading directory contents, 312–313
 - reading from files with, 166–169
 - from terminal in noncanonical mode, 880–884
 - and VFS, 262
- read and write permissions on created file, 151–152
- `readdir()`, 314, 315–316, 341–345, 375
- `readelf`, 55–56, 503–506
- read end of pipe. *See* pipe
- reading
 - from device file, 177–178
 - directory contents, 312–313
 - from file, 165–169
 - from pipe
 - defining characteristics, 647
 - semantics, 651–652
 - POSIX AIO, 808, 810–811
 - universality of, 150
- `readlink()`, 284
- `readlink` command, 171
- read lock, 774
- read mode, 155
- read operations, behavior on pipes, 651–652
- read-write lock
 - diagram, 777
 - example of, 779–787
 - overview, 774
 - in *Pthreads*, 778–779
 - read-write lock API, 775–776
 - rwlock_demo.c* program, 779–787
 - use and semantics of, 776–777
- read/write mode, 155
- real name of shared library, 949
- real time
 - as calendar time, 108
 - in output of time command, 179–181
 - and SIGALRM, 392
- real-time clock (RTC), 439
- real-time library, 462
- real-time signals, 802
- real user ID, 153–154, 161–162
- reaping status of terminated child, 570, 582–590
- record lock, 552
- recursive mutex, 749–750
- recursive_mutex_demo.c* program, 750–751
- recursive tree walk, 341–347
- `refresh()` (Curses), 899, 903, 913
- refreshing screen, 475–478
- registering a signal handler. *See* signal handler
- register method for passing parameters, 59
- regular file, 17
- relative pathname, 25
- relative sleep, 447, 448
- relocatable code, 950
- relocatable file, 499
- relocation tables in ELF file, 498
- repeat interval of a timer, 462
- reporting usage errors, 78–79
- request code in `ioctl()`, 871–872
- research systems, Bell Labs, 41
- resource monitor
 - developing, 472–484
 - program logic, 473–474
 - refreshing screen for, 475–478
 - running, 483–484
 - setting up file list for, 474–475
 - signal handler logic for, 479
 - updating file sizes for, 478–479
 - watchfiles.c* program, 479–480
- resources, system, 5
- restarting system call after signal
 - interruption, 399, 411, 418, 426

- return statement, 557
- RETURN VALUE section, man pages, 36
- rewinddir(), 324–325
- Ritchie, Dennis, 2, 7, 16–18, 29, 41, 45, 150
- root directory (/), 22
- rotational delay, 249–250
- round-trip compatibility, 956
- RTC (real-time clock), 439
- runlevel, 214
- runtime
 - defined, 52
 - increasable limit, 71
 - invariant limit, 71
 - runtime binding, 262
 - timing programs, 179–185
- rwlockdemo.log* file, 787

S

- sa_flags field of sigaction structure, 417, 418, 419, 424–431
- sa_mask field of sigaction structure, 418
- SA_NOCLDWAIT flag, 584–585
- SA_NODEFER flag, 418, 424–425
- SA_RESETHAND flag, 418, 424–425
- SA_RESTART flag, 418, 424–425
- sa_restorer field of sigaction structure, 418
- sa_sigaction field of sigaction structure, 417
- SA_SIGINFO flag, 418, 419
- saved set-user-ID, 153, 154
- save-text-image bit, 273–274
- /sbin directory, 23
- sbrk(), 514
- scan codes, keyboard, 385
- scandir()
 - description and use of, 329–335
 - recursive tree walk using, 345–347
 - use in *spl_top.c*, 935
- scandir_manpage_example.c* program, 331–332
- scanf(), 3–5, 31, 89, 150, 525–527, 633–634, 906
- scanw() (Curses), 906
- screen, 2
- screen (Curses), 896
- SCREEN (Curses), 896
- screen cell, 803
- SCREEN data type (Curses), 896
- screen management
 - in *spl_top.c* program, 919–924
 - in *sprite.c* program, 889

- screen refreshing in Curses, 899
- screen updating in Curses, 898–899
- script in Unicode, 956
- scrolling, disabling in Curses, 921
- scrollok() (Curses), 921
- search path, preprocessor, 197
- secondary storage, 17
- section header table in ELF, 501, 502
- section in ELF, 501, 506
- sector, 248
- SEE ALSO section, man pages, 36
- seekdir(), 325–328
- seeking, 188–190, 249
- seek time, 249–250
- segmentation
 - ELF files, 501, 506, 512
 - of process, 511–512, 513–514
- select(), 820–821. *See also* I/O multiplexing
- select_demo.c* program, 824–830
- select law, 823–824
- SELinux, 57
- semaphore, 614–616. *See also* interprocess communication
- sem_close(), 620
- sem_destroy(), 617
- sem_getvalue(), 617
- sem_init(), 617
- sem_open(), 619–620
- sem_post(), 617, 623–624
- sem_t, 617–619, 621, 623–624
- sem_unlink(), 620
- sem_wait(), 617, 623–624
- sending signals, 400–405
- seqcount_t, 790
- server. *See* client-server applications
- session, 496–497
- session ID (SID), 496–497
- session leader, 496, 684
- setgid bit, 272–273
- setitimer(), 462
- setlocale(), 131, 138–139, 205
- setpgid(), 495–496
- setpwent(), 204
- setsid(), 496
- setuid bit, 154–155
- setuid program, 154
- shareable file, 23–24
- shared library. *See* libraries
- shared memory. *See* interprocess communication
- shared object file (.so file), 52
- shared object file ELF format, 500

- shared resources of threads, 712–715
- sharing of open files, 546–551
- shell
 - bash, 13, 33–34
 - creating named pipes in, 667–669
 - diagram, 658
 - help feature, 33–34
 - overview, 12–13
 - pipe simulation, 656–662
 - shell builtins, 13
 - spl_sh.c* program, 590–593
 - shellpipe_demo.c* program, 661–662
 - shell script, 13
 - shm_consumer.c* program, 625–627
 - shm_creator_demo1.c* program, 608, 609
 - shm_creator_demo2.c* program, 611–612
 - shm_demo1.h* header file, 607
 - shm_open()*, 603, 605–606
 - shm_producer.c* program, 627–629
 - shm_unlink()*, 603, 604
 - shm_user_demo1.c* program, 608–609
 - shm_user_demo2.c* program, 612–613
 - short description in man page, 39
 - short options of a command, 11
 - showallusers.c* program, 204–205
 - showbreak.c* program, 514
 - SHUTDOWN_TIME value, 232
 - sibling of a process, 493
 - SID (session ID), 496–497
 - SIGABRT signal, 390, 391
 - sigact_demo1.c* program, 420–422
 - sigact_demo2.c* program, 422–423
 - sigact_demo3.c* program, 428–431
 - sigaction()
 - effect of *sa_flags* on signal handler execution, 424–431
 - effect of SA_SIGINFO flag on form of handler, 419–424
 - overview, 416–417
 - role of sigaction structure in, 417
 - sigaction structure, 417–419, 584–585
 - SIGALRM signal, 390, 392, 451–452, 744
 - sig_atomic_t, 412–413, 431–432
 - SIGBUS signal, 390, 391
 - SIGCHLD signal
 - and asynchronous waiting, 582–590
 - overview, 390, 393
 - SIGCHLD reaping signal handler, 585–590
 - sighandler_wait_demo.c* program, 585–590
 - SIGCONT signal, 390, 393
 - SIGEMT signal, 391
 - sigevent structure, 463, 467, 635
 - SIGFPE signal, 390, 391
 - sighandler_t, 395, 397
 - SIGHUP signal, 390, 392, 497–498, 684
 - SIGILL signal, 390, 391
 - siginfo_t structure, 415, 417–422, 424, 463, 484–485, 582, 585–586, 635, 637
 - SIGINT signal
 - and background process, 497
 - blocking, 409–414
 - default actions, 390
 - and kill command, 401–403
 - overview, 384, 392
 - programs that catch, 396–397, 398–399
 - signal delivery example, 385–386
 - signal_demo1.c* program, 396–397
 - sysv_signal_demo.c* program, 398–399
 - sigio_counter.c* program, 800–801
 - sigio_demo.c* program, 802–807
 - SIGIO signal, 390, 392, 799, 801
 - SIGKILL signal, 390, 392, 406
 - signal
 - asynchronous, 384
 - basic handling of, 393–399
 - blocking
 - overview, 405–407
 - signal sets, 407–408
 - sigprocmask(), 408–416
 - and clock_nanosleep() function, 448–449
 - concepts, 387–389
 - condition variables, 755–756
 - delivery of, 384
 - design of signal handlers, 431–432
 - disposition of, 389
 - example of signal delivery, 385–386
 - handler. *See* signal handler
 - kill(), 401–402
 - kill command, 400
 - kill_demo.c* program 402
 - lifetime of, 387–389
 - pending, 388
 - process synchronization with, 553–556
 - from program errors, 391
 - real-time signals and timers, 484–487
 - registering handlers, 394
 - role of, 384–387
 - sending, 400–405
 - sigact_demo1.c* program, 420–422
 - sigact_demo2.c* program, 422–423

- signal (*continued*)
 - sigact_demo3.c* program, 428–431
 - sigaction(), 416–417
 - signal(), 395–399
 - signal_demo1.c* program, 396–397
 - signal_demo2.c* program, 399
 - signal_demo3.c* program, 402–403
 - signal types, 389–393
 - sources of, 386–387
 - synchronous, 384
 - synchronously waiting using
 - sigwait(), 415, 733
 - table of types, 390
 - and threads, 727–731
 - use with message queues, 635–636
- signal(), 395–399
- signal definitions, 393
- signal_demo1.c* program, 396–397
- signal_demo2.c* program, 399
- signal_demo3.c* program, 402–403
- signal-driven I/O. *See* alternative methods of I/O
- signal handler. *See also* sigaction()
 - async-signal-safety, 397
 - basic signal handling, 393–399
 - diagram, 394
 - guidance on designing, 431–432
 - installing (registering), 388, 394
 - overview, 388–389
 - program for displaying hardware-generated signal, 422–423
 - program for displaying source of signal, 420–421
 - reentrant, 419, 432, 593
 - sa_flags, effect on execution, 424–431
 - signal(), 395–396
 - signal information passed to, 419–424
- signal.h* header file, 393
- signal mask, 406, 728–731
- signal sets, 407–408
- signal types, 385, 389–393
- signed integer, 167
- sigpending(), 728
- SIGPIPE signal, 390, 662, 674
- SIGPOLL signal, 390
- sigprocmask(), 407, 408–416, 456, 744
- sigprocmask_demo1.c* program, 409–411
- sigprocmask_demo2.c* program, 411–412
- sigprocmask_demo3.c* program, 413–415
- SIGPROF signal, 390
- SIGPWR signal, 390, 393
- SIGQUIT signal, 390, 392, 396–397, 398–399
- SIGRTMAX macro, 484–485
- SIGRTMIN macro, 484–485
- SIGSEGV signal, 390, 391
- sigset_t, 407–411, 413–415, 417, 456
- SIGSTOP signal, 390, 393, 405, 406
- sigsuspend(), 414–415
- SIGSYS signal, 390, 391
- SIGTERM signal, 390, 392, 400, 401–402
- SIGTRAP signal, 390, 391
- SIGTSTP signal, 390, 393, 405
- SIGTTIN signal, 390, 497
- SIGTTOU signal, 390, 497
- SIGURG signal, 390, 392
- SIGUSR1 signal, 390, 393, 464, 467, 484, 553–556
- SIGUSR2 signal, 390, 393, 484, 553–556
- SIGVTALRM signal, 390, 392
- sigwait(), 415, 733
- sigwaitinfo(), 415
- SIGWINCH signal, 390, 393, 891, 913
- SIGXCPU signal, 390
- SIGXFSZ signal, 390
- simple commands, 11, 72
- single-indirect block, 256
- Single UNIX Specification, 44–45, 461
- sizeof(), 199
- size_t, 63, 167, 199, 269
- sleep(), 399, 440
- sleep functions
 - high-resolution
 - clock_nanosleep(), 447–450
 - clock_nanosleep_demo.c* program, 449–450
 - nanosleep(), 441–447
 - nanosleep_demo1.c* program, 442–443
 - nanosleep_demo2.c* program, 445–446
 - overview, 440–441
 - return value of clock_nanosleep() vs. nanosleep(), 449
 - sleep(), 399, 440
 - timer drift with, 445–450
 - and timers, 437–438
 - using absolute time with, 447–450
 - usleep(), 409–411
- sleeping locks, 779
- snake terminal-based game, 884
- sockets, 18
- soft link, 24. *See also* link
- soft resources, 5
- software clock, 440
- software libraries, 944

- software timer. *See* timers
- Solaris UFS, 252
- soname, 949
- source code, xxxvii, 25
- sources of signals, 386–387
- sparse files, 193
- Spec 1170 Initiative, 44
- special bits, 270
- special characters in terminal, 386, 844, 851–852
- special file, 18
- special settings in terminal, 844
- speed_t, 849, 859, 865, 869
- spinning lock, 779
- spl_calc_client.c* program, 690–692
- spl_calc.h* header file, 689–690
- spl_calc_server.c* program, 692–695
- spl_cp1.c* program, 173–179.
 - See also* copying files
- spl_date1.c* program, 114
- spl_date2.c* program, 114–116
- spl_date3.c* program, 116–128
- spl_du1.c* program, 358–359
- spl_du2.c* program, 364–365
- spl_last.c* program, 230–242
- spl_lastlog.c* program, 209–213
- spl_ls1.c* program, 321–323
- spl_ls2.c* program, 328
- spl_ls3.c* program, 333–334
- spl_ls_rec1.c* program, 343–345
- spl_ls_rec2.c* program, 346
- spl_pwd.c* program, 379–380
- spl_sh.c* program, 590–593
- spl_stat.c* program, 285–297.
 - See also* filesystem
- spl_statfs.c* program, 299–308.
 - See also* filesystem
- spl_stty.c* program
 - functions that display attributes, 861–866
 - functions that set attributes, 866–870
 - main() listing, 860–861
 - overview, 856–857
 - program data structures, 857–859
- spl_top.c* program
 - acquiring per-process data, 934–938
 - acquiring summary data, 930–934
 - data structures, 924–928
 - design considerations, 917–918
 - input mode and cursor, 918–919
 - main() listing, 938–940
 - overview, 915–916
 - per-process data, 934–938
 - requirements of, 916–917
 - screen management, 919–924
 - sorting functions, 928–929
- spl_utmpdump.c* program, 225–226
- sprintf(), 100
- sprite, 803
- sprite.c* program
 - Curses version of, 911–915
 - global constants, types, and variables, 888–889
 - main() listing, 893–894
 - overview, 884
 - program features and issues, 885–887
 - sprite_state[] array, 889, 890
 - support functions, 889–892
 - terminal control functions, 887
- spurious wake-up, 755–756
- /srv directory, 23
- sscanf(), 89, 524, 531, 906
- ssize_t, 68, 167–168
- stack segment, 514
- Stallman, Richard, 7, 31, 42
- standard error, 157
- standard file descriptors, 157
- standard input, 157
- standard output, 157
- standards, Unix, 7, 43–45
- standard screen, 900
- standout mode, 920, 921
- start character in terminal, 845
- stat(), 256
- stat command, 264–266
- stat file of /proc subdirectories, 524
- statfs(), 299
- static files, 24
- static library. *See* libraries
- static linking, 944–945
- statm file of /proc subdirectories, 524
- stat structure, 267–269
- status-checking macros, 573–574
- status file of /proc subdirectories, 522, 524
- statvfs()
 - comparison with statfs(), 300–301
 - prototype, 300
 - use in *spl_statf.c*, 301–307
- statx()
 - calling, 281–284
 - prototype, 278
 - use in *spl_stat.c*, 301–307
- statx structure, 278–281
- st_dev field of stat structure, 268
- stdio.h header file, 67–68

- stdscr (Curses), 900
- sticky bit, 273–274
- st_mode field of stat structure, 268, 270–271
- str2int.c listing, 89
- strcmp(), 141, 142
- strcoll(), 141–142, 368
- strcoll_demo.c program, 141–142
- stream buffering, 549–550
- strerror(), 64, 65–66, 102
- strerror_demo.c program, 66
- strftime(), 112–113
- string() man page, 121–122
- strings, environment, 15–16
- strings, extracting numbers from, 86–89
- string tables in ELF file, 499
- strncpy(), 76
- strrchr(), 79–80
- strstr(), 124
- strtok(), 122–123
- strtoul(), 86, 87–89, 97, 100
- strtoul_demo.c program, 87–88
- struct __dirstream, 319
- struct lastlog, 196–199
- struct mq_attr, 630
- struct passwd, 201
- struct tm, 109
- struct tty_struct, 838
- stty command, 841–843, 856
- subwindows in Curses, 896
- suffix of a filename, 22
- sum_reduce() listing, 771–772
- superblock, 255
- superuser, 15
- supervisor mode, 15
- surfaces, disk, 248
- susp character in terminal, 845
- SuSV4 (Single UNIX Specification Version 4), 44
- SVID (System V Interface Definition), 397–399
- _SVID_SOURCE macro, 70
- swap area, 251
- swap partition, 251
- switches, terminal, 842, 845–847, 849–851, 866–868
- switch statement, integer type of expression, 124
- symbolic constants for file mode, 162
- symbolic link, 24, 275–276, 335. *See also* link
- symbol tables in ELF file, 498
- synchronization mechanism. *See also* Pthreads API
 - mutex lock, 614
 - overview, 601, 614
- synchronous I/O, 797
- synchronous signal, 384
- sync_io_demo.c program, 554–555, 578, 598
- SYNOPSIS section, man pages, 36–37
- syscall(), 60–61
- sysconf(), 686
- sys_hdrs.h file in repository, 95
- syslog(), 681–682
- syslog_demo.c program, 682–683
- system(), 593, 663
- system buffering, 182–185, 797
- system call
 - diagram, 60
 - errors, handling, 62–66
 - execution of, 59–61
 - overhead of, 181
 - overview, 6, 58–59
 - passing parameters to, 58–59
 - as path to kernel services, 61
 - return values, 62
 - syscall(), 60
 - system calls without wrappers, 60
 - trap and sysenter machine instructions, 58
 - wrapper functions, 59
- system clock, 440
- system data types, 268, 269
- system_demo.c program, 593
- system libraries, 51–52
- system limits
 - getconf command
 - limits.h, 71
 - overview, 71
 - pathconf(), 71, 317
 - pathname variable, 71
 - runtime increasable, 71
 - runtime invariant, 71
 - sysconf(), 71
- system logging facilities, 681–683
- system programs, 6–7, 121
- system requirements, xxxiii
- system resources, 5–6
- system space, 9
- system time, 179–181
- System V. *See also* interprocess communication
 - IPC facilities, 601–602
 - semaphore API, 616

- System V Interface Definition (SVID), 397–399
- system-wide headers, 95
- `sysv_signal()`, 398
- sysv_signal_demo.c* program, 398

T

- TABSIZE (Curses), 900
- Tanenbaum, Andrew, 43, 252
- target of link, 24
- task, 28
- `task_struct` structure, 518–519, 713–714
- `tcflag_t`, 848–849, 866
- `tcflush()`, 427
- `tcgetattr()`, 852, 854, 856
- TCIFLUSH, 427–429, 482, 855
- `tcsetattr()`, 852, 853, 855, 856, 887
- teletype, 221
- `telltdir()`, 325–328
- terminal. *See also* noncanonical modes of terminal
 - API functions
 - `cfgetispeed()`, 864–865
 - `cfgetospeed()`, 864–865
 - `cfsetispeed()`, 865
 - `cfsetospeed()`, 865
 - `ioctl()`, 870–875
 - `tcflush()`, 427
 - `tcgetattr()`, 852–854, 856
 - `tcsetattr()`, 852–853, 855–856, 887
 - API overview, 847–849
 - API structures
 - special characters, 851–852
 - special settings, 844
 - switches (flags), 849–851
 - TCSADRAIN, 852
 - TCSAFLUSH, 852, 856
 - TCSANOW, 852, 854, 856
 - termios.h* header file, 848
 - `termios` structure, 848, 852–853, 880–884
 - MIN and TIME parameters, effect on reads, 880–884
 - variables not in `termios`, 842–843
 - attribute categories, 844–845
 - and background processes, 497
 - canonical mode, 841, 846–847, 850–851, 877–879
 - canonical processing, 841
 - controlling, 496–497, 529, 558, 684

- control settings of, 844
- CTRL-C in, 385–386
- in Curses, 896
- developing the *spl_stty.c* program. *See spl_stty.c* program
- driver
 - diagram, 839
 - input queue, 839–840
 - line discipline in, 838
 - operations, 838–841
 - output queue, 839–840
 - structure of, 839
- enabling and disabling echoing using `stty`, 842–843
- escape sequence (ANSI), 476–477, 796, 806, 882, 888–890
- example programs
 - fakeinput_demo.c* program, 872–873
 - fillqueue.c* program, 874–875
 - max_input.c* program, 839–840
 - spl_stty.c* program, 856–870
 - toggle_echo.c* program, 853–854
 - upcophchars.c* program, 835–836
 - watchtty.c* program, 873–874
- filling the input queue with `ioctl()`, 874–875
- flushing queues
 - in Curses, 905
 - with `ioctl()`, 872
 - with `tcflush()`, 848, 855–856
- and foreground processes, 497
- getting and changing attributes
 - in a program, 852–856
 - in the shell, 841–844
- getting device name for
 - in a program, 839, 876
 - in the shell, 178
- getting size of input queue, 839–840
- handling excess input in, 855
- input settings of, 844
- and interactive programs, 834
- `isatty()`, 550
- local settings of, 844
- output settings of, 844
- overview, 476, 834–838
- related signals, 390
- reset command, 482
- resetting, 842
- as source of signals, 386, 391–392
- special characters of, 844, 851–852
- special settings of, 844

- terminal (*continued*)
 - stty command, 841–846
 - switches (flags), 849–851
 - termios data structure, 848, 852–853, 880–884
 - tty command, 178
 - ttyname(), 839, 876
 - viewing and changing settings in
 - shell, 841–844
- terminal device driver, 838
- terminal emulator, 476
- terminal I/O, 834–835
- terminal start character, 841
- terminal stop character, 841
- terminal stop signal, 393
- terminal window, 2, 475
- terminating processes, 540, 557–560
- termination signals, 391–392
- terminfo library, 895
- termios data structure, 848, 852–853, 880–884
- termios.h header file, 848
- TES (Thread Execution Scheduling), 776–777
- testdircalls.c program, 312–313
- Texinfo documentation system, 31
- text file, 17
- text segment, 498, 501–505, 511–513
- Thompson, Ken, 7, 13, 16–17, 29, 41, 42, 150, 646, 958
- thread_attr_setdetachstate(), 723
- thread-directed signals, 727
- threaded_progbar.c program
 - design, 743–748
 - diagram of logic, 744
- threaded_upcased.c program, 733–734
- Thread Execution Scheduling (TES), 776–777
- thread identifier (TID), 28–29
- threads. *See also* Pthreads API
 - background, 710–711
 - diagram showing use of task_struct
 - for threads, 714
 - history, 710
 - multithreaded concurrent server,
 - use in, 731–735
 - overview, 27–29, 710–711
 - per-thread resources and attributes, 712–715
 - and processes, 711–715
 - program design considerations
 - with, 715
 - pros and cons of multithreading, 711–712
 - runtime stack, 714–715
 - shared resources and attributes, 712–715
 - and signals, 727–731
 - standardization, 710
 - support in kernel, 711
 - threading models, 711
- thread synchronization. *See* Pthreads API
- throw-away code, 87
- tick, clock
 - definition, 439–440
 - use in stat file in /proc subdirectories, 527–530
- TID (thread identifier), 28–29
- tiled windows (in Curses)
 - spl_top.c program, 919–921
 - tiled_windows.c program, 909–911
- time(), 110–111, 113, 179–180, 438
- Time and the Art of Living (Grudin), 108
- time conversion functions, 112–114
- timed read, 881
- time formats. *See* date/time formats and representations
- time format specifiers, 961–963
- timeout (timer), 436
- timeout parameter, 822
- TIME parameter of terminal, 880–884
- TIMER_ABSTIME flag, 448, 468
- timer_create(), 462–463
- timer_delete(), 464
- timer drift, 445, 447
- timer expiration signal, 392
- timer_getoverrun(), 466
- timer_gettime(), 462
- timer interrupt, 439
- timer overruns, 465–466
- timers. *See also* interval timer
 - alarm() one-shot timer. *See* alarm()
 - comparison with alarm clocks, 436
 - comparison with sleep functions, 436–437
 - deprecated functions, 462
 - hardware clocks and hardware timers, 439
 - High Precision Event Timer (HPET), 439
 - jiffy, 439
 - keeping track of time with, 436–438
 - overruns
 - counting, 466

- definition, 466
- posix_timer_manpage*
 - _example.c*, 466
- `timer_getoverrun()` for counting, 466
- programmable interval timer (PIT), 439
- software timer vs. hardware timer, 437
- system clock, 440
- using absolute time in, 465
- `timer_settime()`, 462, 464–467, 471, 482, 486
- `timer_t`, 464, 466
- `timespec`, 441–442
- Time Stamp Counter, 439
- timestamp format, 288
- `time_t`, 110, 112–113, 119, 198, 207–208, 224, 230, 269
- `__time_t`, 198–199
- time_utils.h* header file, 441–442
- time_utils.h* listing, 442–443
- time zone, 112, 131, 139
- timing programs, 179–185
- TIOCINQ request code of `ioctl()`, 873
- TIOCSI request code of `ioctl()`, 872, 874
- `tm_isdst` field of `tm` structure, 109
- /tmp* directory, 23
- `tm_sec` field of `tm` structure, 109
- toggle_echo.c* program, 853–854
- `top` command, 435
- top-level directories
 - diagram, 23
 - list of common ones, 23
- Torvalds, Linus, 43, 253, 254, 262
- tracks of disk, 248
- trap instruction, 58
- tree, directory
 - constructing from inode numbers and links, 371–373
 - overview, 335–337
- `tree` command, 336
- tree walks. *See* directory hierarchy
- triple-indirect block, 257
- `truncate()`, 59
- `truncate64()`, 59
- Ts'o, Theodore, 253
- Tweedie, Stephen, 253
- `typeflag` argument, 348–349
- TZ environment variable, 130, 131

U

- `__u16`, 280–281
- `__u32`, 280–281
- `__u64`, 280–281
- UCB (University of California at Berkeley), 42
- `ucontext_t`, 418–419
- UCS (Universal Character Set), 956
- UFS (Unix File System), 252
- `uid`. *See* user ID
- `uid2name()` listing, 293
- `uid_t`, 203, 268, 280
- `uintmax_t`, 275–276, 355
- ulogger_client.c* program, 637–639
- ulogger.c* program, 641–642
- `umask()`, 153
- `umask` command, 152
- `umasks`, 150–153
- `umount` command, 337
- `uname()`, 181
- unbuffered streams, 549
- Unicode, 133, 955–960
- unidirectional pipe, 646–647, 650
- uninitialized data segment, 513
- union (C language), 417
- unistd.h*, 54, 64, 83
- unit separator character in terminal, 843
- Universal Character Set (UCS), 956
- universal I/O, 150
- University of California at Berkeley (UCB), 42
- Unix
 - core concepts
 - command, 11–12
 - device-independent I/O,
 - diagram of, 7
 - directory, 19–21
 - directory hierarchy, 22–25
 - environment, 15–16
 - file, 17
 - file attributes and permissions, 18–19
 - filename (link), 20
 - filesystem, 17
 - file types, 18
 - group (of users), 14
 - kernel, 7–10
 - man pages, 29–30
 - mounting, 22
 - pathname, 24–25
 - privileged instruction, 15
 - privileged mode, 15

- Unix (*continued*)
 - core concepts (*continued*)
 - process, 25–27
 - root, 15
 - shell, 12–13
 - superuser, 15
 - symbolic link, 24
 - thread, 27–29
 - top-level directories, diagram
 - of, 23
 - user, 13–14
 - history, xxxiv, 41–43
 - standards, 43–45
- Unix File System (UFS), 252
- UNIX Programmer's Manual, 29, 41
- unmounted filesystems, 337
- unnamed pipe. *See* pipe
- unnamed semaphore, 616–619
- unnamedsem_demo.c* program, 618–619
- unprivileged mode, 15
- unresolved symbol, 4
- unsetenv() (Curses), 897
- unshareable file, 23–24
- upcase.c* program, 698–700
- upcase.h* header file, 696
- upcopychars2 program, 840, 845–846, 878–879
- upcopychars.c* program, 835–837
- updating file sizes, 478–479
- usagecheck_demo.c* program, 78–79
- usage_error() listing, 103
- usage errors, reporting, 78–79
- USAGE section, man pages, 36
- uselocale(), 144–145
- user entries in *passwd* file, printing all, 203–205
- user ID
 - general discussion, 199–205
 - obtaining username from, 293
 - for processes, 153–154
 - setgid bit, 272
 - use in *lastlog.c* program, 206–207
- user mode, 15
- usernames, 199–205
- USER_PROCESS value, 231
- users, Unix, 13–15
- user space, 9
 - buffering of input, 242–245
- user structure, 518
- user time reported by time command, 179–181
- usleep(), 409–410

- /usr directory, 23
- /usr/share/zoneinfo directory, 131
- UTF-8 codeset
 - encoding of Unicode
 - algorithm for, 957–959
 - examples, 959–960
 - summary, 133
- utility programs, xxxiii
- utmp* file
 - historical background, 217
 - and logins/logouts, 220–222
 - program for displaying, 222–227
- utmp.h* file, 197–198
- utmp structure, 216, 217–219
- utmpx API, 217, 220, 225
- ut_type member of utmpx struct, 223

V

- V (semaphore post) operation, 614
- valgrind utility, xxxiii
- /var directory, 23
- variable files, 24
- variables, environment, 15–16
- variables in Curses, 900
- variadic function, 871
- VERSIONS section, man pages, 36
- vfork(), 556–557
- virtual addresses, 501
- virtual address space, 511, 512
- virtual console, 475–476
- virtual dispatch table, 262
- virtual filesystem (VFS), 261–263
- virtual memory
 - calculating total used by processes, 768–771
 - diagram of process layout, 512
 - diagram showing thread stacks in, 715
 - displaying a process's own locations, 515–517
 - displayvm.c* program, 515–517
 - kernel responsibility, 10
 - layout of process, 511–517
 - management, 501
 - vmem_usage.c* program, 765–773
- virtual screen, 898–899
- virtual time, 392
- visible canvas, 898
- vmem_usage.c* program, 765–773
- vmstat command, 436
- vnode, 263
- vnumber, 263

`void*` parameter of `thread` function,
720–721
volatile storage, 17

W

`wait()`, 569–570
wait_demo1.c program, 571–572
wait_demo2.c program, 573–574
`waitid()`, 582
waiting for child of a process
 API functions
 `wait()`, 570–574
 `waitid()`, 582
 `waitpid()`, 570, 574–581
 asynchronous waiting
 effect of `SA_NOCLDWAIT` flag on
 `SIGCHLD` handler, 585
 overview, 582–583
 problems with reaping inside
 `SIGCHLD` handler, 583
 reaping `SIGCHLD` handler,
 585–587
 role of `SIGCHLD` signal in, 584
 diagram, 572
 overview, 569–570
 reaping child's status, 570, 582–583
 synchronous waiting
 disadvantages of, 582
 polling wait in producer-
 consumer program,
 580–581
 polling wait with `WNOHANG`
 flag, 575
 with `wait()`, 570–574
 waitable child, 571
 waiting for all children, 571–572
 waiting for a specific child, 574–575
waiting on condition variables, 754–755
`waitpid()`, 570, 574–581
waitpid_demo.c program, 574–581, 598
`wants_...` field variable, 283
watchfiles.c program
 overview, 471–474
 refreshing screen, 475–478
 setting up file list, 474–475
 updating file sizes, 478–479
watchtty.c program, 873–875
`WCONTINUED` flag, 574
`WCOREDUMP()` macro, 573
well-known FIFOs, 669
werase character, 845
`WEXITSTATUS()` macro, 573

whence parameter of `lseek()`, 189
whitespace characters on
 command line, 72
`who` command, 215
wide library functions, 901
`WIFEXITED()` macro, 573
`WIFSIGNALED()` macro, 573
wildcards in `apropos` command, 39
window, *ncurses*, 896
window configuration, top program, 921
`WINDOW` data structure, 896
`WINDOW` data type, 896
window functions, *ncurses*, 908
winsize.c program, 864
`WNOHANG` flag, 574
`WNOWAIT` flag, 582
words, on command line, 72
`__WORDSIZE_TIME64_COMPAT32` macro,
198–199
wrapper functions, 59, 61
`wrefresh()` (Curses), 899
`write()`, 4, 170–171, 182–183, 260,
652–653, 662–663
`write` command, 272–273
write end of pipe, 646–647
`writelfds` parameter of `select()`, 821
write lock, 774
write mode, 155
writing
 behavior on pipes, 652–653
 to files, 169–171
 locking for, 775
 opening FIFOs, 670–671
 POSIX AIO, 808, 812
`wstandout()` (Curses), 921
`WTERMSIG()`, 573
wtmp file
 analysis of, 227–230
 historical background, 217
 and logins/logouts, 220–222
 program for showing, 222–227
`WUNTRACED` flag, 574

X

`_XOPEN_SOURCE` macro, 70
XSI Curses. *See* Curses library and API
xterm emulator, 834

Z

zombie processes, 570