

INDEX

The following conventions are used in the index:

- Library function and system call prototypes are indexed with a subentry labeled *prototype*. Normally, you'll find the main discussion of a function or system call in the same location as the prototype.
- Definitions of C structures are indexed with subentries labeled *definition*. This is where you'll normally find the main discussion of a structure.
- Implementations of functions developed in the text are indexed with a subentry labeled *code of implementation*.
- Instructional or otherwise interesting examples of the use of functions, variables, signals, structures, macros, constants, and files in example programs are indexed with subentries labeled *example of use*. Not all instances of the use of each interface are indexed; instead, just a few examples that provide useful guidance are indexed.
- Diagrams are indexed with subentries labeled *diagram*.
- The names of example programs are indexed to make it easy to find an explanation of a program that is provided in the source code distribution for this book.
- Citations referring to the publications listed in the bibliography are indexed using the name of the first author and the year of publication, in an entry of the form *Name (Year)*—for example, Rochkind (1985).
- Items beginning with nonalphabetic characters (e.g., `/dev/stdin`, `_BSD_SOURCE`) are sorted before alphabetic items.

Symbols

#! (interpreter script), 572
• (directory entry), 27, 351
.. (directory entry), 27, 351
/ (root directory), 27
/boot/vmlinuz file, 22
/dev directory, 252
/dev/console device, 777
/dev/fd directory, 107–108
/dev/kcore file, 801
/dev/kmem device, 801

/dev/log socket, 776
 diagram, 775
/dev/mem device, 166, 801
/dev/null device, 769
/dev/poll device (Solaris), 1328
/dev/ptmx device, 1381
/dev/pts directory, 1321, 1380, 1382
/dev/ptyxy devices, 1395
/dev/random device, 801
/dev/shm directory, 275, 1090, 1108
/dev/stderr, 108
/dev/stdin, 108

- /dev/stdout, 108
- /dev/tty device, 707, 708, 1321. *See also*
controlling terminal
- /dev/tty devices, 1289
- /dev/ttyxy devices, 1395
- /dev/zero device, 1034
 - used with *mmap()*, 1034
 - example of use*, 1036
- /etc directory, 774
- /etc/fstab file, 263
- /etc/group file, 26, 155–156
- /etc/gshadow file, 156
- /etc/hosts file, 1210
- /etc/inetd.conf file, 1249–1250
- /etc/inittab file, 820
- /etc/ld.so.cache file, 848, 854
- /etc/ld.so.conf file, 847, 848
- /etc/ld.so.preload file, 874
- /etc/localtime file, 198
- /etc/mtab file, 263
- /etc/named.conf file, 1210
- /etc/passwd file, 26, 153–155
- /etc/resolv.conf file, 1211
- /etc/services file, 1212–1213
- /etc/shadow file, 155
- /etc/syslog.conf file, 776, 781–782
 - diagram*, 775
- /lib directory, 847, 848, 854
- /lib/ld-linux.so.2 (dynamic linker),
839, 844
- /lib/libc.so.6 (*glibc* 2), 844, 870
- /proc file system, 42, 223–228
 - diagram*, 227
- /proc/config.gz file, 1418
- /proc/cpuinfo file, 752
- /proc/domainname file, 230
- /proc/filesystems file, 255
- /proc/hostname file, 230
- /proc/kallsyms file, *diagram*, 119
- /proc/kmsg file, 776
- /proc/ksyms file, *diagram*, 119
- /proc/locks file, 1140–1142
- /proc/mounts file, 263
- /proc/net/tcp file, 1276
- /proc/net/tcp6 file, 1276
- /proc/net/udp file, 1276
- /proc/net/udp6 file, 1276
- /proc/net/unix file, 1276
- /proc/partitions file, 254
- /proc/PID directory, 224–226
- /proc/PID/cmdline file, 124, 225
- /proc/PID/coredump_filter file, 449, 615
- /proc/PID/cwd file, 225, 364, 800
- /proc/PID/envIRON file, 126, 225, 801
- /proc/PID/exe file, 225, 564, 800
- /proc/PID/fd directory, 107, 225, 342, 762
- /proc/PID/fdinfo directory, 75
- /proc/PID/limits file, 755
- /proc/PID/maps file, 225, 842, 1006, 1008,
1019, 1025, 1041, 1115
 - example of use*, 1046
- /proc/PID/mem file, 225
- /proc/PID/mounts file, 225, 263
- /proc/PID/oom_adj file, 615, 1040
- /proc/PID/oom_score file, 1040
- /proc/PID/root file, 225, 367, 800
- /proc/PID/smaps file, 1006
- /proc/PID/stat file, 599, 700, 748, 755
- /proc/PID/status file, 115, 172, 224, 225,
764, 799, 806, 1049, 1050
- /proc/PID/task directory, 225
- /proc/PID/task/TID/status file, 799, 806
- /proc/self symbolic link, 225
- /proc/swaps file, 254
- /proc/sys/fs/file-max file, 763, 801
- /proc/sys/fs/inotify/max_queued_events
file, 385
- /proc/sys/fs/inotify/max_user_instances
file, 385
- /proc/sys/fs/inotify/max_user_watches file,
385, 1358
- /proc/sys/fs/mqueue/msg_max file, 1086
- /proc/sys/fs/mqueue/msgsize_max file, 1086
- /proc/sys/fs/mqueue/queues_max file, 1086
- /proc/sys/fs/nr_open file, 762
- /proc/sys/fs/pipe-max-size file, 892
- /proc/sys/fs/suid_dumpable file, 449
- /proc/sys/kernel/acct file, 594
- /proc/sys/kernel/cap-bound file, 815
- /proc/sys/kernel/core_pattern file, 449
- /proc/sys/kernel/msgmax file, 951
- /proc/sys/kernel/msgmnb file, 801, 949, 951
- /proc/sys/kernel/msgmni file, 951
- /proc/sys/kernel/ngroups_max file, 179
- /proc/sys/kernel/osrelease file, 229
- /proc/sys/kernel/ostype file, 229
- /proc/sys/kernel/pid_max file, 115, 228
- /proc/sys/kernel/pty/max file, 1381
- /proc/sys/kernel/pty/nr file, 1381
- /proc/sys/kernel/rtsig-max file, 458
- /proc/sys/kernel/rtsig-nr file, 458
- /proc/sys/kernel/sched_child_runs_first
file, 526
- /proc/sys/kernel/sem file, 992
- /proc/sys/kernel/shmall file, 1015
- /proc/sys/kernel/shmmax file, 1015
- /proc/sys/kernel/shmni file, 1015
- /proc/sys/kernel/threads-max file, 763
- /proc/sys/kernel/version file, 229
- /proc/sys/net/core/somaxconn file, 1157

/proc/sys/net/ipv4/ip_local_port_range file, 1189, 1224
 /proc/sys/net/ipv4/tcp_ecn file, 1267
 /proc/sys/vm/dirty_expire_centisecs file, 241
 /proc/sys/vm/legacy_va_layout file, 793
 /proc/sys/vm/overcommit_memory file, 1038
 /proc/sys/vm/overcommit_ratio file, 1039
 /proc/sysvipc/msg file, 935
 /proc/sysvipc/sem file, 935
 /proc/sysvipc/shm file, 935
 /proc/version file, 229
 /sbin/init file, 33
 /sys directory, 252
 /tmp directory, 300, 791
 /usr/account/pacct file, 592
 /usr/group association, 12
 /usr/lib directory, 847, 848, 854
 /usr/lib/locale directory, 201, 203
 /usr/local/lib directory, 847, 848
 /usr/share/locale directory, 201
 /usr/share/locale/locale.alias file, 201
 /usr/share/zoneinfo directory, 198
 /usr/src/linux directory, 1424
 /var/log directory, 774
 /var/log/lastlog file, 830
 /var/log/messages file, 782
 /var/log/pacct file, 592
 /var/log/wtmp file, 818
 /var/run directory, 1142
 /var/run/utmp file, 818
 <errno.h> header file, 49
 <features.h> header file, 62
 <limits.h> header file, 212
 <sys/types.h> header file, 68
 __GLIBC__ constant, 48
 __GLIBC_MINOR__ constant, 48
 __WALL constant, 610
 __WCLONE constant, 609
 example of use, 602
 __WNOHREAD constant, 610
 _ATFILE_SOURCE feature test macro, 366
 _BSD_SOURCE feature test macro, 62
 _CS_GNU_LIBC_VERSION constant, 48
 _CS_GNU_LIBPTHREAD_VERSION constant, 694
 _CS_PATH constant, 588
 _exit(), 32, 426, 514, 531–532, 692
 example of use, 524, 583, 587, 759
 prototype, 531
 _Exit(), 426
 _FILE_OFFSET_BITS macro, 104, 106
 _fini(), 873
 _GNU_SOURCE feature test macro, 62
 _init(), 873
 _IOFBF constant, 238
 _IOLBF constant, 238
 _IONBF constant, 237
 _LARGEFILE64_SOURCE feature test macro, 105
 _longjmp(), 429
 _PATH_LASTLOG constant, 830
 _PATH_UTMP constant, 818
 _PATH_WTMP constant, 818
 _PC_CHOWN_RESTRICTED constant, 221
 _PC_NAME_MAX constant, 214, 218
 _PC_PATH_MAX constant, 214, 218
 _PC_PIPE_BUF constant, 214, 218
 _PC_VDISABLE constant, 1296
 example of use, 1301
 _POSIX_ASYNCHRONOUS_IO constant, 221
 _POSIX_C_SOURCE feature test macro, 61, 63
 _POSIX_CHOWN_RESTRICTED constant, 221
 _POSIX_JOB_CONTROL constant, 221
 _POSIX_MQ_OPEN_MAX constant, 1085
 _POSIX_MQ_PRIO_MAX constant, 1073
 _POSIX_PIPE_BUF constant, 891
 _POSIX_PRIORITY_SCHEDULING constant, 221
 _POSIX_REALTIME_SIGNALS constant, 221
 _POSIX_RTSIG_MAX constant, 457
 _POSIX_SAVED_ID constant, 221
 _POSIX_SEMAPHORES constant, 221
 _POSIX_SHARED_MEMORY_OBJECTS constant, 221
 _POSIX_SIGQUEUE_MAX constant, 457
 _POSIX_SOURCE feature test macro, 61
 _POSIX_THREAD_KEYS_MAX constant, 668
 _POSIX_THREADS constant, 221
 _REENTRANT macro, 622
 _SC_ARG_MAX constant, 124, 214, 217
 _SC_ASYNCHRONOUS_IO constant, 221
 _SC_ATEXIT_MAX constant, 535
 _SC_CHILD_MAX constant, 217, 763
 _SC_CLK_TCK constant, 206, 214
 example of use, 209
 _SC_GETPW_R_SIZE_MAX constant, 158
 _SC_IOV_MAX constant, 100
 _SC_JOB_CONTROL constant, 221
 _SC_LOGIN_NAME_MAX constant, 214
 _SC_MQ_PRIO_MAX constant, 1073
 _SC_NGROUPS_MAX constant, 179, 214
 _SC_OPEN_MAX constant, 214, 217
 example of use, 771
 RLIMIT_NOFILE resource limit and, 762
 _SC_PAGE_SIZE constant, 214
 _SC_PAGESIZE constant, 214, 215
 _SC_PRIORITY_SCHEDULING constant, 221
 _SC_REALTIME_SIGNALS constant, 221
 _SC_RTSIG_MAX constant, 214
 _SC_SEMAPHORES constant, 221
 _SC_SHARED_MEMORY_OBJECTS constant, 221
 _SC_SIGQUEUE_MAX constant, 214, 457
 _SC_STREAM_MAX constant, 214

- `_SC_THREAD_KEYS_MAX` constant, 668
- `_SC_THREAD_STACK_MIN` constant, 682
- `_SC_THREADS` constant, 221
- `_SC_XOPEN_UNIX` constant, 221
- `_SEM_SEMUN_UNDEFINED` constant, 970
- `_setjmp()`, 429
- `_SVID_SOURCE` feature test macro, 62
- `_sys_errlist` variable, 664
- `_sys_nerr` variable, 664
- `_XOPEN_SOURCE` feature test macro, 62, 63
- `_XOPEN_UNIX` constant, 221

Numbers

- 2MSL, 1274
- 3BSD, 4
- 4.2BSD, 4, 155, 342, 390, 443, 476, 776, 1149, 1180
- 4.3BSD, 4
- 4.4BSD, 4, 17, 1442
- 4.4BSD-Lite, 8
- 386/BSD, 7

A

- a.out* (executable file format), 113
- ABI, 118, 867
- `abort()`, 390, 426, 433–434, 446
 - prototype*, 433
- absolute pathname, 29, 367
- abstract socket binding, 1175
- `ac` command, 818
- `accept()`, 426, 673, 801, 1152, 1157–1158
 - diagram*, 1156
 - example of use*, 1168, 1222
 - inheritance of file flags and socket options, 1281
 - interrupted by signal handler, 444
 - prototype*, 1157
 - RLIMIT_NOFILE resource limit and, 762
- `accept4()`, 1158
 - interrupted by signal handler, 444
- access control list (ACL), 319–337, 800, 1440
 - access ACL, 327
 - ACL entry, 320–321
 - application programming interface, *diagram*, 330
 - default ACL, 327
 - diagram*, 320
 - extended ACL, 321
 - group class, 324–325
 - limits on number of entries, 328–329
 - long text form, 323
 - mask entry, 321, 323, 324–325
 - minimal ACL, 321

- permission set, 320
- permission-checking algorithm, 321–322
- short text form, 323
- tag qualifier, 320, 321, 323, 332
- tag type, 320, 323, 331
- access mode, file, 72, 75, 93, 95
- `access()`, 298–299, 345, 426
 - prototype*, 299
- `acct` structure, 593–594
 - definition*, 593
- `acct()`, 345, 592–593, 801
 - example of use*, 593
 - prototype*, 592
- `acct_on.c`, 592
- `acct_v3` structure, 597–598
 - definition*, 598
- `acct_v3_view.c`, 598
- `acct_view.c`, 596
- `accton` command, 592
- ACK control bit (TCP), 1267
- ACL. *See* access control list
- `acl_add_perm()`, 332
 - diagram*, 330
- `acl_calc_mask()`, 333
- `acl_check()`, 334
- `acl_clear_perms()`, 332
 - diagram*, 330
- `acl_create_entry()`, 332
 - diagram*, 330
- `acl_delete_def_file()`, 334
- `acl_delete_entry()`, 333
 - diagram*, 330
- `acl_delete_perm()`, 332
 - diagram*, 330
- `acl_dup()`, 334
- `acl_entry_t` data type, 331
 - diagram*, 330
 - example of use*, 335
- `acl_error()`, 334
- ACL_EXECUTE constant, 332
- ACL_FIRST_ENTRY constant, 331
- `acl_free()`, 334
 - example of use*, 336
- `acl_from_text()`, 333
 - diagram*, 330
- `acl_get_entry()`, 331
 - diagram*, 330
 - example of use*, 335
- `acl_get_file()`, 331
 - diagram*, 330
 - example of use*, 335
- `acl_get_perm()`, 332
 - diagram*, 330
 - example of use*, 336

acl_get_permset(), 332
 diagram, 330
 example of use, 336
acl_get_qualifier(), 332
 diagram, 330
 example of use, 336
acl_get_tag_type(), 331
 diagram, 330
 example of use, 336
 ACL_GROUP constant, 321, 322, 323
 ACL_GROUP_OBJ constant, 321, 322, 323
acl_init(), 334
 ACL_MASK constant, 321, 322, 323,
 324–325, 333
 ACL_NEXT_ENTRY constant, 331
 ACL_OTHER constant, 321, 322, 323
acl_permset_t data type, 332
 diagram, 330
 example of use, 335
 ACL_READ constant, 332
acl_set_file(), 333
 diagram, 330
acl_set_permset(), 332
 diagram, 330
acl_set_qualifier(), 332
 diagram, 330
acl_set_tag_type(), 331
 diagram, 330
acl_t data type, 331
 diagram, 330
 example of use, 335
acl_to_text(), 333
 diagram, 330
 ACL_TYPE_ACCESS constant, 331, 333
 ACL_TYPE_DEFAULT constant, 331, 333
acl_type_t data type, 331
 diagram, 330
 example of use, 335
acl_update.c, 334
 ACL_USER constant, 320, 321, 322, 323
 ACL_USER_OBJ constant, 320, 321, 322, 323
acl_valid(), 334
acl_view.c, 335
 ACL_WRITE constant, 332
 ACORE constant, 594
 active close (TCP), 1272
 active open (socket), 1155
 address (socket), 1152
 Address Resolution Protocol (ARP), 1181
 address-space randomization, 793
addrinfo structure, 1214, 1215
 definition, 1214
adjtime(), 205, 801
 prototype, 205
adjtimex(), 205, 801
 Advanced Research Projects Agency
 (ARPA), 1180
 advisory file lock, 1119, 1137
 AF_INET constant, 1150, 1151
 AF_INET6 constant, 1150, 1151
 example of use, 1208, 1209
 AF_LOCAL constant, 1150
 AF_UNIX constant, 1150, 1151
 example of use, 1168, 1169, 1172, 1173
 AF_UNSPEC constant, 1162, 1215, 1217
 example of use, 1221, 1224, 1229
 Affero General Public License (GNU),
 xxxiv
 AFORK constant, 594
 Aho (1988), 574, 1437
 Aho, A.V., 1437
 AI_ADDRCONFIG constant, 1216
 AI_ALL constant, 1216
 AI_CANONNAME constant, 1214, 1216
 AI_NUMERICHOST constant, 1216
 AI_NUMERICSERV constant, 1216
 example of use, 1221
 AI_PASSIVE constant, 1216
 example of use, 1221, 1229
 AI_V4MAPPED constant, 1216
 AIO (asynchronous I/O), 613, 1327, 1347
aio_error(), 426
aio_return(), 426
aio_suspend(), 426, 673
 AIX, 5
alarm(), 390, 426, 484–485, 486, 488, 614
 example of use, 487
 prototype, 484
 Albitz (2006), 1210, 1247, 1437
 Albitz, P., 1437
 algorithmic-complexity attack, 794, 1438
 Allman, M., 1194
alloca(), 150–151
 prototype, 150
 allocating memory
 on the heap, 140–144, 147–150
 on the stack, 150–151
 alternate signal stack, 65, 434–437, 578,
 613, 683, 691, 693, 764
 American National Standards Institute
 (ANSI), 11
 Anley (2007), 792, 795, 1437
 Anley, C., 1437
anon_mmap.c, 1036
 anonymous mapping, 35, 882, 886, 1017,
 1033, 1034–1037
 private, 1019, 1035
 shared, 1019, 1035
 anonymous root, DNS, 1210

ANSI (American National Standards Institute), 11

ANSI C, 11

Anzinger, G., xxxix

application binary interface, 118, 867

ar command, 834

archive, 834

ARG_MAX constant, 214

argc argument to *main()*, 31, 123

argv argument to *main()*, 31, 118, 123, 124, 214, 564, 567

diagram, 123

example of use, 123

ARP (Address Resolution Protocol), 1181

ARPA (Advanced Research Projects Agency), 1180

ARPANET, 1180

asctime(), 16, 191, 657

diagram, 188

example of use, 192, 199

prototype, 191

asctime_r(), 191, 658

ASN.1, 1200

ASU constant, 298, 594, 928

async-cancel-safe function, 680

asynchronous I/O, POSIX, 613, 1327, 1347

async-signal-safe function, 425–428

AT_EACCESS constant, 365

AT_FDCWD constant, 290, 366

AT_REMOVEDIR constant, 365

AT_SYMLINK_FOLLOW constant, 365, 366

AT_SYMLINK_NOFOLLOW constant, 290, 365, 366

atexit(), 532, 534–535, 866

example of use, 537, 915, 960, 1393

prototype, 534

atomic_append.c, 1425

atomicity, 90–92, 465

when accessing shared variables, 631

Austin Common Standards Revision Group, 13

Autoconf program, 219, 1444

automatic variables, 116, 122

A/UX, 5

awk program, 574, 1437

AXSIG constant, 594

B

B programming language, 2

Bach (1986), 250, 278, 521, 530, 919, 1422, 1437

Bach, M., 1437

background process group, 700, 708, 714

diagram, 701, 717

bad_exclusive_open.c, 90

bad_longjmp.c, 1426

bad_symlink.c, 1428, 1429

basename(), 370–372, 657

example of use, 371

prototype, 370

bash (Bourne again shell), 25

baud, 1316

bcopy(), 1166

BCPL programming language, 2

Becher, S., xxxix

become_daemon.c, 770

become_daemon.h, 770

becomeDaemon(), 769–771

code of implementation, 770–771

example of use, 774, 1241, 1244

prototype, 769

Bell Laboratories, 2

Benedyczak, K., xxxix

Berkeley Internet Name Domain (BIND), 1210, 1437

Berkeley Software Distribution, 4, 7–8

bg shell command, 715

diagram, 717

Bhattiprolu (2008), 608, 1437

Bhattiprolu, S., 1437

Biddle, R.L., xl

Biederman, E.W., 1437

big-endian byte order, 1198

diagram, 1198

binary semaphores, 988–991

binary_sems.c, 990

binary_sems.h, 989

BIND (Berkeley Internet Name Domain), 1210, 1437

bind mount, 272–274

bind(), 345, 426, 1152, 1153–1154, 1155

diagram, 1156, 1160

example of use, 1166, 1168, 1172, 1173, 1176, 1208, 1222, 1229

prototype, 1153

Bishop (2003), 795, 1437

Bishop (2005), 795, 1437

Bishop, M., 795, 1437

Black, D., 1194

Blaess, C., xxxvi

blkcnt_t data type, 64, 280

casting in *printf()* calls, 107

blksize_t data type, 64, 280

block device, 252, 282

block groups (*ext2* file system), 256

Boolean data type, 51

boot block, 256

BOOT_TIME constant, 820, 822

Borisov (2005), 300, 1438

Borisov, N., 1438

Borman, D., 1194
 Bostic, K., 1442
 Bound, J., 1194
 Bourne again shell (*bash*), 25
 Bourne, S., 25
 Bourne shell (*sh*), 3, 25, 154
 Bovet (2005), 24, 46, 250, 256, 278, 419, 521, 530, 616, 919, 936, 994, 1015, 1044, 1147, 1422, 1438
 Bovet, D.P., 1438
 Braden, R., 1194
 Brahneborg, D., xxxix
 BREAK condition, 1302, 1304, 1318
 Brecht, T., 1439
brk(), 140
 prototype, 140
 RLIMIT_AS resource limit and, 760
 RLIMIT_DATA resource limit and, 761
 BRKINT constant, 1302, 1304
 example of use, 1311
 broken pipe (error message). *See* SIGPIPE
 signal
 broken-down time, 189
 converting to and from printable form, 195–197
 converting to *time_t*, 190
 Brouwer, A.E., xxxix
 BSO constant, 1302
 BS1 constant, 1302
 BSD, 4, 7–8
 BSD file locks, 1120
 BSD Net/2, 7
 BSDi, 8
 BSDLY constant, 1302
 BSD/OS, 8
 bss, 116
Btrfs file system, 261
 buffer cache, 233, 234
 using direct I/O to bypass, 246–247
 buffer overrun, 792
 buffering of file I/O, 233–250
 diagram, 244
 effect of buffer size on performance, 234–236
 in the kernel, 233–236, 239–243
 overview, 243–244
 in the *stdio* library, 237–239, 249
 BUFSIZ constant, 238
Build_ename.sh, 57
 built-in command (shell), 576
 bus error (error message). *See* SIGBUS
 signal
 BUS_ADRLN constant, 441
 BUS_ADRERR constant, 441
 BUS_MCEERR_AO constant, 441
 BUS_MCEERR_AR constant, 441
 BUS_OBJERR constant, 441
 busy file system, 270
 Butenhof (1996), 630, 639, 647, 659, 687, 696, 751, 1105, 1422, 1438
 Butenhof, D.R., xxxvi, 1438
 byte stream, 879, 890
 separating messages in, 910–911
 diagram, 911
bzero(), 1166

C
 C library, 47–48, 1442
 C programming language, 2, 1440, 1444
 ANSI 1989 standard, 11
 C89 standard, 11, 17
 C99 standard, 11, 17
 ISO 1990 standard, 11
 standards, 10–11
 C shell (*cs*h), 4, 25
 C89, 11, 17
 C99, 11, 17
 cache line, 748
 calendar time, 185–187
 changing, 204–205
calendar_time.c, 191
calloc(), 147–148
 example of use, 148
 prototype, 148
 canceling a thread. *See* thread cancellation
 cancellation point, thread cancellation, 673–674
 canonical mode, terminal I/O, 1290, 1305, 1307
 Cao, M., 1441
 CAP_AUDIT_CONTROL capability, 800
 CAP_AUDIT_WRITE capability, 800
 CAP_CHOWN capability, 292, 800, 807
 CAP_DAC_OVERRIDE capability, 287, 299, 800, 807
 CAP_DAC_READ_SEARCH capability, 299, 800, 807
 CAP_FOWNER capability, 76, 168, 287, 288, 300, 303, 308, 800, 807
cap_free(), 808
 example of use, 809
 CAP_FSETID capability, 304, 800, 807, 1432
cap_get_proc(), 807
 example of use, 809
 CAP_IPC_LOCK capability, 800, 999, 1012, 1048, 1051
 CAP_IPC_OWNER capability, 800, 928, 929
 CAP_KILL capability, 402, 800
 CAP_LEASE capability, 800

CAP_LINUX_IMMUTABLE capability, 306, 800, 807
 CAP_MAC_ADMIN capability, 800
 CAP_MAC_OVERRIDE capability, 800, 807
 CAP_MKNOD capability, 252, 368, 800, 807
 CAP_NET_ADMIN capability, 800
 CAP_NET_BIND_SERVICE capability, 800, 1189
 CAP_NET_BROADCAST capability, 800
 CAP_NET_RAW capability, 800
 CAP_SET constant, 807
cap_set_flag(), 807
 example of use, 809
cap_set_proc(), 808
 example of use, 809
 CAP_SETFCAP capability, 799, 800
 CAP_SETGID capability, 172, 800, 1285
 CAP_SETPCAP capability, 801, 806, 807, 812, 814, 815, 816
 CAP_SETUID capability, 172, 801, 1285
 CAP_SYS_ADMIN capability, 254, 262, 312, 607, 763, 801, 929, 1285
 CAP_SYS_BOOT capability, 801
 CAP_SYS_CHROOT capability, 367, 801
 CAP_SYS_MODULE capability, 801, 815
 CAP_SYS_NICE capability, 736, 743, 747, 750, 801
 CAP_SYS_PACCT capability, 592, 801
 CAP_SYS_PTRACE capability, 364, 801
 CAP_SYS_RAWIO capability, 255, 801
 CAP_SYS_RESOURCE capability, 306, 756, 763, 801, 892, 949, 1086
 CAP_SYS_TIME capability, 204, 492, 801
 CAP_SYS_TTY_CONFIG capability, 801
cap_t data type, 807
 example of use, 809
 capability
 file. *See* file capabilities
 process. *See* process capabilities
 capability bounding set, 615, 801, 805–806, 815
capget(), 807
capset(), 807
 Card, R., 255
 catch_rtsigs.c, 462
 catch_SIGHUP.c, 710
catgets(), 202, 533, 657
catopen(), 202, 533
 CBAUD constant, 1302, 1317
 CBAUDEX constant, 1302, 1317
 cbreak mode (terminal I/O), 1309–1316
cc_t data type, 64, 1292
 Cesati, M., 1438
cfgetispeed(), 426, 1316–1317
 prototype, 1316
cfgetospeed(), 426, 1316–1317
 prototype, 1316
cfsetispeed(), 426, 1316–1317
 prototype, 1316
cfsetospeed(), 426, 1316–1317
 prototype, 1316
 Chandra, C., xl
 change_case.c, 1432
 character device, 252, 282
 chattr command, 305
chdir(), 345, 364–365, 426, 604, 607
 example of use, 365
 prototype, 364
 check_password.c, 164
 check_password_caps.c, 808
 Chen (2002), 795, 1438
 Chen, H., 1438
 chiflag.c, 1428
 child process, 31, 513, 515
 signaled on death of parent, 553
 waiting on, 541–553
 child_status.c, 548
chmod(), 286, 303–304, 325, 345, 426, 800
 prototype, 303
 Choffnes, D.R., 1438
 chown command, 292
chown(), 221, 286, 291–293, 345, 426, 800
 example of use, 294
 prototype, 292
 chroot jail, 273, 367, 789
chroot(), 345, 367–368, 580, 604, 607, 801
 prototype, 367
 Church, A.R., xxxix
 Church, D.E., xl
 Church, D.E.M., xl
 Chuvakin, A., 1442
 CIBAUD constant, 1302
 Clare, G.W., xxxvii
 CLD_CONTINUED constant, 441, 551
 CLD_DUMPED constant, 441
 CLD_EXITED constant, 440, 441, 551
 CLD_KILLED constant, 441, 551
 CLD_STOPPED constant, 441, 551
 CLD_TRAPPED constant, 441
 cleanup handler, thread cancellation, 676–679
clearenv(), 129–130
 prototype, 129
 client, 40
 client-server architecture, 40
 CLOCAL constant, 1302
 clock, POSIX. *See* POSIX clock
clock(), 207–208, 210
 example of use, 209
 prototype, 207

clock_getcpuclockid(), 493, 496
 prototype, 493
clock_getres(), 491
 prototype, 491
clock_gettime(), 426, 491
 example of use, 494, 511
 prototype, 491
 CLOCK_MONOTONIC constant, 491, 492, 494, 508
 CLOCK_MONOTONIC_COARSE constant, 492
 CLOCK_MONOTONIC_RAW constant, 492
clock_nanosleep(), 493–494, 673
 example of use, 494
 interrupted by signal handler, 444
 prototype, 493
 CLOCK_PROCESS_CPUTIME_ID constant, 491, 492, 494
 CLOCK_REALTIME constant, 491, 492, 494, 508
 example of use, 501, 507
 CLOCK_REALTIME_COARSE constant, 492
clock_settime(), 492
 prototype, 492
clock_t data type, 64, 206, 207, 208, 438
 CLOCK_THREAD_CPUTIME_ID constant, 491, 492
clockid_t data type, 64, 491, 492, 493, 495
 CLOCKS_PER_SEC constant, 207, 208, 210
 example of use, 209
 clone child, 609
clone(), 598–609, 801, 987
 example of use, 602
 prototype, 599
 RLIMIT_NPROC resource limit and, 763
 speed, 610–612
 CLONE_CHILD_CLEARPID constant, 600, 606
 CLONE_CHILD_SETTID constant, 600, 606
 CLONE_FILES constant, 600, 603
 example of use, 602
 CLONE_FS constant, 600, 604, 607
 CLONE_IDLETASK constant, 608
 CLONE_IO constant, 600, 608
 CLONE_NEWIPC constant, 600, 608
 CLONE_NEWNET constant, 600, 608
 CLONE_NEWNS constant, 261, 600, 607, 801
 CLONE_NEWPID constant, 600, 608
 CLONE_NEWUSER constant, 600, 608
 CLONE_NEWUTC constant, 608
 CLONE_NEWUTS constant, 600
 CLONE_PARENT constant, 600, 608
 CLONE_PARENT_SETTID constant, 600, 606
 CLONE_PID constant, 600, 608
 CLONE_PTRACE constant, 600, 608
 CLONE_SETTID constant, 600, 607
 CLONE_SIGHAND constant, 600, 604, 605
 CLONE_SYSVSEM constant, 600, 607, 987
 CLONE_THREAD constant, 600, 604–606
 CLONE_UNTRACED constant, 600, 608
 CLONE_VFORK constant, 600, 608
 CLONE_VM constant, 600, 604
clone2(), 599
close(), 70, 80–81, 426
 example of use, 71
 prototype, 81
 CLOSE_WAIT state (TCP), 1269
closedir(), 354–355
 example of use, 356
 prototype, 355
closelog(), 777, 780
 prototype, 780
 close-on-exec flag, 74, 96, 98, 355, 377, 576–578, 613, 788, 894, 1110, 1153, 1158, 1175, 1281, 1356
 closeonexec.c, 578
 CLOSING state (TCP), 1269
cmdLineErr(), 53–54
 code of implementation, 57
 prototype, 54
 CMSPAR constant, 1302
 COFF (Common Object File Format), 113
 Columbus UNIX, 922
 Comer (1999), 1235, 1438
 Comer (2000), 1210, 1235, 1438
 Comer, D.E., 1438
 command interpreter, 24
 command-line arguments, 31, 122–124, 225
 Common Object File Format (COFF), 113
comp_t data type, 64, 593, 594, 598
 compressed clock tick, 594
 concurrent server, 912, 957, 1239–1240, 1243–1247
 condition variable, 614, 642–652, 881
 association with mutex, 646
 destroying, 652
 initializing, 651–652
 signaling, 643–644
 statically allocated, 643
 testing associated predicate, 647–648
 waiting on, 643–645
 CONFIG_BSD_PROCESS_ACCT kernel option, 592
 CONFIG_HIGH_RES_TIMERS kernel option, 485
 CONFIG_INOTIFY kernel option, 376
 CONFIG_INOTIFY_USER kernel option, 376
 CONFIG_LEGACY_PTYS kernel option, 1395
 CONFIG_POSIX_MQUEUE kernel option, 1063
 CONFIG_PROC_FS kernel option, 275
 CONFIG_PROCESS_ACCT_V3 kernel option, 597
 CONFIG_RT_GROUP_SCHED kernel option, 744
 CONFIG_SECURITY_FILE_CAPABILITIES kernel option, 814
 CONFIG_SYSVIPC kernel option, 922

CONFIG_UNIX98_PTYS kernel option, 1381
confstr(), 48, 588, 694
 congestion control (TCP), 1192, 1194, 1236, 1443
connect(), 426, 673, 1152, 1158
 diagram, 1156
 example of use, 1169, 1224, 1228
 interrupted by signal handler, 444
 prototype, 1158
 used with datagram sockets, 1162
 connected datagram socket, 1162
 container, 608
 controlling process, 39, 533, 700, 706–708, 712
 controlling terminal, 34, 39, 77, 533, 615, 700, 705, 706–708, 1380, 1385.
 See also /dev/tty device
 diagram, 701
 obtaining name of, 707
 opening, 707
 Cook, L., xl
 cooked mode (terminal I/O), 1309–1310
copy.c, 71
 copy-on-write, 521, 1018
 diagram, 521
 Corbet (2002), 307, 1438
 Corbet (2005), 278, 1422, 1438
 Corbet, J., 1438
 core dump file, 83, 166, 389, 441, 448–450, 530, 546, 594, 692, 789
 circumstances when not produced, 448–449
 naming, 449–450
 obtaining for running process, 448, 1430
 resource limit on size of, 760
 set-user-ID programs and, 789
 Cox, J., 1440
 CPF_CLOEXEC constant, 1143
 CPU affinity, 748
 CPU time. *See* process time
CPU_CLR(), 749
 prototype, 749
CPU_ISSET(), 749
 prototype, 749
CPU_SET(), 749
 example of use, 750
 prototype, 749
CPU_ZERO(), 749
 example of use, 750
 prototype, 749
 CR terminal special character, 1296, 1297, 1298, 1302, 1307
 CR0 constant, 1302
 CR1 constant, 1302
 CR2 constant, 1302
 CR3 constant, 1302
 CRDLY constant, 1302
 CREAD constant, 1303
creat(), 78–79, 286, 345, 426, 673
 prototype, 78
create_module(), 801
create_pid_file.c, 1143
createPidFile(), 1143–1144
 code of implementation, 1144
 credentials. *See* process, credentials
 critical section, 631, 635
 Crosby (2003), 794, 1438
 Crosby, S.A., 1438
 CRTSCTS constant, 1303
crypt(), 162–163, 657
 example of use, 165, 425
 prototype, 163
crypt_r(), 658
 CS5 constant, 1303
 CS6 constant, 1303
 CS7 constant, 1303
 CS8 constant, 1303
cs (C shell), 25
 CSIZE constant, 1303
 CSTOPB constant, 1303
ctermid(), 656, 707–708
 prototype, 707
ctime(), 16, 188–189, 198, 657
 diagram, 188
 example of use, 192, 199
 prototype, 188
ctime_r(), 189, 658
curr_time.c, 194
 current working directory, 29, 225, 363–365, 604, 613
 Currie, A.L., xl
currTime(), 193
 code of implementation, 194–195
 prototype, 193
curses library, 14, 1290, 1444
 Cvetkovic, D., xxxix

D

daemon process, 34, 767–774
 creating, 768–771
 ensuring just one instance runs, 1142–1143
 programming guidelines, 771–772
 reinitializing, 391, 772–775
daemon(), 770
daemon_SIGHUP.c, 774
 dangling link, 28, 342, 349, 360
 DARPA (Defense Advanced Research Projects Agency), 1180

da Silva, D., 1444
 data segment, 116
 resource limit on size of, 761
 Datagram Congestion Control Protocol (DCCP), 1286
 data-link layer, 1182
 diagram, 1181
 DATEMSK environment variable, 196
 Davidson, F., xxxix
 Davidson, S., xxxix
 daylight saving time, 187
daylight variable, 198
dbm_clearerr(), 657
dbm_close(), 657
dbm_delete(), 657
dbm_error(), 657
dbm_fetch(), 657
dbm_firstkey(), 657
dbm_nextkey(), 657
dbm_open(), 657
dbm_store(), 657
 DCCP (Datagram Congestion Control Protocol), 1286
 DEAD_PROCESS constant, 820, 821, 822, 826
 deadlock
 mutex, 639
 when locking files, 1128–1129
 when opening FIFOs, 916
 Dean, D., 1438
 Deering, S., 1194
 Defense Advanced Research Projects Agency (DARPA), 1180
 Deitel (2004), 1147, 1438
 Deitel, H.M., 1438
 Deitel, P.J., 1438
delete_module(), 801
demo_clone.c, 603
demo_inotify.c, 382
demo_sched_fifo.c, 1432
demo_SIGFPE.c, 452
demo_sigio.c, 1348
demo_SIGWINCH.c, 1320
demo_timerfd.c, 510
 denial-of-service attack, 793, 920, 1140, 1167, 1438
detached_attr.c, 628
dev_t data type, 64, 280, 281
devfs file system, 253
 device, 252–253
 major ID, 253, 281
 minor ID, 253, 281
 device control operations, 86
 device driver, 252, 1438
 de Weerd, P., xl
 Diamond, D., 1444
diet libc, 47
 Dijkstra (1968), 994, 1438
 Dijkstra, E.W., 989, 1438
 Dilger, A., 1441
DIR data type, 64, 352, 353, 354, 355, 357
 direct I/O, 246–248
direct_read.c, 247
 directory, 27, 282, 339–342
 creating, 350–351
 diagram, 340
 opening, 76
 permissions, 297
 reading contents of, 352–357
 removing, 351, 352
 set-group-ID permission bit, 291
 sticky permission bit, 300
 synchronous updates, 264, 265, 267, 305, 307
 directory stream, 64, 352, 613
 closed on process termination, 533
 file descriptor associated with, 355
dirent structure, 353
 definition, 353
 example of use, 356
dirfd(), 15, 355
 prototype, 355
dirname(), 370–372, 657
 example of use, 371
 prototype, 370
disc_SIGHUP.c, 712
 DISCARD terminal special character, 1296, 1297
 discretionary locking, 1138
 disk drive, 253
 disk partition, 254
 diagram, 255
 disk quotas, 794, 801
display_env.c, 127
Dl_info structure, 866
 definition, 866
dladdr(), 866
 prototype, 866
dlclose(), 860, 861, 866, 876
 example of use, 865
 prototype, 866
dllerror(), 657, 862, 863
 example of use, 865
 prototype, 862
dlopen(), 860–862
 example of use, 865
 prototype, 860
dlsym(), 862–864
 example of use, 865
 prototype, 863
dlusym(), 863

dmalloc (*malloc* debugger), 147

dnotify (directory change notification), 386, 615

DNS (Domain Name System), 1209–1212, 1437

- anonymous root, 1210
- domain name, 1210
- iterative resolution, 1211
- name server, 1210
- recursive resolution, 1211
- root name server, 1211
- round-robin load sharing, 1247
- top-level domain, 1212

Domaigné, L., xxxvii

domain name, 1210

Domain Name System. *See* DNS

domainname command, 230

Döring, G., xxxvii

dotted-decimal notation (IPv4 address), 1186

DragonFly BSD, 8

drand48(), 657

Drepper (2004a), 638, 1438

Drepper (2004b), 857, 868, 1439

Drepper (2007), 748, 1439

Drepper (2009), 795, 1439

Drepper, U., 47, 689, 1438, 1439

DST, 187

DSUSP terminal special character, 1299

DT_DIR constant, 353

DT_FIFO constant, 353

DT_LNK constant, 353

DT_NEEDED tag (ELF), 839

DT_REG constant, 353

DT_RPATH tag (ELF), 853, 854

DT_RUNPATH tag (ELF), 853, 854

DT_SONAME tag (ELF), 840

dumb terminal, 714

dump_utmxx.c, 824

Dunchak, M., xli

dup(), 97, 426, 1425

- prototype*, 97
- RLIMIT_NOFILE resource limit and, 762

dup2(), 97, 426, 899, 900, 1426

- example of use*, 771, 901
- prototype*, 97
- RLIMIT_NOFILE resource limit and, 762

dup3(), 98

- prototype*, 98

Dupont, K., xxxix

dynamic linker, 36, 839

dynamic linking, 839, 840

dynamically allocated storage, 116

dynamically loaded library, 859–867

dynload.c, 865

E

E2BIG error, 565, 943, 991

EACCES error, 77, 312, 564, 702, 928, 952, 1031, 1127

eaccess(), 300

EADDRINUSE error, 1166, 1279

EAGAIN error, 57, 103, 270, 379, 460, 471, 473, 509, 761, 763, 764, 917, 918, 941, 979, 980, 1065, 1073, 1075, 1095, 1127, 1139, 1259, 1260, 1330, 1347, 1367

EAI_ADDRFAMILY constant, 1217

EAI_AGAIN constant, 1217

EAI_BADFLAGS constant, 1217

EAI_FAIL constant, 1217

EAI_FAMILY constant, 1217

EAI_MEMORY constant, 1217

EAI_NODATA constant, 1217

EAI_NONAME constant, 1217, 1219

EAI_OVERFLOW constant, 1217

EAI_SERVICE constant, 1217

EAI_SOCKTYPE constant, 1217

EAI_SYSTEM constant, 1217

EBADF error, 97, 762, 1126, 1334, 1344, 1345

Ebner, R., xl

EBUSY error, 270, 637, 1078, 1396

ECHILD error, 542, 556, 903

ECHO constant, 1303, 1304

- example of use*, 1306, 1310, 1311

ECHOCTL constant, 1303, 1304

ECHOE constant, 1303, 1304

ECHOK constant, 1303, 1304

ECHOKE constant, 1303, 1304

ECHONL constant, 1296, 1303

ECHOPRT constant, 1303

ecvt(), 656, 657

edata variable, 118

- diagram*, 119

EDEADLK error, 636, 1129, 1139, 1431

edge-triggered notification, 1329–1330, 1366–1367

- preventing file-descriptor starvation, 1367

EEXIST error, 76, 315, 345, 349, 350, 924, 932, 938, 969, 999, 1059, 1109, 1357

EF_DUMPCORE environment variable, 52

EFAULT error, 187, 465

EFBIG error, 761

effective group ID, 33, 168, 172, 173, 175, 177, 613

effective user ID, 33, 168, 172, 174, 175, 177, 613

- effect on process capabilities, 806

- EIDRM error, 933, 947, 971, 979
- EINTR error, 418, 442, 443, 486, 489, 941, 944, 979, 1095, 1334, 1339
- EINVAL error, 179, 216, 246, 247, 349, 381, 750, 762, 933, 950, 952, 969, 991, 999, 1000, 1014
- EIO error, 709, 718, 727, 730, 764, 1382, 1388, 1389, 1396
- EISDIR error, 78, 346, 349
- elapsed time, 185
- Electric Fence (*malloc* debugger), 147
- ELF (Executable and Linking Format), 113, 565, 837, 1441
- ELF interpreter, 565
- Elliston, B., 1444
- ELoop error, 77
- EMFILE error, 78, 762
- EMPTY constant, 820
- EMSGSIZE error, 1073, 1075
- ename.c.inc, 58
- encapsulation, in networking protocols, 1182
- encrypt()*, 657
- end* variable, 118
 - diagram*, 119
- endgrent()*, 161, 657
- end-of-file character, 1296, 1297
- end-of-file condition, 30, 70
- endpwent()*, 160–161, 657
 - prototype*, 161
- endspent()*, 161
 - prototype*, 161
- endutxent()*, 657, 821
 - example of use*, 824, 830
 - prototype*, 821
- ENFILE error, 78, 763
- enforcement-mode locking, 1138
- ENODATA error, 315, 316
- ENOENT error, 78, 158, 346, 349, 565, 823, 924, 932, 1059, 1357, 1396, 1429
- ENOEXEC error, 565
- ENOMEM error, 760, 761, 1037
- ENOMSG error, 944
- ENOSPC error, 950, 991, 1014, 1206
- ENOTDIR error, 76, 345, 349, 351, 379
- ENOTEMPTY error, 349
- ENOTTY error, 727, 825, 1292
- env* command, 126
- envargs.c, 566
- environ* variable, 34, 124, 126, 568
 - diagram*, 126
 - example of use*, 127, 566
- environment list, 34, 125–131, 214, 225, 570–571, 612, 791
 - accessing from a program, 126–128
 - diagram*, 126
 - modifying, 128–131
- environment variable, 125
- envp* argument to *main()*, 127
- ENXIO error, 707, 916, 1388
- EOF terminal special character, 1296, 1297, 1305, 1307
- EOL terminal special character, 1296, 1297, 1305, 1307
- EOL2 terminal special character, 1296, 1297, 1305, 1307
- EOVERFLOW error, 106
- EPERM error, 76, 173, 346, 403, 435, 702, 705, 762, 929, 1357, 1435
- ephemeral port, 1189, 1224, 1263
- EPIPE error, 895, 912, 1159, 1256, 1260
- Epoch, 40, 186
- epoll*, 1327, 1355–1367, 1439
 - creating *epoll* instance, 1356
 - duplicated file descriptors and, 1363–1364
 - edge-triggered notification, 1366–1367
 - events, 1359
 - waiting for, 1358–1359
 - interest list, 1355
 - modifying, 1356–1358
 - performance, 1365–1366
 - ready list, 1355
- EPOLL_CLOEXEC constant, 1356
- epoll_create()*, 801, 1355, 1356, 1363
 - example of use*, 1358, 1362
 - prototype*, 1356
 - RLIMIT_NOFILE resource limit and, 762
- epoll_ctl()*, 1356–1358, 1364
 - example of use*, 1358, 1362
 - prototype*, 1356
- EPOLL_CTL_ADD constant, 1357
- EPOLL_CTL_DEL constant, 1357
- EPOLL_CTL_MOD constant, 1357
- epoll_event* structure, 1357, 1358
 - definition*, 1357
 - example of use*, 1362
- epoll_input.c*, 1362
- epoll_pwait()*, 1370
 - interrupted by signal handler, 444
 - interrupted by stop signal, 445
- epoll_wait()*, 1356, 1358–1360, 1364, 1366–1367
 - example of use*, 1362
 - interrupted by signal handler, 444
 - interrupted by stop signal, 445
 - prototype*, 1358
- EPOLLERR constant, 1359
- EPOLLET constant, 1359, 1366
- EPOLLHUP constant, 1359

- EPOLLIN constant, 1359
- EPOLLONESHOT constant, 1359, 1360
- EPOLLOUT constant, 1359
- EPOLLPRI constant, 1359
- EPOLLRDHUP constant, 1359
- ERANGE error, 315, 363, 991
- Eranian, S., 1442
- ERASE terminal special character, 1296, 1297, 1303, 1304, 1305, 1307
- Erickson (2008), 792, 795, 1439
- Erickson, J.M., 1439
- EROFS error, 78
- err_exit()*, 52–53
 - code of implementation, 56
 - prototype, 52
- errExit()*, 52
 - code of implementation, 55
 - prototype, 52
- errExitEN()*, 52–53
 - code of implementation, 56
 - prototype, 52
- errMsg()*, 52
 - code of implementation, 55
 - prototype, 52
- errno* variable, 45, 49, 53, 620, 780
 - in threaded programs, 621
 - use inside signal handler, 427, 556
- error handling, 48–50
- error number, 49
- error_functions.c*, 54
- error_functions.h*, 52
- error-diagnostic functions, 51–58
- ESPIPE error, 83
- ESRCH error, 158, 402, 403, 702
- ESTABLISHED state (TCP), 1269
- etxt* variable, 118
 - diagram, 119
- ethercal* command, 1277
- ETIMEDOUT error, 637, 645, 1077, 1096
- ETXTBSY error, 78, 373, 565
- eidaccess()*, 300
- event (I/O), 1327
- event_flags.c*, 1434
- eventfd()*, 882
- EWOULDBLOCK error, 57, 103, 1119, 1330, 1347, 1367
- example programs, xxxiv, 50–61, 100
- EXDEV error, 349
- exec* shell command, 713
- exec()*, 32, 286, 345, 514, 563–579, 690, 801
 - effect on process attributes, 612–615
 - file descriptors and, 575–578
 - in multithreaded process, 605
 - process capabilities and, 805
 - set-user-ID program and, 169
 - signals and, 578–579
 - threads and, 686
- execl()*, 426, 567–568, 570–571
 - example of use, 571, 583, 587
 - prototype, 567
- execlp()*, 426, 567–568, 570
 - example of use, 570
 - prototype, 567
- execlp()*, 567–569, 570, 575, 589
 - avoid in privileged programs, 788
 - example of use, 570, 901, 1392
 - prototype, 567
- execp.c*, 1430
- Executable and Linking Format (ELF), 113, 565, 837, 1441
- execute permission, 29, 282, 295, 297
- execv()*, 426, 567–568, 570
 - prototype, 567
- execve()*, 32, 426, 514, 563–566, 567–568, 593
 - diagram, 515
 - example of use, 566
 - prototype, 564
- execvp()*, 567–570, 575, 1430
 - avoid in privileged programs, 788
 - prototype, 567
- execvpe()*, 568
- exit handler, 532, 533–537, 615
- exit status, 32, 545
- exit()*, 32, 390, 513, 531–533, 692
 - diagram, 515
 - example of use, 537
 - prototype, 532
 - threads and, 687
- EXIT_FAILURE constant, 532
- exit_group()*, 692
- exit_handlers.c*, 536
- exit_status* structure, 819
 - definition, 819
- EXIT_SUCCESS constant, 532
- expect* command, 1379
- explicit congestion notification (TCP), 1194, 1267, 1439
- export* shell command, 125
- ext2* file system, 234, 255, 257–259
 - i-node flag, 304–308
- ext3* file system, 260
 - i-node flag, 304–308
- ext4* file system, 261, 1441
 - i-node flag, 304–308
- extended attribute, 311–318
 - implementation limits, 314
 - name, 312
 - namespace, 312
 - os2 (JFS)*, 312

- security*, 312, 801
- system*, 312, 321, 327
- trusted*, 312, 316, 801
- user*, 312
- extended file attribute (i-node flag), 304–308
- extended network ID, 1187
 - diagram*, 1187

F

- F_DUPFD constant, 97
 - RLIMIT_NOFILE resource limit and, 762
- F_DUPFD_CLOEXEC constant, 98
- F_GETFD constant, 577
 - example of use*, 578
- F_GETFL constant, 93–94, 96
 - example of use*, 518, 917, 1349
- F_GETLK constant, 1127
 - example of use*, 1131, 1135
- F_GETOWN constant, 1350–1351
- F_GETOWN_EX constant, 1351, 1354, 1355
- F_GETPIPE_SZ constant, 892
- F_GETSIG constant, 1352, 1353
- F_NOTIFY constant, 386, 615
- F_OK constant, 299
- f_owner_ex* structure, 1354, 1355
 - definition*, 1354
- F_OWNER_PGRP constant, 1354
- F_OWNER_PID constant, 1354
- F_OWNER_TID constant, 1355
- F_RDLCK constant, 1125
 - example of use*, 1131
- F_SETFD constant, 577
 - example of use*, 578
- F_SETFL constant, 93–94, 96
 - example of use*, 519, 917, 1347, 1349
- F_SETLEASE constant, 615, 800, 1142
- F_SETLK constant, 1126–1127
 - example of use*, 1131, 1134
- F_SETLKW constant, 673, 1127
 - example of use*, 1131, 1134
- F_SETOWN constant, 1281, 1283, 1347
 - example of use*, 1349
- F_SETOWN_EX constant, 1354
- F_SETPIPE_SZ constant, 891
- F_SETSIG constant, 1281, 1352–1353
- F_UNLCK constant, 1125
 - example of use*, 1131
- F_WRLCK constant, 1125
 - example of use*, 1131
- Fabry, R.S., 1442
- faccessat()*, 365, 426
- fallocate()*, 83
- FALSE constant, 51
- FAM (File Alteration Monitor), 375

- FASYNC constant, 1347
- fatal()*, 54
 - code of implementation*, 56
 - prototype*, 54
- fchdir()*, 364
 - example of use*, 364
 - prototype*, 364
- fchmod()*, 286, 303, 426, 1110
 - prototype*, 303
- fchmodat()*, 365, 426
- fchown()*, 221, 286, 291–293, 426, 1110
 - prototype*, 292
- fchownat()*, 365, 426
- fchroot()*, 368
- fcntl()*, 92–93, 426, 673, 1124, 1134
 - changing signal associated with a file descriptor, 1352–1353
 - duplicating file descriptors, 97–98
 - example of use*, 518, 578, 1131, 1349
 - interrupted by signal handler, 444
 - prototype*, 93
 - retrieving and setting file descriptor flags, 577–578
 - retrieving and setting open file status flags, 93–94
 - setting file descriptor owner, 1347
 - setting pipe capacity, 891–892
- fcvt()*, 656, 657
- FD_CLOEXEC constant, 75, 98, 355, 377, 472, 508, 577, 894, 1110, 1153, 1158, 1175, 1281, 1356
 - example of use*, 578
- FD_CLR(), 1331–1332
 - prototype*, 1332
- FD_ISSET(), 1331–1332
 - example of use*, 1336
 - prototype*, 1332
- fd_set* data type, 64, 1331, 1332, 1344, 1369
- FD_SET(), 1331–1332
 - example of use*, 1335
 - prototype*, 1332
- FD_SETSIZE constant, 1332
- FD_ZERO(), 1331–1332
 - example of use*, 1335
 - prototype*, 1332
- fdatasync()*, 240–241, 242, 244, 426, 673, 1032
 - prototype*, 240
- fdisk* command, 254
- fdopen()*, 248–249, 892, 906
 - prototype*, 248
- fdopendir()*, 15, 353
 - prototype*, 353
- feature test macro, 61–63

- fenableexcept()*, 391
- Fellinger, P., xxxix, xl
- Fenner, B., 1421, 1444
- fexecve()*, 15, 426, 571
 - prototype*, 571
- FF0 constant, 1302
- FF1 constant, 1302
- FFDLY constant, 1302
- fflush()*, 239, 244, 538
 - prototype*, 239
- fg* shell command, 715
 - diagram*, 717
- fgetxattr()*, 315
 - prototype*, 315
- FIBMAP constant, 255
- FIFO, 282, 392, 882, 883, 886, 906–918.
 - See also* pipe
 - creating dual pipeline with *tee(1)*, *diagram*, 908
 - deadlock during open by two processes, *diagram*, 917
 - open()* semantics, 907, 915–916
 - poll()* on, 1342
 - read()* semantics, 917–918
 - select()* on, 1342
 - write()* semantics, 918
- fifo_seqnum.h*, 911
- fifo_seqnum_client.c*, 914
- fifo_seqnum_server.c*, 912, 920
- file, 27
 - appending output to, 92
 - blocks allocated to, 282
 - compression, 306
 - control operations, 92
 - creating, 76
 - creating exclusively, 76, 90–92
 - deleting, 346, 352
 - descriptor. *See* file descriptor
 - holes in, 83, 259, 283
 - lease, 615, 800, 1135, 1142
 - lock. *See* file lock
 - mapping. *See* file mapping
 - maximum size of, 258
 - offset. *See* file offset
 - on-disk structure
 - diagram*, 258
 - opening, 72–79
 - optimal I/O block size, 283
 - randomly accessing, 81–86
 - reading, 79–80
 - renaming, 348–349
 - resource limit on size, 761
 - retrieving metadata, 279–285
 - sharing of open file by parent and child, 517–520
 - size, 282
 - synchronous updates, 264, 267, 307
 - temporary, 108–109
 - timestamps. *See* file timestamps
 - truncating, 103
 - truncation on *open()*, 77
 - type, 27, 95, 256, 281
 - diagram*, 281
 - writing, 80
- file access mode, 72, 75, 93, 95
- File Alteration Monitor (FAM), 375
- file capabilities, 799, 803–804, 1440
 - effective, 799, 802
 - inheritable, 799, 803
 - permitted, 799, 802
- file creation flags, 75
- file descriptor, 30, 69, 94, 530, 603, 613
 - closed on process termination, 533
 - diagram*, 95, 520
 - duplicating, 96–98
 - multiplexing, 1327, 1330–1346
 - passing via UNIX domain socket, 1284
 - for POSIX shared memory object, 1108
 - ready, 1327
 - refers to same open file in forked child, 517
 - relationship to open file, 94–96
 - resource limit on number of open, 762
- file descriptor set, 64, 1331
- file hole, 83, 259, 283
- file I/O, 29
 - advising kernel about access patterns, 244
 - benchmarking, 236
 - buffering, 233–250
 - diagram*, 244
 - large files, 104–107
 - performing at a specified offset, 98–99
 - scatter-gather I/O, 99–102
 - speed, 235, 236, 242
- file lease, 615, 800, 1135, 1142
- file lock, 533, 881, 882, 884, 886, 1117–1144
 - advisory, 1119, 1137
 - comparison of semantics of *flock()* and *fcntl()*, 1136–1137
 - deadlock, 1128–1129
 - with *fcntl()*, 614, 1124–1137
 - semantics of lock inheritance and release, 1136–1137
 - with *flock()*, 614, 1119–1124
 - limitations, 1123–1124
 - semantics of lock inheritance and release, 1122–1123
 - limits, 1135–1136

- LinuxThreads nonconformance, 691
- mandatory, 265, 293, 1119, 1137–1140
- priority of queued lock requests, 1137
- speed, 1135–1136
- starvation, 1137
- file mapping, 35, 882, 886, 1017, 1024–1031
 - diagram*, 1025
 - private, 1018, 1024–1025
 - shared, 1019, 1025–1029
- file mode creation mask (umask), 301–303, 328, 351, 604, 613, 790, 907, 923, 1060, 1065, 1091, 1110, 1174
- file offset, 81, 94, 613
 - changing, 81
- file ownership, 29, 281, 291–294, 800
 - changing, 291–294
 - of new files, 291
- file permissions, 29, 281, 282, 294–299, 800
 - changing, 303–304
 - diagram*, 281
 - permission-checking algorithm, 297–299
- file status flags, open, 75, 93–94, 95, 96, 518, 613
- file system, 22, 254–256
 - busy, 270
 - diagram*, 27, 255
 - mount point, 261
 - diagram*, 262
 - mounting, 264–269
 - at multiple mount points, 271
 - retrieving information about mounted, 276–277
 - stacking multiple mounts, 271–272
 - unmounting, 269–270
- file timestamps, 257, 283, 285–287
 - changing, 286, 287–290
 - last access time, 74, 76–77, 257, 264, 265, 266, 267, 283, 285, 286, 287, 289, 305, 306
 - last modification time, 257, 283, 285, 286, 287
 - last status change time, 257, 283, 285, 286
 - nanosecond precision, 287
- file tree walking, 358–363
- file_perms.c, 296
- file_perms.h, 296
- file_type_stats.c, 1429
- file-based mapping. *See* file mapping
- filename, 28, 341
 - maximum length, 214, 340
- fileno()*, 248
 - prototype*, 248
- filePermStr()*, 295–296
 - code of implementation*, 296
 - example of use*, 284, 303
- file-system group ID, 171–172, 178, 298, 615
- Filesystem in Userspace (FUSE), 255, 267
- file-system user ID, 171–172, 178, 615, 800
 - effect on process capabilities, 807
- filter, 31, 899
- FIN control bit (TCP), 1267
- FIN_WAIT1 state (TCP), 1269
- FIN_WAIT2 state (TCP), 1269
- finger* command, 154
- FIOCLEX constant, 577
- FIOGETOWN constant, 1350
- FIONCLEX constant, 577
- FIONREAD constant, 381, 892, 1153, 1291
- FIOSETOWN constant, 1350
- FIPS 151-1, 12
- FIPS 151-2, 12
- Fletcher, G., xl
- flistxattr()*, 316
 - prototype*, 316
- floating-point environment, 615, 620
- floating-point exception (error message). *See* SIGFPE signal
- flock* structure, 1124–1126
 - definition*, 1124
 - example of use*, 1130
- flock()*, 1119–1122, 1147, 1435
 - example of use*, 1121
 - interrupted by signal handler, 444
 - prototype*, 1119
- flow control (TCP), 1192
- Floyd (1994), 1267, 1439
- Floyd, S., 1194, 1439
- FLUSHO constant, 1303
- footprint.c, 522
- FOPEN_MAX constant, 215
- fopen64()*, 105
- For portability* comment in function prototypes, 67
- foreground process group, 39, 700, 708
 - diagram*, 701, 717
 - signaled on terminal window size change, 1319–1320
 - terminal-generated signals and, 1290
- Forero Cuervo, A., xl
- fork bomb, 793
- fork handler, 609, 687
- fork()*, 31, 426, 513, 515–522, 589, 609, 690, 1430
 - copy-on-write semantics, 521
 - diagram*, 515
 - effect on process attributes, 612–615

fork(), *continued*
example of use, 516, 517, 519, 526, 543, 554, 582, 587, 770, 900, 1387
file descriptors and, 96, 517–520
glibc wrapper invokes *clone()*, 609
memory semantics, 520–521
prototype, 516
RLIMIT_NPROC resource limit and, 763
scheduling of parent and child after, 525
speed, 610
stdio buffers and, 537–538
threads and, 686
fork_file_sharing.c, 518
fork_sig_sync.c, 528
fork_stdio_buf.c, 537
fork_whos_on_first.c, 526
format-string attack, 780
Fox, B., 25
fpathconf(), 217–218, 425, 426
example of use, 218
prototype, 217
FPE_FLTDIV constant, 441
FPE_FLTINV constant, 441
FPE_FLTUVF constant, 441
FPE_FLTRES constant, 441
FPE_FLTUND constant, 441
FPE_INTDIV constant, 441
FPE_INTOVF constant, 441
FPE_SUB constant, 441
FQDN (fully qualified domain name), 1210
fragmentation of free disk space, 257
Franke (2002), 638, 1439
Franke, H., 1439
Free Software Foundation, 5
free(), 140–142, 144, 423
example of use, 143
implementation, 144–146
diagram, 145
prototype, 141
free_and_sbrk.c, 142
freeaddrinfo(), 1217
example of use, 1222
prototype, 1217
FreeBSD, 7, 1442
fremovexattr(), 286, 316
prototype, 316
Frisch (2002), 616, 818, 1439
Frisch, A., 1439
FS_APPEND_FL constant, 305, 306
FS_COMPR_FL constant, 305, 306
FS_DIRSYNC_FL constant, 265, 305, 306, 307
FS_IMMUTABLE_FL constant, 305, 306, 307
FS_IOC_GETFLAGS constant, 308
FS_IOC_SETFLAGS constant, 308
FS_JOURNAL_DATA_FL constant, 305
FS_JOURNAL_FL constant, 306
FS_NOATIME_FL constant, 77, 265, 305, 306
FS_NODUMP_FL constant, 305, 307
FS_NOTAIL_FL constant, 305, 307
FS_SECRM_FL constant, 305, 307
FS_SYNC_FL constant, 305, 307
FS_TOPDIR_FL constant, 305, 307
FS_UNRM_FL constant, 305, 307
fsblkcnt_t data type, 64, 276
fsck command, 260, 263
fsetxattr(), 286, 314–315
prototype, 314
fsfilent_t data type, 64, 276
fstab file format, 263
fstat(), 279–283, 426, 907, 1110
example of use, 1023, 1113
prototype, 279
fstatat(), 365, 426
fstatfs(), 277
fstatvfs(), 276–277
prototype, 276
fsync(), 240–241, 242, 244, 265, 426, 673, 1240
prototype, 240
fstok(), 925–927, 936
prototype, 925
example of use, 930
ftruncate(), 103, 286, 426, 800, 1139
example of use, 1111, 1112
prototype, 103
use with POSIX shared memory object, 1110
fts_open(), 358
FTW structure, 360
definition, 360
example of use, 360
ftw(), 16, 358, 657
FTW_ACTIONRETRVAL constant, 362
FTW_CHDIR constant, 359
FTW_CONTINUE constant, 362
FTW_D constant, 359
FTW_DEPTH constant, 359
FTW_DNR constant, 359
FTW_DP constant, 359
FTW_F constant, 359
FTW_MOUNT constant, 359
FTW_NS constant, 359
FTW_PHYS constant, 359, 360
FTW_SKIP_SIBLINGS constant, 363
FTW_SKIP_SUBTREE constant, 363
FTW_SL constant, 359, 360
FTW_SLN constant, 360
FTW_STOP constant, 363
fully qualified domain name (FQDN), 1210

FUSE (Filesystem in Userspace), 255, 267
fuser command, 342
 futex (fast user space mutex), 605, 607, 638, 1438, 1439
futex(), 638, 1090
 interrupted by signal, 444
 interrupted by stop signal, 445
 FUTEX_WAIT constant, 444, 445
futimens(), 15, 286, 426
futimes(), 15, 286, 288–289, 426
 prototype, 289

G

gai_strerror(), 1217–1218
 prototype, 1218
 Gallmeister (1995), 222, 512, 751, 1087, 1327, 1439
 Gallmeister, B.O., 1439
 Gamin, 375
 Gammo (2004), 1374, 1439
 Gammo, L., 1439
 Gancarz (2003), 1422, 1439
 Gancarz, M., 1439
 Garfinkel (2003), 20, 795, 1439
 Garfinkel, S., 1439
 gather output, 102
gcore (*gdb*) command, 448, 1430
gcvt(), 656, 657
gdb program, 1442
 General Public License (GPL), 5
get_current_dir_name(), 364
get_num.c, 59
get_num.h, 59
get_robust_list(), 801
get_thread_area(), 692
getaddrinfo(), 1205, 1213–1217
 diagram, 1215
 example of use, 1221, 1224, 1228, 1229
 prototype, 1213
 GETALL constant, 971, 972
 example of use, 974
getc_unlocked(), 657
getchar_unlocked(), 657
getconf command, 215
getcontext(), 442
getcwd(), 363–364
 prototype, 363
getdate(), 196, 657
getdate_r(), 196
getdents(), 352
getdomainname(), 230
getdtablesize(), 215
getegid(), 172–173, 426
 prototype, 173
getenv(), 127–128, 657
 example of use, 1392
 prototype, 127
geteuid(), 172–173, 426
 prototype, 173
getfacl command, 325
getfattr command, 312
getfsent(), 263
getgid(), 172–173, 426
 prototype, 173
getgrent(), 161, 657
getgrgid(), 158–159, 657
 example of use, 160
 prototype, 158
getgrgid_r(), 158, 658
getgrnam(), 158–159, 657
 example of use, 160
 prototype, 158
getgrnam_r(), 158, 658
getgroups(), 179, 426
 example of use, 183
 prototype, 179
gethostbyaddr(), 16, 656, 657, 1205, 1231–1232
 prototype, 1231
gethostbyname(), 16, 656, 657, 1205, 1231–1232,
 example of use, 1233
 prototype, 1231
gethostbyname_r(), 658
gethostent(), 657
gethostname(), 230
getInIt(), 58–59
 code of implementation, 60–61
 prototype, 58
getitimer(), 16, 481
 example of use, 483
 prototype, 481
getlogin(), 657, 825, 826
 prototype, 825
getlogin_r(), 658, 825
getLong(), 58–59
 code of implementation, 60
 prototype, 58
getmntent(), 263
getmsg(), 673
getnameinfo(), 1205, 1218–1219
 example of use, 1230
 prototype, 1218
 GETNCNT constant, 972
 example of use, 974
getnetbyaddr(), 657
getnetbyname(), 657
getnetent(), 657

getopt(), 657, 1405–1411
 example of use, 1409
 prototype, 1406
getopt_long(), 1411
getpagesize(), 215
getpass(), 164, 166
 example of use, 165
 prototype, 164
getpeername(), 426, 1263–1264
 example of use, 1265
 prototype, 1263
getpgid(), 704
getpgrp(), 426, 701–702, 704
 example of use, 706, 720
 prototype, 701
 GETPID constant, 972
 example of use, 974
getpid(), 114, 426, 604, 690
 prototype, 114
getpmsg(), 673
getppid(), 115, 426, 553, 608, 690
 prototype, 115
getpriority(), 735–736
 example of use, 737
 prototype, 735
getprotobyname(), 657
getprotobynumber(), 657
getprotoent(), 657
getpwent(), 161, 657
 prototype, 161
getpwnam(), 157–158, 657
 example of use, 160, 165
 prototype, 157
getpwnam_r(), 158, 658
getpwuid(), 157–158, 657
 example of use, 159
 prototype, 157
getpwuid_r(), 158, 658
getresgid(), 176–177
 prototype, 177
getresuid(), 176–177
 example of use, 182
 prototype, 177
getrlimit(), 755–757, 759
 example of use, 758
 prototype, 756
getrusage(), 560, 619, 691, 694,
 753–755, 765
 prototype, 753
getservbyname(), 657, 1205, 1234
 prototype, 1234
getservbyname_r(), 658
getservbyport(), 657, 1205, 1234–1235
 prototype, 1234
getservent(), 657
getsid(), 704–705
 example of use, 706, 720
 prototype, 704
getsocname(), 426, 1263–1264
 example of use, 1265
 prototype, 1263
getsockopt(), 426, 1278–1279
 prototype, 1278
getspent(), 161–162
 prototype, 161
getspnam(), 161–162
 example of use, 165
 prototype, 161
gettext API, 202
gettid(), 226, 497, 605, 625, 749, 1355
gettimeofday(), 16, 186–187
 diagram, 188
 example of use, 192, 482, 490
 prototype, 186
getty command, 820
getuid(), 172–173, 426
 prototype, 173
getutent_r(), 658, 823
getutid_r(), 658, 823
getutline_r(), 658, 823
getutxent(), 657, 822
 example of use, 824
 prototype, 822
getutxid(), 657, 822–823
 prototype, 822
getutxline(), 657, 822–823
 prototype, 822
 GETVAL constant, 971, 972
getwd(), 364
getxattr(), 315, 329, 345
 example of use, 318
 prototype, 315
 GETZCNT constant, 972
 example of use, 974
 GID (group ID), 26, 64, 153, 156
gid_t data type, 64, 157, 158, 159, 173,
 174, 175, 177, 178, 179, 280, 292,
 330, 927
 Gilligan, S., 1194
glibc. *See* GNU C library
 globbing, 903
 Gloger, W., xxxvii
 Gmelch, T., xl
gmtime(), 189, 657
 diagram, 188
 example of use, 192
 prototype, 189
gmtime_r(), 189, 658
 GNU C library, 47–48
 determining version, 47–48
 manual, 1421

GNU project, 5–6, 1422
 GNU/Linux, 6
gnu_get_libc_version(), 48
 prototype, 48
 Göllersch, N., xxxix
 Gont (2008), 1235, 1439
 Gont (2009a), 1235, 1439
 Gont (2009b), 1283, 1439
 Gont, F., xxxvii, 1439
 Goodheart (1994), 20, 24, 250, 278, 419,
 530, 936, 1044, 1440
 Goodheart, B., 1440
 Goralski (2009), 1235, 1440
 Goralski, W., 1440
 Gorman (2004), 138, 1440
 Gorman, M., xxxix, 1440
 GPL (General Public License), 5
grantpt(), 1380–1381
 example of use, 1384
 prototype, 1381
 Gratzl, C., xxxix, xl
 group file, 155–156
 retrieving records from, 158–160
 group ID, 26, 64, 153, 156
 group structure, 159
 definition, 159
 example of use, 160
groupIdFromName(), 159
 code of implementation, 160
groupNameFromId(), 159
 code of implementation, 160
 groups command, 155
 Grünbacher (2003), 337, 1440
 Grünbacher, A., xxxviii, 337, 1440
 Gutmann (1996), 307, 1440
 Gutmann, P., 1440

H

h_errno variable, 1231
 Haig, B., xl
 Hallyn (2007), 814, 1440
 Hallyn, S., xxxix, 1437, 1440
 handle, 331
 Handley, M., 1194
 handling_SIGTSTP.c, 724
 Harbison (2002), xxxii, 30, 1440
 Harbison, S.P., 1440
 hard link. *See* link
 hard realtime, 738
 HARD_MSGMAX constant, 1086
 Hartinger, M., xxxix, xl
 Hauben, R., 20
hcreate(), 657
hdestroy(), 657

heap, 31, 116, 612
 Heasman, J., 1437
 Hellwig, C., xxxviii
 Henderson, R., xxxix
 Herbert (2004), 1235, 1440
 Herbert, T.F., 1440
herror(), 1232–1233
 prototype, 1233
 hex-string notation (IPv6 address), 1188
 high-resolution timers, 485
 Hinden, R., 1194
 Hoffman, R., xli
 home directory, 26, 154
 HOME environment variable, 34, 154
 host byte order, 1198
 host ID, 1186
 hostent structure, 1232
 definition, 1232
 example of use, 1233
 hostname, 1204
 canonical, 1210, 1216
 hostname command, 230
 Howard, M., xl
 HP-UX, 5
hsearch(), 657
hstrerror(), 1232–1233
 prototype, 1233
htonl(), 1199
 prototype, 1199
htons(), 1199
 prototype, 1199
 Hubička (2003), 837, 1440
 Hubička, J., 1440
 huge page, 999
 HUPCL constant, 1303
 HURD, 6, 1443
 HZ constant, 205

I

i_fcntl_locking.c, 1130
i6d_ucase.h, 1207
i6d_ucase_cl.c, 1209
i6d_ucase_sv.c, 1208
 I18N, 200
 IA-64, 10, 1442
 IANA (Internet Assigned Numbers
 Authority), 1189
 ICANON constant, 1296, 1297, 1303,
 1305, 1307
 example of use, 1310, 1311
 ICMP (Internet Control Message
 Protocol), 1181
 ICRNL constant, 1296, 1297, 1298, 1302
 example of use, 1310, 1311

id_echo.h, 1240
 id_echo_cl.c, 1242
 id_echo_sv.c, 1241
 id_t data type, 64, 550, 735
 idshow.c, 182
 IEEE (Institute of Electrical and Electronic Engineers), 11
 IETF (Internet Engineering Task Force), 1193
 IEXTEN constant, 1296, 1297, 1298, 1299, 1303, 1305, 1307
 example of use, 1311
 IF5 environment variable, 581, 791
 IGMP (Internet Group Management Protocol), 1181
 IGNBRK constant, 1302, 1304
 example of use, 1311
 IGNCR constant, 1296, 1297, 1302
 example of use, 1311
 ignore_pending_sig.c, 1429
 IGMPAR constant, 1302, 1305
 ILL_BADSTK constant, 441
 ILL_COPROC constant, 441
 ILL_ILLADR constant, 441
 ILL_ILLOPC constant, 441
 ILL_ILLOPN constant, 441
 ILL_ILLTRP constant, 441
 ILL_PRVOPC constant, 441
 ILL_PRVREG constant, 441
 IMAXBEL constant, 1302, 1305
 IN_ACCESS constant, 378
 in_addr structure, 1202, 1231, 1232
 in_addr_t data type, 64, 1202
 IN_ALL_EVENTS constant, 378
 IN_ATTRIB constant, 378, 379
 IN_CLOEXEC constant, 377
 IN_CLOSE constant, 378
 IN_CLOSE_NOWRITE constant, 378
 IN_CLOSE_WRITE constant, 378
 IN_CREATE constant, 378
 IN_DELETE constant, 378, 379
 IN_DELETE_SELF constant, 378, 379
 IN_DONT_FOLLOW constant, 378, 379
 IN_IGNORED constant, 378, 380, 381
 IN_ISDIR constant, 378, 380
 IN_MASK_ADD constant, 378, 379
 IN_MODIFY constant, 378
 IN_MOVE constant, 378
 IN_MOVE_SELF constant, 378, 379
 IN_MOVED_FROM constant, 378, 379, 381
 IN_MOVED_TO constant, 378, 379, 381
 IN_NONBLOCK constant, 377
 IN_ONESHOT constant, 378, 379, 380
 IN_ONLYDIR constant, 378, 379
 IN_OPEN constant, 378
 in_port_t data type, 64, 1202, 1203
 IN_Q_OVERFLOW constant, 378, 385
 IN_UNMOUNT constant, 378, 381
 in6_addr structure, 1202, 1203, 1232
 in6addr_any variable, 1203
 IN6ADDR_ANY_INIT constant, 1203
 in6addr_loopback variable, 1203
 IN6ADDR_LOOPBACK_INIT constant, 1203
 INADDR_ANY constant, 1187
 INADDR_LOOPBACK constant, 1187
 INET_ADDRSTRLEN constant, 1206
 inet_aton(), 1204, 1230–1231
 prototype, 1231
 inet_ntoa(), 657, 1204, 1231
 prototype, 1231
 inet_ntop(), 1205, 1206
 example of use, 1208, 1234
 prototype, 1206
 inet_pton(), 1205, 1206
 example of use, 1209
 prototype, 1206
 inet_sockets.c, 1228
 inet_sockets.h, 1226
 INET6_ADDRSTRLEN constant, 1206
 inetAddressStr(), 1227
 code of implementation, 1230
 example of use, 1265
 prototype, 1227
 inetBind(), 1227
 code of implementation, 1230
 example of use, 1241
 prototype, 1227
 inetConnect(), 1226
 code of implementation, 1228
 example of use, 1242, 1258, 1265
 prototype, 1226
 inetd (Internet superserver daemon), 768, 1247–1251
 inetListen(), 1226–1227
 code of implementation, 1230
 example of use, 1245, 1265
 prototype, 1226
 info documents, 1421
 init process, 33, 115, 402, 768, 805, 815, 820
 adopts orphaned processes, 553
 cleans up utmp file during system boot, 826
 sends SIGTERM to children on system shutdown, 772
 sent SIGPWR on power failure, 392
 signals and, 402
 updates login accounting files, 820
 init_module(), 801
 INIT_PROCESS constant, 820, 821, 822

- initgroups()*, 179–180, 800
 - prototype*, 179
- initial thread, 622
- initialized data segment, 116, 117, 118, 1019, 1025
- initSemAvailable()*, 989–990
 - code of implementation*, 990
 - example of use*, 1004
- initSemInUse()*, 989–990
 - code of implementation*, 990
 - example of use*, 1004
- INLCR constant, 1296, *diagram*, 878
 - example of use*, 1311
- ino_t* data type, 64, 280, 353
- i-node, 95, 256–259
 - diagram*, 95, 258, 340
- i-node flag, 304–308
- i-node number, 64, 256, 281, 341
- i-node table, 256, 340
- inotify* (file system event notification)
 - read()* interrupted by signal handler, 444
 - read()* interrupted by stop signal, 445
- inotify* (notification of file-system events), 375–385
 - inotify_add_watch()*, 376, 377
 - example of use*, 383
 - prototype*, 377
 - inotify_event* structure, 379–381
 - definition*, 379
 - diagram*, 380
 - example of use*, 382
 - inotify_init()*, 376–377
 - example of use*, 383
 - prototype*, 376
 - inotify_init1()*, 377
 - inotify_rm_watch()*, 376, 378
 - prototype*, 378
- INPCK constant, 1302, 1305
 - example of use*, 1311
- Institute of Electrical and Electronic Engineers (IEEE), 11
- int32_t* data type, 472, 593, 819
- International Standards Organization (ISO), 11
- internationalization, 200
- internet, 1179
- Internet Assigned Numbers Authority (IANA), 1189
- Internet Control Message Protocol (ICMP), 1181
- Internet Engineering Task Force (IETF), 1193
- Internet Group Management Protocol (IGMP), 1181
- Internet protocol (IP). *See* IP
- Internet Society, 1193
- Internet superserver daemon (*inetd*), 768, 1247–1251
- Internet Systems Consortium, 1210
- interpreter, 572
- interpreter script, 572–575
- interprocess communication (IPC), 37, 877–887
 - performance, 887
 - persistence of IPC objects, 886
 - taxonomy of facilities, *diagram*, 878
- interrupt* character, 392, 1296, 1297
- interruptible sleep state, 451
- interval timer, 479–485, 614
 - scheduling and accuracy, 485
- intmax_t* data type, 66
- intquit.c*, 401
- INTR terminal special character, 1296, 1297, 1303, 1305
- invalid memory reference, 393
- I/O
 - asynchronous I/O, POSIX, 613, 1327, 1347
 - buffering. *See* buffering of file I/O
 - direct, 246–248
 - event, 1327
 - file. *See* file I/O
 - large file, 76, 104–107
 - memory-mapped, 1019, 1026–1027
 - multiplexed, 1327, 1330–1346
 - nonblocking, 77, 103–104, 915–917, 1326, 1330
 - signal-driven, 75, 95, 1327, 1346–1355, 1367
 - synchronous, 241–243
- io_getevents()*, interrupted by signal handler, 444
- ioctl()*, 72, 86, 308, 1293, 1319
 - example of use*, 1320, 1387
 - interrupted by signal handler, 443
 - prototype*, 86
- ioperm()*, 801
- iopl()*, 801
- IOPRIO_CLASS_RT constant, 801
- ioprio_set()*, 801
- IOV_MAX constant, 100
- iovec* structure, 99–100, 102
 - definition*, 99
 - example of use*, 101
- IP (Internet protocol), 1184–1186, 1193, 1439
 - address, 1186–1188
 - datagram, 1184
 - duplication of, 1185

IP (Internet protocol), *continued*
diagram, 1181
fragmentation, 1185, 1440
header, 1185
checksum, 1185
minimum reassembly buffer size, 1185
unreliability, 1185

IPC. *See* interprocess communication

ipc(), 922

IPC_CREAT constant, 924, 925, 932, 938, 969, 998
example of use, 939

IPC_EXCL constant, 924, 925, 928, 932, 938, 969, 999
example of use, 940

IPC_INFO constant, 936, 951, 992, 1015

IPC_NOWAIT constant, 941, 943, 979
example of use, 942, 946, 983

ipc_perm structure, 927–928, 948, 972, 1012
definition, 927

IPC_PRIVATE constant, 925, 928
example of use, 939, 960

IPC_RMID constant, 801, 924, 929, 947, 971, 1011
example of use, 948

IPC_SET constant, 801, 927, 928, 929, 947, 948, 949, 971, 973, 1011, 1013
example of use, 927, 950

IPC_STAT constant, 927, 929, 947, 971, 1011
example of use, 927, 950, 974, 975

IPCMNI constant, 951, 992, 1015

ipcrm command, 934

ipcs command, 934, 952

IPPROTO_SCTP constant, 1286

IPv4, 1184
address, 1186–1187
loopback address, 1187
socket address, 1202
wildcard address, 1187

IPv4-mapped IPv6 address, 1188
diagram, 1188

IPv5, 1184

IPv6, 1184, 1194
address, 1188
loopback address, 1188, 1203
socket address, 1202
wildcard address, 1188, 1203

IS_ADDR_STR_LEN constant, 1227

is_echo_cl.c, 1258, 1287

is_echo_inetd_sv.c, 1251

is_echo_sv.c, 1244, 1252

is_echo_v2_sv.c, 1435

is_seqnum.h, 1220

is_seqnum_cl.c, 1224

is_seqnum_sv.c, 1221

is_seqnum_v2.h, 1435

is_seqnum_v2_cl.c, 1435

is_seqnum_v2_sv.c, 1435

isalpha(), 202

isatty(), 1321
example of use, 720
prototype, 1321

ISIG constant, 1296, 1297, 1298, 1299, 1303
example of use, 1310, 1311

ISO (International Standards Organization), 11

ISO/IEC 9899:1990, 11

ISO/IEC 9899:1999, 11

ISO/IEC 9945:2002, 13

ISO/IEC 9945-1:1990, 11

ISO/IEC 9945-1:1996, 12

ISO/IEC 9945-2:1993, 12

ISTRIP constant, 1302
example of use, 1311

iterative resolution (DNS), 1211

iterative server, 912, 1239–1242

ITIMER_PROF constant, 480

ITIMER_REAL constant, 480
example of use, 484

ITIMER_VIRTUAL constant, 480

itimerspec structure, 498, 499, 508, 509
definition, 498

itimerspec_from_str.c, 502

itimerspecFromStr(), 502
code of implementation, 502–503

itimerval structure, 480, 481
definition, 480

IUCLC constant, 1302, 1303, 1305

IUTF8 constant, 1302, 1305

IXANY constant, 1299, 1302

IXOFF constant, 1296, 1298, 1299, 1302

IXON constant, 1296, 1298, 1302
example of use, 1311

J

Jacobsen, V., 1194

Jaeger, A., xxxviii

jail() (BSD), 368

Java Native Interface (JNI), 837, 1441

JFS file system, 261
i-node flag, 304–308

jiffy, 205–206

Jinmei, T., 1194

JNI (Java Native Interface), 837, 1441

job. *See* process group

job control, 39, 221, 714–725
diagram, 717
implementation, 717–718
shell commands, 714–717

- job_mon.c, 719
- job-control signal, 717
 - handling within applications, 722–725
- jobs shell command, 715
- Johnson (2005), 1440
- Johnson, M.K., 1440
- Johnson, R., 1438
- Johnson, S., 4
- Jolitz, L.G., 7
- Jolitz, W.F., 7
- Jones, R., xxxviii
- Josey (2004), 20, 222, 1440
- Josey, A., xxxix, 1440
- journaling file system, 260–261
- Joy, W.N., 4, 25, 1442
- jumbogram, 1185

K

- K&R C, 10
- Kahabka, T., xl
- Kara, J., xl
- Kegel, D., 1374
- Kegel, K., xxxix
- Kent (1987), 1186, 1440
- Kent, A., 1440
- kernel, 21
 - configuration, 1417
 - source code, 1424
- kernel mode, 23, 44
- kernel scheduling entity (KSE), 603, 687
- kernel space, 23
- kernel stack, 44, 122
- kernel thread, 241, 608, 768
- Kernighan (1988), xxxii, 10, 11, 30, 1440
- Kernighan, B.W., 1437, 1440
- kexec_load(), 801
- key_t data type, 64, 925, 927, 938, 969, 998
- KEYCTL_CHOWN constant, 801
- KEYCTL_SETPERM constant, 801
- KILL terminal special character, 1296, 1298, 1303, 1304, 1305, 1307
- kill(), 401–403, 426, 439, 441, 458, 800
 - example of use, 405, 413
 - prototype, 402
- killable sleep state, 451
- killpg(), 405, 458
 - prototype, 405
- Kirkwood, M., 1439
- Kleen, A., xxxviii
- Kleikamp, D., xl
- klogctl(), 776
- klogd daemon, 776
 - diagram, 775
- Kopparapu (2002), 1247, 1440

- Kopparapu, C., 1440
- Korn shell (*ksh*), 25
- Korn, D., 25
- Korrel, K., xl
- Kozierok (2005), 1235, 1441
- Kozierok, C.M., 1441
- kqueue API (BSD), 375, 1328, 1441
- Kroah-Hartman (2003), 253, 1441
- Kroah-Hartman, G., 1438, 1441
- KSE (kernel scheduling entity), 603, 687
- ksh (Korn shell), 25
- Kumar (2008), 307, 1441
- Kumar, A., 1441
- kupdated kernel thread, 241, 1032
- Kuznetsov, A., 1443

L

- L_ctermid constant, 708
- L_INCR constant, 82
- L_SET constant, 82
- L_XTND constant, 82
- l64a(), 657
- Landers, M., xxxviii, xl
- LANG environment variable, 203
- large file I/O, 76, 104–107
- Large File Summit (LFS), 104
- large_file.c, 105
- last access time, file timestamp, 74, 76–77, 257, 264, 265, 266, 267, 283, 285, 286, 287, 289, 305, 306
- last command, 817
- last modification time, file timestamp, 257, 283, 285, 286, 287
- last status change time, file timestamp, 257, 283, 285, 286
- LAST_ACK state (TCP), 1270
- lastcomm command, 591
- lastlog command, 830
- lastlog file, 830
 - example of use, 831
- lastlog structure, 830
 - definition, 830
 - example of use, 831
- Lawyer, D., 1322
- lazy swap reservation, 1038
- LC_ADDRESS locale category, 202
- LC_ALL environment variable, 203
- LC_ALL locale category, 203
- LC_COLLATE environment variable, 203
- LC_COLLATE locale category, 202
- LC_CTYPE environment variable, 203
- LC_CTYPE locale category, 202
- LC_IDENTIFICATION locale category, 202
- LC_MEASUREMENT locale category, 202
- LC_MESSAGES environment variable, 203

LC_MESSAGES locale category, 202
 LC_MONETARY environment variable, 203
 LC_MONETARY locale category, 202
 LC_NAME locale category, 202
 LC_NUMERIC environment variable, 203
 LC_NUMERIC locale category, 202
 LC_PAPER locale category, 202
 LC_TELEPHONE locale category, 202
 LC_TIME environment variable, 203
 LC_TIME locale category, 202
lchown(), 286, 292–293, 345
 prototype, 292
ld command, 833
 LD_ASSUME_KERNEL environment variable, 695
 LD_BIND_NOW environment variable, 861
 LD_DEBUG environment variable, 874–875
 LD_DEBUG_OUTPUT environment variable, 875
 LD_LIBRARY_PATH environment variable,
 840, 853, 854
 LD_PRELOAD environment variable, 873
 LD_RUN_PATH environment variable, 851
ldconfig command, 848–849
ldd command, 843
 lease, file, 615, 800, 1135, 1142
 least privilege, 784
 Leffler, S.J., 1442
 LEGACY (SUSv3 specification), 15
 Lemon (2001), 1328, 1441
 Lemon (2002), 1185, 1441
 Lemon, J., 1441
 Leroy, X., 689
 level-triggered notification, 1329–1330
 Levine (2000), 857, 1441
 Levine, J., 1441
 Lewine (1991), 222, 1441
 Lewine, D., 1441
 Lezcano, D., 1437
 LFS (Large File Summit), 104
lgamma(), 657
lgammaf(), 657
lgammal(), 657
lgetxattr(), 315, 345
 prototype, 315
 Liang (1999), 837, 1441
 Liang, S., 1441
libcap package, 807
libcap-ng package, 808
 Libenzi, D., xxxix
 Libes (1989), 20, 1441
 Libes, D., 1441
libevent library, 1328
 library function, 46
 error handling, 50
Libtool program, 857
limit C shell command, 448
 Lindner, F., 1437
 link, 27, 257, 339–342
 creating, 344–346
 diagram, 343
 removing, 346–348
 link count (file), 281, 341
 link editing, 840
link(), 286, 344–346, 426, 1145
 prototype, 344
linkat(), 365, 426
 linker, 833, 1441
 linking, 840
 Linux
 distributions, 10
 hardware ports, 10
 history, 5–10, 1443
 kernel, 6–7
 mailing list, 1423
 version numbering, 8–9
 programming-related newsgroups, 1423
 standards conformance, 18
 Linux Documentation Project, 1422
 Linux Foundation, 18
 Linux Standard Base (LSB), 19
 LinuxThreads, 457, 592, 603, 604, 609,
 687, 688, 689–692, 695
 Pthreads nonconformances, 690
 Linux/x86-32, 5
 Lions (1996), 24, 1441
 Lions, J., 1441
list_files.c, 356, 373
list_files_readdir_r.c, 1429
 LISTEN state (TCP), 1269
listen(), 426, 1152, 1156–1157
 diagram, 1156
 example of use, 1168, 1222, 1229
 prototype, 1156
listxattr(), 316, 345
 example of use, 318
 prototype, 316
 little-endian byte order, 1198
 diagram, 1198
 Liu, C., 1437
 LKML (Linux kernel mailing list), 1423
llistxattr(), 316, 345
 prototype, 316
ln command, 341
 LNEXT terminal special character, 1296,
 1298, 1305, 1307
 locale, 200–204, 615
 specifying to a program, 203–204
locale command, 203
localeconv(), 202, 657
 localhost, 1187
 locality of reference, 118

localization, 200
localtime(), 189, 198, 657
 diagram, 188
 example of use, 192, 195, 199
 prototype, 189
localtime_r(), 189, 658
 lock (file). *See* file lock
 LOCK_EX constant, 1120
 example of use, 1121
 LOCK_NB constant, 1119, 1120
 example of use, 1121
 LOCK_SH constant, 1120
 example of use, 1121
 LOCK_UN constant, 1120
lockf(), 673, 1127
 interrupted by signal handler, 444
lockRegion(), 1133
 code of implementation, 1134
lockRegionWait(), 1133
 code of implementation, 1134
 LOG_ALERT constant, 779
 LOG_AUTH constant, 778, 779
 LOG_AUTHPRIV constant, 778, 779
 LOG_CONS constant, 777
 LOG_CRIT constant, 779
 LOG_CRON constant, 779
 LOG_DAEMON constant, 779
 LOG_DEBUG constant, 779
 LOG_EMERG constant, 779
 LOG_ERR constant, 779
 LOG_FTP constant, 778, 779
 LOG_INFO constant, 779
 LOG_KERN constant, 779
 LOG_LOCAL* constants, 779
 LOG_LPR constant, 779
 LOG_MAIL constant, 779
 LOG_MASK(), 781
 LOG_NDELAY constant, 778
 LOG_NEWS constant, 779
 LOG_NOTICE constant, 779
 LOG_NOWAIT constant, 778
 LOG_ODELAY constant, 778
 LOG_PERROR constant, 778
 LOG_PID constant, 778
 LOG_SYSLOG constant, 778, 779
 LOG_UPTO(), 781
 LOG_USER constant, 779
 LOG_UUCP constant, 779
 LOG_WARNING constant, 779
logger command, 780
 logical block, 255
 login accounting, 817–832
 login name, 26, 153, 154
 retrieving with *getlogin()*, 825
 login session, 700, 825
 login shell, 24, 26, 154
login(), 827
 LOGIN_NAME_MAX constant, 214
 LOGIN_PROCESS constant, 820, 821, 822
 LOGNAME environment variable, 825
logout(), 827
logrotate program, 772
logwtmp(), 827
 Lokier, J., xxxviii
 London, T., 4
longjmp(), 131–133, 135, 151, 360, 1426
 example of use, 134, 136, 432
 handling of signal mask, 429
 incorrect use of, 135
 prototype, 132
longjmp.c, 134
lookup_dcookie(), 801
 loopback address (IP), 1187
 Love (2010), 46, 210, 250, 278, 530, 751,
 1422, 1441
 Love, R., xxxix, 1441
brand48(), 657
removexattr(), 286, 316, 345
 prototype, 316
lsattr command, 305
 LSB (Linux Standard Base), 19
lseek(), 30, 81–83, 96, 257, 426
 diagram, 82
 example of use, 85, 519
 prototype, 81
lseek64(), 105
lsetxattr(), 286, 314–315, 345
 prototype, 314
lsuf command, 342
lstat(), 279–283, 345, 426
 example of use, 285, 370
 prototype, 279
ltrace command, 1403
 Lu (1995), 857, 1441
 Lu, H.J., xxxix, 1441
lutimes(), 286, 288–289, 345
 prototype, 289
 lvalue, 53

M

Mach, 6
 MADV_DOFORK constant, 1055
 MADV_DONTFORK constant, 612, 1055
 MADV_DONTNEED constant, 1055
 MADV_HWPOISON constant, 1055
 MADV_MERGEABLE constant, 1055
 MADV_NORMAL constant, 1054
 MADV_RANDOM constant, 1054
 MADV_REMOVE constant, 1055
 MADV_SEQUENTIAL constant, 1055

MADV_SOFT_OFFLINE constant, 1055
MADV_UNMERGEABLE constant, 1055
MADV_WILLNEED constant, 764, 1055
advise(), 1054–1055
 prototype, 1054
 RLIMIT_RSS resource limit and, 764
advise_dontneed.c, 1434
magic SysRq key, 1300
Mahdavi, J., 1194
main thread, 622
major(), 281
 example of use, 284
make program, 1442
make_zombie.c, 554, 562
makecontext(), 442
mallinfo(), 147
malloc debugging library, 147
malloc(), 140–142, 423, 1035
 debugging, 146–147
 example of use, 143
 implementation, 144–146
 diagram, 145
 prototype, 141
MALLOC_CHECK_ environment variable, 146
MALLOC_TRACE environment variable, 146
mallopt(), 147, 1035
mandatory file lock, 265, 293, 1119, 1138
Mane-Wheoki, J., xl
Mann (2003), 1250, 1442
Mann, S., 1442
manual pages, 1419–1421
MAP_ANON constant, 1034
MAP_ANONYMOUS constant, 1033, 1034
 example of use, 1036
MAP_FAILED constant, 1020, 1037
MAP_FIXED constant, 1033, 1040–1041, 1049
MAP_HUGETLB constant, 800, 1033
MAP_LOCKED constant, 1033, 1048
MAP_NORESERVE constant, 612, 999, 1033,
 1038–1040
MAP_POPULATE constant, 1033
MAP_PRIVATE constant, 1009, 1018,
 1021, 1033
 example of use, 1023
MAP_SHARED constant, 1009, 1021, 1031,
 1033, 1139
 example of use, 1029, 1036
MAP_UNINITIALIZED constant, 1033, 1034
mapped file. *See* file mapping
Margolin, B., xxxviii
Marshall, P., xxxix, xl
marshalling, 1200
Mason, C., xxxix
Mathis, M., 1194
Matloff (2008), 393, 1442
Matloff, N., 1442
Matz, M., xxxix
max(), *code of implementation*, 51
MAX_CANON constant, 1291
MAX_INPUT constant, 1291
maximum segment lifetime (MSL),
 TCP, 1274
Maximum Transmission Unit (MTU),
 1182, 1185
Maxwell (1999), 24, 46, 419, 936, 994, 1442
Maxwell, S., 1442
mbind(), 801
McCann, J., 1194
McGrath, R., 47
McGraw, G., 1445
mcheck(), 146
McKusick (1984), 276, 1442
McKusick (1996), 8, 20, 1044
McKusick (1999), 20, 1442
McKusick (2005), 20, 1442
McKusick, M.K., 1442
MCL_CURRENT constant, 1051
MCL_FUTURE constant, 761, 1051
Mecklenburg (2005), 431, 1442
Mecklenburg, R., 1442
mem_segments.c, 117
memalign(), 149–150
 example of use, 248
 prototype, 149
memlock.c, 1052
memory footprint, 121
 controlling with *fork()* plus *wait()*, 521
memory leak, 146
memory locking, 612, 800, 1012, 1033,
 1047–1051
 locks removed on process
 termination, 533
 resource limit on, 761
memory management, 22
memory mapping, 35, 225, 612,
 1017–1044. *See also* *mmap()*
 anonymous. *See* anonymous mapping
 creating, 1020–1023
 file-based. *See* file mapping
 nonlinear, 1041–1043
 private, 35, 1018, 1021
 remapping, 1037–1038
 removed on process termination, 533
 shared, 35, 1018, 1021
 synchronizing, 1031–1032
 unmapping, 1023–1024
memory protection, 1020
 changing, 1045–1047
 interaction with file access mode,
 1030–1031

memory residence, 1051–1054

memory usage (advising patterns of), 1054–1055

memory-based semaphore. *See* POSIX semaphore, unnamed

memory-mapped file. *See* file mapping

memory-mapped I/O, 1019, 1026–1027

message queue descriptor. *See* POSIX message queue, descriptor

message queue. *See* POSIX message queue; System V message queue

metadata, 239

migrate_pages(), 801

Miller, R., 4

Mills (1992), 205, 1442

Mills, D.L., 1442

MIN terminal setting, 1307

min(), *code of implementation*, 51

mincore(), 1051–1052

example of use, 1053

prototype, 1051

mingetty command, 820

Minix, 6, 1422

minor(), 281

example of use, 284

MINSIGSTKSZ constant, 435

Mitchell, E.L., 1442

mkdir(), 286, 350–351, 426

example of use, 302

prototype, 350

mkdirat(), 365, 426

mkdtemp(), 15, 351

mkfifo(), 286, 426, 907

example of use, 913

prototype, 907

mkfifoat(), 365, 426

mkfs command, 254

mknod command, 252

mknod(), 252, 286, 426, 800, 907

mknodat(), 365, 426

mkstemp(), 108–109, 791

example of use, 518

prototype, 108

mkswap command, 254

mktemp(), 109

mktime(), 190, 198

diagram, 188

example of use, 192

prototype, 190

mlock(), 800, 1048, 1049–1050

example of use, 1053

prototype, 1049

 RLIMIT_MEMLOCK resource limit and, 761

mlockall(), 761, 800, 1048, 1050–1051

prototype, 1050

 RLIMIT_MEMLOCK resource limit and, 761

mmap(), 286, 761, 1020–1023, 1058, 1139.

See also memory mapping

 compared with other shared memory APIs, 1115–1116

diagram, 1025, 1029, 1030

example of use, 1023, 1029, 1036, 1111, 1112, 1113

prototype, 1020

 RLIMIT_AS resource limit and, 760

 RLIMIT_MEMLOCK resource limit and, 761

MMAP_THRESHOLD constant, 1035

mmap64(), 105

mmcat.c, 1022

mmcopy.c, 1434

MNT_DETACH constant, 270, 272

MNT_EXPIRE constant, 270

MNT_FORCE constant, 270

Mochel (2005), 253, 1442

Mochel, P., 1442

mode_t data type, 64, 72, 78, 280, 301, 303, 350, 365, 907, 1064, 1090, 1109, 1146

modify_env.c, 131

Mogul, J.C., 1193, 1440

Molnar, I., 689

Mosberger (2002), 10, 1442

Mosberger, D., 1442

mount command, 169, 263, 267

mount namespace, 225, 261, 263, 607

mount point, 225, 261, 263, 264

diagram, 262

mount(), 246–267, 607, 801

example of use, 269

prototype, 264

move_pages(), 801

MPOL_MF_MOVE_ALL constant, 801

mprobe(), 146

mprotect(), 1022, 1045–1046

example of use, 1047

prototype, 1046

mq_attr structure, 1064, 1068, 1070, 1072

definition, 1068

example of use, 1069, 1071

mq_close(), 1058, 1064, 1066

prototype, 1066

mq_getattr(), 1058, 1064, 1070–1071

example of use, 1071

prototype, 1070

mq_notify(), 1058, 1064, 1078–1079

example of use, 1081, 1083

prototype, 1078

mq_notify_sig.c, 1080

mq_notify_sigwaitinfo.c, 1434

mq_notify_thread.c, 1082

mq_open(), 1058, 1064–1065
 example of use, 1070
 prototype, 1064
 RLIMIT_MSGQUEUE resource limit and, 762
 MQ_OPEN_MAX constant, 1085
 MQ_PRIO_MAX constant, 1073, 1085
mq_receive(), 673, 1058, 1064, 1074–1075
 example of use, 1077
 interrupted by signal handler, 444
 prototype, 1075
mq_send(), 673, 1058, 1064, 1073
 example of use, 1074
 interrupted by signal handler, 444
 prototype, 1073
mq_setattr(), 1058, 1064, 1072
 prototype, 1072
 example of use, 1072
mq_timedreceive(), 673, 1077
 interrupted by signal handler, 444
 prototype, 1077
mq_timedsend(), 673, 1077
 interrupted by signal handler, 444
 prototype, 1077
mq_unlink(), 1058, 1064, 1066
 example of use, 1067
 prototype, 1066
mqd_t data type, 64, 882, 1058, 1059,
 1064, 1065, 1066, 1070, 1072,
 1073, 1075, 1077, 1078, 1083
rand48(), 657
mremap(), 761, 1037–1038
 prototype, 1037
 RLIMIT_AS resource limit and, 760
 MREMAP_FIXED constant, 1037
 MREMAP_MAYMOVE constant, 1037
 MS_ASYNC constant, 1032
 MS_BIND constant, 264, 265, 266, 272
 MS_DIRSYNC constant, 264, 265, 306
 MS_INVALIDATE constant, 1032
 MS_MANDLOCK constant, 264, 265, 1138
 MS_MOVE constant, 264, 265
 MS_NOATIME constant, 77, 264, 265, 272
 MS_NODEV constant, 264, 266, 272
 MS_NODIRATIME constant, 264, 266, 272
 MS_NOEXEC constant, 264, 266, 272, 564
 MS_NOSUID constant, 264, 266, 272
 MS_PRIVATE constant, 267
 MS_RDONLY constant, 264, 266, 272
 MS_REC constant, 264, 266, 273
 MS_RELATIME constant, 264, 266, 272
 MS_REMOUNT constant, 264, 266
 MS_SHARED constant, 267
 MS_SLAVE constant, 267
 MS_STRICTATIME constant, 264, 267
 MS_SYNC constant, 1032
 example of use, 1029
 MS_SYNCHRONOUS constant, 264, 267
 MS_UNBINDABLE constant, 267
 MSG_DONTWAIT constant, 1259, 1260
 MSG_EXCEPT constant, 944
 example of use, 946
 MSG_INFO constant, 952
 example of use, 953
 MSG_MORE constant, 1260, 1263
 MSG_NOERROR constant, 943, 944
 example of use, 946
 MSG_NOSIGNAL constant, 1260
 MSG_OOB constant, 1259, 1260
 MSG_PEEK constant, 1259
 MSG_R constant, 923
 MSG_STAT constant, 952
 example of use, 953
 MSG_TRUNC constant, 1161
 MSG_W constant, 923
 MSG_WAITALL constant, 1259
msgctl(), 922, 947
 example of use, 948, 950, 953, 959, 961
 prototype, 947
msgget(), 922, 938, 950
 example of use, 940, 958, 960
 prototype, 938
msginfo structure, 951, 952
 example of use, 953
msglen_t data type, 64, 948
 MSGMAX limit, 950, 951
 MSGMNB limit, 949, 950, 951
 MSGMNI limit, 950, 951
 MSGPOOL limit, 950
msgqnum_t data type, 65, 948
msgrcv(), 673, 922, 943–944, 947, 948, 949
 example of use, 946, 959, 961
 interrupted by signal handler, 444
 interrupted by stop signal, 445
 prototype, 943
msgsnd(), 673, 922, 940–941, 947, 948,
 949, 950
 example of use, 942, 958, 960
 interrupted by signal handler, 444
 interrupted by stop signal, 445
 prototype, 941
 MSGTQL limit, 950
 MSL (maximum segment lifetime),
 TCP, 1274
msqid_ds structure, 922, 947, 948–949, 950
 definition, 948
 example of use, 949
msync(), 286, 673, 1022, 1024, 1031–1032
 example of use, 1029
 prototype, 1031

mtrace(), 146
 MTU (Maximum Transmission Unit), 1182, 1185
 Mui, L., 1444
multi_descriptors.c, 1426
multi_SIGCHLD.c, 557
multi_wait.c, 543
 MULTICS, 2
 multihomed host, 1180, 1187, 1220
 multiplexed I/O, 1327, 1330–1346
munlock(), 1049–1050
 prototype, 1049
munlockall(), 1050–1051
 prototype, 1050
munmap(), 1022, 1023–1024, 1058
 example of use, 1036
 prototype, 1023
muntrace(), 146
 mutex, 614, 631–642, 881
 attributes, 640
 deadlock, 639
 diagram, 639
 destroying, 640
 initializing, 639–640
 locking, 636, 637–638
 performance, 638
 statically allocated, 635
 type, 640–642
 unlocking, 636
 used with a condition variable, 646
 mutual exclusion, 634

N

N_TTY constant, 1292, 1294
 NAME_MAX constant, 214, 215
named daemon, 1210
 named semaphore. *See* POSIX semaphore, named
nanosleep(), 488–489, 673
 example of use, 490
 interrupted by signal handler, 444
 interrupted by stop signal, 445
 prototype, 488
 Native POSIX Thread Library (NPTL). *See* NPTL
 NCCS constant, 1292
necho.c, 123
 NetBSD, 7
netstat command, 1182, 1275–1276
 network byte order, 1198–1199
 Network File System (NFS), Linux implementation, 254
 network ID, 1186
 network layer, 1184–1186
 diagram, 1181
 network mask, 1186
 Network Time Protocol (NTP), 204, 205, 1442
 networking protocol, 1180
 Neville-Neil, G.V., 1442
new_intr.c, 1301
 NEW_TIME constant, 820, 822
newgrp command, 155, 156
 newline character, 30, 1298
 Next Generation POSIX Threads (NGPT), 689
 NeXTStep, 5
nfs_t data type, 65, 1337
 NFS (Network File System), Linux implementation, 254
nftw(), 358–360, 657
 example of use, 361
 prototype, 358
nftw_dir_tree.c, 360
 NGPT (Next Generation POSIX Threads), 689
 NGROUPS_MAX constant, 179, 214
 NI_DGRAM constant, 1218
 NI_MAXHOST constant, 1218
 NI_MAXSERV constant, 1218
 NI_NAMEREQD constant, 1219
 NI_NOFQDN constant, 1219
 NI_NUMERICHOST constant, 1219
 NI_NUMERICSERV constant, 1219
nice command, 735
 nice value, 614, 733–737, 801
 diagram, 734
 LinuxThreads nonconformance, 691
 NPTL nonconformance, 693
 resource limit, 762
nice(), 735, 801
 RLIMIT_NICE resource limit and, 762
 NL terminal special character, 1296, 1298, 1302, 1303, 1307
nl_langinfo(), 657
 NLO constant, 1302
 NL1 constant, 1302
 NLDLY constant, 1302
nlink_t data type, 65, 280
nm command, 844
no_echo.c, 1306
 NOFLSH constant, 1303, 1305, 1307
nohup command, 710
 nonblocking I/O, 77, 103–104, 915–917, 1326, 1330
 noncanonical mode (terminal I/O), 1290, 1307–1309
 nonlocal goto, 131–137
 nonprivileged (unprivileged) process, 33
 nonreentrant function, 116, 423

nonreentrant.c, 424
 Nordmark, E., 1194
 NPTL (Native POSIX Threads Library),
 457, 592, 600, 603, 606, 607, 609,
 668, 682, 687, 688, 689, 692–694,
 696, 987
 Pthreads nonconformances, 693
 NSIG constant, 408
 niohl(), 1199
 prototype, 1199
 ntohs(), 1199
 prototype, 1199
 NTP (Network Time Protocol), 204,
 205, 1442
 NULL pointer, casting inside variadic
 function call, 1413–1415
 null signal, 403, 458
 numbers-and-dots notation (IPv4
 address), 1231
 NX (no execute) protection (x86-32),
 793, 1022

O

O_ACCMODE constant, 93
 O_APPEND constant, 74, 75, 92, 93, 96,
 110, 306
 example of use, 519
 O_ASYNC constant, 74, 75, 93, 1281, 1347.
 See also signal-driven I/O
 example of use, 1349
 O_CLOEXEC constant, 74, 75, 98, 894
 O_CREAT constant, 74, 76, 90, 107, 1059,
 1065, 1109, 1145, 1146
 example of use, 71, 84
 O_DIRECT constant, 74, 76, 93, 246
 example of use, 248
 O_DIRECTORY constant, 74, 76
 O_DSYNC constant, 74, 76, 243
 O_EXCL constant, 74, 76, 90, 109, 791, 1059,
 1065, 1109, 1145
 O_FSYNC constant, 242
 O_LARGEFILE constant, 74, 76, 93, 105
 O_NDELAY constant, 104
 O_NOATIME constant, 74, 76, 93, 265, 800
 O_NOCTTY constant, 74, 77, 706, 707,
 768, 1380
 O_NOFOLLOW constant, 74, 77
 O_NONBLOCK constant, 74, 77, 93, 96,
 103–104, 377, 472, 508, 894,
 915–918, 1065, 1068, 1071, 1072,
 1073, 1075, 1139, 1153, 1158,
 1175, 1254, 1260, 1281, 1308,
 1326, 1330, 1347. *See also*
 nonblocking I/O
 example of use, 917, 1349, 1372

O_RDONLY constant, 73, 74, 1060, 1065, 1109
 example of use, 71
 O_RDWR constant, 73, 74, 1060, 1065,
 1109, 1380
 example of use, 84
 O_RSYNC constant, 243
 O_SYNC constant, 74, 77, 93, 241, 250, 267
 performance impact, 242
 O_TRUNC constant, 74, 77, 1109, 1139, 1146
 example of use, 71
 O_WRONLY constant, 73, 74, 1060, 1065
 example of use, 71
 objdump command, 844
 object library, 834
 OCRNL constant, 1296, 1302
 OFDEL constant, 1302, 1303
 off_t data type, 65, 66, 81, 82, 98, 102, 103,
 104, 106, 244, 280, 757, 759,
 1020, 1125, 1261
 casting in *printf()* calls, 107
 off64_t data type, 105
 offsetof(), 357
 OFILL constant, 1302, 1303
 OLCUC constant, 1302, 1303
 OLD_TIME constant, 820, 822
 on_exit(), 532, 535–536, 866
 example of use, 537
 prototype, 535
 one_time_init.c, 1431
 ONLCR constant, 1296, 1298, 1302
 ONLRET constant, 1296, 1302
 ONOCR constant, 1296, 1302
 OOM killer, 1039
 opaque (data type), 621
 open file description, 94
 diagram, 95
 open file status flags, 75, 93–94, 95, 96,
 518, 613
 open file table, 94
 Open Group, The, 13
 Open Software Foundation (OSF), 13
 Open Source Development Laboratory
 (OSDL), 18
 open(), 70, 72–78, 96, 286, 345, 426, 673,
 801, 1139, 1142, 1145, 1146
 directories and, 76
 example of use, 71, 73, 84, 302
 FIFOs and, 916
 interrupted by signal handler, 444
 prototype, 72
 returns lowest available file
 descriptor, 73
 RLIMIT_NOFILE resource limit and, 762
 symbolic links and, 77
 OPEN_MAX constant, 214, 215

open64(), 105
openat(), 15, 365–366, 426, 674
 prototype, 365
 OpenBSD, 7
opendir(), 345, 352, 355
 example of use, 356
 prototype, 352
openlog(), 777–779
 example of use, 780
 prototype, 777
openpty(), 1386
 operating system, 21, 1438, 1444
 oplock (opportunistic lock), 1142
 OPOST constant, 1296, 1298, 1302, 1305
 example of use, 1311
 opportunistic lock, 1142
optarg variable, 1406
opterr variable, 1406
optind variable, 1406
optopt variable, 1406
 O'Reilly, T., 1444
 ORIGIN (in *rpath* list), 853
 Orlov block-allocation strategy, 307, 1438
 orphan.c, 1430
 orphaned process, 553
 orphaned process group, 533, 725–730
 diagram, 726
 terminal *read()* and, 730
 terminal *write()* and, 730
 orphaned_pgrp_SIGHUP.c, 728
 OSDL (Open Source Development Laboratory), 18
 OSF (Open Software Foundation), 13
 OSF/1, 4
 ouch.c, 399
 out-of-band data, 394, 1259, 1260, 1283, 1288, 1331, 1343

P

P_ALL constant, 550
 P_PGID constant, 550
 P_PID constant, 550
 packet mode (pseudoterminal), 1342, 1389
 Padhye, J., 1194
 page (virtual memory), 119
 page fault, 119
 page frame, 119
 diagram, 120
 page size
 determining at run time, 214
 on various hardware architectures, 119
 page table, 224, 879
 diagram, 120, 521, 1026
 paged memory management unit (PMMU), 120

Pai, R., 1445
 PARENB constant, 1303, 1305
 parent directory, 27
 parent process, 31, 513, 515, 553
 signaled on death of child, 555
 parent process ID, 32, 114–115, 608, 613
 Pariag, D., 1439
 parity (terminal I/O), 1305
 PARMRK constant, 1302, 1305
 example of use, 1311
 PARODD constant, 1303, 1305
 partial write, 80, 891, 1254
 Partridge, C., 1194, 1444
 PASC (Portable Applications Standards Committee), 11
 passive close (TCP), 1272
 passive open (socket), 1155
 passive socket, 1156
passwd command, 169
passwd structure, 157
 definition, 157
 example of use, 159
 password encryption, 162–166
 password file, 153–155
 retrieving records from, 157–158, 160
 PATH environment variable, 34, 567, 568–570, 791
 path MTU, 1185
 PATH_MAX constant, 214, 215, 350
pathconf(), 217–218, 345, 425, 426
 prototype, 217
 pathname, 28
 absolute, 29, 367
 maximum length of, 214
 parsing, 370–372
 relative, 29, 363
 resolving, 369–370
pause(), 418, 426, 673
 example of use, 401
 prototype, 418
 Paxson, V., 1194
pclose(), 902–903, 919
 example of use, 905
 prototype, 902
pdfflush kernel thread, 241, 768, 1032
 PDP-11, 2, 3, 391
 Peach, J., xxxix
 Peek (2001), xxxii, 1442
 Peek, J., 1442
 Peikari (2004), 795, 1442
 Peikari, C., 1442
 PENDIN constant, 1303
perror(), 49–50
 prototype, 49
 persistence, 886

personality(), 1334
 PGID (process group ID), 39, 613, 699, 705
 Phillips, M., xxxix
 physical block, 253
 PID (process ID), 32, 114, 604, 608, 613, 705
pid_t data type, 65, 114, 115, 402, 405, 438, 458, 493, 496, 516, 523, 542, 544, 552, 599, 605, 699, 700, 701, 702, 704, 705, 708, 741, 742, 744, 747, 749, 750, 819, 948, 1012, 1125, 1354, 1385
 Piggin, N., xxxix
 pipe, 3, 214, 282, 392, 882, 883, 886, 889–906

- atomicity of *write()*, 891
- bidirectional, 890
- capacity, 891
- closing unused file descriptors, 894
- connecting filters with, 899–902
- creating, 892
- diagram*, 879, 890
- poll()* on, 1342
- read()* semantics, 917–918
- select()* on, 1342
- to a shell command, 902–906
- stdio* buffering and, 906
- used for process synchronization, 897–899
- write()* semantics, 918

pipe(), 286, 426, 801, 892, 1175

- diagram*, 892
- example of use*, 896, 898, 900
- prototype*, 892
- RLIMIT_NOFILE resource limit and, 762

 PIPE_BUF constant, 214, 891, 918, 1343, 1351
pipe_ls_wc.c, 900
pipe_sync.c, 897
pipe2(), 894
pivot_root(), 345, 801
 Plauger (1992), 30, 1442
 Plauger, P.J., 1442
 Pluzhnikov, P., xxxix
 PMMU (paged memory management unit), 120
pmsg_create.c, 1069
pmsg_getattr.c, 1071
pmsg_receive.c, 1076
pmsg_send.c, 1074
pmsg_unlink.c, 1066
 Podolsky, M., 1194
 poll, 1326
poll(), 426, 673, 1337–1339, 1389, 1439

- comparison with *select()*, 1344–1345
- example of use*, 1341
- interrupted by signal handler, 444
- interrupted by stop signal, 445
- performance, 1365
- problems with, 1346
- prototype*, 1337

 POLL_ERR constant, 441, 1353
 POLL_HUP constant, 441, 1343, 1353
 POLL_IN constant, 440, 441, 1353
 POLL_MSG constant, 440, 441, 1353
 POLL_OUT constant, 440, 441, 1353
poll_pipes.c, 1340
 POLL_PRI constant, 441, 1353
 Pollard, J., xl
 POLLERR constant, 1337, 1338, 1342, 1343, 1353
pollfd structure, 1337–1338

- definition*, 1337

 POLLHUP constant, 1337, 1338, 1342, 1343, 1353
 POLLIN constant, 1337, 1338, 1342, 1343, 1353
 POLLMSG constant, 1337, 1338, 1353
 POLLNVAL constant, 1337, 1338, 1339
 Pollock, W., xli
 POLLOUT constant, 1337, 1338, 1342, 1343, 1353
 POLLPRI constant, 1337, 1338, 1343, 1353, 1389
 POLLRDBAND constant, 1337, 1338
 POLLRDHUP constant, 1337, 1338, 1339, 1343
 POLLRDNORM constant, 1337, 1338, 1353
 POLLWRBAND constant, 1337, 1338, 1353
 POLLWRNORM constant, 1337, 1338, 1353
popen(), 902–903, 919

- avoid in privileged programs, 788
- diagram*, 902
- example of use*, 905
- prototype*, 902

popen_glob.c, 904
 port number, 64, 1188–1189

- ephemeral, 1189, 1224, 1263
- privileged, 800, 1189
- registered, 1189
- well-known, 1189

 portability, xxxiv, 10, 61–68, 211, 1420

- source code vs. binary, 19

 Portable Application Standards Committee (PASC), 11
 portable filename character set, 28
 Portable Operating System Interface (POSIX), 11
 position-independent code, 837, 838

- POSIX, 11
- POSIX 1003.1-2001, 13
- POSIX asynchronous I/O, 613, 1327, 1347
- POSIX clock, 491-493
 - obtaining clock ID for process or thread, 493
 - retrieving value of, 491-492
 - setting value of, 492
- POSIX conformance, 14
- POSIX file locking, 1124
- POSIX IPC, 885, 1057-1062
 - compared with System V IPC, 1061-1062
 - object
 - creating, 1059-1060
 - deleting, 1060-1061
 - listing, 1061
 - name, 1058-1059
 - permissions, 1060
 - persistence, 1060-1061
 - portability, 884, 1061
- POSIX message queue, 614, 882, 883, 886, 1063-1088
 - attributes, 1068-1072
 - modifying, 1072
 - retrieving, 1070-1071
 - closed on process termination, 533
 - closing, 1066
 - compared with System V message queue, 1086-1087
 - creating, 1064-1065
 - descriptor, 64, 1065
 - relationship to open message queue, 1067-1068
 - limits, 1085-1086
 - Linux-specific features, 1083-1085
 - message notification, 1077-1083
 - via a signal, 1079-1081
 - via a thread, 1082-1083
 - opening, 1064
 - priority of messages, 1073
 - receiving messages, 1074-1077
 - with timeout, 1077
 - resource limit on bytes allocated for queues, 761
 - sending messages, 1073-1074
 - with timeout, 1077
 - unlinking, 1066-1067
- POSIX semaphore, 1089-1105
 - closed on process termination, 533
 - compared with System V semaphore, 1103-1104
 - limits, 1104-1105
 - named, 614, 882, 886, 1089, 1090-1093
 - closing, 1093
 - initializing, 1091
 - opening, 1090-1091
 - unlinking, 1093
 - post (increment) operation, 1096-1097
 - process-shared, 1100
 - retrieving current value, 1097
 - thread-shared, 1100
 - unnamed, 614, 882, 886, 1089, 1099-1103
 - destroying, 1102
 - initializing, 1100-1101
 - wait (decrement) operation, 1094-1096
- POSIX shared memory, 275, 614, 882, 886, 1107-1116
 - compared with other shared memory APIs, 1115-1116
 - creating, 1108, 1109-1111
 - removing, 1114
- POSIX thread, 614, 617-697, 1438. *See also* thread
- POSIX timer, 495-507, 614
 - arming, 498-499
 - creating, 495-497
 - deleting, 499
 - disarming, 498-499
 - notification via a signal, 499-503
 - notification via a thread, 504-507
 - retrieving current value, 499
 - timer overrun, 502, 503-504, 505
- POSIX.1, 11, 17, 1442
- POSIX.1-2001, 13, 17
- POSIX.1-2008, 17
- POSIX.1b, 12, 17, 41, 61, 456, 491, 495, 738, 1057
- POSIX.1c, 12, 17, 61, 620
- POSIX.1d, 12, 17, 1077, 1096
- POSIX.1e, 319, 337, 798, 1369
- POSIX.1g, 12, 16, 17, 1149
- POSIX.1j, 12, 17
- POSIX.2, 12, 17
- POSIX.2c, 319, 337
- POSIX.4, 12, 1439
- POSIX_ARG_MAX constant, 124
- POSIX_FADV_DONTNEED constant, 245, 1032
- POSIX_FADV_NOREUSE constant, 245
- POSIX_FADV_NORMAL constant, 245
- POSIX_FADV_RANDOM constant, 245
- POSIX_FADV_SEQUENTIAL constant, 245
- POSIX_FADV_WILLNEED constant, 245, 1055
- posix_fadvise()*, 244-246, 1032
 - prototype*, 244
- posix_fallocate()*, 83
- POSIX_MADV_DONTNEED constant, 1055
- POSIX_MADV_NORMAL constant, 1055
- POSIX_MADV_RANDOM constant, 1055

POSIX_MADV_SEQUENTIAL constant, 1055
 POSIX_MADV_WILLNEED constant, 1055
posix_madvise(), 1055
posix_memalign(), 149–150
 example of use, 150
 prototype, 149
posix_openpt(), 1380–1381
 example of use, 1384
 prototype, 1380
posix_spawn(), 514
posix_trace_event(), 426
 POSIXLY_CORRECT environment variable, 1410
 Postel, J., 1193, 1194
 Potthoff, K.J., xxxix
 PPID (parent process ID), 32, 114–115, 608, 613
ppoll(), 1370
 interrupted by signal handler, 444
 PR_CAPBSET_DROP constant, 806
 PR_CAPBSET_READ constant, 806
 PR_GET_SECUREBITS constant, 812
 PR_SET_DUMPABLE constant, 449, 615
 PR_SET_KEEPCAPS constant, 813, 816
 PR_SET_NAME constant, 615
 PR_SET_PDEATHSIG constant, 553, 615
prctl(), 449, 553, 806, 813
pread(), 98–99, 286, 673
 prototype, 98
preadv(), 102, 286
 prototype, 102
 preforked server, 1246
 prethreaded server, 1246
 print_rlimit.c, 758
 print_rusage.c, 1432
 print_wait_status.c, 546
printenv command, 126
printf()
 buffering. *See* buffering of file I/O
 use within signal handlers, 427–428
printfk() (kernel function), 776
printPendingSigs(), 408
 code of implementation, 409
printSigMask(), 408
 code of implementation, 409
printSigset(), 408
 code of implementation, 409
printWaitStatus(), 546
 code of implementation, 546–547
 PRIO_PGRP constant, 735
 PRIO_PROCESS constant, 735
 PRIO_USER constant, 735
 private, copy-on-write mapping, 1018
 privileged process, 33, 168
 privileged program, 783. *See also* process capabilities
 process, 22, 31, 113
 accounting. *See* process accounting
 capabilities. *See* process capabilities
 checking for existence of, 403–404
 controlling memory footprint with
 fork() plus *wait()*, 521
 CPU affinity, 615, 748–750
 creating, 31, 515–525
 credentials, 167–184
 passing via socket, 1284–1285
 current working directory, 29, 225, 363–365, 604, 613
 exit status, 32, 545
 ID, 32, 114, 604, 608, 613, 705
 memory layout, 31, 115–118
 diagram, 119, 1007
 memory policy, 615
 mount namespace, 225, 261, 263, 607
 nice value (priority). *See* nice value
 priority. *See* realtime scheduling, priority
 privileged, 33, 168
 realtime scheduling. *See* realtime scheduling
 resource limit. *See* resource limit
 resource limit on number of, 763
 resource usage, 552, 614, 753–755
 root directory, 225, 367–368, 604, 613
 setting as owner of a file descriptor, 1347, 1350–1351
 signal mask, 38, 388, 410, 578, 613, 683
 speed of creation, 610–612
 system-wide limit on number of, 763
 termination, 32, 531–533
 abnormal, 389, 433, 441, 531
 normal, 531
 from signal handler, 549–550
 termination status, 32, 513, 531, 545
 umask, 301–303, 328, 351, 604, 613, 790, 907, 923, 1060, 1065, 1091, 1110, 1174
 unprivileged, 33
 process accounting, 591–598, 801
 Version 3, 597–598
 process capabilities, 33, 615, 798–799
 changing, 807–808
 effect on, when changing process user IDs, 806–807
 effective, 799, 802, 807
 inheritable, 799, 803, 807
 permitted, 798, 802, 807
 transformation during *exec()*, 805
 process group, 39, 699, 701–704
 background. *See* background process group

- changing capabilities of all
 - processes in, 815
- changing membership of, 702
- creating, 702
- diagram*, 701
- foreground. *See* foreground process
 - group
- leader, 39, 699, 702, 705
- lifetime, 699
- orphaned. *See* orphaned process group
- sending signal to, 402, 405
- setting as owner of a file descriptor,
 - 1347, 1350–1351
- waiting on member of, 544, 550
- process group ID, 39, 613, 699, 705
- process ID, 32, 114, 604, 608, 613, 705
- process scheduling, 22. *See also* realtime
 - scheduling
- process time, 40, 185, 206–209, 614
 - resource limit on, 761
- process_time.c*, 208
- procfs_pidmax.c*, 228
- procfs_user_exe.c*, 1428
- prod_condvar.c*, 645
- prod_no_condvar.c*, 642
- program, 30, 113
 - executing, 32, 563–571
- program break, 116, 139–140
 - adjusting, 139–140
- program counter, 133
- program termination routine (exit
 - handler), 532, 533–537, 615
- program_invocation_name* variable, 124
- program_invocation_short_name*
 - variable, 124
- PROT_EXEC constant, 1020, 1021, 1030
- PROT_NONE constant, 1020, 1021
- PROT_READ constant, 1020, 1021, 1030
 - example of use*, 1023, 1029
- PROT_WRITE constant, 1020, 1021, 1030
 - example of use*, 1029
- protocol stack (TCP/IP), 1181
- Provos, N., 1328
- pselect()*, 426, 673, 1369
 - example of use*, 1370
 - interrupted by signal handler, 444
 - prototype*, 1369
- psem_create.c*, 1092
- psem_getvalue.c*, 1098
- psem_post.c*, 1097
- psem_timedwait.c*, 1434
- psem_unlink.c*, 1094
- psem_wait.c*, 1095
- pset_bind()*, 748
- pseudoheader (TCP), 1267
- pseudoterminal, 39, 1375–1399
 - BSD, 1379, 1395–1397
 - device pair, 1376
 - diagram*, 1377, 1378
 - I/O, 1388–1389
 - master, 1376
 - opening, 1380–1381, 1383–1384
 - master clone device, 1381
 - packet mode, 1342, 1389
 - poll()* on, 1342
 - select()* on, 1342
 - slave, 1376
 - changing ownership and
 - permissions of, 1381, 1396
 - obtaining name of, 1382
 - opening, 1383
 - unlocking, 1382
 - terminal attributes, 1394
 - UNIX 98, 1379, 1380
 - window size, 1394
- PSH control bit (TCP), 1267
- pshm_create.c*, 1110
- pshm_read.c*, 1113
- pshm_unlink.c*, 1114
- pshm_write.c*, 1112
- psiginfo()*, 440
- psignal()*, 15, 406
 - prototype*, 406
- pstree* command, 115
- pt_chown* program, 784, 1381, 1396
- pthread_atfork()*, 687
- pthread_attr_destroy()*, 628
- pthread_attr_init()*, 628
- pthread_attr_setdetachstate()*, 628
- pthread_attr_setstack()*, 681
- pthread_attr_t* data type, 497, 620, 622,
 - 623, 628, 1079
- pthread_cancel()*, 671–672, 680
 - example of use*, 675, 679
 - prototype*, 671
- PTHREAD_CANCEL_ASYNCHRONOUS constant,
 - 672, 680
- PTHREAD_CANCEL_DEFERRED constant, 672
- PTHREAD_CANCEL_DISABLE constant, 672
- PTHREAD_CANCEL_ENABLE constant, 672
- PTHREAD_CANCELED constant, 622, 674
 - example of use*, 675, 679
- pthread_cleanup_pop()*, 676–677
 - example of use*, 678
 - prototype*, 676
- pthread_cleanup_push()*, 676–677
 - example of use*, 678
 - prototype*, 676
- pthread_cond_broadcast()*, 643–644
 - prototype*, 644

pthread_cond_destroy(), 652
 prototype, 652
pthread_cond_init(), 651–652
 prototype, 651
 PTHREAD_COND_INITIALIZER constant, 643
pthread_cond_signal(), 643–644
 example of use, 645, 650
 prototype, 644
pthread_cond_t data type, 620, 643, 644, 645, 651, 652
pthread_cond_timedwait(), 644–645, 673
 interrupted by signal handler, 444
 prototype, 645
pthread_cond_wait(), 643–644, 673, 683
 example of use, 647, 651
 interrupted by signal handler, 444
 prototype, 644
pthread_condattr_t data type, 620, 651
pthread_create(), 622–623
 example of use, 627, 628, 650, 675, 679
 prototype, 622
pthread_detach(), 627–628
 example of use, 627
 prototype, 627
pthread_equal(), 624–625, 1431
 prototype, 624
pthread_exit(), 623–624
 prototype, 623
pthread_getcpuclockid(), 493, 496
 prototype, 493
pthread_getspecific(), 662–663
 example of use, 667
 prototype, 662
pthread_join(), 606, 607, 625–626, 673, 674, 1431
 example of use, 627, 651, 675, 679
 prototype, 625
pthread_key_create(), 661–662
 example of use, 666
 prototype, 661
pthread_key_t data type, 620, 661, 662,
 PTHREAD_KEYS_MAX constant, 668
pthread_kill(), 683, 684, 690
 prototype, 684
 PTHREAD_MUTEX_DEFAULT constant, 641
pthread_mutex_destroy(), 640
 prototype, 640
 PTHREAD_MUTEX_ERRORCHECK constant, 641
pthread_mutex_init(), 639–640
 example of use, 642
 prototype, 639
 PTHREAD_MUTEX_INITIALIZER constant, 635, 640, 641
pthread_mutex_lock(), 635–636, 683
 example of use, 636, 647, 650
 interrupted by signal handler, 444
 prototype, 636
 PTHREAD_MUTEX_NORMAL constant, 641
 PTHREAD_MUTEX_RECURSIVE constant, 641
pthread_mutex_t data type, 620, 635, 636, 639, 640, 644, 645
pthread_mutex_timedlock(), 637
 interrupted by signal handler, 444
pthread_mutex_trylock(), 637, 639
 interrupted by signal handler, 444
pthread_mutex_unlock(), 635–636
 example of use, 637, 651
 prototype, 636
pthread_mutexattr_destroy(), 642
pthread_mutexattr_init(), 641
pthread_mutexattr_settype(), 642
pthread_mutexattr_t data type, 620, 639, 640
pthread_once(), 658–659
 example of use, 667
 prototype, 658
 PTHREAD_ONCE_INIT constant, 659
pthread_once_t data type, 620, 658
pthread_self(), 624
 example of use, 627
 prototype, 624
pthread_setcancelstate(), 672, 680
 prototype, 672
pthread_setcanceltype(), 672–673, 680
 prototype, 672
pthread_setspecific(), 662–663
 example of use, 667
 prototype, 662
pthread_sigmask(), 683, 684
 prototype, 684
pthread_sigqueue(), 683, 685
 prototype, 685
pthread_t data type, 493, 605, 620, 622, 623, 624, 625, 627, 671, 684, 685
pthread_testcancel(), 673, 675–676
 prototype, 676
 Pthreads, 617
 ptmr_null_evpc.c, 1429
 ptmr_sigevev_signal.c, 500
 ptmr_sigevev_thread.c, 506
ptrace(), 394, 545, 608, 801
ptrdiff_t data type, 65
ptsname(), 657, 1380, 1382
 example of use, 1384
 prototype, 1382
ptsname_r(), 658, 1383
 pty, 1376
 pty_fork.c, 1386
 pty_master_open.c, 1384
 pty_master_open_bsd.c, 1396

ptyFork(), 1385–1386
code of implementation, 1386–1388
example of use, 1392
prototype, 1385
ptyMasterOpen(), 1383, 1396
code of implementation, 1384, 1396–1397
example of use, 1387
prototype, 1383
putc_unlocked(), 657
putchar_unlocked(), 657
putenv(), 128, 130, 657
example of use, 131
prototype, 128
putmsg(), 673
putpmsg(), 673
pututxline(), 657, 826
example of use, 829
prototype, 826
pwwrite(), 98–99, 286, 673
prototype, 98
pwrtv(), 102, 286
prototype, 102

Q

quantum, 733
 Quartermann (1993), 20, 1442
 Quartermann, J.S., 1442
 quit character, 1296, 1298
 QUIT terminal special character, 1296,
 1298, 1303, 1305
quotactl(), 345, 801

R

R_OK constant, 299
 race condition, 90–92, 465, 525–527, 897,
 975, 1118, 1368
 time-of-check, time-of-use, 790
 Rago, S.A., 1421, 1444
raise(), 404, 426, 441, 458
example of use, 720, 724
prototype, 404
 Ramakrishnan, K., 1194
 Ramey, C., 25
 Rampp, F., xxxix
rand(), 657
rand_r(), 658
 Randow, D., xl
 raw I/O, 246–248
 raw mode (terminal I/O), 1309–1316
 raw socket, 1184
rdwrn.c, 1255
 read permission, 29, 282, 294, 297
read(), 70, 79–80, 286, 426, 673, 1138
example of use, 71, 85, 487
 FIFO, 918
 interrupted by signal handler, 443
 pipe, 918
prototype, 79
 terminal input
 by background job, 394
 canonical mode, 1307
 noncanonical mode, 1307–1309
 by orphaned process group, 730
read_line.c, 1201
read_line_buf.c, 1435
read_line_buf.h, 1435
readahead(), 245, 1055
readdir(), 286, 353–354, 657
example of use, 356
prototype, 353
readdir_r(), 357, 658
prototype, 357
readelf command, 844
readLine(), 1200–1202
code of implementation, 1201
example of use, 1222, 1225
prototype, 1200
readlink(), 345, 349–350, 426
example of use, 370
prototype, 350
readlinkat(), 365, 426
readn(), 1254
code of implementation, 1255
prototype, 1254
readv(), 99–100, 286, 673
example of use, 101
 interrupted by signal handler, 443
prototype, 99
 read-write offset. *See* file offset
 ready file descriptor, 1327
 real group ID, 32, 167, 172, 173, 175,
 177, 613
 real time, 40, 185
 real user ID, 32, 167, 172, 173, 175, 177
real_timer.c, 482
realloc(), 148–149, 1038
example of use, 149
prototype, 148
realpath(), 369
example of use, 370
prototype, 369
 realtime, 41
 realtime scheduling, 737–747, 801
 FIFO policy (SCHED_FIFO), 740
 policy, 614, 738
 changing, 741–744

realtime scheduling, *continued*
 priority, 614, 738, 740
 changing, 741–744
 relinquishing CPU, 747
 resource limit for CPU time, 764
 resource limit for priority, 764
 round-robin policy (SCHED_RR), 739
 round-robin time slice, 747
 realtime signal, 214, 221, 388, 456–463
 handling, 460–463
 limit on number queued, 457, 764
 sending, 458–460
 used by LinuxThreads, 690
 used by NPTL, 693
reboot(), 801
 receiving TCP, 1191
 record lock. *See* file lock
 recursive bind mount, 273–274
 recursive resolution, DNS, 1211
recv(), 426, 673, 1259–1260
 interrupted by signal handler, 444
 prototype, 1259
recvfrom(), 426, 673, 1160–1161
 diagram, 1160
 example of use, 1172, 1173, 1208, 1209, 1241
 interrupted by signal handler, 444
 prototype, 1161
recvmmsg(), 1284
recvmmsg(), 426, 673, 1284
 interrupted by signal handler, 444
 reentrancy, 556
 reentrant function, 422–425, 622, 657
region_locking.c, 1134
regionIsLocked(), 1134
 code of implementation, 1134–1135
 regular file, 27, 282
 poll() on, 1342
 select() on, 1342
 Reiser, J., xxxix, 4
 Reiserfs file system, 260
 i-node flag, 304–308
 tail packing, 260, 307
 relative pathname, 29, 363
releaseSem(), 989–991
 code of implementation, 991
 example of use, 1004, 1005
 reliable signal, 390
 relocation (of symbols), 837
remap_file_pages(), 1041–1043
 prototype, 1041
remove(), 286, 345, 352
 prototype, 352
removexattr(), 286, 316, 345
 prototype, 316
rename(), 286, 300, 345, 348–349, 426, 800
 prototype, 348
renameat(), 365, 426
renice command, 735
 REPRINT terminal special character, 1296, 1298, 1305, 1307
 Request for Comments (RFC), 1179, 1193–1194. *See also individual RFC entries*
 reserved blocks (file system), 277, 801
 reserved port, 1189
reserveSem(), 989–990
 code of implementation, 990
 example of use, 1004, 1005
 resident set, 119
 resource limit on size of, 763
 resource limit, 34, 614, 755–764, 801
 details of specific limits, 760–764
 LinuxThreads nonconformance, 691
 NPTL nonconformance, 694
 unrepresentable, 759–760
 resource usage, 552, 614, 753–755
 Ressler, S., 1441
 retransmission timeout (RTO), 1191
rewinddir(), 354
 prototype, 354
 RFC (Request For Comments), 1179, 1193–1194
 RFC 768, 1194
 RFC 791, 1193
 RFC 793, 1194, 1270, 1283
 RFC 862, 1240
 RFC 950, 1193
 RFC 993, 1267
 RFC 1014, 1200
 RFC 1122, 1194, 1274
 RFC 1123, 1194
 RFC 1305, 1442
 RFC 1323, 1194
 RFC 1819, 1184
 RFC 2018, 1194
 RFC 2460, 1194, 1203
 RFC 2581, 1194
 RFC 2861, 1194
 RFC 2883, 1194
 RFC 2988, 1194
 RFC 3168, 1194, 1267
 RFC 3257, 1286
 RFC 3286, 1286
 RFC 3390, 1194
 RFC 3493, 1194, 1203, 1213
 RFC 3513, 1194
 RFC 3542, 1194
 RFC 3697, 1203
 RFC 4007, 1203

RFC 4219, 1188
 RFC 4291, 1203
 RFC 4336, 1286
 RFC 4340, 1286
 RFC 4960, 1286
 RFC Editor, 1193
 Richarte, G., 1437
 Ritchie (1974), 3, 1443
 Ritchie (1984), 20, 1442
 Ritchie, D.M., 2, 4, 1440, 1442, 1443
 RLIM_INFINITY constant, 736, 756
 RLIM_SAVED_CUR constant, 759
 RLIM_SAVED_MAX constant, 759
rlim_t data type, 65, 756, 759–760
 casting in *printf()* calls, 757
rlimit structure, 756
 definition, 756
 example of use, 758
 RLIMIT_AS resource limit, 757, 760, 1039
 RLIMIT_CORE resource limit, 448, 757,
 760, 789
 RLIMIT_CPU resource limit, 395, 746, 757, 761
 RLIMIT_DATA resource limit, 140, 757, 761
 RLIMIT_FSIZE resource limit, 80, 395, 448,
 757, 760, 761
 RLIMIT_MEMLOCK resource limit, 757, 761,
 1012, 1048–1049, 1051
 RLIMIT_MSGQUEUE resource limit, 757,
 761, 1086
 RLIMIT_NICE resource limit, 736, 757, 762
 RLIMIT_NOFILE resource limit, 78, 217,
 757, 762
 RLIMIT_NPROC resource limit, 217, 516, 757,
 763, 801
 example of use, 759
rlimit_nproc.c, 758
 RLIMIT_RSS resource limit, 757, 763
 RLIMIT_RTPRIO resource limit, 743, 757, 764
 RLIMIT_RTTIME resource limit, 746, 757, 764
 RLIMIT_SIGPENDING resource limit, 458,
 757, 764
 RLIMIT_STACK resource limit, 124, 217, 434,
 436, 682, 757, 764, 793, 1006
rmdir(), 286, 300, 345, 351, 426, 800
 prototype, 351
 Robbins (2003), 630, 1327, 1443
 Robbins, K.A., 1443
 Robbins, S., 1443
 Robins, A.V., xxxix, xl
 Rochkind (1985), xxxv, 1421, 1443
 Rochkind (2004), xxxv, 837, 1421, 1443
 Rochkind, M.J., 1443
 Romanow, J., 1194
 root directory, 27, 340
 of a process, 225, 367–368, 604, 613
 root name server, 1211
root user, 26
 Rosen (2005), 6, 1443
 Rosen, L., 1443
 Rothwell, S., xxxix
 round-robin time-sharing, 733
 router, 1180
 RST control bit (TCP), 1267
rt_tsigqueueinfo(), 685
 RTLD_DEEPBIND constant, 862
 RTLD_DEFAULT constant, 864
 RTLD_GLOBAL constant, 861, 862, 864
 RTLD_LAZY constant, 861
 RTLD_LOCAL constant, 861
 RTLD_NEXT constant, 864
 RTLD_NODELETE constant, 861
 RTLD_NOLOAD constant, 862
 RTLD_NOW constant, 861
 RTO (retransmission timeout), 1191
 RTS/CTS flow control, 1299
 RTSIG_MAX constant, 214, 457
 Rubini, A., 1438
 Rudoff, A.M., 1421, 1444
 RUN_LVL constant, 820, 822
 run-time linker (dynamic linker), 36, 839
rusage structure, 552, 753, 754–755
 definition, 754
rusage.c, 1432
 RUSAGE_CHILDREN constant, 560, 754,
 755, 765
 RUSAGE_SELF constant, 754
 RUSAGE_THREAD constant, 754
rusage_wait.c, 1432
 Rusling, D., 255
 Russell, R., 1439

S

S_IFBLK constant, 282
 S_IFCHR constant, 282
 S_IFDIR constant, 282
 S_IFIFO constant, 282, 907
 S_IFLNK constant, 282
 S_IFMT constant, 281
 S_IFREG constant, 282
 S_IFSOCK constant, 282, 1166
 S_IRGRP constant, 295
 S_IROTH constant, 295
 S_IRUSR constant, 295
 S_IRWXG constant, 295
 S_IRWXO constant, 295
 S_IRWXU constant, 295
 S_ISBLK(), 282
 S_ISCHR(), 282
 S_ISDIR(), 282

S_ISFIFO(), 282
 S_ISGID constant, 295, 351
 S_ISLNK(), 282
 S_ISREG(), 282
 S_ISSOCK(), 282
 S_ISUID constant, 295, 351
 S_ISVTX constant, 295, 300, 351
 S_IWGRP constant, 295
 S_IWOTH constant, 295
 S_IWUSR constant, 295
 S_IXGRP constant, 295
 S_IXOTH constant, 295
 S_IXUSR constant, 295
 sa command, 591
 sa_family_t data type, 65, 1154, 1165, 1202, 1203, 1204
 SA_NOCLDSTOP constant, 417
 SA_NOCLDWAIT constant, 417, 560
 SA_NODEFER constant, 417, 427, 455
 example of use, 455
 SA_NOMASK constant, 417
 SA_ONESHOT constant, 417
 SA_ONSTACK constant, 417, 578
 example of use, 437
 SA_RESETHAND constant, 417, 454
 example of use, 455
 SA_RESTART constant, 417, 443, 486, 941, 944
 example of use, 455, 486
 SA_SIGINFO constant, 417, 437–442, 458, 1352, 1353
 example of use, 463, 501
 Salus (1994), 3, 20, 1443
 Salus (2008), 20, 1443
 Salus, P.H., 1443
 Salzman, P.J., 1442
 Santos, J., 1441
 Sarolahti (2002), 1236, 1443
 Sarolahti, P., 1443
 Sastry, N., 1438
 saved set-group-ID, 170, 173, 177, 613
 saved set-user-ID, 170, 173, 177, 613
 saved-text bit. *See* sticky permission bit
 sbrk(), 140, 761
 example of use, 142
 prototype, 140
 RLIMIT_AS resource limit and, 760
 RLIMIT_DATA resource limit and, 761
 ScalmaZZi, C., xl
 scandir(), 354
 scatter input, 100
 scatter-gather I/O, 99–102
 SCHED_BATCH constant, 740, 742
 SCHED_FIFO constant, 739, 740, 742, 801
 sched_get_priority_max(), 740–741
 prototype, 741
 sched_get_priority_min(), 740–741
 prototype, 741
 sched_getaffinity(), 750
 prototype, 750
 sched_getparam(), 744
 example of use, 745
 prototype, 744
 sched_getscheduler(), 744–745
 example of use, 745
 prototype, 744
 SCHED_IDLE constant, 740, 742
 SCHED_OTHER constant, 738, 742
 sched_param structure, 741–742, 744
 definition, 741
 SCHED_RESET_ON_FORK constant, 615, 746, 801
 SCHED_RR constant, 739, 742, 801
 sched_rr_get_interval(), 747
 prototype, 747
 sched_set.c, 743
 sched_setaffinity(), 749, 801
 prototype, 749
 sched_setparam(), 742, 801
 prototype, 742
 RLIMIT_RTPRIO resource limit and, 764
 sched_setscheduler(), 741–742, 801
 example of use, 743
 prototype, 741
 RLIMIT_NICE resource limit and, 762
 RLIMIT_RTPRIO resource limit and, 764
 sched_view.c, 745
 sched_yield(), 747
 prototype, 747
 Schimmel (1994), 748, 1443
 Schimmel, C., 1443
 Schröder, M., xxxix
 Schüpbach, W.M.M., xl
 Schwaiger, M., xxxix
 Schwartz, A., 1439
 scm_cred_recv.c, 1285
 scm_cred_send.c, 1285
 SCM_CREDENTIALS constant, 800, 801
 scm_rights_recv.c, 1284
 scm_rights_send.c, 1284
 screen command, 1379
 script, 572
 script program
 diagram, 1390
 implementation, 1390–1394
 script.c, 1392
 SCTP (Stream Control Transmission Protocol), 1285, 1444
 search permission, 29
 SECBIT_KEEP_CAPS constant, 615, 812, 813, 816
 SECBIT_KEEP_CAPS_LOCKED constant, 812

SECBIT_NO_SETUID_FIXUP constant, 812, 813
 SECBIT_NO_SETUID_FIXUP_LOCKED constant, 812
 SECBIT_NOROOT constant, 812, 816
 SECBIT_NOROOT_LOCKED constant, 812
 secure programming, 783–796, 1437, 1445
 Secure Sockets Layer (SSL), 1190
 securebits flags, 615, 801, 812–813
 SEEK_CUR constant, 82, 1126
 SEEK_END constant, 82, 1126
 seek_io.c, 84
 SEEK_SET constant, 82, 1126
 segment (virtual memory), 115
 segmentation fault (error message). *See*
 SIGSEGV signal
 SEGV_ACCERR constant, 441
 SEGV_MAPERR constant, 441
 select(), 426, 673, 1331–1334, 1389, 1439
 comparison with *poll()*, 1344–1345
 example of use, 1335, 1393
 interrupted by signal handler, 444
 performance, 1365
 problems with, 1346
 prototype, 1331
 select_mq.c, 1436
 self_pipe.c, 1371
 self-pipe trick, 1370–1372
 SEM_A constant, 923
 sem_close(), 1058, 1093
 prototype, 1093
 sem_destroy(), 1058, 1102–1103
 prototype, 1103
 SEM_FAILED constant, 1090, 1091
 sem_getvalue(), 1058, 1097
 example of use, 1098
 prototype, 1097
 SEM_INFO constant, 952, 993
 sem_init(), 1058, 1100–1101
 example of use, 1102
 prototype, 1100
 SEM_NSEMS_MAX constant, 1104
 sem_open(), 1058, 1090–1091
 example of use, 1093
 prototype, 1090
 sem_post(), 426, 1058, 1096
 example of use, 1097, 1102
 prototype, 1096
 SEM_R constant, 923
 SEM_STAT constant, 952
 sem_t data type, 882, 1058, 1059, 1090,
 1091, 1093, 1094, 1095, 1096,
 1097, 1099, 1100, 1101, 1103
 sem_timedwait(), 673, 1095–1096
 interrupted by signal handler, 444
 interrupted by stop signal, 445
 prototype, 1096
 sem_trywait(), 1095
 prototype, 1095
 SEM_UNDO constant, 986–988
 example of use, 983, 990
 sem_unlink(), 1058, 1093
 example of use, 1094
 prototype, 1093
 SEM_VALUE_MAX constant, 1105
 sem_wait(), 673, 1058, 1094–1095
 example of use, 1095, 1101
 interrupted by signal handler, 444
 interrupted by stop signal, 445
 prototype, 1094
 semadj value (System V semaphore undo
 value), 533, 607, 614, 619, 691,
 693, 986–988, 991
 SEMAEM limit, 991, 992
 semaphore, 881. *See also* POSIX
 semaphore; System V semaphore
 sembuf structure, 978, 979, 980
 definition, 979
 example of use, 981
 semctl(), 922, 969–972
 example of use, 974, 975, 977, 990, 1004
 prototype, 969
 semget(), 922, 969, 991
 example of use, 977, 1003, 1005
 prototype, 969
 semid_ds structure, 922, 970, 971,
 972–973, 976
 definition, 972
 example of use, 973
 seminfo structure, 970, 992, 993
 SEMMNI limit, 991, 992
 SEMMNS limit, 991, 992
 SEMMNU limit, 991
 SEMMSL limit, 991, 992
 semncnt value, 972, 974, 985
 semop(), 922, 971, 972, 973, 978–980, 991
 example of use, 977, 981, 983, 990
 interrupted by signal handler, 444
 interrupted by stop signal, 445
 prototype, 978
 SEMOPM limit, 991, 992
 sempid value, 972, 985
 semtimedop(), 980
 interrupted by signal handler, 444
 interrupted by stop signal, 445
 prototype, 980
 SEMUME limit, 991
 semun union, 969, 970
 definition, 970
 example of use, 973, 974, 976, 977
 semun.h, 970
 SEMVMX limit, 988, 991, 992

semzcnt value, 972, 974, 985
send(), 426, 673, 1259–1260
 interrupted by signal handler, 444
 prototype, 1259
sendfile(), 286, 1260–1263
 diagram, 1261
 prototype, 1261
sendfile.c, 1435
 sending TCP, 1191
sendip command, 1184
sendmmsg(), 1284
sendmsg(), 426, 673, 1284
 interrupted by signal handler, 444
sendto(), 426, 673, 1160–1161
 diagram, 1160
 example of use, 1172, 1173, 1208, 1209, 1241
 interrupted by signal handler, 444
 prototype, 1161
servent structure, 1234
 definition, 1234
 server, 40
 affinity, 1247
 design, 1239–1252
 farm, 1247
 load balancing, 1247
 pool, 1246
 service name, 1204, 1212
 session, 39, 700, 704–706
 diagram, 701
 leader, 39, 700, 705
 session ID, 39, 613, 700, 705, 819
set_mempolicy(), 615
set_thread_area(), 607, 692
 SETALL constant, 971, 972, 973, 987
 example of use, 975
setbuf(), 238, 532
 prototype, 238
setbuffer(), 238
 prototype, 238
setcontext(), 442
setdomainname(), 229, 801
setegid(), 174–175, 181, 785, 800
 prototype, 174
setenv C shell command, 125
setenv(), 128–130, 657, 1426
 example of use, 131
 prototype, 128
setenv.c, 1426
seteuid(), 174–175, 181, 784, 801
 prototype, 174
setfacl command, 326
setfattr command, 312
setfsuid(), 178, 181, 800
 example of use, 182
 prototype, 178
setfsuid(), 178, 181, 801
 example of use, 182
 prototype, 178
setgid(), 173–174, 181, 426, 786, 800
 prototype, 173
setgrent(), 161, 657
 set-group-ID permission bit, 168, 291, 292, 294, 295, 300, 304, 351, 564, 788, 800, 1138, 1432
 propagated from parent directory to new subdirectory, 351
 set-group-ID program, 146, 147, 168–170, 266, 564, 569, 581, 615, 784, 854, 874, 875
 core dump files and, 449
 dropping and reacquiring privileges, 784
 dropping privileges permanently, 785
setgroups(), 179–180, 181, 800
 prototype, 179
sethostname(), 229, 801
setitimer(), 16, 390, 392, 395, 479–481, 485, 486, 488, 691, 694
 example of use, 484
 prototype, 480
setjmp(), 131–135
 example of use, 134, 136, 433
 handling of signal mask, 429
 prototype, 132
 restrictions on use of, 134–135
setjmp_vars.c, 136
setkey(), 657
setlocale(), 203
 example of use, 199
 prototype, 203
setlogmask(), 780–781
 prototype, 781
setpgid(), 426, 691, 693, 702–704
 example of use, 703, 711, 713
 prototype, 702
setpgrp(), 704
setpriority(), 691, 735–736, 801
 example of use, 737
 prototype, 735
setpwnent(), 160–161, 657
 prototype, 161
setregid(), 175–176, 181, 800
 prototype, 175
setresgid(), 177–178, 181, 800
 prototype, 177
setresuid(), 177–178, 181, 801
 prototype, 177

setreuid(), 175–176, 181, 786, 801
 prototype, 175
setrlimit(), 755–757, 801
 example of use, 759
 prototype, 756
setrlimit64(), 105
setsid(), 426, 691, 693, 705, 768, 1377
 example of use, 706, 770, 1387
 prototype, 705
setsockopt(), 426, 1278–1279
 example of use, 1222
 prototype, 1278
setspent(), 161
 prototype, 161
settimeofday(), 204–205, 801
 diagram, 188
 prototype, 204
 set-UID bit. *See* set-user-ID permission bit
 set-UID program. *See* set-user-ID program
setuid(), 173–174, 181, 426, 801
 prototype, 173
 set-user-ID permission bit, 168, 292, 294,
 295, 300, 564, 788, 800, 1432
 set-user-ID program, 33, 129, 146, 147,
 168–170, 266, 564, 569, 581, 615,
 690, 718, 784, 854, 874, 875
 core dump files and, 449
 dropping and reacquiring
 privileges, 784
 dropping privileges permanently, 785
 set-user-ID-root program, 169, 783
setutxent(), 657, 821
 example of use, 824, 829
 prototype, 821
 SETVAL constant, 971, 972, 973, 987
 example of use, 990
setubuf(), 237–238, 532
 prototype, 237
setxattr(), 286, 314–315, 329, 345
 prototype, 314
 Seventh Edition UNIX, 3
 SFD_CLOEXEC constant, 472
 SFD_NONBLOCK constant, 472
sh (Bourne shell), 25
 shadow group file, 156
 shadow password file, 155
 retrieving records from, 161–162,
 164–165
 shared library, 35, 1439
 compared with static library, 856
 compatibility, 850
 controlling symbol visibility, 867–870
 creating, 837–838, 841–842
 diagram, 842
 dependency tree, 860
 dynamic dependency list, 839
 dynamic loading, 859–867
 —export-dynamic linker option, 867
 finalization (destructor) function,
 872–873
 finding at run time, 854
 initialization (constructor) function,
 872–873
 installing, 847–849
 interdependencies
 diagram, 852
 linker name, 845, 846
 loading run-time, *diagram*, 843
 major version, 844
 minor version, 844
 names, 846–848
 diagram, 846
 overview, 836–837
 preloading, 873–874
 real name, 840, 846
 —rpath linker option, 851–854
 soname, 840–843, 846–847
 symbol resolution at run time, 854–856
 upgrading, 850–851
 using, 839–840
 versions and naming conventions,
 844–847
 shared memory, 880. *See also*
 memory mapping;
 POSIX shared memory;
 System V shared memory
 shared object. *See* shared library
 shared subtree, 267, 1445
 shell, 24–25
 shell command execution, 579–582
 SHELL environment variable, 125, 154
 example of use, 1392
 shell layers, 1300
 shell script, 25
 SHM_DEST constant, 1013
 SHM_HUGETLB constant, 800, 999
 SHM_INFO constant, 952, 1015
shm_info structure, 1015
 SHM_LOCK constant, 800, 1012, 1048, 1050
 SHM_LOCKED constant, 1013
 SHM_NORESERVE constant, 999
shm_open(), 801, 1058, 1109–1110
 example of use, 1111, 1112, 1113
 prototype, 1109
 RLIMIT_NOFILE resource limit and, 762
 SHM_R constant, 923
 SHM_RDONLY constant, 1000, 1001
 SHM_REMAP constant, 1000, 1001
 SHM_RND constant, 999, 1001
 SHM_STAT constant, 952

shm_unlink(), 1058, 1114
 example of use, 1114
 prototype, 1114
 SHM_UNLOCK constant, 800, 1012
 SHM_W constant, 923
 SHMALL limit, 1014, 1015
shmat(), 922, 999–1001, 1013, 1014
 example of use, 1004, 1005
 prototype, 999
 RLIMIT_AS resource limit and, 760
shmatt_t data type, 65, 1012, 1014
shmctl(), 922, 1011–1012
 example of use, 1004
 prototype, 1011
 RLIMIT_MEMLOCK resource limit and, 761
shmdt(), 922, 1000–1001, 1013, 1014
 example of use, 1004, 1005
 prototype, 1001
shmget(), 922, 998–999, 1014
 example of use, 1004, 1005
 prototype, 998
shmid_ds structure, 922, 1011, 1012–1013
 definition, 1012
shminfo structure, 1015
 SHMLBA constant, 999, 1001
 SHMMAX limit, 1014, 1015
 SHMMIN limit, 1014
 SHMMNI limit, 1014, 1015
 SHMSEG limit, 1014
show_time.c, 199
 Shukla, A., 1439
 SHUT_RD constant, 1256, 1273
 SHUT_RDWR constant, 1256, 1273
 SHUT_WR constant, 1256, 1273
 example of use, 1258
shutdown(), 426, 1256–1257
 example of use, 1258
 prototype, 1256
 on TCP socket, 1273
 SI_ASYNCIO constant, 441
 SI_KERNEL constant, 440, 441
 SI_MESGQ constant, 441, 1079
 SI_QUEUE constant, 441, 460
 example of use, 462
 SI_SIGIO constant, 440, 441
 SI_TIMER constant, 441, 500
 SI_TKILL constant, 441
 SI_USER constant, 441
 example of use, 462
 SID (session ID), 39, 613, 700, 705, 819
 SIG_ATOMIC_MAX constant, 428
 SIG_ATOMIC_MIN constant, 428
sig_atomic_t data type, 65, 428
 example of use, 432, 466, 774
 SIG_BLOCK constant, 410
 example of use, 409, 411
 SIG_DFL constant, 398, 412, 416, 578
 SIG_ERR constant, 397, 398, 456
 example of use, 399, 455
 SIG_HOLD constant, 475
 SIG_IGN constant, 398, 412, 416, 419, 578
sig_receiver.c, 414, 419
sig_sender.c, 412
 SIG_SETMASK constant, 410
 example of use, 411, 415
sig_speed_sigsuspend.c, 478
 SIG_UNBLOCK constant, 410
 SIGABRT signal, 390, 392, 396, 433
sigaction structure, 416–417, 437–438
 definition, 416, 437
 example of use, 425, 433
sigaction(), 416–417, 426, 604
 example of use, 433, 437, 452, 455, 463, 587
 prototype, 416
sigaddset(), 407, 426
 example of use, 411, 466
 prototype, 407
 SIGALRM signal, 390, 396, 480, 484, 486, 488
 example of use, 483, 487
sigaltstack(), 417, 434–435, 578, 691, 693
 example of use, 437
 prototype, 434
sigandset(), 408
 prototype, 408
sigblock(), 476–477
 prototype, 476
 SIGBUS signal, 390, 396, 439, 440, 441, 453, 683, 1021, 1030
 correct handling of, 452
 diagram, 1030
 SIGCHLD signal, 390, 391, 396, 440, 441, 514, 545, 551, 555–561, 583, 590, 605, 609, 697, 717, 755, 1431
 change of disposition across *exec()*, 578
 contrasted with System V SIGCLD, 561
 delivery for resumed children, 559
 delivery for stopped children, 559
 designing handler for, 556
 diagram, 515
 disabling generation for stopped child processes, 417
 example of use, 558
 handling, 555–559
 ignoring, 559–561
 SIGCLD signal, 391, 561
 SIGCONT signal, 391, 396, 450, 489, 544, 545, 546, 550, 559, 717, 718, 720, 727
 diagram, 717
 establishing handler for, 478

- example of use*, 728
 - sent to foreground process group
 - when controlling process terminates, 707, 712–714
 - sent to orphaned process group
 - containing stopped processes, 533, 727
- `sigdelset()`, 407, 426
 - example of use*, 463
 - prototype*, 407
- `sigemptyset()`, 407, 426
 - example of use*, 411, 415, 466
 - prototype*, 407
- SIGEMT signal, 391, 396, 397, 453
- SIGEV_NONE constant, 496, 1079
- SIGEV_SIGNAL constant, 496, 497, 499–503, 1079
 - example of use*, 501, 1081
- SIGEV_THREAD constant, 496, 497, 504–507, 1079
 - example of use*, 507, 1083
- SIGEV_THREAD_ID constant, 496, 497
- `sigevent` structure, 495, 496–497, 1078–1079
 - definition*, 496, 1079
 - example of use*, 501, 506, 1080, 1082
- `sigfillset()`, 407, 426
 - example of use*, 415, 463
 - prototype*, 407
- SIGFPE signal, 391, 396, 439, 440, 441, 453, 683
 - correct handling of, 452
- `sighandler_t` data type, 398
- `sighold()`, 475
 - prototype*, 475
- SIGHUP signal, 39, 391, 396, 451, 700, 706, 709–714, 725–729, 772–775
 - example of use*, 711, 713, 728, 774
 - handling by job-control shells, 710–712
 - sent on closure of master side of pseudoterminal, 1388
 - sent on terminal disconnect, 709
 - sent to foreground process group
 - when controlling process terminates, 533, 707, 712–714
 - sent to orphaned process group
 - containing stopped processes, 533, 727
 - sent when master side of pseudoterminal is closed, 709
 - stopped shell background jobs and, 710
 - used to reinitialize daemon, 772–775
- `sigignore()`, 475–476
 - prototype*, 475
- SIGILL signal, 391, 396, 439, 440, 441, 453, 683
 - correct handling of, 452
- SIGINFO signal, 391, 1299
- `siginfo_t` structure, 65, 437, 438–442, 460, 468, 471, 472, 499–500, 550, 551–552, 1079, 1353–1354
 - definition*, 438
 - example of use*, 462, 470, 500, 552
- SIGINT signal, 392, 396, 451, 583, 700, 720, 725, 1296, 1297, 1302, 1304
 - example of use*, 399, 401
- `siginterrupt()`, 16, 419, 444–445
 - prototype*, 445
- `siginterrupt.c`, 1429
- SIGIO signal, 392, 396, 397, 440, 441, 1347
- SIGIOT signal, 392
- `sigisemptyset()`, 408
 - prototype*, 408
- `sigismember()`, 407, 426
 - example of use*, 409
 - prototype*, 407
- SIGKILL signal, 392, 393, 396, 411, 450, 761, 764, 772, 1040
 - disposition can't be changed, 450
- `siglongjmp()`, 151, 429–430, 452
 - example of use*, 432
 - prototype*, 430
- SIGLOST signal, 392
- `sigmask()`, 476–477
 - prototype*, 476
- `sigmask_longjmp.c`, 432
- signal, 37, 387–478
 - accepting, 468
 - asynchronous generation, 453
 - blocked, 38, 388, 389
 - blocking, 410–411
 - broadcast, 402
 - BSD API, 476–477
 - caught, 389
 - default action, 389, 390–397
 - delivery, 388
 - diagram*, 399, 454
 - order when multiple signals
 - pending, 454
 - disposition, 389, 613
 - changing, 397–398, 416–417
 - of pending signal, 412
 - default, 389
 - generation, 388
 - handler. *See* signal handler
 - hardware-generated, 452
 - ignored, 389, 398
 - job-control, 717
 - LinuxThreads nonconformances, 690

signal, *continued*
 list of all signals, 390–397
 mask, 38, 388, 410, 578, 613, 683
 names (descriptions), 406
 pending, 38, 388, 389, 411–415, 578, 613, 683
 permission required for sending, 402–403, 800
 diagram, 403
 queuing, 412–414, 422, 456, 457
 reading via a file descriptor, 471–474
 realtime. *See* realtime signal
 reliable, 390, 455
 semantics in multithreaded process, 682–683
 sending, 401–405
 synchronous generation, 453
 System V API, 475–476
 timing and order of delivery, 453–454, 464
 unreliable, 454
 used for IPC, 474
 used for synchronization, 527–529
 waiting for, 418, 464–471
 signal catcher. *See* signal handler
 signal handler, 38, 389, 398–401, 421–446
 design, 422–428
 diagram, 399, 454
 employing *printf()* in example programs, 427
 invocation in multithreaded process, 683
 terminating, 428–433
 terminating process from, 549–550
 use of *errno* within, 427
 use of global variables within, 428
 use of nonlocal goto within, 429–433
 signal mask, 38, 388, 410, 578, 613, 683
 signal set, 65, 406–409. *See also* *sigset_t* data type
 signal stack, alternate, 65, 434–437, 578, 613, 683, 691, 693, 764
signal(), 397–398, 426, 604
 code of implementation, 455
 example of use, 399, 401, 415
 obsolete in favor of *sigaction()*, 456
 portability problems, 454–456
 prototype, 397
 System V, 475
 signal.c, 455
 signal_functions.c, 408
 signal-driven I/O, 75, 95, 1327, 1346–1355, 1367
signalfd(), 471–472
 example of use, 473
 prototype, 471
 signalfd_siginfo structure, 472
 definition, 472
 example of use, 473
 signalfd_sigval.c, 473
 sigorset(), 408
 prototype, 408
 sigpause(), 426, 475–477, 673, 674
 prototype (BSD), 476
 prototype (System V), 475
 sigpending(), 411–412, 426, 683
 example of use, 409, 415
 prototype, 411
 SIGPIPE signal, 392, 396, 683, 895, 903, 912, 918, 1159, 1220, 1256, 1260
 example of use, 913
 SIGPOLL signal, 392, 441
 sigprocmask(), 410–411, 426, 684
 example of use, 409, 411, 415, 466, 473, 587
 prototype, 410
 SIGPROF signal, 392, 396, 480
 SIGPWR signal, 391, 392, 396
 sigqueue(), 426, 439, 441, 458–460, 800
 example of use, 459
 prototype, 458
 RLIMIT_SIGPENDING resource limit and, 764
 SIGQUEUE_MAX constant, 214, 457
 SIGQUIT signal, 393, 396, 451, 583, 700, 725, 1296, 1298
 example of use, 401
 sigrelse(), 475
 prototype, 475
 SIGRTMAX constant, 457
 SIGRTMIN constant, 457
 SIGSEGV signal, 120, 140, 146, 151, 393, 396, 439, 440, 441, 453, 523, 683, 764, 1000, 1021, 1030, 1046, 1051
 correct handling of, 452
 delivering on an alternate signal stack, 434–435
 diagram, 1029, 1030
 example of use, 437
 sigset(), 426, 475
 prototype, 475
 sigset_t data type, 65, 407, 408, 410, 411, 416, 437, 465, 468, 471, 684, 685, 1369
 example of use, 411, 415, 463, 464
 sigsetjmp(), 429–430
 example of use, 433
 prototype, 430
 sigsetmask(), 476–477
 prototype, 476

SIGSTKFLT signal, 393, 396
 SIGSTKSZ constant, 435
 example of use, 437
 SIGSTOP signal, 393, 396, 411, 445, 450,
 716, 717, 790
 diagram, 717
 disposition can't be changed, 450
 sigsuspend(), 426, 465, 673
 example of use, 467
 prototype, 465
 SIGSYS signal, 393, 396
 SIGTERM signal, 393, 396, 772
 sigtimedwait(), 471, 673
 interrupted by stop signal, 445
 prototype, 471
 SIGTRAP signal, 394, 396, 442
 SIGTSTP signal, 394, 396, 445, 450, 451,
 700, 715, 717, 720, 725, 790,
 1296, 1299, 1312
 diagram, 717
 example of use, 724, 1313, 1315
 handling within applications, 722
 orphaned process group and, 730
 SIGTTIN signal, 394, 396, 445, 450, 451,
 717, 718, 725
 diagram, 717
 orphaned process group and, 730
 SIGTTOU signal, 394, 396, 445, 450, 451,
 717, 718, 725, 1293, 1303
 diagram, 717
 orphaned process group and, 730
 SIGUNUSED signal, 394
 SIGURG signal, 394, 396, 397, 1283
 SIGUSR1 signal, 394, 396
 used by LinuxThreads, 690
 SIGUSR2 signal, 395, 396
 used by LinuxThreads, 690
 signal union, 459, 496, 1078
 sigval_t data type, 459
 sigvec structure, 476
 definition, 476
 sigvec(), 476
 prototype, 476
 SIGVTALRM signal, 395, 396, 480
 sigwait(), 685–686, 673
 prototype, 685
 sigwaitinfo(), 468, 673
 example of use, 470
 interrupted by stop signal, 445
 prototype, 468
 SIGWINCH signal, 395, 396, 1319, 1320, 1395
 example of use, 1320
 SIGXCPU signal, 395, 396, 746, 761, 764
 SIGXFSZ signal, 395, 396, 761
 simple_pipe.c, 896
 simple_system.c, 582
 simple_thread.c, 626
 single directory hierarchy, *diagram*, 27
 Single UNIX Specification (SUS), 13
 version 2 (SUSv2), 13, 17
 version 3 (SUSv3), 13–15, 17, 1440
 Technical Corrigenda, 14
 version 4 (SUSv4), 15–17
 SIOCGPRP constant, 1350
 SIOCSPGRP constant, 1350
 size command, 116
 size_t data type, 65, 66, 79, 80, 98, 99, 141,
 148, 149, 150, 179, 193, 237, 238,
 314, 315, 316, 350, 363, 435, 749,
 750, 941, 943, 998, 1012, 1020,
 1023, 1031, 1037, 1041, 1046,
 1049, 1051, 1054, 1073, 1075,
 1077, 1161, 1200, 1206, 1214,
 1218, 1254, 1259, 1261
 sleep(), 426, 487–488, 673
 interrupted by signal handler, 444
 prototype, 488
 sleeping, 487–494
 high-resolution, 488–491, 493–494
 sliding window (TCP), 1192
 slow-start algorithm (TCP), 1193, 1194
 Smith, M., xli
 Snader (2000), 1235, 1275, 1443
 Snader, J.C., xl, 1443
 SO_RCVBUF constant, 1192
 SO_REUSEADDR constant, 1220, 1279–1281
 example of use, 1222, 1229, 1281
 SO_SNDBUF constant, 1171
 SO_TYPE constant, 1279
 SOCK_CLOEXEC constant, 1153, 1158, 1175
 SOCK_DGRAM constant, 1152
 example of use, 1172, 1208
 SOCK_NONBLOCK constant, 1153, 1158, 1175
 SOCK_RAW constant, 1153, 1184
 SOCK_SEQPACKET constant, 1285
 SOCK_STREAM constant, 1151
 example of use, 1168, 1169, 1173, 1209,
 1221, 1224
 sockaddr structure, 1153, 1154–1155, 1157,
 1158, 1161
 definition, 1154
 sockaddr_in structure, 1151, 1202
 definition, 1202
 sockaddr_in6 structure, 1151, 1202–1203
 definition, 1203
 example of use, 1208, 1209
 sockaddr_storage structure, 1204
 definition, 1204
 example of use, 1221, 1241

- sockaddr_un* structure, 1151, 1165–1166
 - definition*, 1165
 - example of use*, 1168, 1176
- socketmark()*, 426
- socket, 282, 392, 883, 1149–1163
 - abstract binding, 1175
 - accepting a connection, 1157
 - active, 1155
 - active close (TCP), 1272
 - address structure, 1154
 - asynchronous error, 1254, 1351, 1352
 - binding to an address, 1153
 - broadcasting, 800, 1282
 - connecting to peer, 1158
 - connection termination, 1159
 - connectionless, 1152
 - connection-oriented, 1152
 - creating, 1153
 - datagram, 1152, 1159–1162
 - sending and receiving, 1160
 - domain, 1150
 - half-close, 1256
 - identified by 4-tuple, 1280
 - Internet domain, 882, 886, 1150, 1197–1237
 - address structure, 1202–1204
 - maximum datagram size, 1190
 - I/O system calls, 1259–1260
 - listening for connections, 1156
 - local, 1152
 - multicasting, 800, 1282
 - options, 1278–1279
 - out-of-band data, 394, 1259, 1260, 1283, 1288, 1331, 1343
 - pair, 1174
 - partial reads and writes, 1254–1255
 - passing credentials via, 800, 801, 1284–1285
 - passing file descriptor via, 1284
 - passive, 1155
 - passive close (TCP), 1272
 - peer, 1152
 - pending connection, 1156
 - poll()* on, 1343
 - port number. *See* port number
 - raw, 800, 1184
 - receive buffer, 1276
 - diagram*, 1190
 - remote, 1152
 - select()* on, 1343
 - send buffer, 1276
 - diagram*, 1190
 - sequenced-packet, 1285
 - stream, 1151, 1155–1159
 - I/O, 1159
 - type, 1151
 - UNIX domain, 882, 884, 886, 1150, 1165–1177
 - address structure, 1165–1167
 - maximum datagram size, 1171
 - socket permissions, 1174
- socket()*, 426, 801, 1150, 1152, 1153
 - diagram*, 1156, 1160
 - example of use*, 1166, 1169, 1172, 1173, 1208, 1209, 1221, 1224, 1228
 - prototype*, 1153
 - RLIMIT_NOFILE resource limit and, 762
- socketcall()*, 1152
- socketpair()*, 426, 1174–1175
 - prototype*, 1175
- socklen_t* data type, 65, 1153, 1154, 1157, 1158, 1161, 1218, 1231, 1263, 1278
- socknames.c, 1265
- soft link. *See* symbolic link
- soft realtime, 738
- software clock, 205–206
- SOL_SOCKET constant, 1278
- Solaris, 4
- SOMAXCONN constant, 1157
- soname, shared library, 840–843, 846–847
- source code (of example programs), xli
- Spafford, G., 1439
- sparse array, 1038
- spawn, 514
- Spec 1170, 13, 17
- speed_t* data type, 65, 1292, 1316, 1317
- splice()*, 1262
- Spraul, M., xxxix
- spurious readiness notification, 1330
- spurious wake-up, 648
- spwd* structure, 161, 162
 - definition*, 162
 - example of use*, 164, 810
- SS_DISABLE constant, 435
- SS_ONSTACK constant, 435
- ssh* program, 1378
- ssize_t* data type, 65, 66, 79, 80, 98, 99, 102, 315, 316, 350, 943, 1075, 1077, 1161, 1259, 1261
- SSL (Secure Sockets Layer), 1190
- stack, 31, 116, 121–122, 612, 764, 1051
 - diagram*, 122
 - direction of growth, 121
 - resource limit on size of, 764
 - unwinding, 133
- stack crashing, 792
- stack frame, 116, 121–122, 133, 151
- stack pointer, 121, 133, 150

stack_t data type, 65, 434, 435
 example of use, 436
 Stallman, R.M., 5, 6, 11, 20, 1445
 standard error, 30
 standard input, 30
 standard output, 30
 START terminal special character, 1296, 1298, 1319
stat structure, 279, 280–283
 definition, 280
 example of use, 284
stat(), 106, 279–283, 325, 345, 426, 907, 1428
 example of use, 285, 303
 prototype, 279
stat64 structure, 105
stat64(), 105
statfs(), 277, 345
 static (used to control symbol visibility), 867
 static library, 35, 834–836
 use in preference to a shared library, 856
 static linking, 840
 statically allocated variable, 116
 function reentrancy and, 423
 STATUS terminal special character, 1299
statvfs structure, 276–277
 definition, 276
statvfs(), 276–277, 345
 prototype, 276
stderr variable, 30, 70
 STDERR_FILENO constant, 70
stdin variable, 30, 70
 STDIN_FILENO constant, 70
stdio buffers, 237–239
 diagram, 244
 fork() and, 537–538
stdio library, 30
 mixing use with I/O system calls, 248
stdout variable, 30, 70
 STDOUT_FILENO constant, 70
 Steele, G.L., 1440
 Stevens (1992), 1322, 1421, 1443, 1444
 Stevens (1994), 1190, 1210, 1235, 1256, 1267, 1268, 1272, 1443
 Stevens (1996), 1282, 1444
 Stevens (1998), 1443
 Stevens (1999), 20, 975, 1087, 1105, 1108, 1143, 1146, 1421, 1443
 Stevens (2004), 1151, 1162, 1184, 1188, 1203, 1210, 1213, 1246, 1254, 1270, 1272, 1275, 1278, 1279, 1282, 1283, 1285, 1286, 1328, 1330, 1374, 1421, 1444
 Stevens (2005), 20, 30, 222, 487, 527, 561, 731, 821, 1118, 1146, 1383, 1421, 1444
 Stevens, D.L., 1438
 Stevens, W.R., xl, 1194, 1421, 1443, 1444, 1445
 Stewart (2001), 1286, 1444
 Stewart, R.R., 1444
 sticky permission bit, 294, 295, 300, 800
 acting as restricted deletion flag, 300
 user extended attributes and, 313
 STICKY_TIMEOUTS constant, 1334
stime(), 204, 801
 diagram, 188
 St. Laurent (2004), 6, 1443
 St. Laurent, A.M., 1443
 Stone (2000), 1190, 1444
 Stone, J., 1444
 stop signal, 450
 STOP terminal special character, 1296, 1298, 1299, 1319
strace command, 394, 1401–1403
 Strang (1986), 1290, 1444
 Strang (1988), 1289, 1444
 Strang, J., 1442, 1444
strcoll(), 202
 Stream Control Transmission Protocol (SCTP), 1285, 1444
 stream pipe, 890, 1175
 STREAM_MAX constant, 214, 215
 STREAMS (System V), 86, 237, 1338
strerror(), 50, 657
 prototype, 50
strerror.c, 664
strerror_r(), 658
strerror_test.c, 665
strerror_tls.c, 669
strerror_tsd.c, 666
strftime(), 193, 194, 198, 203
 diagram, 188
 example of use, 195, 197, 199
 prototype, 193
strip command, 834
strncpy(), 793
strncpy(), 793
 Strongman, K., xl
strptime(), 195–196
 diagram, 188
 example of use, 197
 prototype, 195
strsignal(), 15, 406, 656
 example of use, 409
 prototype, 406
strtime.c, 197
strtok(), 657

- strtok_r()*, 658
- strxfrm()*, 202
- stty* command, 1294–1295
- su* command, 169
- subnet, 1179
- subnet broadcast address, 1187
- subnet mask, 1187
 - diagram*, 1187
- subnetted address (IP), 1187, 1193
- Suisted, R., xl
- Sun Microsystems, 4
- SunOS, 4
- superblock, 256
- superuser, 26
- supplementary group IDs, 33, 172, 178–180, 613
- SUS. *See* Single UNIX Specification (SUS)
- useconds_t* data type, 65, 186, 480, 1333
- SUSP terminal special character, 1296, 1299, 1303, 1305
- suspend* character, 394, 1296, 1299
- SV_INTERRUPT constant (BSD), 476
- SVID (System V Interface Definition), 17, 62
- svmsg_chqbytes.c*, 949
- svmsg_create.c*, 938
- svmsg_demo_server.c*, 930
- svmsg_file.h*, 956
- svmsg_file_client.c*, 960
- svmsg_file_server.c*, 957
- svmsg_info.c*, 952
- svmsg_ls.c*, 953
- svmsg_receive.c*, 945
- svmsg_rm.c*, 947
- svmsg_send.c*, 941
- SVR4 (System V Release 4), 4, 17, 1440
- svsem_bad_init.c*, 976
- svsem_create.c*, 984
- svsem_demo.c*, 968
- svsem_good_init.c*, 977
- svsem_info.c*, 993
- svsem_mon.c*, 973
- svsem_op.c*, 982
- svsem_rm.c*, 985
- svsem_setall.c*, 974
- svshm_attach.c*, 1007
- svshm_create.c*, 1007
- svshm_info.c*, 1015
- svshm_mon.c*, 1434
- svshm_rm.c*, 1007
- svshm_xfr.h*, 1002
- svshm_xfr_reader.c*, 1005
- svshm_xfr_writer.c*, 1003
- swap area, 119, 254
- swap space overcommitting, 1038–1040
- swapcontext()*, 442
- swapoff()*, 254, 345, 801
- swapon()*, 254, 345, 801
- Sweet, M., 1322
- Swift, J., 1198
- Swigg, T., xxxix
- SWTCH terminal special character, 1300
- symbol relocation, 837
- symbol versioning, 870–872
- symbolic link, 28, 77, 282, 342–344
 - changing ownership of, 292
 - creating, 342, 349
 - dangling, 28, 342, 349, 360
 - diagram*, 343
 - following (dereferencing), 28
 - interpretation by system calls, 344
 - permissions and ownership, 344
 - reading contents of, 349
 - representation in file system, 342
- symlink()*, 286, 349, 426
 - prototype*, 349
- SYMLINK_MAX constant, 350
- symlinkat()*, 365, 426
- SYN control bit (TCP), 1267
- SYN_RECV state (TCP), 1269
- SYN_SENT state (TCP), 1269
- sync()*, 241, 242, 1032
 - prototype*, 241
- sync_file_range()*, 241, 1027
- synchronized I/O completion, 239
- synchronized I/O data integrity completion, 240
- synchronized I/O file integrity completion, 240
- synchronous I/O, 241–243
- SYN-flooding, 1185, 1441
- sys_siglist* array, 406
- sysconf()*, 215–216, 425, 426
 - example of use*, 216
 - prototype*, 215
- sysfs* file system, 252, 1442
- syslog* logging facility, 775–782
- syslog()*, 776, 779–780
 - diagram*, 775
 - example of use*, 780, 1241, 1244, 1245, 1251
 - prototype*, 779
- syslog(2)* system call, 776, 801
- syslogd* daemon, 776
 - diagram*, 775
- system call, 23, 43–46
 - diagram*, 46
 - error handling, 48–50
 - interrupted by signal handler, 442–445
 - interrupted by stop signal plus SIGCONT, 445

- restarting, 442–445
 - setting timeout on, 486–487
 - system clock, updating, 204–205
 - system CPU time, 40, 206
 - system data types, 63–66
 - casting in *printf()* calls, 66
 - system limits, SUSv3, 212–215
 - indeterminate limits, 219
 - retrieving, 215–217
 - file-related limits, 217–218
 - system options, SUSv3, 219–221
 - retrieving, 215–217
 - file-related options, 217–218
 - system programming, xxxi
 - System V, 4
 - System V Interface Definition (SVID), 17, 62
 - System V IPC, 921–936
 - algorithm employed by *get* calls, 931–933
 - compared with POSIX IPC, 1061–1062
 - control operations, 924
 - design problems, 884
 - identifier, 923, 931
 - key, 64, 923, 925–927
 - limits, 935–936
 - object
 - associated data structure, 927–929
 - diagram*, 932
 - creating, 923–924
 - deleting, 924
 - listing, 934–935
 - permissions, 800, 927–929
 - persistence, 924
 - re-creating after server crash, 930
 - removing, 934
 - portability, 884, 1061
 - System V message queue, 882, 883, 886, 937–964
 - associated data structure, 948–950
 - compared with POSIX message queue, 1086–1087
 - control operations, 947
 - creating, 938–940
 - deleting, 947
 - disadvantages, 961–962
 - limits, 950–951
 - messages, 940
 - receiving, 943–946
 - nonblocking, 943
 - sending, 940–942
 - nonblocking, 941
 - use in client-server applications, 953–961
 - System V Release 4 (SVR4), 4, 17, 1440
 - System V semaphore, 882, 886, 965–995
 - adjustment on process termination, 533
 - associated data structure, 972–973
 - compared with POSIX semaphore, 1103–1104
 - control operations, 969–972
 - creating, 969
 - deleting, 971
 - disadvantages, 993
 - initialization, 971, 974, 975–978
 - limits, 991–993
 - order of handling of multiple blocked operations, 986
 - performing operations on, 978–983
 - starvation, 986
 - undo value (*semadj*), 533, 607, 614, 619, 691, 693, 986–988, 991
 - System V shared memory, 614, 882, 886, 997–1016
 - associated data structure, 1012–1014
 - attaching, 999
 - compared with other shared memory APIs, 1115–1116
 - control operations, 1011–1012
 - creating, 998–999
 - deleting, 1011
 - detaching, 1000
 - on process termination, 533
 - limits, 1014–1015
 - location in process virtual memory, 1006–1009
 - locking into memory, 1012
 - storing pointers in, 1010
 - system()*, 582–588, 656, 673
 - avoid in privileged programs, 788
 - code of implementation*, 582–583, 586–587
 - diagram*, 584
 - example of use*, 581
 - implementation, 582–588
 - prototype*, 579
 - system.c*, 586
 - sys_signal()*, 456
 - prototype*, 456
- ## T
- t_ prefix (in names of example programs), 100
 - t_chown.c, 293
 - t_clock_nanosleep.c, 1429
 - t_clone.c, 601
 - t_dirbasename.c, 371
 - t_exec1.c, 571
 - t_execle.c, 570

t_execlp.c, 570
 t_execve.c, 566
 t_flock.c, 1121
 t_fork.c, 517
 t_fpathconf.c, 218
 t_ftok.c, 1433
 t_gethostbyname.c, 1233
 t_getopt.c, 1408
 t_getservbyname.c, 1235
 t_kill.c, 405
 t_mmap.c, 1028
 t_mount.c, 268
 t_mprotect.c, 1046
 t_nanosleep.c, 490
 t_readv.c, 101
 t_sched_getaffinity.c, 750
 t_sched_setaffinity.c, 750
 t_select.c, 1334
 t_setpriority.c, 736
 t_setsid.c, 706
 t_sigaltstack.c, 436
 t_sigqueue.c, 459, 461
 t_sigsuspend.c, 466
 t_sigwaitinfo.c, 470
 t_stat.c, 284
 t_statfs.c, 277
 t_statvfs.c, 277
 t_sysconf.c, 216
 t_syslog.c, 1432
 t_system.c, 581
 t_umask.c, 302
 t_uname.c, 230
 t_unlink.c, 347
 t_utimes.c, 288
 t_vfork.c, 524
 TAB0 constant, 1302
 TAB1 constant, 1302
 TAB2 constant, 1302
 TAB3 constant, 1302, 1303
 TABDLY constant, 1302, 1303
 Tanenbaum (2002), 1235, 1444
 Tanenbaum (2006), 24, 1422, 1444
 Tanenbaum (2007), 24, 138, 278, 630, 1147, 1444
 Tanenbaum, A.S., 6, 1444
 TASK_INTERRUPTIBLE process state, 451
 TASK_KILLABLE process state, 451
 TASK_UNINTERRUPTIBLE process state, 451
 TASK_UNMAPPED_BASE constant, 1006
 Taylor, I.L., 1444
 tcdrain(), 426, 673, 718, 727, 1293, 1316–1317
 prototype, 1318
 tcf_{lag}_t data type, 65, 1292
 tcflow(), 426, 718, 727, 1293, 1316–1317
 prototype, 1318
 tcflush(), 426, 718, 727, 1293, 1316–1318
 prototype, 1318
 tcgetattr(), 426, 1291–1292
 example of use, 1301, 1306, 1310, 1311, 1313, 1314, 1392
 prototype, 1291
 tcgetpgrp(), 426, 708–709
 example of use, 713, 720
 prototype, 708
 tcgetsid(), 706
 TCIFLUSH constant, 1318
 TCIOFF constant, 1319
 TCIOFLUSH constant, 1318
 TCION constant, 1319
 TCOFLUSH constant, 1318
 TC00FF constant, 1319
 TCOON constant, 1319
 TCP (Transmission Control Protocol), 1152, 1190–1193, 1194, 1266–1275, 1439
 acknowledgements, 1191, 1267, 1268
 diagram, 1268
 checksum, 1267
 connection establishment, 1191, 1270–1272
 diagram, 1272
 connection termination, 1272–1273
 diagram, 1273
 delayed ACK, 1191
 diagram, 1181
 endpoint, 1190
 flow control, 1192
 initial sequence number, 1192
 options, 1268
 receiving, 1191
 retransmission, 1191, 1194
 segment, 1191
 format, 1266–1268
 sending, 1191
 sequence number, 1191, 1266, 1268
 state machine, 1269
 state transition diagram, 1271
 three-way handshake, 1270
 diagram, 1272
 timeouts, 1191
 vs. UDP, 1282–1283
 urgent pointer, 1268, 1283
 window size, 1192, 1267
 TCP_CORK constant, 1262
 TCP_NOPUSH constant, 1263
 tcpd daemon, 1250
 tcpdump command, 1276–1278

TCP/IP, 1179–1195, 1438, 1440, 1441, 1443, 1444, 1445

TCSADRAIN constant, 1293

TCSAFLUSH constant, 1293

example of use, 1301, 1311, 1313, 1314, 1315

TCSANOW constant, 1293

example of use, 1306, 1387

tcsendbreak(), 426, 718, 727, 1293, 1316–1318

prototype, 1318

tcsattr(), 426, 718, 727, 1291–1293

example of use, 1301, 1306, 1311, 1313, 1314, 1315, 1387, 1392

prototype, 1291

tcsgrp(), 426, 708–709, 718, 727

prototype, 708

tee command, 87, 908

tee(), 1262

tell(), 82

telldir(), 355

TEMP_FAILURE_RETRY macro, 443

tempnam(), 109

temporary file, 108–109

termcap database, 1289, 1444

terminal, 392, 1289–1323

- background process group. *See* background process group
- canonical mode, terminal I/O, 1290, 1305, 1307
- disabling echoing of input, 1306
- disconnect, 709
- flags, 1301–1306
- flow control, 1299
- foreground process group. *See* foreground process group
- generating BREAK condition, 1318
- identification, 1321
- input queue, 1291
 - flushing, 1318
- line control, 1317–1319
- line speed, 1316–1317
- noncanonical mode, terminal I/O, 1290, 1307–1309
- obtaining device name associated with file descriptor, 1321
- output queue, 1291
 - flushing, 1318
- poll()* on, 1342
- resuming output, 1296, 1319
- retrieving and modifying attributes, 1291–1293
- select()* on, 1342
- special character, 64, 1296–1301
 - stopping output, 1296, 1319
 - window size, 395, 1319–1321

termination signal, 599, 605

termination status, process, 32, 513, 531, 545

terminfo database, 1289, 1444

termios structure, 1291, 1292, 1296, 1301–1306, 1316

- definition*, 1292
- example of use*, 1293, 1301, 1306, 1310–1311, 1313

test_become_daemon.c, 771

test_tty_functions.c, 1313

text segment, 115, 118, 612, 1019, 1024

- sharing between processes, 116, 521

TFD_CLOEXEC constant, 508

TFD_NONBLOCK constant, 508

TFD_TIMER_ABSTIME constant, 508

TGID (thread group ID), 604

tghkill(), 441, 684

Thomas, M., 1194

Thompson, K.L., 2, 4, 1443

Thomson, J., 1194

thread, 38, 225, 617–697

- attributes, 623, 628
- canceling. *See* thread cancellation
- compared to process, 629
- creating, 609, 622–623, 626–627
- dealing with asynchronous signals, 685
- detached, 627, 628
- exec()* and, 673, 686
- exit()* and, 687
- fork()* and, 673, 686
- ID. *See* thread ID
- implementation models, 687–689
- interactions with signals, 682–683
- joinable, 627
- joining, 625–627
- Linux implementation, 689–699
- maximum number of, 682, 763
- memory layout, *diagram*, 618
- one-time initialization, 658–659
- return value, 623, 625
- sending a signal to, 684
- signal mask, 683, 684
- stack, 681–682
- termination, 623–624

thread cancellation, 671–680

- asynchronous cancelability, 680
- cancelability state, 672
- cancelability type, 672
- cancellation point, 673–674
- cleanup handler, 676–679
- sending cancellation request, 671
- testing for, 675

- thread group, 225, 604, 610
 - diagram*, 605
- thread group ID, 604
- thread group leader, 605
 - diagram*, 605
- thread ID (kernel), 605
- thread ID (Pthreads), 623, 624
 - comparing IDs, 624
- thread of execution, 422
- thread_cancel.c, 674
- thread_cleanup.c, 678
- thread_incr.c, 632
- thread_incr_mutex.c, 636
- thread_incr_psem.c, 1101
- thread_multijoin.c, 649
- thread-local storage, 668–669
- thread-safe function, 655
- thread-specific data, 659–668
 - implementation, 662–663
- three-way handshake, TCP, 1270
 - diagram*, 1272
- TID (thread ID, kernel), 605
- Tilk, K., xl
- time* command, 206
- time slice, 733
- TIME terminal setting, 1307
- time()*, 187, 426
 - diagram*, 188
 - example of use*, 192
 - prototype*, 187
- time_t* data type, 65, 186, 187, 188, 189, 190, 280, 283, 287, 290, 471, 480, 488, 498, 747, 830, 948, 972, 1012, 1333
 - converting to and from broken-down time, 189–190
 - converting to printable form, 188–189
- TIME_WAIT state (TCP), 1269, 1274–1275
 - assassination, 1275
- timed_read.c, 486
- timeout on blocking system call, 486–487
- timer
 - high-resolution, 485
 - POSIX. *See* POSIX timer
 - profiling, 392, 480
 - real, 390, 480
 - virtual, 395, 480
- timer overrun, 495, 503–504, 505
- TIMER_ABSTIME constant, 494, 498
- timer_create()*, 495–497
 - example of use*, 501, 507
 - prototype*, 495
- timer_delete()*, 495, 499
 - prototype*, 499
- timer_getoverrun()*, 426
 - example of use*, 501, 506
 - prototype*, 504
- timer_gettime()*, 426, 499
 - prototype*, 499
- timer_settime()*, 426, 495, 498–499
 - example of use*, 501, 507
 - prototype*, 498
- timer_t* data type, 65, 494, 496, 498, 499, 504
- timerfd* timers, 507–511, 615
- timerfd_create()*, 508
 - example of use*, 511
 - prototype*, 508
- timerfd_gettime()*, 509
 - prototype*, 509
- timerfd_settime()*, 508–509
 - example of use*, 511
 - prototype*, 508
- times()*, 206–207, 210, 426, 560, 619, 691, 694, 755
 - example of use*, 209
 - prototype*, 206
- timespec* structure, 289, 290, 471, 488, 491, 492, 493, 498, 645, 747, 980, 1077, 1096, 1369
 - definition*, 290, 471, 488, 498, 747
 - example of use*, 290, 490
- timeval* structure, 186, 188, 204, 205, 288, 289, 480, 754, 819, 1331, 1333
 - definition*, 186, 480, 1333
- timezone, 197–200
 - specifying to a program, 198–200
- timezone* structure, 186, 187, 204
- timezone* variable, 198
- TIOCCONS constant, 801
- TIOCGPRP constant, 709
- TIOCGSID constant, 706
- TIOCGWINSZ constant, 1319, 1392, 1395
 - example of use*, 1320
- TIOCNOTTY constant, 692, 707
- TIOCPKT constant, 1389
- TIOCSCTTY constant, 707, 1385
 - example of use*, 1387
- TIOCSGRP constant, 709
- TIOCSWINSZ constant, 1320, 1395
 - example of use*, 1387
- tkill()*, 441
- TLI (Transport Layer Interface), 16
- tlpi_hdr.h, 51
- tm* structure, 188, 189, 190, 191, 193, 195, 196
 - definition*, 189
 - example of use*, 192

tmpfile(), 109, 346
 prototype, 109
tmpfs file system, 274–275, 1009,
 1090, 1108
tmpnam(), 109, 656
tms structure, 206–207
 definition, 206
 Todino-Gonguet, G., 1442
 top-level domain, 1212
 Törring, J.T., xxxix
 Torvalds (2001), 20, 1444
 Torvalds, L.B., 2, 6, 18, 20, 1444
 TOSTOP constant, 394, 716, 718, 1303, 1379
 translation look-aside buffer, 527,
 999, 1027
 Transmission Control Protocol. *See* TCP
 transport layer, 1188–1193
 diagram, 1181
 Transport Layer Interface (TLI), 16
 TRAP_BRANCH constant, 442
 TRAP_BRKPT constant, 442
 TRAP_HWBKPT constant, 442
 TRAP_TRACE constant, 442
 Troan, E.W., 1440
 Tromey, T., 1444
 Tru64 UNIX, 5
 TRUE constant, 51
truncate(), 103, 286, 345, 395, 1139, 1142
 prototype, 103
 RLIMIT_FSIZE resource limit and, 761
truncate64(), 105
 Tsafir (2008), 786, 787, 795, 1444
 Tsafir, D., 1444
 tty, 1289
tty command, 1321
tty group, 169
tty_functions.c, 1310
ttyname(), 657, 1321
 example of use, 829
 prototype, 1321
ttyname.c, 1436
ttyname_r(), 658, 1321
ttySetCbreak(), 1310
 code of implementation, 1310–1311
 example of use, 1314, 1349
ttySetRaw(), 1310
 code of implementation, 1311
 example of use, 1315, 1393
 tuple (identifying a socket), 1280
 Tweedie, S., xxxix
 TZ environment variable, 198
 TZDIR environment variable, 198
tzfile file format, 198
tzname variable, 198
tzset(), 198

U

u_int16_t data type, 593, 598
u_int32_t data type, 593, 598
uClibc, 47
ucontext_t data type, 442
ud_ucase.h, 1171
ud_ucase_cl.c, 1173
ud_ucase_sv.c, 1172
udev (user-space device file system
 daemon), 252, 1441
 UDP (User Datagram Protocol), 1152,
 1189–1190, 1194
 checksum, 1189
 datagram size, 1190
 diagram, 1181
 vs. TCP, 1282–1283
 UDP_CORK constant, 1260
ugid_functions.c, 159
 UID (user ID), 26, 153
uid_t data type, 65, 157, 173, 174, 175,
 177, 178, 280, 292, 330, 438, 927
uint8_t data type, 1202, 1203
uint16_t data type, 1199
uint32_t data type, 377, 378, 379, 472,
 1199, 1203, 1204, 1357
uintmax_t data type, 66
ulimit command, 448, 755
 Ultrix, 4
umask(), 301, 309, 426, 604. *See also*
 process, *umask*
 example of use, 302
 prototype, 301
 UML (User-Mode Linux), 789
umount command, 169, 263
umount(), 269–270, 607, 801
 prototype, 269
 UMOUNT_NOFOLLOW constant, 270
umount2(), 270
 prototype, 270
uname(), 229, 426
 example of use, 230
 prototype, 229
unbuffer.c, 1436
 undo value, System V semaphore (*semadj*),
 533, 607, 614, 619, 691, 693,
 986–988, 991
 uninitialized data segment, 116, 117, 118
 uninterruptible sleep state, 451
 universality of I/O, 29, 72
 UNIX, 1, 1437, 1441, 1444
 editions, 3
 history, 2–5, 1442, 1443
 standards, 10–19
 UNIX 03, 14, 17

- UNIX 95, 13, 17
- UNIX 98, 13, 17
- UNIX International, 13
- UNIX System Laboratories, 8
- unix_sockets.c, 1435
- unix_sockets.h, 1435
- unlink(), 109, 286, 300, 345, 346, 426, 800, 1145, 1146
 - example of use*, 347
 - prototype*, 346
- unlinkat(), 365, 426
- unlockpt(), 1380, 1382
 - example of use*, 1384
 - prototype*, 1382
- unnamed semaphore. *See* POSIX semaphore, unnamed
- unprivileged process, 33
- UNSAFE comment inside signal handler, 428
- unset shell command, 125
- unsetenv C shell command, 125
- unsetenv(), 129, 657, 1426
 - example of use*, 131
 - prototype*, 129
- unshare(), 603, 801
- unspecified (in standard description), 15
- updwtmpx(), 827
 - example of use*, 829
 - prototype*, 827
- URG control bit, TCP, 1267, 1283
- urgent data (socket), 394, 396, 1267, 1268, 1283, 1439
- urgent mode (TCP), 1283
- us_abstract_bind.c, 1176
- us_xfr.h, 1167
- us_xfr_cl.c, 1169
- us_xfr_sv.c, 1168
- us_xfr_v2_cl.c, 1435
- us_xfr_v2_sv.c, 1435
- usageErr(), 53–54
 - code of implementation*, 56
 - prototype*, 54
- usageError(), 54
- uselib(), 345
- user authentication, 162–166
- user CPU time, 40, 206
- User Datagram Protocol. *See* UDP
- user ID, 26, 153
- user mode, 23, 44
- user space, 23
- user stack, 122
- USER_HZ constant, 207
- USER_PROCESS constant, 820, 821, 822, 825
- userIdFromName(), 159
 - code of implementation*, 159–160
- User-Mode Linux (UML), 789
- username, 154
- userNameFromId(), 159
 - code of implementation*, 159
- user-uninitialized data segment, 116
- USL (UNIX System Laboratories), 8
- usleep(), 673, 674
- UT_HOSTSIZE constant, 830
- UT_NAMESIZE constant, 830
- utimbuf structure, 287
 - definition*, 287
 - example of use*, 288
- utime(), 285, 286, 287–288, 345, 426, 800
 - prototype*, 287
- UTIME_NOW constant, 290
- UTIME_OMIT constant, 290
- utimensat(), 15, 286, 289–290, 365, 426
 - prototype*, 289
- utimes(), 286, 345, 288, 426
 - prototype*, 288
- utmp file, 817
 - example of use*, 828
 - retrieving information from, 821
 - updating, 825
- UTMP_FILE constant, 818
- utmpx structure, 819–820, 822, 825, 826, 827
 - definition*, 819
 - example of use*, 824, 829
- utmpx_login.c, 828
- utmpxname(), 823
 - example of use*, 824
 - prototype*, 823
- utsname structure, 229
 - definition*, 229
 - example of use*, 230

V

- Vahalia (1996), 24, 138, 250, 278, 342, 630, 919, 1044, 1422, 1444
- Vahalia, U., 1444
- van der Linden (1994), xxxii, 1444
- van der Linden, P., 1444
- vanilla kernel, 234
- Vargas, B.L., xxxix, xli
- Vargas, C.E.K., xli
- variadic function, 1413
- Vaughan (2000), 857, 1444
- Vaughan, G.V., 1444
- VDISCARD constant, 1296
- VEOF constant, 1296, 1309

- VEOL constant, 1296, 1309
- VEOL2 constant, 1296
- VERASE constant, 1296
- version script (*ld*), 868–872
- vfork()*, 16, 523–525, 530, 609
 - example of use*, 524
 - prototype*, 523
 - RLIMIT_NPROC resource limit and, 763
 - scheduling of parent and child
 - after, 523
 - speed, 610
- vfork_fd_test.c*, 1430
- VFS (virtual file system), 259
 - diagram*, 259
- vhangup()*, 801
- Viega (2002), 795, 1445
- Viega, J., 1445
- view_lastlog.c*, 831
- view_symlink.c*, 369
- VINTR constant, 1296
- Viro (2006), 267, 1445
- Viro, A., 1445
- virtual address space, 120
 - diagram*, 120
- virtual device, 252
- virtual file switch, 259
- virtual file system (VFS), 259
 - diagram*, 259
- virtual memory
 - resource limit on, 760
 - unified, 1032
- virtual memory management, 22, 118–121, 1440
- virtual server, 789
- virtual time, 206
- virtualization, 608, 789
- VKILL constant, 1296
- VLNEXT constant, 1296
- WMIN constant, 1307, 1309
 - example of use*, 1311
- vmsplice()*, 1262
- volatile variables, 137
- VQUIT constant, 1296
- VREPRINT constant, 1296
- VSTART constant, 1296
- VSTOP constant, 1296
- VSUSP constant, 1296
- vsyslog()*, 777
- VT0 constant, 1302
- VT1 constant, 1302
- VTDLY constant, 1302
- VTIME constant, 1307, 1309
 - example of use*, 1311
- WVERASE constant, 1296

W

- W_OK constant, 299
- Wagner, D., 1438, 1444
- wait morphing, 647
- wait status, 545–547, 580
- wait()*, 32, 426, 514, 541–542, 673, 690
 - diagram*, 515
 - example of use*, 543, 901
 - interrupted by signal handler, 443
 - prototype*, 542
- wait3()*, 552–553, 609, 754
 - interrupted by signal handler, 443
 - prototype*, 552
- wait4()*, 552–553, 609, 754
 - interrupted by signal handler, 443
 - prototype*, 552
- waitid()*, 550–552, 610, 673
 - interrupted by signal handler, 443
 - prototype*, 550
- waitpid()*, 426, 544–545, 609, 673
 - example of use*, 549, 583, 587, 602
 - interrupted by signal handler, 443
 - prototype*, 544
- wall clock time, 185
- wall* command, 169
- Wallach, D.S., 1438
- watch descriptor (*inotify*), 376, 377
- Watson (2000), 798, 1445
- Watson, R.N.M., 1445
- WCONTINUED constant, 544, 545, 550
- WCOREDUMP(), 546
 - example of use*, 547
- wcrtomb()*, 656
- wcsrtombs()*, 656
- wcstombs()*, 657
- wctomb()*, 657
- weakly specified (in standard description), 15
- Weinberger, P.J., 1437
- well-known address, 909
- WERASE terminal special character, 1296, 1299, 1305, 1307
- WEXITED constant, 550
- WEXITSTATUS(), 546
 - example of use*, 547
- Wheeler, D., 795, 857
- who* command, 817
- WIFCONTINUED(), 546
 - example of use*, 547
- WIFEXITED(), 546
 - example of use*, 547
- WIFSIGNALED(), 546
 - example of use*, 547

WIFSTOPPED(), 546
 example of use, 547
 wildcard address (IP), 1187
 Wilhelm, S., 1442
 Williams (2002), 20, 1445
 Williams, S., 1445
 winsize structure, 1319, 1385, 1394–1395
 definition, 1319
 example of use, 1320, 1386, 1392
 wireshark command, 1277
 WNOHANG constant, 544, 551
 example of use, 557
 WNOWAIT constant, 551
 Woodhull, A.S., 1444
 working directory, current, 29, 225,
 363–365, 604, 613
 Wright (1995), 1235, 1272, 1445
 Wright, C., xxxix
 Wright, E.A., xl
 Wright, G.R., 1445
 write permission, 29, 282, 294, 297
 write(), 70, 80, 286, 395, 426, 673,
 800, 1138
 example of use, 71, 85
 FIFO, 918
 interrupted by signal handler, 443
 pipe, 918
 prototype, 80
 RLIMIT_FSIZE resource limit and, 761
 terminal output
 by background job, 394
 by orphaned process group, 730
 write_bytes.c, 236, 242, 250
 writen(), 1254
 code of implementation, 1255
 prototype, 1254
 writev(), 99–100, 102, 286, 673
 interrupted by signal handler, 443
 prototype, 99
 Wronski, M., xxxix
 WSTOPPED constant, 550
 WSTOPSIG(), 546
 example of use, 547
 WTERMSIG(), 546
 example of use, 547
 wtmp file, 817
 example of use, 828
 updating, 825
 WTMP_FILE constant, 818
 WUNTRACED constant, 544, 545, 552
 example of use, 549

X

X_OK constant, 299
 XATTR_CREATE constant, 315
 XATTR_REPLACE constant, 315
 xattr_view.c, 317
 XBD, 14
 XCASE constant, 1303
 XCU, 14
 XCURSES, 14
 XDR (External Data Representation), 1200
 Xen, 789
 XENIX, 5
 XFS file system, 261
 i-node flag, 304–308
 Xie, Q., 1444
 xinetd daemon, 1248
 XNS (X/Open Networking Specification),
 13, 16, 17
 X/Open, 13
 X/Open Networking Specification (XNS),
 13, 16, 17
 X/Open Portability Guide (XPG), 13,
 16, 17
 Issue 3 (XPG3), 13, 17
 Issue 4 (XPG4), 13, 17
 Issue 4, version 2 (XPG4v2), 13, 17
 Issue 5 (XPG5), 13, 17
 X/Open Transport Interface (XTI), 16
 XPG. *See* X/Open Portability Guide
 XRAT, 14
 XSH, 14
 XSI conformance, 14
 XSI extension, 15, 62, 63, 221
 XSI IPC, 922
 XTI (X/Open Transport Interface), 16

Y

Yourtchenko, A., 1439

Z

Zamuner, U., xxxix
 Zemlin, J., xl
 zero-copy transfer, 1261
 zero-uninitialized data segment, 116
 zic command, 198
 zombie process, 553, 554, 555, 556, 559,
 1431