

Index

COPYRIGHTED MATERIAL

Index

SYMBOLS AND NUMERICS

`#define` statements in `config.h` file, 74–76, 79
 % (percent sign) for PHP code blocks, 412
 ? (question mark) for PHP code blocks, 412
 ; (semicolon) terminating SQL statements, 406
 64-bit cleanliness, 89

A

Abatron BD12000 CPU-level debugger, 289
ABIs (Application Binary Interfaces), 33
abstraction. See **layers of abstraction**
`accept()` function, 124, 127
`aclocal` utility, 71
`aclocal.m4` macro file, 71
`add` command (Bazaar), 110
`add_three_inits()` function call, 349
`alien` command (Debian), 64
`alignment` attribute (GCC), kernel use of, 45
Apache
 basic authentication, 402
 installing, 399–400
 overview, 396
 SSL encryption, 402–403
 SSL integration with HTTP, 403
 versions available, 396
 virtual hosting, 400
API documentation, 251–252
APIs. See **kernel interfaces; Linux kernel modules (LKMs)**

Application Binary Interfaces (ABIs), 33
arrays (PHP)
 optimizing, 423–424
 overview, 415–416
 pass-by-value vs. pass-by-reference approach, 418–420
ASCII character encoding, 83, 86, 140
assembly language, 35–36, 44
attributes in MySQL, 409, 411
audio devices, 225
audio support
 CD-ROM access using SDL, 391–394
 need for, 361
 playing an audio file using SDL, 388–391
`audiotest.c` program, 390–391
authentication (Apache), 402–403
automated code analysis tools, 299
Autotools online book (GNU), 70

B

`b` or `break` command (GDB), 41, 291–292
`backtrace` or `bt` command (GDB), 293–295
`badclient.c` program, 135–137
`badserver.c` program, 133–135, 136–137
Bazaar-NG (bzz)
 adding files to track, 110
 checking in modifications, 110–111
 `commit` command, 110
 configuring, 109
 creating a branch, 109–110
 GNU Arch vs., 109
 merging branches, 111–112

Berkeley DB package

- adding data, 157–159
- building programs, 154
- `c_close()` function, 165
- `c_del()` function, 163
- `c_get()` function, 163, 165
- compiling applications, 154
- `c_put()` function, 163
- creating a database, 155
- creating a DB handle, 155
- `cursor()` function, 163
- cursors, 163–165
- database types, 155
- `db_create()` function, 155, 157
- `del()` function, 161, 163
- deleting data, 161–163
- described, 152
- directories, 154
- downloading and installing, 153–154
- `employees.db` database, 161–163
- `get()` function, 159, 161, 163–165
- `getemployee.c` program, 160–161
- history of, 152–153
- `newemployee.c` program, 157–159
- `open()` function, 155–156
- opening a database, 155–157
- Oracle acquisition of, 153
- `put()` function, 157
- retrieving data, 159–161
- retrieving multiple records, 163–165
- `betterserver.c` program, 138–139
- big endian mode, 33, 89–90. See also endianness**
- `bind()` function for sockets, 124, 130
- `bisect` command (git), 115
- BitKeeper SCM tool (deprecated), 112**
- boot kernel, overriding default, 197**
- branching in CVS, 94, 102**
- breakpoints, setting**
 - with DDD, 300–301
 - with GDB, 41, 291–292
- `bt` or `backtrace` command (GDB), 293–295
- `build_gstreamer_pipeline()` function, 331
- `build_gui()` function
 - connecting callbacks to buttons, 329
 - constructing the music player GUI, 322
 - variables, 327–328
- `bzr`. See **Bazaar-NG**

C

C language, GCC options for, 31

`c` or `continue` command (GDB), 42

Cairo graphics library (GNOME), 316

callbacks

- GLUT functions, 370, 377–378, 379–380
- HAL example, 352–353
- music player project, 328–329
- NetworkManager functions, 359

case in SQL, 406

`c_close()` function (Berkeley DB), 165

`c_del()` function (Berkeley DB), 163

CD-ROM access using SDL, 391–394

`cdtest.c` program, 392–394

centralized SCM. See also SCM (software configuration management)

- choosing tools, 95
- CVS tools with, 95, 96–104
- decentralized vs., 93, 95
- Subversion with, 104–108

certification (LAMP), 398

`c_get()` function (Berkeley DB), 163, 165

changesets in SCM

- with CVS, 104
- defined, 94
- with Subversion, 107

character devices

- audio devices, 225
- building drivers, 229–231
- `char.c` example, 226–229
- confirming correct loading of driver, 230
- creating, 225–229
- defined, 225
- driver initialization, 231–232
- driver payload, 232–233
- driver uninitialization, 232
- dynamic, 231
- installing drivers, 230
- loading drivers, 230
- in `ls` listing, 225
- unloading drivers, 231

character sets

- native language output, 85–88
- UTF-8 encoding, 83
- wide characters, 84–85

`char.c` character device, 226–229

- `char_exit()` **function**, 228, 232
- `char_init()` **function**, 227–228, 231–232
- `char.ko` **driver**, 229–231
- `char_open()` **function**, 226–227, 233
- `char_read()` **function**, 227, 232
- checking in modifications**
 - to Bazaar-NG, 110–111
 - to CVS, 99
 - to Subversion, 106–107
- checking out sources**
 - from CVS, 98–99
 - from Subversion, 106
- `class_create()` **function**, 232
- client interface (MySQL)**, 405
- clients**
 - connectionless sockets, 130, 132–133
 - connection-oriented sockets, 123, 127–129
- `clone` **command (git)**, 113–114
- `close()` **function for sockets**, 129
- Cogito utility**, 113
- command line**, 11
- `commit` **command**
 - Bazaar-NG, 110
 - CVS, 100, 104
 - git, 114
 - Subversion, 107
- community of Linux users**, 15–16
- compatibility**
 - hardware, 50
 - of Linux kernel modules, 251
 - LSB, 4, 54
- compiling. See also GCC (GNU Compiler Collection); Makefiles**
 - cross-compilation, 46–47, 193
 - multiple source files, 26–28
 - single source file, 24–25
- Concurrent Version System. See CVS**
- `config.h` **header file**, 74–76, 79
- `configure` **scripts**
 - for cross-compilation, 47
 - generation from `configure.in` file, 71, 79, 80
 - GNU Autoconf tool for, 21–22, 71, 78–80
 - GNU Libtool reference in, 81
 - for source-level portability, 72–74
- `configure.in` **file**
 - AC_PREREQ line in, 79
 - AM_PROG_LIBTOOL line in, 81
 - binary location in, 77
 - configure script generation from, 71, 79, 80
 - example, 79
- conflict resolution with CVS**, 96, 100–101
- `connect()` **function**, 128, 130
- connectionless sockets**
 - `bind()` function, 130
 - `connect()` function, 130
 - functions, 130–131, 132, 133
 - `recvfrom()` function, 130–131, 132
 - `sendto()` function, 130, 133
 - simple UDP client, 132–133
 - simple UDP server, 131–132
- connection-oriented sockets**
 - client functions, 127–129
 - closing the connection, 129–130
 - defined, 123
 - sequence of functions for creating, 123
 - server functions, 124–127
 - simple TCP client, 128–129
 - simple TCP server, 125–127
 - terminating sessions, 130
- `continue` **or c command (GDB)**, 42
- control buttons (music player)**, 325–326
- converting**
 - IP address format, 127–128
 - RPMs to Debian packages, 64
- copying**
 - kernel to test system, 197–198
 - object code using `objcopy`, 37
- core dumps**
 - debugging using GDB, 42, 296–297
 - kernel crash dumps, 306
 - `proc/kcore` file (fake core dump), 43
 - user limits preventing, 296
- core file debugging**, 296–297
- Covert code analysis tool**, 299
- CPU**
 - Abatron BD12000 CPU-level debugger, 289
 - endianness of, 92
 - kernel development requirements, 186
 - targeting when compiling with GCC, 33
- `c_put()` **function (Berkeley DB)**, 163
- `CREATE TABLE` **statement (SQL)**, 406–407
- cross-compilation**, 46–47, 193
- Cross-Site-Scripting (XSS) attacks**, 429
- crosstool script**, 48

Curl package, 140. See also libCurl library

`curl_easy_cleanup()` **function, 143**

`curl_easy_setopt()` **function, 142–143**

`cursor()` **function (Berkeley DB), 163**

cursors with Berkeley DB databases, 163–165

CVS (Concurrent Version System)

- adding files and directories, 99–100
- branching, 94, 102
- browsing repository history, 100
- bundled with distributions, 96
- changesets, 104
- checking in modifications, 99
- checking out sources, 98–99
- commit activity, 100, 104
- concurrent defined, 96
- conflict resolution, 96, 100–101
- creating a project, 97–98
- creating a repository, 96–97
- distributed development with, 103–104
- Eclipse functionality, 115–117
- editor for, 98
- file rename issues, 104
- importing a project, 97–98
- limitations, 104
- over SSH connection, 103
- overview, 96
- project website, 96
- with remote pserver, 103–104
- removing files and directories, 99–100
- tagging, 94, 102
- tools with SCM, 95
- update process, 101

D

Data Display Debugger (DDD)

- data presentation capabilities, 301–302
- described, 299
- inserting breakpoints, 300–301
- starting, 299

data integrity in MySQL, 410

data modeling in MySQL, 410–411

databases. See also specific database software

- Berkeley DB package, 152–165
- implementing via built-in engine, 151
- implementing via servers, 151–152
- MySQL, 396, 404–411
- overview, 150–152

for persistent data storage, 149

PHP support for, 432

PostgreSQL server, 165–184

datagrams vs. streams, 133–137

`db_create()` **function (Berkeley DB), 155, 157**

D-Bus (Desktop Bus)

- bindings available, 342
- communication with objects, 343
- described, 341–342
- filter function, 344, 345–346, 348–349
- Glib binding, 342–343
- HAL API in, 351
- headers, 342
- methods, 346–350
- NetworkManager use of, 358
- owning a namespace, 344
- polling for signals, 345
- preventing “spam,” 345
- sending and receiving signals, 343–345
- types of messages, 343

`dbus_message_append_args()` **method, 347**

`dbus_message_new_method_call()` **method, 346**

`dbus_message_new_signal()` **method, 346**

DDD (Data Display Debugger)

- data presentation capabilities, 301–302
- described, 299
- inserting breakpoints, 300–301
- starting, 299

Debian Developers (DDs), 70

Debian GNU/Linux, 7, 14

Debian-derived distributions

- building packages, 65–70
- converting RPMs to Debian packages, 64
- Debian Developers (DDs) for, 70
- Debian website, 70
- default `udev` permissions, 266–267
- dependencies, 67
- further information, 70
- overview, 8
- portability issues, 56

debugging. See also testing; specific tools

- automated code analysis tools for, 299
- core dumps using GDB, 42
- core files, 296–297
- DDD for, 299–302
- Eclipse functionality for, 302–305
- finding GPL violations in source code, 299

- GCC options for, 32
- GDB for, 40–43, 289–297
- git tool as aid for, 114–115
- hardware, 289
- inserting breakpoints, 41
- kernel, 305–313
- memory leaks, 288
- memory-handling problems, 288
- overview, 287–288
- PHP, 424–426
- segmentation faults, 288
- Valgrind for, 298
- decentralized SCM. See also SCM (software configuration management)**
 - Bazaar-NG for, 109–112
 - centralized project management with, 108
 - centralized vs., 93, 95
 - git tool for, 95, 108, 112–115
 - overview, 108–109
- DECLARE_WORK **macro**, 275–276
- #define **statements in config.h file**, 74–76, 79
- del() **function (Berkeley DB)**, 161, 163
- DELETE **statement (SQL)**, 408–409
- deleting or removing**
 - CVS files and directories, 99–100
 - data from Berkeley DB database, 161–163
 - records in MySQL, 408–409
- dependency resolution**
 - for Debian dependencies, 67
 - for packages, 53–54
- depmod **command**, 198
- Desktop Bus. See D-Bus**
- development environment setup**
 - graphical desktop, 10
 - sudo utility, 13
 - system terminal, 11
 - text editors, 11–12
- development process**
 - building sources, 22–23
 - configuring the local environment, 21–22
 - need for understanding, 19
 - working with source code, 20
- development releases of distributions, 13–14**
- device files**
 - for block devices, 233
 - for character devices, 225–233
 - directory for, 224
 - mechanism vs. policy, 233–234
 - network devices not files, 233
 - overview, 224–225
 - procfs, 234–238
 - sysfs class for, 231, 238
 - viewing with ls utility, 224–225
- device objects (HAL)**
 - defined, 353
 - info.capabilities property, 354
 - key/value properties, 353
 - mapping to physical devices, 354
 - namespaces for property keys, 353
 - optical drives example, 354–357
 - property key categories, 354
- dh_make **script (Debian)**, 65–67
- distributed development**
 - with CVS, 103–104
 - with Subversion, 107–108
- distributing Linux kernel modules, 284–285**
- distribution portability**
 - building your own distribution, 57
 - Debian distributions, 56
 - Linux Standard Base, 54–55
 - Linux vendor role in, 55–56
 - need for, 52
 - overview, 52
 - packages, 53–54
 - RPM-based distributions, 56
 - source-based distributions, 57
- distributions**
 - building your own, 8, 57
 - choosing, 6–8
 - classes of, 7–8
 - currently popular, 6
 - Debian GNU/Linux, 7
 - Debian-derived, 8
 - development releases, 13–14
 - Distrowatch website information for, 5
 - history of, 4
 - installing, 8–15
 - locales and popularity of, 81
 - Novell, 6–7
 - obtaining, 9
 - portability of, 52–57
 - Red Hat, 6
 - RPM-based, 7
 - source-based, 8
 - Ubuntu GNU/Linux, 7
- Distrowatch website, 5**

DKMS project, 284

`down()` function, 267

`down_interruptible()` function, 268

`dpkg` utility (Debian), 65, 68–69

drivers. See character devices; device files; Linux kernel modules (LKMs)

DVD library LAMP example

adding logging and exceptions, 437–441

applying templating framework, 441–442

with DB-specific data layer, 434–437

developer's nightmare, 433–434

dynamic devices, 231

E

Eclipse IDE environment

bundled with distributions, 10

CVS functionality, 115–117

debugging functionality, 302–305

ELF (Executable and Linking Format) files

as de facto binary format, 248

further documentation, 40

in GNU `objdump` tool output, 38–39

overview, 36

emacs text editor, 11–12

embedded devices, 52

`empinfo.c` program (PostgreSQL), 182–184

`employees.db` database (Berkeley DB), 161–163

endianness

endian neutrality, 89–92

GCC options for, 33

“holy wars” concerning, 92

importance of, 90–91

origins of term, 92

error handling in PHP, 420–421

exceptions in PHP, 421–422

execution contexts, 203–204

`EXPORT_SYMBOL` macro, 258

`EXPORT_SYMBOL_GPL` macro, 258

external kernel interfaces

defined, 218, 219

device files, 224–238

ignoring kernel protections, 239–243

kernel events, 238–239

overview, 219

system calls, 219–224

F

Factory development release (SuSE), 14

Fedora Linux

default `udev` permissions, 267

Fedora project website, 6

GNOME as default graphical desktop for, 10

rawhide development release, 14

file descriptors, 120, 121–122

files

everything as, 224, 233

`.ko` extension for LDM object file, 247

for persistent data storage, 150

`.php` suffix for PHP files, 412

steps required for interaction with, 150

`filesrc` element (GStreamer), 330

floating point instructions, GCC options for, 33

foreign key, 409

foreign languages. See internationalization

Free Software Foundation, 4–5

FreeDesktop project. See also specific projects

Desktop Bus, 341–350

Hardware Abstraction Layer, 350–358

NetworkManager, 358–359

other projects, 360

websites, 360

functions in PHP, 416

G

GCC (GNU Compiler Collection)

attributes, 45

command line options, 30–34

compiling a single source file, 24–25

compiling multiple source files, 25–27

compiling supporting library code, 26

compiling wrapper program, 26

creating executable from object code, 26–27

debugging options, 32

described, 24

`-fPIC` option, 29

further documentation, 34

`-g` flag for debugging information, 290

`gcc` driver as front end for, 24, 25

general options, 31

hardware options, 33–34

history of, 23–24

inline assembly support by, 44

kernel reliance on, 44–46

-L flag, 29

language options, 31

-lm option, 28

optimization options, 32–33

-shared flag, 29

using external libraries with, 27–30

warning level options, 32

GConf library (GNOME), 318

GDB (GNU debugger)

backtrace or bt command, 293–295

break or b command, 41, 291–292

buggy program example, 294–296

compiling code for use with, 289–290

continue or c command, 42

core file debugging, 42, 296–297

described, 40, 289

hardware debugging support, 289

help command, 42, 291

inserting breakpoints before using, 41

list command, 40–41

print command, 41–42, 292–293

redirecting IO using tty command, 42

run command, 292–293

running an application under, 40

setting breakpoints, 41, 291–292

starting, 290–291

viewing source code, 40–41

viewing variable values, 41–42

visualizing data, 292–293

GDK (GIMP Drawing Kit) graphics library (GNOME), 317

General Public License. See GPL

generic nature of Linux, 17–18

Gentoo Linux, 8

get() **function (Berkeley DB), 159, 161, 163–165**

getemployee.c **program (Berkeley DB), 160–161**

getemployees.c **program (PostgreSQL), 179–181**

getheaders.c **program, 146–147**

gettext **library, translation using, 85–88**

getvals.c **program (PostgreSQL), 177–179**

getweb.c **program, 143–145**

GFP_KERNEL **flag for kmalloc function, 260**

GFP_USER **flag for kmalloc function, 260**

git SCM tool

bisect command, 115

cloning an existing repository, 113–114

Cogito utility, 113

commit command, 114

creating a repository, 113

as debugging aid, 114–115

as decentralized SCM tool, 95, 108

managing sources, 114

as object-based, 112–113

obtaining, 112

overview, 112

project website, 112

update-index command, 114

gtk graphical tree browser, 213

glBegin() **function, 373–374**

glClear() **function, 371**

glClearColor() **function, 371–372**

glColor3f() **function, 372**

glEnd() **function, 373–374**

glFlush() **function, 372, 382**

Glib library (GNOME)

D-Bus bindings, 342

GMainLoop, 351

g_signal_connect() **function, 321**

g_snprintf() **function, 334**

g_timeout_add() **function, 333**

overview, 316

GLIBC (GNU C library), 27–28, 202

glLoadIdentity() **function, 375**

glMatrixMode() **function, 375**

glRotate() **function, 372**

glRotatef() **function, 372, 382**

gluOrtho2D() **function, 375**

GLUT (OpenGL Utility Toolkit). See also OpenGL (Open Graphics Library)

adjusting coordinate system to window, 375

animation, 380–382

basic principles for using, 368

built-in shapes, 372

callback functions, 370, 377–378, 379–380

clearing windows, 371

creating a window, 368–373

creating points, 373

defining set of points, 373–374

described, 365

displaying 3-D objects on 2-D monitor, 375

drawing objects, 373–377

establishing the matrix mode, 375

glutMainLoop() **function, 370**

GLUT (OpenGL Utility Toolkit) (continued)

- handling keyboard events, 377–380
- handling mouse events, 378
- initialization and window establishment functions, 368–369
- processing user input, 377–380
- rotating images, 372
- sending drawing to display, 372
- setting object color, 372
- setting origin point on viewport, 375
- setting the viewport upon resize, 374–375
- setting window background color, 371–372
- `testdraw.c` program, 375–377
- `testkey.c` program, 378–380
- `testtimer.c` program, 380–382
- `testwin.c` program, 370–373
- workspace for, 368

`glutCreateWindow()` **function, 368, 369**

`glutGetModifiers()` **function, 378**

`glutInit()` **function, 368–369**

`glutInitDisplayMode()` **function, 368, 369, 382**

`glutInitWindowPosition()` **function, 368, 369**

`glutInitWindowSize()` **function, 368, 369**

`glutKeyboardFunc()` **function, 377**

`glutMainLoop()` **function, 370**

`glutMotionFunc()` **function, 378**

`glutMouseFunc()` **function, 378**

`glutPostRedisplay()` **function, 380, 382**

`glutReshapeFunc()` **function, 374–375**

`glutSpecialFunc()` **function, 377**

`glutSwapBuffer()` **function, 382**

`glutTimerFunc()` **function, 380, 382**

`glutWireCube()` **function, 382**

`glutWireTetrahedron()` **function, 372**

`glVertex()` **function, 373, 374**

`glViewport()` **function, 375**

GNOME desktop environment, 10, 315

GNOME Developer Platform. See also music player project

- Cairo graphics library, 316
- GConf library, 318
- GDK graphics library, 317
- Glib library, 316
- GNOME desktop environment based on, 315
- GObject library, 316
- GStreamer library, 318

- GTK+ library, 317
- layered interaction of libraries, 316
- libglade library, 318
- Pango text-rendering library, 317

GNU Arch, 109

GNU assembler, 34–36

GNU Autoconf tool

- commands, 80
- for `configure` script generation from
 - `configure.in` file, 71
- `configure` scripts, 21–22, 78–80

GNU Autotools

- Autoconf, 21–22, 71, 78–80
- Autoheader, 72
- Automake, 72, 80–81
- building projects, 71–78
- example for, 71
- further information, 70
- GNU Build System formed by, 70
- Libtool, 72, 81

GNU binutils

- assembler, 34–36
- linker, 25–27, 36, 40
- `objcopy` and `objdump`, 37–39

GNU Build System, 70

GNU C library (GLIBC), 27–28, 202

GNU Compiler Collection. See GCC

GNU debugger. See GDB

GNU info system for GCC, 34

GNU Libtool, 72, 81

GNU linker

- further documentation, 40
- invoked by GCC, 25–27
- overview, 36

GNU Make tool, 39–40

GNU project, 2, 19

GNU Toolchain. See also specific components

- binutils, 34–39
- building, 47–48
- cross toolchains, 46–47
- GCC overview, 23–34
- GDB overview, 40–43
- kernel reliance on, 44–46
- Make overview, 39–40
- as most common toolchain, 19

GObject library (GNOME), 316

GPL (General Public License)

- extended to kernel modules, 258
- finding violations in source code, 299
- for MySQL, 396
- overview, 4–5

graphics

- need for support, 361
- OpenGL, 364–382
- prevalence in Linux distributions, 361
- SDL, 365, 382–388
- X Windows specification, 362–363

GRUB bootloader, 197

- `g_signal_connect()` **function, 321**
- `g_snprintf()` **function, 334**
- `gst_bus_add_watch()` **function, 331–332**
- `gst_element_factory_make()` **function, 331**
- `gst_element_query_duration()` **function, 334**
- `gst_element_query_position()` **function, 334**
- `gst_element_seek()` **function, 338–339**

GStreamer library (GNOME)

- `build_gstreamer_pipeline()` function, 331
- `filesrc` element, 330
- `gst_bus_add_watch()` function, 331–332
- `gst_element_factory_make()` function, 331
- `gst_element_query_duration()` function, 334
- `gst_element_query_position()` function, 334
- `gst_element_seek()` function, 338–339
- `gst_tag_list_free()` function, 337
- `gst_tag_list_get_string()` function, 338
- music player use of, 329–335
- overview, 318
- pads, 330
- pipelines, 329–330
- `playbin` element, 330, 331
- `play_file()` function, 333
- `stop_playback()` function, 332–334
- `gst_tag_list_free()` **function, 337**
- `gst_tag_list_get_string()` **function, 338**
- `g_timeout_add()` **function, 333**

GTK+ library (GNOME). See also specific widgets

- accelerator syntax, 327–328
- global header file, 319
- initializing, 320
- menu and toolbar construction objects, 326
- music player main window using, 319–321
- overview, 317
- window layout widgets, 321–322

- `GtkAboutDialog` **object, 329**
- `GtkAction` **objects, 326–327**
- `GtkActionGroup` **variable, 327–328**
- `GtkBin` **class, 322**
- `gtk_box_pack_end()` **function, 323**
- `gtk_box_pack_start()` **function, 323**
- `gtk_button_set_use_stock()` **function, 325**
- `GtkFileChooserDialog` **widget, 335–337**
- `GtkFixed` **widget, 321**
- `GtkHBox` **widget, 321, 323, 325**
- `GtkHScale` **widget, 324**
- `gtk_init()` **function, 320**
- `GtkLabel` **class, 323**
- `gtk_label_set_use_markup()` **function, 324**
- `gtk_main()` **function, 321**
- `GtkScale` **widget, 338**
- `gtk_scale_set_draw_value()` **function, 324**
- `GtkTable` **widget, 321**
- `gtk_toggle_button_new_from_stock()` **function, 325**
- `GtkUIManager` **variable, 327**
- `GtkVBox` **widget, 321, 322, 326**
- `GtkVScale` **widget, 324**
- `gtk_window_new()` **function, 320**
- `gtk_window_set_title()` **function, 320–321**
- GTP_ATOMIC **flag for kmalloc function, 260**
- `gui_status_update()` **function, 335**
- `gui_update_time()` **function, 334–335, 339**
- Gulliver's Travels (Swift)*, 89, 92

H**HAL (Hardware Abstraction Layer)**

- API in D-Bus, 351
- callback functions, 352–353
- development of, 350–351
- device objects, 353–358
- GMainLoop for events, 351
- libhal C helper library, 351
- LibHalContext for events, 351–352
- listing properties with `lshal`, 357–358
- NetworkManager use of, 358
- starting the main loop, 352

hardware

- compatibility, 50
- debugging, 289
- GCC options for, 33–34

hardware (continued)

- kernel development requirements, 186
- portability, 51–52, 89–92

Hardware Abstraction Layer. See HAL

hardware portability

- endian “holy wars” concerning, 92
- endian neutrality for, 89–92
- of Linux kernel, 51–52
- overview, 88
- 64-bit cleanliness for, 89

help command (GDB), 42, 291

history

- of Berkeley DB package, 152–153
- of distributions, 4
- of GCC, 23–24
- of General Public License, 4–5
- of GNU project, 2
- of Linux kernel, 3

history command (CVS), 100

HTML, mixing PHP with, 412–413

include function (PHP), 413–414

inet_addr() function, 127–128

info command (GCC), 34

init command (Bazaar), 109

initdb program (PostgreSQL), 166

init.h header, 256

initial ramdisk for booting, 198–199

initializing

- character device driver, 231–232
- GTK+ library (GNOME), 320
- libCurl library, 142
- PostgreSQL default database, 166

initrd command, 199

inline assembly, 44

INSERT statement (SQL), 407

insmod utility, 230

installing

- Apache, 399–400
- Berkeley DB package, 153–154
- binary with `make install` command, 77–78
- character device driver, 230
- kernel, 199
- libCurl library, 141
- Linux kernel module man pages, 252

- Linux kernel modules, 198

- MySQL, 404

- OpenGL, 366–367

- `pear`, 427

- PHP add-on packages, 427

- PHP modules, 401

- PHP5, 401

- PostgreSQL database server, 165–167

- SDL, 383

installing Linux distributions

- development environment setup, 10–13

- development releases, 13–14

- multiple distributions for testing, 8–9

- obtaining a distribution, 9

- package selection at install time, 9–10

- RPMS on non-RPM-based systems, 64

- scratch boxes, 14

- virtualization products, 14–15

integrity rules (MySQL), 410–411

internal kernel interfaces

- defined, 218

- further information, 243

- kernel ABI, 244

- kernel API, 243–244

internationalization

- abstraction from users, 83

- character sets, 83–88

- languages and locales, 82–83

- Linux support for, 81–82

- native language output, 85–88

- Open Internationalisation Initiative, 81

- for Perl and Python programs, 88

- translation using `gettext` library, 85–88

- wide characters, 84–85

Internet Relay Chat (IRC) networks, 16

Internet resources

- Berkeley DB package, 153

- `crossstool` script, 48

- CVS project website, 96

- Debian GNU/Linux website, 7

- Debian website, 70

- Distrowatch website, 5

- DKMS project, 284

- Fedora project, 6

- FreeDesktop project websites, 360

- Gentoo Linux website, 8

- git project website, 112

GNU Autotools information, 70
 junit information, 430
 kernel API changes, 244
 kernel sources, 187
 libCurl library, 141
 Linux From Scratch project, 8
Linux Journal, 15
 Linux kernel module man pages, 251
Linux Weekly News, 216, 283
 “Making Hardware Just Work,” 350
 Mesa project website, 366
 Netcraft survey, 395
 Novell website, 7
 Open18N website, 81
 OpenSUSE project, 7
 PEAR, 432
 PHP error-handling information, 420
 PHP unit testing information, 430
 PHP websites, 412
 PostgreSQL database server, 165
 PTXdist website, 8
 Qemu virtualization product, 15
 quilt patch-tracking utility, 215
 Red Hat website, 6
 SDL website, 382
 SimpleTest, 430
 Subversion project website, 104
 Ubuntu project, 7
 VMware virtualization product, 14
 xdebug, 424
 Xen virtualization product, 15

interrupts, protecting spinlocks against, 268–269

IP addresses

converting to client format, 127–128
 referencing, 122–123

IRC (Internet Relay Chat) networks, 16

iterators in PHP, 417

J

junit-derived test harnesses for PHP, 430–432

K

kABI (kernel ABI), 244–245

KDE graphical desktop, 10

kdump mechanism, 306

Kegel, Dan (crosstool creator), 48

kernel. See also kernel interfaces; Linux kernel modules (LKMs)

building, 193–196
 built, location for, 196–197
 challenges for working with, 185–186
 configuring, 190–193
 copying to text system, 197–198
 crash dumps, 306
 cross-compilation configuration, 193
 debugging, 305–313
 development process, 211–216
 execution contexts, 203–204
 GCC attributes used by, 45
 getting modules into, 284
 git SCM tool with, 95, 108, 112–115, 212
 gitk graphical tree browser, 213
 GLIBC not used by, 27
 hacking, 208–211
 history of, 3
 inline assembly by, 44
 installing, 199
 linker scripts used by, 45–46
Linux Weekly News, 216, 283
 LKML (Linux Kernel Mailing List), 213–215
 making initial ramdisk for booting, 198–199
 memory mapping, 207
 “-mm” kernel tree, 215
 module_init function, 45
 overview, 305–306
 packaging, 199
 portability of, 51–52
 prerequisites for working with, 186
 quilt patch-tracking utility, 215
 reliance on GNU Toolchain, 44–46
 scheduler, 201–202
 segmentation fault detection by, 288
 sources, 187–190
 “-stable” team, 215–216
 submitting patches to LKML, 214–215
 supporting files, 197
 system calls, 202–203
 task abstraction, 200–205
 testing, 197–199
 threads, 201, 204–205, 276–283
 top-level directories, 188–189
 transferring control to, 221
 User Mode Linux (UML), 309–312

kernel (*continued*)

- vendor configuration options, 193
- vendor kernels, 190
- versioning, 189–190
- virtual memory, 205–208
- warning about working with, 200

kernel ABI (kABI), 244–245

kernel debugging

- anecdotal example, 312–313
- crash dumps for, 306
- example oops, 307–308
- fatal vs. nonfatal errors, 306
- in-kernel debuggers, 313
- overview, 305–306
- reporting bugs upstream, 309
- subsequent oopses, not trusting, 308
- User Mode Linux (UML) for, 309–312

kernel interfaces

- definition for this book, 217
- device files, 224–238
- external vs. internal, 218
- further information, 243
- ignoring kernel protections, 239–243
- kernel ABI, 244
- kernel API, 243–244
- kernel events, 238–239
- system calls, 219–224
- undefined, 218

kernel symbol table, 250

kernel threads

- creating, 276–277
- defined, 204
- identifying in `ps` output, 204–205
- LKM example using, 277–283
- starting, 276
- stopping, 276–277
- task allocation using, 201, 204–205
- for Work Queues, 276

Kernighan, Brian W. (UNIX co-creator), 288

`kexec()` interface, 222, 306

keyboard shortcuts (music player), 327–328

keys (MySQL), 409

klogd tool, 256–257

`kmalloc` function, 259–260

`.ko` extension for module object file, 247

`kthread_create()` function, 276

`kthread_run()` function, 276

`kthread_stop()` function, 276–277

L

LAMP (Linux-Apache-MySQL-PHP). *See also specific components*

- Apache, 396, 399–403
- certification, 398
- compared to commercial products, 396, 397
- components, 395–396
- cost and licensing, 398
- defined, 396
- DVD library example, 433–442
- integration with other technologies, 399
- MySQL, 396, 404–411
- PHP, 397, 411–433
- prevalence of, 395
- as rebel platform, 397
- support and services, 398
- vendor/developer community, 399

languages, foreign. *See internationalization*

layers of abstraction

- device files, 224
- Hardware Abstraction Layer (HAL), 350–358
- for internationalization, 83
- kernel task abstraction, 200–205
- overview, 51–52
- system calls, 219
- virtual memory pages, 205–206

`ld` command, 36

`ldd` command, 30

`LD_LIBRARY_PATH` environment variable, 30

Lerdorf, Rasmus (PHP developer), 397

Levine, John R. (*Linkers and Loaders*), 40

libCurl library

- compiling applications, 142
- connecting to network resources, 142–145
- `curl_easy_cleanup()` function, 143
- `curl_easy_setopt()` function, 142–143
- downloading, 141
- easy C function format, 142
- `getheaders.c` program example, 146–147
- `getweb.c` program example, 143–145
- initializing, 142
- installing, 141
- multi C function format, 142
- options, 143, 146, 147
- storing retrieved data, 146–147

libglade library (GNOME), 318, 324

libhal C helper library, 351

libm (math library), 28**libraries. See also specific libraries**

- built-in functions vs., 31
- compiling supporting code with GCC, 26
- database implementation via, 151, 152
- discovering application requirements for, 30
- in GNOME Developer Platform, 316–318
- GNU Libtool for generating, 72, 81
- locations for, 29–30
- network programming libraries, 140
- for network protocols, 140–147
- portability aided by, 51–52
- shared, creating, 29–30
- shared vs. static, 28–29
- system calls by, 202, 219–220
- system, prefix for, 28

Linkers and Loaders (Levine), 40**linking object files**

- automatically, GCC capabilities for, 25–26
- creating executable from object code, 26–27
- ELF files, 36
- by GNU linker, 36
- linker scripts used by kernel, 45–46

Linux From Scratch project, 8**Linux Journal, 15****Linux Kernel Mailing List (LKML)**

- kernel janitors group, 215
- kernelnewbies group, 215
- overview, 213–214
- submitting kernel patches, 214–215

Linux kernel modules (LKMs)

- bare minimum example, 253–255
- building character device drivers, 229–231
- compatibility issues, 251
- deferring work, 275–283
- defined, 209
- distributing, 284–285
- exported symbols, 257–259
- extending the kernel namespace, 250–251
- finding API documentation, 251–252
- further reading, 283
- getting into upstream kernel, 284
- header macros and functions, 256–257
- init.h header for, 256
- installing, 198
- kernel threads, 276–283
- .ko extension for module object file, 247

- listing with `lsmod`, 248–250
- loaded by system calls, 247
- locking considerations, 267–275
- logging, 256–257
- making available to `modprobe`, 198
- man pages, 251–252
- memory allocation, 259–267
- `module.h` header for, 255–256
- mutexes, 267
- overview, 247–250
- `plp.c` example, 209–211
- `plp_kloc.c` example, 269–275
- `plp_kmem.c` example, 260–267
- `plp_kwork.c` example, 277–283
- `plp_min.c` example, 253–255
- prevalence of, 248
- requirements essential for, 253–256
- semaphores, 267–268
- similarity to drivers in other OS, 247
- spinlocks, 268–269
- symbols in `procf`s kernel symbol table, 250
- test machine for, 253
- unexported symbols, avoiding, 258–259
- user tracking by, 269
- Work Queues, 275–276

Linux Standard Base (LSB), 4, 54–55**Linux user groups (LUGs), 15****Linux Weekly News (LWN), 216, 283****Linux-Apache-MySQL-PHP. See LAMP****list command (GDB), 40–41****listen() function, 124, 126–127****little endian mode, 33, 89–90. See also endianness****LKML. See Linux Kernel Mailing List****LKMs. See Linux kernel modules****load_file() function, 337****local build environment, configuring, 21–22****locales**

- `gettext` library use of, 86–88
- internationalization using, 82–83
- support for, 88

locking for Linux kernel modules

- `plp_kloc.c` example, 269–275
- semaphores for, 267–268
- spinlocks for, 268–269

logging

- kernel mechanism for, 256–257
- in PHP, 427–428

loops in PHP, 417

`ls` utility, 224–225

LSB (Linux Standard Base), 4, 54–55

`lshal` command, 357–358

`lsmod` command, 248–250

LUGs (Linux user groups), 15

LWN (Linux Weekly News), 216, 283

M

magic typecasts (PHP), 412, 414–415

mailing lists, 16

`main.c` program, 320–321

`make` command

building sources, 22, 76–77

cleaning source code for distribution, 22

configuring the kernel, 190–192

installing binary with, 77–78

`INSTALL_MOD_PATH` variable, 198

for `libCurl` library installation, 141

`make config` target, 191–192

`make_modules` target, 230

`Makefile.am` file, 80–81

Makefiles

for building sources, 22–23

for Debian packages, 68

defining rules, 39

further documentation, 40

generation by automated tools, 23

GNU Automake tool for, 80–81

GNU Make tool for, 39–40

for music player main window, 321

“Making Hardware Just Work” (Pennington), 350

`malloc` function, 259

man pages

for GCC, 34

for Linux kernel modules, 251–252

math library (libm), 28

memory. See RAM; virtual memory

memory allocation in kernel modules

`kmalloc` function for, 259–260

`malloc` not available for, 259

`vmalloc` function for, 260–267

Memory Management Unit (MMU), 205, 206

menu bar (music player), 326–327

`merge` command (Bazaar), 111–112

Mesa project, 364, 366–367

metadata (music player), 337–338

MIME types, mapping files to, 360

“-mm” kernel tree, 215

`mmap()` system call, 206

`mm_struct` structures, 207–208

MMU (Memory Management Unit), 205, 206

`modprobe` command, 198, 231

modular nature of Linux, 17

`module.h` header, 255–256

`module_init` function, 45, 228, 232

`Module.symvers` file, 257–258

monitor, computer, 362, 363

Morton, Andrew (“-mm” kernel tree maintainer), 215

music CDs, playing using SDL, 391–394

music player project

adding music playing capability, 329–335

callbacks, 328–329

control buttons, 325–326

creating directory for, 319

GStreamer library use, 329–335

keyboard shortcuts, 327–328

main window, 319–321

menu bar, 326–327

metadata, 337–338

opening files, 335–337

`playback.c` program, 330–335

`playback.h` file, 330

requirements, 319

seeking within files, 338–340

slider control, 324

text labels, 323–324

top-level container, 322

mutexes, 267

MySQL

changing default password, 405

client interface, 405

configuring and starting the database, 404–405

creating tables, 406–407

deleting records, 408–409

editing records, 408

inserting records, 407

installing, 404

licenses, 396

overview, 396

PHP support for, 432
 relational databases, 405–406
 relational model, 409–411
 searching for records, 407–408
 SQL with, 406–409

`my_work()` **function**, 275–276

N

network programming

datagrams vs. streams, 133–137
 libCurl library for, 140–147
 Linux socket programming, 119–133
 marking message boundaries, 137–140
 moving data, 133–140

networking devices (netdevs), 225, 233

NetworkManager, 358–359

`newemployee.c` **program (Berkeley DB)**, 157–159

`new_from_stock()` **constructor**, 325

`nonnull` **attribute (GCC)**, 45

normalization in MySQL, 411

Novell, 6–7

O

`objcopy` **tool (GNU)**, 37

`objdump` **tool (GNU)**, 37–39

objects in PHP, 417–418

opaque, defined, 85

`open()` **function (Berkeley DB)**, 155–156

open source, defined, 5

`open()` **system call, tracing**, 220–221

OpenGL (Open Graphics Library)

animation, 380–382
 creating a window, 368–373
 described, 364
 downloading and installing, 366–367
 drawing objects, 373–377
 GLUT toolkit, 365, 368–382
 license required for, 365
 Mesa project implementation, 364, 366
 multiple versions in separate locations, 367
 processing user input, 377–380
 programming environment, 367
 software implementation, 364
 video card support for, 364

OpenGL Utility Toolkit. See GLUT

Open18N (Open Internationalisation Initiative), 81

OpenSUSE Linux, 6–7, 14

optical drives, HAL device object for, 354–357

optimization

costs of, 33
 GCC options for, 32–33
 PHP techniques, 422–426

overriding kernel protections, 239–243

P

package management tools

dependency resolution by, 54
 installing development tools using, 10

packages. See also distributions

building for portability, 57–70
 Debian, building, 65–70
 dependency resolution for, 53–54
 distribution portability issues, 53–54
 kernel, packaging, 199
 modifying selection at install time, 9–10
 RPM, building, 58–64
 tarball for, 58

Pango text-rendering library (GNOME), 317

parameter handling in PHP, 428–429

password, changing default for MySQL, 405

patch files

kernel patches, 214–215
 for RPM-based distributions, 61
 for system call table, avoiding, 258–259

PEAR (PHP Extension and Application Repository), 432

`pear` **with PHP**, 427

Pennington, Havoc (“Making Hardware Just Work”), 350

percent sign (%), for PHP code blocks, 412

performance optimization with GCC, 32–33, 34

permissions

for CVS repository, 96–97
 udev defaults, 266–267

persistent data storage, 149–150. See also databases

`pg_ctl` **utility (PostgreSQL)**, 166–167

PHP (PHP Hypertext Preprocessor)

alternatives, 397
 arrays, 415–416, 418–420, 423–424
 configuring, 401

PHP (PHP Hypertext Preprocessor) (continued)

- countering XSS attacks, 429
- database support, 432
- defined, 411
- error handling, 420–421
- exceptions, 421–422
- frameworks, 432–433
- functions, 416
- further information, 412, 420, 430
- `include` function, 413–414
- installing additional software, 427
- installing modules, 401
- installing PHP5, 401
- language basics, 412–413
- logging, 427–428
- loops and iterators, 417
- magic typecasts, 412, 414–415
- mixing with HTML, 412–413
- objects, 417–418
- optimization techniques, 422–426
- overview, 397
- parameter handling, 428–429
- pass-by-value vs. pass-by-reference approach, 418–420
- pear with, 427
- `.php` suffix for files, 412
- PHP5 as focus of this book, 411
- `register_globals` setting in `php.ini`, 429
- `require` function, 413–414
- scope of variables, 416–417
- session handling, 429–430
- Smarty project, 432–433
- strings, 422–423
- unit testing, 430–432
- variables, 412
- websites, 412
- `xdebug` module, 424–426
- PHPUnit2 **PHP testing framework, 430**
- physical memory. See RAM**
- platform portability. See portability**
- `playbin` **element (GStreamer), 330**
- `play_file()` **function, 333**
- `plp.c` **Linux kernel module, 209–211**
- `plp_kloc.c` **Linux kernel module**
 - code listing, 270–275
 - locking considerations, 269–270
 - user tracking by, 269

- `plp_kmem.c` **Linux kernel module**
 - building and testing, 266
 - code listing, 261–265
 - overview, 260–261
 - `udev` and default permissions, 266–267
- `plp_kwork.c` **Linux kernel module, 277–283**
- `plp_min.c` **Linux kernel module, 253–255**
- portability**
 - building packages for, 57–70
 - cross-compilation for, 46–47
 - of distributions, 52–57
 - hardware, 88–92
 - internationalization, 81–88
 - layers of abstraction for, 51–52
 - libraries as aids for, 51–52
 - of Linux, 17, 51–52
 - LSB issues, 54–55
 - meanings of, 51
 - need for, 50–51
 - pure software vs. hardware, 49
 - software, defined, 50
 - of source code, 70–81
 - of UNIX, 50–51
- ports, referencing for socket programming, 122–123**
- PostgreSQL database server. See also**
 - specific functions**
 - building programs, 167
 - command execution functions, 173–174
 - connecting to, 169–172
 - creating an application database, 167–169
 - described, 165
 - downloading and installing, 165–167
 - `empinfo.c` program, 182–184
 - executing SQL commands, 173–181
 - `getemployees.c` program, 179–181
 - `getvals.c` program, 177–179
 - Group roles, 167–168
 - handling column data, 179–181
 - helper functions, 175
 - initializing the default database, 166
 - inserting records, 176–177
 - Login roles, 167, 168
 - `pg_ctl` utility, 166–167
 - `psql` command line login, 167
 - querying records, 177–179
 - retrieving multiple records, 179–181
 - SQL commands returning data, 177–179
 - SQL commands returning no data, 176–177

status functions, 171–172
 stopping, starting, and reloading the configuration, 166–167
 update.c program, 176–177
 user account for, 166
 using parameters, 181–184
 version.c program, 172–173

PQclear() **function**, 174, 179

Pqconnectdb() **function**, 169–170

PqconnectPoll() **function**, 169, 170–171

PqconnectStart() **function**, 169, 170

PQexec() **function**, 173, 174, 179, 181

PQexecParams() **function**, 173, 181–182, 184

Pqfinish() **function**, 169

PQgetvalue() **function**, 175, 177, 178–179, 184

PQnfields() **function**, 175, 179

PQntuples() **function**, 175, 179, 181

PQparameterStatus() **function**, 171, 173

Pqreset() **function**, 169

PQresultStatus() **function**, 174–175, 177

PQstatus() **function**, 171, 172, 173

PQuser() **function**, 171, 172

primary key, 409

print **command (GDB)**, 41–42, 292–293

printf() **function**, 129

print_message **routine**, 71

private Linux communities, 16

procfs **pseudo-filesystem**
 building a driver, 238
 creating entries, 235–238
 described, 234
 Linux kernel module symbols in, 250
 querying system information, 235
 setting system variables, 235
 typical entries, 234

proc/kcore **file (fake core dump)**, 43

ps **command**, 204–205

pserver, remote, 103–104

psql **command line program (PostgreSQL)**, 167

PTXdist website, 8

put() **function (Berkeley DB)**, 157

Q

Qemu virtualization product, 15

A Quarter Century of UNIX (Salus), 3

question mark (?) for PHP code blocks, 412

quilt patch-tracking utility, 215

R

RAM. See also virtual memory
 allocation in kernel modules, 259–267
 kernel development requirements, 186
 kernel memory mapping, 207
 memory leaks, 288
 memory-related errors, 288
 opening and memory mapping ranges, 239–243
 segmentation faults, 288
 task address spaces, 207–208
 viewing programs' address space, 206–207

rawhide development release (Fedora), 14

RDBMS (relational database management systems), 405–406

recv() **function**
 in bad client/server programs, 136, 137
 in better server program, 139
 client program use of, 128, 129
 server program use of, 125
 with TCP protocol, 133

recvfrom() **function**
 for connectionless sockets, 130–131, 132
 with UDP protocol, 133

Red Hat, 6, 13–14

redirecting IO using tty command, 42

register_globals **setting in php.ini**, 429

relational model (MySQL), 409–411

relational operators (MySQL), 410

relations (MySQL), 409

relationships (MySQL), 409

remote pserver, 103–104

removing. See deleting or removing

repository for Bazaar-NG
 checking in modifications, 110–111
 creating, 109–110
 merging branches, 111–112

repository for CVS
 adding files and directories, 99–100
 branching, 94, 102
 browsing history, 100
 checking in modifications, 99
 checking out sources, 98–99
 commit activity, 100
 conflict resolution, 100–101
 creating, 96–97
 cvsroot directory for, 96–97
 defined, 94

repository for CVS (*continued*)

- Eclipse functionality, 115–117
- editor for, 98
- importing a project, 97–98
- permissions, 96–97
- removing files and directories, 99–100
- tagging, 94, 102
- update process, 101

repository for git

- cloning, 113–114
- creating, 113
- managing sources, 114

repository for Subversion

- checking out sources, 106
- creating, 105
- defined, 94
- importing a project, 105

`require` **function (PHP)**, 413–414

`rmod` **utility**, 231

root account, sudo utility for, 13

RPM-based distributions

- building packages, 58–64
- configuring the build environment, 62
- converting to Debian packages, 64
- example build, 50–54
- installing on non-RPM-based systems, 64
- invoking the `rpmbuild` command, 62–64
- LSB compliance, 54
- overview, 7
- patching sources, 61
- portability issues, 56
- writing spec files, 58–60

`rpmbuild` **command, invoking**, 62–64

`rpmmacros` **file**, 62

`rtag` **command (CVS)**, 102

`run` **command (GDB)**, 292–293

runtime dynamic linker/loader, 29–30

S

Salus, Peter H. (*A Quarter Century of UNIX*), 3

`scanf()` **function**, 129

`schedule()` **function**, 201

`schedule_delayed_work()` **function**, 276

scheduler, kernel, 201–202

`schedule_work()` **function**, 276

SCM (software configuration management). See **also CVS (Concurrent Version System)**

- activities associated with, 94
- Bazaar-NG for, 109–112
- branching, 94, 102
- centralized tools, 95–108
- centralized vs. decentralized, 93, 95
- changesets, defined, 94
- CVS tools for, 95, 96–104
- decentralized tools, 108–115
- Eclipse functionality, 115–117
- git tool for, 95, 108, 112–115
- GNU Arch for, 109
- integrated tools, 115–117
- need for, 94
- overview, 94
- repository, 94
- Subversion for, 104–108
- tools for distributing sources, 20

scope of variables (PHP), 416–417

scratch boxes, 14

SDL (Simple Directmedia Layer)

`audiotest.c` program, 390–391

`cdtest.c` program, 392–394

closing the library, 383

described, 365

displaying an image, 384–388

downloading and installing, 382–383

`imagetest.c` program, 386–388

initializing the library, 383–384

playing an audio file, 388–391

playing music CDs, 391–394

programming environment, 383

`SDL_AudioSpec` structure, 388–389

`SDL_Surface` objects, 384, 385

website, 382

`SDL_BlitSurface()` **function**, 385–386, 388

`SDL_CDOpen()` **function**, 391–392, 393–394

`SDL_CDPlayTracks()` **function**, 392, 394

`SDL_CDStatus()` **function**, 394

`SDL_CloseAudio()` **function**, 391

`SDL_Delay()` **function**, 394

`SDL_FreeWAV()` **function**, 391

`SDL_Init()` **function**, 383–384

`SDL_LoadBMP()` **function**, 385, 388

`SDL_LoadWAV()` **function**, 389

- SDL_MixAudio() **function**, 389–390
- SDL_OpenAudio() **function**, 391
- SDL_PauseAudio() **function**, 390, 391
- SDL_Quit() **function**, 383
- SDL_SetVideoMode() **function**, 384–385, 387
- SDL_UpdateRects() **function**, 386, 388
- section **attribute (GCC)**, 45
- seeking within files (music player)**, 338–340
- seek_to() **function**, 338
- seek_value_changed() **function**, 338
- segmentation faults**, 288
- SELECT **clause (SQL)**, 407
- semaphores (sleeping locks)**, 267–268
- semicolon (;) terminating SQL statements**, 406
- send() **function**
 - in bad client/server programs, 137
 - client program use of, 128, 129
 - server program use of, 125
 - with TCP protocol, 133
- sendto() **function**, 130, 133
- servers. See also PostgreSQL database server**
 - functions, 124–127
 - implementing databases via, 151–152
 - remote pserver, 103–104
 - TCP, 125–127, 134–135, 138–139
 - UDP, 131–132
- session handling in PHP**, 429–430
- shared libraries**, 28–30
- shutdown() **function**, 130
- Shuttleworth, Mark (Ubuntu project founder)**, 7
- Simple Directmedia Layer. See SDL**
- SimpleTest **PHP testing framework**, 430
- 64-bit cleanliness**, 89
- sleeping locks (semaphores)**, 267–268
- slider control (music player)**, 324
- Smarty PHP project**, 432–433
- sockaddr **structure**, 122
- sockaddr_in **structure**, 122–123
- socket() **function**
 - client program use of, 128
 - creating a socket, 120–121
 - server program use of, 126
- socket programming**
 - connectionless sockets, 130–133
 - connection-oriented sockets, 123–130
 - creating sockets, 121–122
 - domain values for sockets, 120–121
 - file descriptors for sockets, 120, 121–122
 - network address/port pairs, 122–123
 - referencing sockets, 122
 - sockets, defined, 120
 - type values for sockets, 121
- software configuration management. See SCM**
- Software in the Public Interest (SPI)**, 8
- software portability. See portability**
- source code**
 - cleaning for distribution, 22
 - examining using GNU objdump tool, 37–39
 - finding GPL violations, 299
 - installing with make command, 77–78
 - kernel sources, 187–190
 - patch file for, 61
 - providing for Linux kernel modules, 284
 - source-level portability, 70–81
 - viewing, 40–41
- source-based distributions**
 - INSTALL file with, 22
 - overview, 8
 - portability issues, 57
 - README file with, 22
 - unpacking, 20
- spec file for RPM packages**, 58–60
- SPI (Software in the Public Interest)**, 8
- spinlocks**, 268–269
- SQL (Structured Query Language)**
 - CREATE TABLE statement, 406–407
 - defined, 152
 - DELETE statement, 408–409
 - INSERT statement, 407
 - with MySQL, 406–409
 - overview, 406
 - SELECT clause, 407
 - statements, notes about, 406
 - UPDATE statement, 408
 - WHERE clause, 408
- SSH connection, CVS over**, 103
- SSL with Apache**, 402–403
- “-stable” kernel team**, 215–216
- Stallman, Richard (GNU developer)**, 2, 4, 5, 23
- static libraries**, 28
- stop_playback() **function**, 332–334
- strace **command**, 202–203, 220–221
- strings, optimizing in PHP**, 422–423
- Structured Query Language. See SQL**

Subversion

- changesets, 107
- checking in modifications, 106–107
- checking out sources, 106
- commit activity, 107
- creating a project, 105
- creating a repository, 105
- distributed development, 107–108
- importing a project, 105
- overview, 104–105
- project website, 104

sudo utility, 13

SuSE Linux, 7, 10

Swift, Jonathan (*Gulliver's Travels*), 89, 92

`sysfs` interface, 231, 238

`sys_open()` system call, 221

system call table

- overview, 222–223
- patching, avoiding, 258–259

system calls. *See also specific system calls*

- implementing your own, 223–224
- library use of, 202, 219–220
- Linux kernel modules loaded by, 247
- overview, 202, 219–220
- tracing, 202–203, 220–221
- transferring control to the kernel, 221
- typical examples, 219
- `vsyscall` optimization, 224

system terminal, 11

T

`tag` command (CVS), 102

Tanenbaum, Andrew (Minix developer), 3

tarball, 20, 58

task abstraction by kernel

- execution contexts, 203–204
- `mm_struct` structures, 207–208
- overview, 200–201
- scheduler, 201–202
- system calls, 202–203
- `task_struct` structures, 200–201, 207
- threads, 201, 204–205

`task_struct` structures

- for kernel threads, 276
- `mm_struct`, 207–208
- overview, 200–201

TCP communications

- bad client program, 135–137
- bad server program, 134–135
- better server program, 138–139
- simple client program, 128–129
- simple server program, 125–127
- streaming technology used by, 133
- UDP communications vs., 133
- using message boundary markers, 140
- using message size to identify boundaries, 138–139

`tcpclient.c` program, 128–129

`tcpserver.c` program, 125–127

terminal, 11

`testdraw.c` program, 375–377

testing. *See also debugging*

- kernel, 197–199
- Linux kernel modules, 253
- unit testing in PHP, 430–432
- virtualization products for, 9, 14–15

`testkey.c` program, 378–380

`testtimer.c` program, 380–382

`testwin.c` program, 370–373

text editors

- for CVS, 98
- `vi` vs. `emacs`, 11–12
- word processors as, avoiding, 11

text labels (music player), 323–324

threads. *See kernel threads*

`time()` system call, 223

toolchains, 9–10, 19. *See also GNU Toolchain*

Torvalds, Linus (father of Linux)

- as `git` tool creator, 95, 108
- on kernel development, 313
- as Linux creator, 3

translation using `gettext` library, 85–88

`trig.c` program example, 27–28

`tty` command (GDB), 42

tuples, defined, 409

typecasts

- magic, in PHP, 412, 414–415
- on 64-bit machines, 89

U

Ubuntu GNU/Linux, 7, 10, 14

`udev` permissions, default, 266–267

UDP communications

- datagrams sent by, 133
- simple client program, 132–133
- simple server program, 131–132
- TCP communications vs., 133

`udpclient.c` **program**, 132–133

`udpserver.c` **program**, 131–132

`ulimit` **command**, 296

UML (User Mode Linux), 309–312

undefined kernel interfaces, 218

Unicode, 83–84, 88

unit testing in PHP, 430–432

universal installer utilities, 58

unpacking a tarball, 20

`up()` **function**, 268

update process for CVS, 101

`UPDATE` **statement (SQL)**, 408

`update.c` **program (PostgreSQL)**, 176–177

`update-index` **command (git)**, 114

user limits, viewing, 296

User Mode Linux (UML), 309–312

user tracking by kernel modules, 269

UTF-8 character encoding

- for Unicode, 83
- wide characters with, 84–85

V

Valgrind runtime diagnostic tool, 298

variables

- `build_gui()` **function**, 327–328
- in PHP 412, 416–417
- viewing values with GDB, 41–42

vendors of Linux

- kernel configuration options, 193
- kernels developed by, 190
- portability issues, 55
- quality issues, 56

`version.c` **program (PostgreSQL)**, 172–173

versioning, kernel, 189–190

video cards, 362, 363, 364

vim text editor, 11–12

virtual hosting (Apache), 400

virtual memory

- defined, 205
- kernel memory mapping, 207
- mapping, 206–207
- page abstraction, 205–206
- task address spaces, 207–208

virtualization products, 9, 14–15

`vmalloc` **function**

- overview, 260
- `plp_kmem.c` **example using**, 260–267

VMware virtualization product, 9, 14

vsyscall optimization, 224

W

`wchar_t` **data type**, 84–85

Web resources. See **Internet resources**

`WHERE` **clause (SQL)**, 408

wide characters, 84–85

word processors, avoiding as text editors, 11

Work Queues, 275–276

X

X Windows specification

- common interface provided by, 362
- described, 362
- software packages implementing, 363–364

xdebug module for PHP, 424–426

Xen virtualization product, 9, 15

XFree86 X Windows implementation, 363–364

X.org X Windows implementation, 363–364

XSS (Cross-Site-Scripting) attacks, 429

Y

YaST package management tool (SuSE), 10

yumex package management tool (Fedora), 10

