

Index

A

abstract binary model class, 97, **97**, 98, **98–99**, 99, **99–100**
acknowledgment for ping, using `socket_recvfrom()`, 36, **36–37**
address model class, 97, **97–98**
Address object, 100–101, **100**, **101**
address type constants for hit counter, 23, *23t*
`addValue()`, in custom structured file, 298, **298–299**
aggregator for UDP hit counter, 22–25
Ajax, 397
 credit card processor application and, 219, 235
Alternative PHP Cache (APC), 385
American Standard Code for Information Interchange. *See* ASCII
AND, bitwise, 49*t*, 52–53, **52**, **53**
AND, Boolean, vs. bitwise, 52
Apache, 324, 398
`appendIndexSegment()`, in custom structured file, 311, **311–313**
application layer, OSI, 7
`array_diff()`, 61
`array_shift()`, 93
ArrayAccess interface, SPL and, 202, 203, 212–214, **212–214**
ArrayObject, 211–214, **211**
arrays, and SPL, 201
ASCII, 7, 122–124, **124**, 134, **134**
 ASCII bytes in, 136, **136–137**

EBCDIC-to-ASCII converter, 124, **124–126**
hex representation and, 48
high-bit characters in, 135, **135**
Telnet and, 77
asynchronous operations with encryption, 217–244.
 See also credit card processor application,
 Asynchronous
asynchronous processing, 396
Authorize.net, 240, 244

B

backups, 392–393
Berkeley DB, 220
big-endian, 73, 254–256, **254**, **256**
binary large objects (BLOBs), streams and, 187
binary protocols, 47–120. *See also* hacking a
 network protocol
 AND, bitwise, and, 52–53, **52**, **53**
 bitwise operations and, 49–50, 49*t*
 Domain Name Service (DNS) as, 47
 hacking a network protocol and, 55–120. *See also*
 hacking a network protocol
hexadecimal representation and, 48
low-level programming and, 47
Network File System (NFS) as, 47
NOT, bitwise, and, 54–55, **55**
OR, bitwise, and, 50–52, **50–52**
Telnet as, 47
XOR, bitwise, and, 53–54, **54**

- bindColumn(), 187
- bindParam(), 364
- bitmaps, 264–266, **264**, **265–266**
- bitwise operations, 49–50, 49*t*
 - broadcast address calculation using, 39–40, **39**, **40**
 - endianness and, 73, 254–256, **254**, **256**
 - examining bits in, 52, **52**
 - reading bits in, 51–52, **51**, **52**
 - testing for bits in, 53, **53**
 - writing bits in, 50–51, **50**, **51**
- blocks, block groups, Ext2 and, 266–269, 267*t*, **268–269**, 282–284
- Boolean vs. bitwise operators, 50, 52
- breakpoints, 355, 356–359, **356**, **357**, **358**
- broadcast addresses, sending to, 39–43
 - calculating, using bitwise OR, 39–40, **39**, **40**
 - heartbeat monitor application using, 40–43, **43**
 - sending to, 40, **41–42**
 - socket_bind() and, 42
 - socket_sendto() and, 42
 - socket_set_option() and, 42
 - TCP and, 43
 - UDP and, 40, 43
- browser. *See* Web browsers
- bytes, endianness and, 73, 254–256, **254**, **256**
- BZip2, 156, 167
- C**
- calculateChecksum(), 35
- calculating needs before writing code, 394–396, **394–395**
- calculating which network a host is on, 13–15, **13**, **14**
- canonical name (CNAME) query, in hacking a network protocol, 68–69, **68**
- Carmack, John, 47
- Cascading Style Sheets (CSS), 48, 397
- character encoding, 121–154. *See also* Unicode and UTF-8
 - ASCII and, 122–123, 124, **124**, 134, **134**
 - bit-level code in, 133, **133–134**
 - conversion tables and, 124
 - EBCDIC and, 122–123, **123**
 - EBCDIC-to-ASCII converter, 124, **124–126**
 - English vs. other languages in, 121–122, 126
 - EUC-JP character set in, 129–133, **129**
 - high-bit characters and, 126, **127**, 135, **135**
 - HTML and, 129
 - ISO-8859-1 encoding and, 127, **127**
 - ISO-8859-7 encoding and, 128, **128**
 - Japanese character sets in, 129–133, **129**
 - JIS-X-0201 in, 130–132, **132**
 - lead bytes in, 130, **130**, 136, **136–137**
 - multi-byte (variable-width) character sets and, 129–133
 - Unicode and UTF-8 in, 133–154
 - Windows-1255 encoding and, 128, **128**
- character large objects (CLOBs), streams and, 187
- checksum calculation, ICMP and, calculateChecksum(), 35–36, **35**
- chunks, 257
- CIFS, 8
- class, in Ext2, 276–279, **276–277**
- classes of networks, 11–18, 12*t*
 - calculating which network a host is on in, 13–15, **13**, **14**
 - Classless Inter-Domain Routing (CIDR) and, 12
 - Internet Assigned Numbers Authority (IANA) and, 12
 - IP addresses and, 12
 - subnet masks/netmasks in, 13–18
 - subnetting and, 15–18, 16*t*, 17–**18**
- Classless Inter-Domain Routing (CIDR), 12
 - reserved IP addresses and, 18–19, 18–19*t*
- client/server networks, packet routing in, 9–11, **10**
- close(), in credit card processor application, 227
- Code Tracing, 349, 363–366, **365**, **366**
- code, good coding concepts for, 350–353
- command bytes, Telnet, 77*t*
- Common Internet File System (CIFS), 8
- compression, in WAV files, 257
- compression on-the-fly, 163–167, **163–167**
- constructor creation, for credit card processor application, 225–226, **225**
- context, streams and, 160–167
- conversion tables, character encoding, 124
- count(), 204
- Countable interface, in, 202–204, **203**, **204**
- countBits(), Ext2 and, 266
- crawler for HTTP, streams and, 160–163, **160–163**

- CREATE, **208**
- Create, Read, Update, Delete (CRUD) operations, in
hacking a network protocol, 94–95, 117–119
- createDb(), 198
- createHttpJob(), 238
- createNewKeyLocation(), in custom structured file,
294, **294**
- credit card processor application, asynchronous,
217–244
- Ajax in, 219, 235
 - Authorize.net and, 240, 244
 - back end construction for, 238–242, **238–239**
 - file_get_contents() in, 242
 - POST in, 239–242, **240–241**
 - credit card processing in, 232–235
 - POST credit card to queue, 234–235, **234**
 - submit credit card, 233, **233**
 - user interface for, 234–235, **234**
 - Zend_Form component for, 232–233
- encrypting data in, 235–238
- createHttpJob() in, 238
 - JobQueue connection and, 237–238
 - openssl_private_decrypt() in, 235
 - OpenSSL extension for, 235
 - openssl_get_privatekey() in, 235
 - openssl_get_publickey() in, 235
 - openssl_public_encrypt() in, 235
 - POST in, 237–238, 237
 - private key creation for, 236, **236**
 - public key creation for, 236, **236**
 - request encryption for, 237, **237**
- front-end machine that connects with JobQueue
in, 237–238
- HTML in, 219
- HTTP request in, 219
- JavaScript in, 219–220, 235
- JavaScript Object Notation (JSON) in, 219
- Job Queue and queuing system flow in, 218, **218**,
220, 237–238
- jQuery in, 219, 235
- message queue setup for, 220–232
- checking data in, 226–227, **226**
 - close() in, to close socket in, 227
 - connection object in, 221–223, **223**
 - constructor creation in, 225–226, **225**
 - GET and POST in, 221, 226–229, **229**, 230,
232, **232**
 - getConnection() in, 225–226, **225**
 - getReadConnection() in, 224, **224**
 - getURI() in, 223–224, **223**
 - getWriteConnection() in, 224, **224**
 - handleHeaderReceived() in, 227, 228, **228**
 - handlePostReceived() in, 230–231, **230**
 - host file entries in, 220, **220**
 - HTTP headers in, 227
 - HTTP response in, 231, **231**
 - mainline code for, **224–225**, 225
 - property declaration for connections in, 221, **221**
 - reading data from socket in, 226, **226**
 - setData() in, 229, 231, **231**
 - setPost() in, 229, **229–230**
 - socket_write() in, 227, **227–228**
 - testing POST to the queue, 232, **232**
- overview of, 217–220, **218**
- POST and GET in, 244
- testing, 242–244, **243, 244**
- cryptography, 53–54
- CSS, 48, 397
- Curl wrapper, streams and, 156
- current(), 205
- Cyrus, 45
- D**
- daemons, 321–348
- fork(), forking, prefork and, 324
 - HTTP and, 324
 - inter-process communications using, 344–347,
344–347
 - Linux and, 327
 - locks and, 325
 - Maximum Transmission Unit (MTU) and, 341
 - Multi-Processing Module (MPM) in Apache and,
324
 - multi-tasking with, 323–326
 - PHP used as, 321–323
 - prefork spider, 326–344
 - child process request handler in, 333, **333**
 - Daemon class in, 330, **330**
 - Daemon object creation for, 341–342, **341**
 - execute() for, 331–332, **331, 333**

daemons, *continued*

- farmLinks() in, 338–340, **338–339**
- file_get_contents() in, 334
- for() loop in, 332
- front-end form for, in HTML from Zend_Form, 329, **329**
- getLinks() in, 336, **336–337**
- indexing page and gathering links in, 335, **335–336**
- inter-process communications for, 344–347, **344–347**
- Maximum Transmission Unit (MTU) and, 341
- output after spidering site in, 343, **343**
- output after submitting URL to, 343, **343**
- pcntl_fork() in, 332–333
- pcntl_wait() in, 332–333
- processPage() in, 334, 335–336, **335–336**
- readData() in, 334, **334**, 340–341
- search results from, 343–344, **344**
- sending link-parse requests to child workers in, 338–340, **338–339**
- sendPacket() in, 340, **340**
- socket_accept() in, 334
- socket_read() in, 335
- socket_select() in, 340
- socket_set_option() in, 332
- starting the daemon, 341–342, **342**
- submitting URL to daemon socket in, 329, **329–330**
- unserialize() in, 335
- Url class for, 327, **327–328**
- UrlResponse() in, 328, **328**
- willDisconnect() in, 334
- Zend_Search_Lucene search engine and, 330–331
- reasons to use, 321–323
- security and, 325–326
- serialization and, 325–326
- storing data and, avoiding, 325

data chunk, WAV files and, 257

data link layer, OSI, 9–10

databases

- CREATE, **208**, 208
- data added to, **208–209**
- lazy loading in, **209–210**

- date(), Ext2 and, 263
- dbObj2BinObj(), 114, **114**
- debug variables in MVC request and, 354–355, **354**
- Debugger, 349, 353. *See also* debugging
- debugging, 353–359
 - breakpoints in, 355, 356–359, **356**, **357**, **358**
 - Code Tracing and, 363–366, **365**, **366**
 - debug variables in MVC request, 354–355, **354**
 - starting code for, **353**
 - step into in, 355
 - step over in, 355
 - step return in, 355
 - steps in, 353, **353**
 - var_dump() and, 354–355
 - watch lists in, 355
- Defense in Depth security concept, 18
- delete(), hacking a network protocol and, server side, 118–120, **118–119**
- DHCP, 7
- diagnosing problems, 400–401. *See also* system-level diagnostics
- diagnostics. *See* system-level diagnostics
- dir_closedir(), streams and, 196, **196**
- dir_opendir(), streams and, 194, **195**
- dir_readdir(), streams and, 195–196, **196**
- directories, Ext2 and, 269–276, 275*t*, 283
- DNS. *See* Domain Name Service (DNS)
- Domain Name Service (DNS), 47
 - hacking a network protocol and, 57, 61–74, **61–73**
- Dynamic Host Configuration Protocol (DHCP), 7

E

- EBCDIC, 122–123, **123**
 - EBCDIC-to-ASCII converter, 124, **124–126**
 - hex representation and, 48
- Eclipse PHP Development Tools (PDT), 352
- encoding. *See* character encoding
- encrypting data, 53–54, 235–238. *See also* credit card processor application, asynchronous
- endianness, 73, 254–256, **254**, **256**
- English language and character encoding, 121–122, 126
- error handling
 - in hacking a network protocol, 106–107, **106**, **107**, 116, **116**

- in hit counter using UDP, 25
 - OSI transport layer and, 8
 - try/catch blocks for, 213–214, **214**
 - UDP, 19
- EUC-JP character set, 129–133, **129**
- execute(), in prefork spider daemon, 331–332, **331**, 333
- Execution Statistics, profiling and, 359–360, **360**
- explode(), 91, 92
- Ext2, 259–286
 - bitmaps in, 264–266, **264**, **265–266**
 - blocks, block groups in, 266–269, 267*t*, **268–269**, 282–284
 - class and, 276–279, **276–277**
 - countBits() in, 266
 - date() and, 263
 - directory information/structure in, 269–276, 275*t*, 283
 - execution flow for reading files in, 285, 285–286*t*
 - file manipulation in, 279–286, **279**, **280**, **281–282**, **284**, **285**, 285–286*t*
 - formatting the partition in, 260–261, **260**
 - fread() in, 266
 - getFile() in, 279, **279**, 285, 286
 - inodes in, 269–276, 270*t*, 271*t*, **271–272**, 273*t*, **274**
 - main() in, 285
 - mkfs.ext2 program and, 260, 264, 264*t*
 - pack() and, 263
 - readBlocks() in, 284, **284**, 286
 - readFileInode() in, 280, 281, **281–282**, 283, 284, 285
 - reading the superblock in, 262–263, **262**, **263–264**
 - readInode() in, 280, **280**, 286
 - substr() in, 273, 275
 - superblock position in, 260–261, 261*t*
 - testing code for, 284, **284**, **285**
 - unpack() in, 265, 266, 273
 - var_dump() in, 277, **277–278**
- Extended Binary Coded Decimal Interchange Code. *See* EBCDIC
- Extensible Markup Language (XML), 47
- F**
- fact chunk, WAV files and, 257
- failure, expecting, 392–393
- failure, single points of, removing, 397–400, **399**
- farmLinks(), in prefork spider daemon, 338–340, **338–339**
- fclose(), 27
 - streams and, 170, 182, **182**
- fetch and retrieve functionality, in hacking a network protocol, client class, 111, **111**
- fetch(), in hacking a network protocol, 99, 102, **102**, 112
- file access, structured. *See* structured file access
- file_exists(), streams and, 170, **171–173**
- file_get_contents()
 - credit card processor application and, 242
 - networking, sockets, and, 5, **5**
 - prefork spider daemon and, 334
 - streams and, 155, 163, **163**, 170, 178
- file-based vs. resource-based functions, in streams, 170
- fill bytes, UTF-8 and, 136, **136–137**
- filters, streams and, 159, **159**, 165–166
- findIndexOffset(), in custom structured file, 315
- findLastValueLocation(), in custom structured file, 297–298, **298**
- Flash, 322
- flow control in OSI transport layer, 8
- fopen(), 364
 - custom structured file and, 291
 - streams and, 155, **155**, 157, 170, 178
- for() loop, prefork spider daemon and, 332
- foreach loops, 204, 206
- fork() and forking, daemons and, 324
- format chunk, WAV files and, 257
- frameworks, 351–353
- fread(), 326
 - in custom structured file and, 293–294, 304, 308, 309, 317
 - Ext2 and, 266
 - streams and, 170
- freeBlock(), in custom structured file, 317, **317–318**
- front-end form, in HTML from Zend_Form, 329, **329**
- fseek(), in custom structured file, 291, 293, 295
- fsockopen(), 21–22, 27, 29, 42
 - streams and, 157

`fwrite()`, 326
streams and, 178

G

GD Graphics Library, 47
GET, in credit card processor application, 221,
226–229, **229**, 244
`getConnection()`, 225–226, **225**
`getFile()`, Ext2 and, 279, **279**, 285*t*, 286
`getFirstFreeSpace()`, 315, **315–316**
in custom structured file and, 292–293, **292–293**,
308
`getIndexLocationForSize()`, in custom structured file,
314, **314–315**
`getKeyValueLocation()`
in custom structured file, 295–297, **296**, 298, 308
performance profiling for, 360–362, **360**, **361**, **362**
`getLinks()`, in prefork spider daemon, 336, **336–337**
`getReadConnection()`, 224, **224**
`getValue()`, in custom structured file, 317, **317**
`getWriteConnection()`, 224, **224**
good code concepts, 350–353

H

hacking a network protocol, 55–120. *See also* binary
protocols
abstract binary model class in, 97, **97**, 98, **98–99**,
99, **99–100**
address model class in, 97, **97–98**
Address object in, 100–101, **100**, **101**
`array_diff()` in, 61
`array_shift()` in, 93
binary fetch method in, 102, **102**
building your own, 94–120
canonical name (CNAME) query in, 68–69, **68**
checking for response in, 83–84, **83**, **84**
converting database object to binary
representation in, 114, **114**
Create, Read, Update, Delete (CRUD) operations
in, 94–95, 117–119
Create, Update, and Delete functionality in, 117,
117
`dbObj2BinObj()` in, 114, **114**
`delete()` functionality in, server side, 118–120,
118–119

DNS and, 61–74, **61–73**
error handling in, 106–107, **106**, **107**, 116, **116**
`explode()` in, 91, 92
fetch and retrieve functionality in, client class,
111, **111**
`fetch()` in, 99, 102, **102**, 112
good reasons for, 55–56
`handle()` in, 112, 118
`handleCommand()` in, 83, 85
`handleTcp()` in, 79
handling request on server side, 112, **112–113**
helper models in, 94, **94–95**
hex values for IP address in, 71, **71**, 71
`interpretPacket` function in, 79, **79**, 83, **83**, 84–85,
84–85, 87–88, **88–89**, 93, **93**
iterating over socket operations in, 87, **87**
known functionality in, starting code to handle,
82–83, **82**
LISTEN_IP in, 57, 59
Mail Exchanger (MX) records in, 67–68, **67**, **68**
minimal negotiation connection in, 81–82,
81, **82**
model classes for, 96–102, **96–102**
packet creation in, 102, **102**
packet ID creation in, 69, **69**
php.net query in, 70, **70**
`printBytes()` in, 80, 103, **103**
query constants in, 69, **69**
receive binary model object in, server, 104,
104–105
REPLAY_IP in, 57
response packet in, 70, **70**
retrieving table object in, 113, **113–114**
retrieving type for model in, 101–102, **101**
send and receive binary model data in, client,
105, **105**
serialization function in, 100, **100**, **101**
serialize/unserialize objects in, 109–110,
109, **110**
SERVICE_PORT in, 57, 59
`socket_accept()` in, 61
`socket_read()` in, 85, 87
`socket_select()` in, 60
`socket_set_nonblock()` and, 60
table definition in, 96, **96**

- TCP/UDP packet replayer in, 56, **56–60**
 - Telnet and, 74–93
 - Telnet output in, 74, **74–76**
 - test code output for, 106, **106**, 115–116, **115**, **116**
 - unserialize() in, 107–109, **107–108**
 - verifying packet in, 72–73, **72–73**
 - handle(), in hacking a network protocol, 112, 118
 - handleCommand(), 83, 85
 - handleHeaderReceived(), 227, 228, **228**
 - handlePostReceived(), 230–231, **230**
 - handleTcp(), 79
 - handshake
 - OR, bitwise, and, 50
 - TCP, SYN–ACK, 26
 - Telnet, 76, **76**
 - hardware backup, 392–393
 - hashing function, in custom structured file, 291, **291**, 318
 - Heartbeat Monitor using broadcast addresses, 40–43, **43**
 - helper classes, 211
 - helper models, in hacking a network protocol, 94, **94–95**
 - hexadecimal representation, binary protocols and, 48
 - high-bit characters, in character encoding, 126, **127**, 135, **135**
 - hit counter
 - using non-blocking I/O, 30–32, **30–31**, **32**
 - using TCP, 27–29, **27–28**
 - using UDP, 20–25, **20–25**, 23*t*
 - hosts, calculating which network a host is on, 13–15, **13**, **14**
 - HTML, 48
 - character encoding and, 129
 - credit card processor application and, 219
 - HTML form for submission, **160**
 - HTTP, 47, 74, 397, 399
 - chunks and, 257
 - credit card processor application and, 219, 227
 - daemons and, 324
 - OSI network layers and, 6–7
 - TCP and, 27
 - HTTP crawler application, 160–163, **160–163**
 - hubs, 9
 - Hypertext Markup Language. *See* HTML
 - Hypertext Transfer Protocol. *See* HTTP
- ## I
- ICMP. *See* Internet Control Message Protocol
 - iconv, 147–149, **148**, **149**
 - if(), 53, **53**
 - IMAP, 7, 74
 - indexes, in custom structured file, 309–315
 - ini_set(), 25
 - innerHTML, 235
 - inodes, Ext2 and, 269–276, 270*t*, 271*t*, **271–272**
 - inter-process communication, prefork spider daemon and, 344–347, **344–347**
 - Internet Assigned Numbers Authority (IANA), 12, 18
 - Internet Control Message Protocol (ICMP), 8, 32–38
 - acknowledgment for ping in, using socket_recvfrom(), 36, **36–37**
 - checksum calculation for, using calculateChecksum() 35–36, **35**
 - pack() and, 33, 35
 - packet structure in, and creation of 33, 33*t*
 - ping application using, 33–38, **34**, **36–37**
 - ports and, 32
 - sending a packet in, and socket_create(), 34, **34**
 - TCP and, 32
 - UDP and, 32
 - unpacking ping response with, using unpack(), 37–38, **37–38**
 - Internet Message Access Protocol (IMAP), 7, 74
 - Internet Protocol (IP), 8, 9. *See also* IP addresses
 - Internetwork Packet Exchange (IPX), streams and, 158
 - interpretPacket function, in hacking a network protocol, 79, **79**, 83, **83**, 84–85, **84–85**, 87–88, **88–89**, 93, **93**
 - IP addresses, 8–9, 12
 - broadcast, 39–43
 - hex values for, using network protocol hack, 71, **71**
 - long2ip() and, 73
 - reserved, 18–19, 18–19*t*
 - IPv4, 12

ISO-8859-1 character encoding, 127, **127**
ISO-8859-7 character encoding, 128, **128**
Iterator interface, SPL and, 204–208, **205–208**
IteratorAggregate interface, SPL and, 202

J

Japanese character sets, 129–133, **129**
Java, 321, 322
JavaScript, 48, 219–220
 credit card processor application and, 235
JavaScript Object Notation (JSON), 219
JavaServer Pages (JSP), 321
JIS-X-0201 character encoding, 130–132, **132**
Job Queue, 396
 credit card processor application and, 218, **218**,
 220, 237–238
JOIN, in Stream Handler project, 173–176
jQuery, 219
 credit card processor application and, 235

K

KeepAlive, UDP performance of, 20
KeepAliveTimeout setting, Linux and, 383
Kevin's Framework Observation, 351
key(), 205

L

lazy loading, 202, 208–210, **209–210**
lead bytes, in character encoding, 130, **130**, 136,
 136–137
length check, UTF-8 and, 144, **144**
lighttpd, 399
linked lists, in custom structured file, 289
Linux, 259, 327. *See also* Ext2
 Alternative PHP Cache (APC) and, 385
 KeepAliveTimeout setting for, 383
 strace in, 379–387, **380–387**
 system-level diagnostics for, 376–387
 vmstat in, 376–379, **376**
 Zend Optimizer+ and, 385
LISTEN_IP, 57, 59
little-endian, 73, 254–256, **254**, **256**
loadHTMLfile(), 161
loading, lazy. *See* lazy loading
locks, daemons and, 325

long2ip() and, 73
low-level programming, 47
Lucene search engine, for prefork spider daemon,
 330–331

M

MAC, MAC addresses, 9
Mail Exchanger (MX) records, in hacking a network
 protocol, 67–68, **67**, **68**
Maximum Transmission Unit (MTU), 341
mbstring, 147–149, **148**, **149**
Media Access Control. *See* MAC, MAC addresses
Memcache, 400
memory, 400
message queue setup, 220–232. *See also* credit card
 processor application, asynchronous
middle character of Unicode blocks, 139, **139–143**
minimal negotiation connection, in hacking a
 network protocol, 81–82, **81**, **82**
mkdir() calls, in Stream Handler project, 176,
 176–178
mkfs.ext2 program. *See* Ext2
model classes, for hacking a network protocol,
 96–102, **96–102**
multi-byte (variable-width) character sets, 129–133
 iconv and mbstring for, 147–149, **148**, **149**
 UTF-8 and, 143, **143–144**
Multi-Processing Module (MPM), 324
MySQL, 45, 187
mysql_execute(), 364

N

.NET, 321, 322
NetBIOS, 8
netmask. *See* subnet masks/netmasks
Network Basic Input/Output System (NetBIOS), 8
Network File System (NFS), 47, 397–400, **399**
network layer, OSI, 8–9
networking and sockets, 5–45
 broadcast addresses and, sending to, 39–43
 classes of networks in, 11–18, 12*t*
 file_get_contents() and, 5, **5**
 Internet Control Message Protocol (ICMP) and,
 32–38
 non-blocking I/O and, 29–32

- OSI model and layers in, 6–11, **6**
- reserved IP addresses in, 18–19, 18–19*t*
- routers and routing in, 11–18
- streams API and, 5, **5**
- TCP in, 25–29
- Unix Domain Sockets and, 43–45, **44**
- User Datagram Protocol (UDP) in, 19–25
- Windows and, 43
- next(), 205
- NFS. *See* Network File System (NFS)
- non-blocking I/O, 29–32
 - hit counter using, 30–32, **30–31, 32**
 - socket_accept() and, 31
 - socket_read() and, 29, 31
 - socket_select() and, 29–31
 - socket_set_nonblock() and, 60
- NOT, bitwise, 49*t*, 54–55, **55**
- O**
- octdec(), tar files and, 249
- offsetExists(), 212
- offsetGet(), 212
- offsetSet(), 213, 214, **214**
- offsetUnset(), 213
- on-the-fly compression, 163–167, **163–167**
- Open Systems Interconnection. *See* OSI network model and layers
- opendir(), streams and, 194
- openssl_private_decrypt(), in credit card processor application, 235
- OpenSSL, 156
 - in credit card processor application, 235
- openssl_get_privatekey(), in credit card processor application, 235
- openssl_get_publickey(), in credit card processor application, 235
- openssl_public_encrypt(), in credit card processor application, 235
- operating systems and system-level diagnostics, 376. *See also* system-level diagnostics
- OR, bitwise, 49*t*, 50–52, **50–52**
 - broadcast address calculation using, 39–40, **39, 40**
- OR, Boolean, vs. bitwise, 50
- ord(), in custom structured file, 294
- OSI network model and layers, 6–11, **6**
 - application layer in, 7
 - data link layer in, 9–10
 - HTTP and, 6
 - network layer of, 8–9
 - physical layer of, 10
 - presentation layer in, 7
 - session layer in, 7–8
 - transport layer in, 8
- overloaded PHP functions, 149–150, **149, 150**
- P**
- pack(), 33, 35
 - in custom structured file, 299
 - in Ext2, 263
 - in WAV files, 253, 253–254*t*
- packet creation, in hacking a network protocol, 102, **102**
- packet replayer. *See* hacking a network protocol
- packet routing, 9–11, **10**
- packets, ICMP structure and creation of, 33, 33*t*
- parseMode(), streams and, 181
- partitioning, 393–396, **394–395**. *See also* Ext2
- pcntl_fork(), 332–333
- pcntl_wait(), 332–333
- performance, Execution Statistics for, 359–360, **360**
- Perl Compatible Regular Expressions (PCRE), UTF-8 and, 149, 150–152, **151**
- pfsocketopen(), 20
- PHP constants for Telnet command bytes, 78, **78**
- PHP Data Objects (PDOs), streams and, 187
- phpinfo(), 156, **156**
- physical layer, OSI, 10
- ping application using ICMP, 33–38, **34, 36–37**
- Point to Point Protocol (PPP), 9
- polling, 396–397
- POP3, 7
- ports, 8, 9, 74
 - hacking a network protocol and, 57
 - ICMP and, 32
 - TCP, 26–27
- POST
 - in credit card processor application and, 221, 226–229, **229, 230, 232, 232, 237–242, 240–241**
 - streams and, 161–162

Post Office Protocol 3 (POP3), 7
PPP, 9
prefork, daemons and, 324
prefork spider daemon, 326–344. *See* daemons
presentation layer, OSI, 7
printBytes(), 80
 hacking a network protocol and, 103, **103**
private key creation, 236, **236**
problem solving, 349
Process Explorer, Windows and, 387, **388, 389**
Process Monitor, Windows and, 387
processPage(), in prefork spider daemon, 334,
 335–336, **335–336**
Profile URL, 359
Profile, Zend Studio toolbar, 349, **349**
Profiler, 359. *See also* profiling
profiling, 359–362, **360, 361, 362**
 Execution Statistics in, 359–360, **360**
 Profile URL button in, 359
protocols, binary. *See* binary protocols
public key creation for, 236, **236**
Python, 322

Q

QueryBrowser (MySQL), UTF-8 and, 153–154, **154**

R

RAM, 400
readBlocks(), Ext2 and, 284, **284, 286**
readData(), 334, **334**
 in prefork spider daemon, 340–341
readFileNode(), Ext2 and, 280, 281, **281–282, 283,**
 284, 285*t*, 286*t*
readHeader(), tar files and, 250
readInode(), Ext2 and, 280, **280, 286**
RecursiveIterator interface, SPL and, 202, 203
REPLAY_IP, 57
replication, 393–394
request handling, in hacking a network protocol,
 server side, 112, **112–113**
require_once(), 375
reserved IP addresses, 18–19, 18–19*t*
Resource Interchange File Format (RIFF), 255
resource-based functions and streams, 170
Rethans, Derick, 353

rewind(), 205, 206
routers and routing, 9–18
 classes of networks and, 11–18, 12*t*
 Classless Inter-Domain Routing (CIDR) and, 12
 security issues and, 18
 subnet masks/netmasks and, 13–18
 subnetting in, 15–18, 16*t*, 17–**18**
Rsync, 398
Ruby on Rails, 321–322

S

Samba, 8
Secure Copy (SCP), 8
secure HTTP (SHTTP), 27
Secure Shell (SSH), 8, 80
Secure Sockets Layer (SSL), 7
security, 375
 routers, routing and, 18
 serialization and, 325–326
SeekableIterator interface, SPL and, 202, 203
Sendmail, 45
sendPacket(), in prefork spider daemon, 340, **340**
Serial Line Interface Protocol (SLIP), 9
Serializable interface, SPL and, 202, 203
serialization and security, 325–326
serialization function, 100, **100, 101**
serialize/unserialize objects, in hacking a network
 protocol, 109–110, **109, 110**
Server Message Block (SMB), 8
SERVICE_PORT, 57, 59
session layer, OSI, 7–8
setData(), 229, 231, **231**
setPost(), 229, **229–230**
sha1() hash, 291, 318
Shift Left/Right, bitwise, 49*t*
SHTTP, 27
Simple Mail Transfer Protocol (SMTP), 27, 74
single points of failure, 397–400, **399**
SLIP. *See* Serial Line Interface Protocol
SMB. *See* Service Message Block.
SMTP. *See* Simple Mail Transfer Protocol
socket constants for hit counter, 24, 24*t*
socket_accept(), 29, 61
 non-blocking I/O and, 31
 in prefork spider daemon, 334

- socket_bind(), 22–25, **22–25**
 - broadcast addresses and, 42
- socket_create(), 22–25, **22–25**
 - ICMP and, 34
- socket_read(), 22–25, **22–25**
 - in hacking a network protocol, 85, 87
 - non-blocking I/O and, 29, 31
 - in prefork spider daemon, 335
- socket_recvfrom(), 36, **36–37**
- socket_select()
 - in hacking a network protocol, 60
 - non-blocking I/O and, 29–31
 - in prefork spider daemon, 340
- socket_sendto(), broadcast addresses and, 42
- socket_set_nonblock() and, 60
- socket_set_option()
 - broadcast addresses and, 42
 - in prefork spider daemon, 332
- socket_write(), in credit card processor application, 227, **227–228**
- sockets. *See* networking and sockets
- software backup, 392–393
- spiders. *See* daemons, prefork spider
- SPL. *See* Standard PHP Library
- sprintf(), 33
- SSH. *See* Secure Shell
- SSL. *See* Secure Sockets Layer
- Standard PHP Library (SPL), 155, 201–215
 - ArrayAccess interface in, 202, 203, 212–214, **212–214**
 - ArrayObject and, 211–214, **211**
 - arrays and, 201
 - Countable interface in, 202, 203–204, **203, 204**
 - current() in, 205
 - disabling, 202–203
 - foreach loops used with, 204, 206
 - helper classes and, 211
 - interfaces and base functionality of, 202–203
 - Iterator interface in, 204–208, **205–208**
 - IteratorAggregate interface in, 202, 203
 - key() in, 205
 - lazy loading and, 202, 208–210, **209–210**
 - next() in, 205
 - offsetExists() in, 212
 - offsetGet() in, 212
 - offsetSet()/offsetUnset() in, 213, 214, **214**
 - RecursiveIterator interface in, 202, 203
 - rewind() in, 205, 206
 - SeekableIterator interface in, 202, 203
 - Serializable interface in, 202, 203
 - Traversable interface in, 202, 203
 - try/catch block and, 213–214, **214**
 - uses for, 201–202
 - valid() in, 205
- stat() data, in Stream Handler project, **171–173**, 173–175, **174, 175**
- step into debugging, 355
- step over debugging, 355
- step return debugging, 355
- strace, Linux and, 379–387, **380–387**
- Stream Handler project, 168–200. *See also* streams
 - binary large objects (BLOBs) and character large objects (CLOBs) in, 187
 - bindColumn() and, 187
 - context creation in, 197–200, **198–200**
 - createDb() in, 198
 - database table definition for, 168, **168**
 - delete directories in, 189–190, **190**
 - delete files in, 189, **189**
 - dir_closedir() in, 196, **196**
 - dir_opendir() in, 194, **195**
 - dir_readdir() in, 195–196, **196**
 - fclose() and, 170, 182, **182**
 - file system structure for, 168, **168**
 - file_exists() and, 170, **171–173**
 - file_get_contents() and, 170, 178
 - file-based vs. resource-based functions in, 170
 - fopen() in, 170, 178
 - fread() in, 170
 - fwrite() in, 178
 - insert and update functionality in, 186, **186**
 - JOIN in, 173–176
 - mkdir() calls in, 176, **176–178**
 - open a stream in a database using, 182, **182–185**
 - open a stream on a database row in, 179, **179–181**
 - open/close directory in, test of, 196–197, **196, 197**
 - opendir() in, 194
 - parseMode() in, 181
 - PHP Data Objects (PDOs) and, 187
 - read operations in, 186, **186–187, 188, 189**

NOTE: **Boldface** indicates illustrations and code; *t* indicates a table.

- Stream Handler project, *continued*
 - registering stream handler with database in, 169–170, **169**, **170**
 - rename files/nodes in, 191, **191–192**
 - search directories in, 194–197
 - stat() data in, **171–173**, 173–175, **174**, **175**
 - stream_close() in, 181
 - stream_context_create() in, 198
 - stream_context_get_options() in, 199, **199**
 - stream_eof() in, 187
 - stream_flush() in, 182, **182**
 - stream_open() in, 181, 185, 187
 - stream_read() in, 187
 - stream_write() in, 181
 - testing functionality of, 192, **192–193**, **194**
 - unlink() in, 189, **189**
 - url_stat() in, for file existence checking, 178
 - validate inserting different values in same node, 185, **185**
- stream_close(), 181
- stream_context_create(), 160, 162, 198
- stream_context_get_options(), 199, **199**
- stream_context_set_options(), 160
- stream_eof(), 187
- stream_filter_append(), 165–166
- stream_filter_remove(), 165–166
- stream_flush(), 182, **182**
- stream_get_filters(), 159, **159**
- stream_get_transports(), 158, **158**, **159**
- stream_get_wrappers(), 158, **158**
- stream_open(), 181, 185, 187
- stream_read(), 187
- stream_write(), 181
- streams, 155–200
 - binary large objects (BLOBs) and character large objects (CLOBs) in, 187
 - compression on-the-fly and, 163–167, **163–167**
 - configuration of, 156–159
 - context creation for, using stream_context_create(), 160, 162
 - context options for, using stream_context_set_options(), 160
 - contexts for, 160–167
 - Curl wrapper for, 156
 - decompression in, 166–167, **167**
 - fclose() and, 170
 - file_exists() and, 170
 - file_get_contents() and, 155, 163, **163**, 170
 - file-based vs. resource-based functions in, 170
 - filters for, using stream_filter_append() and stream_filter_remove(), 165–166
 - fopen() and, 155, **155**, 157, 170
 - fread() and, 170
 - fsockopen() and, 157
 - HTML form for submission in, **160**
 - HTTP crawler application for, 160–163, **160–163**
 - Internetwork Packet Exchange (IPX) and, 158
 - list available filters for, using stream_get_filters(), 159, **159**
 - list available transports for, using stream_get_transports(), 158, **158**, **159**
 - list available wrappers for, using stream_get_wrappers(), 158, **158**
 - loadHTMLfile() in, 161
 - optional modules for, 156
 - parseMode() in, 181
 - PHP Data Objects (PDOs) and, 187
 - phpinfo() to view, 156, **156**
 - POST and, 161–162
 - registering a stream handler for, 169–170, **169**, **170**
 - standard available, 155–156
 - Stream Handler project for, 168–200. *See* Stream Handler project
 - stream_close() in, 181
 - stream_eof() in, 187
 - stream_open() in, 181, 185, 187
 - stream_read() in, 187
 - stream_write() in, 181
 - TCP connection and, 157, **157**
 - unlink() in, 189, **189**
 - URI naming for, 157
 - URLs and, 155
 - var_dump() and, 163
 - writing your own transport for, 157–158
 - XPath to find form elements and submit code in, 161, **161–162**
- streams API, 5, **5**
- structured file access, 245–319
 - custom structured file and, 286–319

- adding a record in, 301, **301, 302**
 - adding new value with different key in, 302, **302**
 - addValue() in, 298, **298–299**
 - appendIndexSegment() in, 311, **311–313**
 - base class creation for, 290, **290**
 - bytes for first data record in, 301, **301**
 - bytes for new key record header in, 300, **300**
 - class members for index in, 310, **310**
 - constructor for, 290, **290–291**, 311, **311**
 - createNewKeyLocation() in, 294, **294**
 - execution flow in, 288–289, **288**
 - findIndexOffset() in, 315
 - findLastValueLocation() in, 297–298, **298**
 - fopen() in, 291
 - fread() in, 293–294, 304, 308, 309, 317
 - freeBlock() in, 317, **317–318**
 - fseek() in, 291, 293, 295
 - getFirstFreeSpace() in, 292–293, **292–293**, 308, 315, **315–316**
 - getIndexLocationForSize() in, 314, **314–315**
 - getKeyValueLocation() in, 295–297, **296**, 298, 308
 - getValue() in, 317, **317**
 - hash value after header bytes in, 300, **300**
 - hashing function for, 291, **291**, 318
 - header value of new record in, 302, **302**
 - hex editor results on code, 299, **299**
 - indexes and, 309–315, **310**
 - multiple key values in, 303, **303**
 - ord() in, 294
 - pack() in, 299
 - performance issues of, 306–309, **307, 308**, 316–317, **316**. *See also* indexes, above
 - purpose of, 287
 - read and delete records in, 303–305, **303–304**
 - reasons for, 286–287
 - record header format for 289, 289*t*
 - sha1() hash in, 291, 318
 - storage and linked lists in, 289
 - testing the code for, 305–306, **305–306**
 - touch() in, 291
 - unpack() in, 294
 - updated primary header field in, 302, **302**
 - values stored in a record in, 301, **301**
 - writeValue() in, 295, **295**, 298
 - Ext2 and, 259–286. *See also* Ext2
 - tar files and, 246–253. *See also* tar files
 - WAV files and, 253–259. *See also* WAV files
 - strunpack(), in tar files, 248
 - subnet masks/netmasks, 13
 - subnetting, 15–18, 16*t*, 17–18
 - substr(), in Ext2, 273, 275
 - superblock, in Ext2, 260–263, 261*t*, **262, 263–264**
 - switches, 9
 - SYN–ACK TCP handshake, 26
 - Sysinternals utilities in Windows, 387
 - Syslog, 45
 - system-level diagnostics, 375–389
 - for Linux, 376–387
 - Alternative PHP Cache (APC) and, 385
 - KeepAliveTimeout setting for, 383
 - strace in, 379–387, **380–387**
 - vmstat in, 376–379, **376**
 - Zend Optimizer+ and, 385
 - operating systems and, 376
 - require_once() and, 375
 - security and, 375
 - for Windows, 387–389, **388, 389**
 - Process Explorer for, 387, **388, 389**
 - Process Monitor for, 387
 - Sysinternals utilities for, 387
- T**
- table definition
 - in hacking a network protocol and, 96, **96**
 - including UTF-8 in, 152, **152**
 - tape archive files. *See* tar files
 - tar files, 246–253
 - block size in, 248
 - header record format for, 246, 246*t*
 - octdec() and, 249
 - readHeader() and, 250
 - reading a single header record for, 247, **247–248**
 - reading subsequent records in, 249–250, **249, 250**
 - record types in, 247, 247*t*
 - strunpack() and, 248
 - Uniform Standard Tape Archive (UStar)
 - information in, 250–253, 251*t*, **251, 252**
 - unpack() and, 248

- TCP, 8, 9, 25–29
 - broadcast addresses and, 43
 - flow control in, 26
 - fsocketopen() and, 27, 29
 - handleTcp() in, 79
 - handshakes in, SYN–ACK, 26
 - hit counter using, 27–29, **27–28**
 - HTTP and, 27
 - ICMP and, 32
 - packet delivery in, 25–26
 - packet replayer for. *See* hacking a network protocol
 - ports for, 26–27
 - secure HTTP (SHTTP) and, 27
 - Simple Mail Transfer Protocol (SMTP) and, 27
 - speed of, vs. UDP, 20
 - streams and, connecting to, 157, **157**
 - UDP vs., 26
 - Unix Domain Sockets and, 44
- Telnet, 47
 - ASCII and, 77
 - bytes interpreted code, 79, **79–80**
 - command bytes for, 77*t*
 - defining constants in, 87, **97**
 - hacking a network protocol and, 74–93
 - handshake, 76, **76**
 - interacting with terminal in, 86–87, **86**
 - negotiation phase for, 90–91
 - PHP constants for command bytes in, 78, **78**
 - ports for, 74
 - state in, TELNET_STATE_AUTHED and, 91, **91–92**
 - TELNET_STATE_AUTHED, 91, **91–92**
 - testing, unit, 366–374, **368–373**
 - Time to Live (TTL), 400
 - touch(), custom structured file and, 291
 - trace, strace, 379–387, **380–387**
 - tracing, code, 363–366, **365, 366**
 - Transmission Control Protocol (TCP). *See* TCP
 - transport layer, OSI, 8
 - Traversable interface, SPL and, 202, 203
 - try/catch blocks, 213–214, **214**
- U**
- UDP, 8, 9, 19–25
 - address type constants for hit counter using, 23, 23*t*
 - aggregator for hit counter using, 22–25
 - broadcast addresses and, 40, 43
 - drawbacks to use of, 19
 - error reporting and, 25
 - fsocketopen() used with, 21–22
 - hit counter example of using, 20–25, **20–25**, 23*t*
 - ICMP and, 32
 - ini_set() and, 25
 - KeepAlive performance in, 20
 - packet replayer for. *See* hacking a network protocol
 - psocketopen() vs., 20
 - service candidates for use of, 20
 - socket constants for hit counter using, 24, 24*t*
 - socket_create(), socket_bind(), and socket_read() in, 22–25, **22–25**
 - speed of, vs. TCP, 20
 - TCP vs., 26
 - Wireshark output for hit counter using, 21, **21**
- Unicode and UTF-8, 7, 133–154
 - ASCII bytes in, 136, **136–137**
 - bit-level code in, 133, **133–134**
 - bit mapping for, 136, 136*t*
 - converting high-bit characters in, 135, **135**
 - fill bytes in, 136, **136–137**
 - header verifying use of, 145, **145–146**
 - hex representation and, 48
 - inserting strings of, into database, 152–153, **152, 153**
 - lead bytes in, 136, **136–137**
 - length check in, 144, **144**
 - middle character of Unicode blocks, 139, **139–143**
 - multi-byte character sets and, iconv and mbstring for, 147–149, **148, 149**
 - multi-byte output in, 143, **143–144**
 - Perl Compatible Regular Expressions (PCRE) and, 149, 150–152, **151**
 - QueryBrowser (MySQL) use with, 153–154, **154**
 - setting database to use, 154, **154**
 - setting overload internal string function in, 149–150, **149, 150**
 - setting which charset php.ini should use, 145, **145**
 - table definition including, 152, **152**
 - types of bytes in, 136, **136–137**
 - utf8_encode()/utf8_decode() and, 146–147, **147**

- Unicode Transformation Format. *See* Unicode and UTF-8
- Uniform Resource Identifier (URI), 157
credit card processor application and, 223–224, **223**
- Uniform Resource Locator (URL)
streams and, 155
- Uniform Standard Tape Archive (UStar) information, 250–253, 251*t*, **251**, **252**
- unit testing, 366–374, **368–373**
- Unix Domain Sockets, 43–45, **44**
- unlink(), 189, **189**
- unpack(), 37–38, **37–38**
custom structured file and, 294
Ext2 and, 265, 266, 273
for ping response, unpack(), 37–38, **37–38**
tar files and, 248
WAV files and, 253, 253–254*t*, 256, **256**
- unserialize(), 107–109, **107–108**
prefork spider daemon and, 335
- URI. *See* Uniform Resource Identifier
- URL. *See* Uniform Resource Locator
- url_stat(), Stream Handler project and, for file existence checking, 178
- UrlResponse(), prefork spider daemon and, 328, **328**, 328
- User Datagram Protocol. *See* UDP
- user interface (UI), 397
- UTF-8. *See* Unicode and UTF-8
- utf8_encode()/utf8_decode(), 146–147, **147**
- V**
- valid(), 205
- var_dump(), 354–355
Ext2 and, 277, **277–278**
streams and, 163
- variable-width (multi-byte) character sets, 129–133
- vmstat, in Linux, 376–379, **376**
- W**
- watch lists, 355
- WAV files, 253–259
chunk length in, 257
compression and, 257
data chunk in, 257
endianness in, 254–256, **254**, **256**
fact chunk in, 257
file header for, 255*t*, 255
format chunk in, 257
metadata format of, 257, 257*t*
pack() and, 253, 253–254*t*
raw data in, **255**
reading metadata in, 258–259, **258**, **259**
Resource Interchange File Format (RIFF) and, 255
unpack() and, 253, 253–254*t*, 256, **256**, 258
- Waveform Audio Format. *See* WAV files
- Web browsers, packet routing in, 9–11, **10**
- willDisconnect(), prefork spider daemon and, 334
- Windows
networking and sockets with, 43–44
Process Explorer for, 387, **388**, **389**
Process Monitor for, 387
Sysinternals utilities for, 387
system-level diagnostics for, 387–389, **388**, **389**
- Windows-1255 character encoding, 128, **128**
- Wireshark output for hit counter using UDP, 21, **21**
- writeValue(), in custom structured file, 295, **295**, 298
- writing your own structured file system. *See* structured file access, custom
- X**
- XDebug, 352, 353
- XML, 47, 48
- XOR, bitwise, 49*t*, 53–54, **54**
- XPath to find form elements and submit code, streams and, 161, **161–162**
- Z**
- Zend Code Tracing. *See* Code Tracing
- Zend Debugger. *See* Debugger
- Zend Framework, 385, **386**, **–387**
- Zend Optimizer+, 385
- Zend Platform, 396
- Zend Profiler, 359
- Zend Server, 363, 364, 396
- Zend Studio, 352, 359
- Zend_Form, 232–233, 329, **329**
- Zend_Search_Lucene search engine, prefork spider daemon and, 330–331
- Zip, 156
- zlib, 156