

Index

Symbols and Numerics

... (ellipsis). *See* `vararg`
 . (dot) character class, 193, 209
 ' (apostrophe character, single quote), 33
 _ (underscore), 30
 * (asterisk) character, 195–198, 200, 209
 ^ (carat) character
 for anchoring patterns, 188, 203
 description of, 209
 \$ (dollar sign) character, 209
 - (hyphen) character
 in bracket classes, 190
 description of, 209
 in pattern matching, 197, 198, 200
 # (length operator) character, 46, 123
 # (number) character, 139, 470
 + (plus) character, 194, 198, 200, 209
 ? (question mark) character, 198, 200, 209
 % (percent) character classes
 in bracket classes, 192
 descriptions of, 209
 as formatting placeholder, 175–178
 in patterns, 187, 192
 = (equal) sign, 29, 34
 *1 pattern, 511
 1stThing identifier, 30
 1tn12 module, 525
 2-D action game, 544
 8-bit clean, 383

A

*a pattern, 510
 absolute value, function for, 266, 358
 abstraction
 benefits of, 70
 defined, 69
 discussed, 69–70

 functions and, 70
 statements and, 70
 accept method, 511, 512
 accumulator, 101
 activeLines field, 312
 actual argument, 84
 addition, 24
 address, 507
 adjusted value, 40
 adjustment function, 358–360
 aggregate function, 409
 AJAX
 time server implementation with, 476–478
 toolkit, 476–478
 algebraic equal sign, 29
 ambiguous syntax error, 215
 anchor, 188
 and operator
 discussed, 42–43
 nil value from, 43
 side effects and, 93
 de Andrade, Marcio Migueletto, 565
 angle conversion function, 361
 anonymous function, 104
 ANSI C, 281, 338
 Apache, 471–472
 defined, 1
 on Windows system, 472
 API (application program interface), 1, 379
 APOP (Authenticated Post Office Protocol), 532
 apostrophe character (single quote), 33
 application program interface (API), 1, 379
 application protocol, 524–536
 Aquario, 566, 577
 archive (mailing list)
 downloading, 598–599
 viewing and searching, 597–598
 argument. *See also* Actual argument
 actual, 84
 formal, 84
 keyword, 143–144
 variable scope of, 84

arithmetical operation

- addition, 24
- discussed, 23
- division, 24
- exponentiation, 24
- interpreter interaction for, 24–25
- multiplication, 24
- nonintuitive actions of, 27–28
- notations for, 25–26
- rounding, 28
- subtraction, 24
- in tables, 249–257

array(s)

- with gaps, 123
- length of, 123–124
- receive, 523
- send, 523
- splicing, with concatenation operator, 250–253
- tables and, 121–124

ASCII characters, 174, 192

assert function

- described, 333–334
- for errors, 220

assertion failed message, 220

assignment(s)

- defined, 29
- discussed, 29–30
- functions as, 103–105
- of global variables, 244–245
- multiple, 31
- variables on right side of, 31

associative table, 121

associativity, 49–51

assumption of errors, 219–221

asynchronous call, 476–478

Authenticated Post Office Protocol (APOP), 532

autoexec.bat, 5, 6

average function

- discussed, 74, 137
- returning values from, 73–74

B

backslash escaping

- for pattern matching, 189
- with quoted strings, 35–37

backtracking, 197

balanced delimiter, 202

base case, 97

Base64 encoding, 525

base-two number system, 44

bash shell, 4

basic output, of core library, 333

Berkeley sockets interface, 510

binary mode, 182

binary packages, prebuilt

- discussed, 18
- installing, on Unix-type systems, 19–20
- installing, on Windows systems, 20–21
- selection of, 18–19

binding, 375, 407

bitting, 551

block

- as chunk, 89
- defined, 86

blocking function, 288

Boole, George, 37

Boolean operator

- discussed, 41
- not unary operator (unary minus), 44–45
- and operator, 42–43
- or operator, 43–44

Boolean value

- comparing numbers with, 37–38
- comparing strings with, 38–40
- defined, 37
- discussed, 37

Boutell, Thomas, 395

bracket classes, 187, 190–191

- hyphen character in, 190
- percent character in, 192

break statement

- discussed, 63–65, 75
- return statement vs., 75

building

- discussed, 2–3
- gd graphics library, 395–397
- libcurl, 389–392
- luacurl, 392–393
- lua-gd, 397–399
- lua-sqlite3, 407–409
- with Microsoft Visual C++, 12–14
- with MinGW package, 16–18
- pack library, 383–384
- SQLite 3, 405–407
- with Tiny C Compiler (TCC), 14–16
- in Windows, 12–18

built-in function(s)

- discussed, 102–103
- replacing, 102–103

byte, 37, 46–47

byte order, 303

bytecode(s)

- chunks and, 81
- discussed, 299
- with luac, 299–303
- modules and, 241–242
- source codes and, 81, 302
- for troubleshooting, 302

bytecode interpreter, 81

C

C++. See Microsoft Visual C++

C compiler, 1, 2

C programming language

- calling Lua from, 421–423
- communicating between Lua and, 415–421
- defined, 1
- embedding Lua in, 414–415
- extension library of, 441–447
- format placeholders in, 179
- function environments in, 439
- indexing values in, 436–438
- libraries written in, 234
- modules with, 247
- obtaining functions in, 421
- retaining values in, 438–441
- stack of, 415
- upvalues in, 439–440
- userdata basic type in, 423–436
- using Lua in, 413–414

- C runtime library, 375**
- C stack, 96**
- calendars, dynamic, 478–481**
- call stack. See stack**
- callback, 130, 394**
- calls and calling**
 - of anonymous functions, 104
 - asynchronous, 476–478
 - functions, 108
 - of functions, in C, 421–423
 - for Lua, from C, 421–423
 - to web server, 476–478
- captures, 198–201**
- Carregal, Andre, 602**
- case-insensitive tables, 259–261**
- case-sensitivity**
 - in filenames, 241
 - of identifiers, 30
- cc command, 8**
- C/C++ compilers, 12**
- Celes, Waldemar, 596**
- CGI application, interactive**
 - discussed, 489
 - for forms, 497
 - helper routines, 489–498
 - Kepler for, 499
 - security issues of, 498
- CGI scripting**
 - developing, 498
 - executing, 469–470
 - for images, 482–484
 - privileges in, 475–476
 - problems with, 475–476
 - on Unix-type systems, 470
 - on Windows systems, 470–471
- character(s)**
 - absence of, 39
 - conversion of, to character codes, 173–174
 - defined, 32
 - discussed, 173–174
 - magic, 209
 - for pattern-matching, 209
- character class, 189, 197–198**
- character code, 173–174**
- chart(s)**
 - dynamic, on Web, 481–489
 - gd graphics library for, 481
 - in web pages, 485
- chat room, for Lua, 601**
- chunk(s)**
 - as block, 86
 - compiling and, 81
 - defined, 75, 89
 - discussed, 81–82
 - as functions, 81–83
 - in `lua.exe`, 82–83
 - mechanics of, 82–83
 - strings executed as, 83
- chunk-loading function, 328–330**
- ChunkSpy, 301, 302**
- churn, memory pool, 305**
- client connections, 523**
- Client for URLs. See cURL file transfer library**
- client object, 511**
- client side networking, 538–541**
- client socket, 512**
- close (filehandle), 366**
- closure**
 - defined, 109
 - discussed, 108–109
 - storage in, 109
 - for upvalues, 440
- cmd shell, 27**
- Coco (patchset), 280**
- code error, 220**
- “collect” action, 332**
- collectgarbage function, 304–305**
- colon syntax, 136**
- color, 401, 486**
- columnar data, 174–179**
- columns**
 - data in, 174–179
 - packaging, 441–442
- command-line argument, 141–142**
- command-line compiler, 576–577**
- command-line interface (shell)**
 - defined, 3
 - discussed, 3
 - environment (variables) of, 4–6
 - features of, 4
 - on Unix and Unix-like systems, 3
 - on Windows systems, 3
- command-line interpreter, 81**
- comma-separated value (CSV) format, 441–447**
- comments, 52–53**
- Common Gateway Interface. See under CGI**
- communication**
 - discussed, 503
 - with LuaSocket, 503–518
 - network, 512–518
 - by networking, 536–541
- community resources**
 - chat room, 601
 - discussion group, 566
 - forums, 601
 - LuaForge, 602–603
 - for LuaSocket, 508–509
 - mailing lists, 597–600
 - mirror sites for, 598
 - question and answer, 597
 - reference manual, 596
 - web site, 596
 - wiki pages, 601–602
 - workshops, 603
- compile-time error, 215**
- compiling**
 - applications, in Plua application, 572–573
 - chunks and, 81
 - defined, 7
 - discussed, 2–3, 7–18
 - libraries, in Plua application, 573–576
 - on Linux and Unix-like systems, 8–12
 - Lua tarball and, 7–8
 - LuaSocket, 503–505
 - in Palm OS emulator, 580
 - on Windows systems, 12–18
- compound statement**
 - break statement, 63–65
 - discussed, 54–65
 - do statement, 63–65
 - if statement, 54–58
 - for loop, 60–62
 - repeat loop, 62–63
 - while loop, 58–59

concatenation

concatenation. See **string concatenation operator**

concatenation operator, 250–253

concurrent tasks

client connections, 523

management of, with coroutines, 281–282

Configure function, 531

connection handling (LuaSocket), 518–524

connection-oriented (stream) socket, 510

constants, 363–364

control

altering, with return values, 74–76

coroutines for, 271–272

flow of, 74–76

transferring, with coroutines, 273

control character, 32

control sharing, coroutines for, 273–275

control structure, 54

control variable, 159

conversion

automatic, 48–49

of characters to character codes, 173–174

functions for, 171–173

of operands, 48–49

of strings, 171–173, 173–174

cooperation, of coroutines, 273–278

copying tables, 148–152

core library

basic output of, 333

chunk-loading functions in, 328–330

discussed, 325–335

environment functions in, 326

error-condition functions in, 333–334

error-containment functions in, 330–331

garbage-collections functions in, 332

metatable functions in, 326–328

module functions in, 331

table traversal functions in, 334–335

type and conversion functions in, 333

vararg-related functions in, 335–336

coroutine(s)

call stack of, 272

for concurrent tasks management, 281–282

for control sharing, 273–275, 277–278

control transference by, 273

cooperation of, 273–278

defined, 271, 272

discussed, 271

dispatching, 280

for event handling, 287–297

event loop, 288–295

functions vs., 272

instantiated, 272

iterating, 286–287

for multitasking, 273–278

for program control, 271–272

programs vs., 272–273

recursion and, 281

requirements for, 279–281

restrictions on, 280

for retaining state, 282–287

with `select` function, 518–522

status of, 278–279

thread of, 272–273

tokenizing, 282–286

variable scope in, 280

wrapping of, 273

yielding of, 279–280, 296–297

coroutine library, 336–337

coroutine.create function, 272, 273, 278, 336

coroutine.resume function, 272, 273, 336

coroutine.running function, 273, 336

coroutine.status function, 336

coroutine.wrap function, 272, 279, 336

coroutine.yield function, 272, 273, 275

cosine functions, 346, 352

“count” action, 332

CPU timing, 368

create key, 528

credit card numbers

formatting type for, 186

validating, 187–193

CSV (comma-separated value) format, 441–447

csv.parse function, 447

cURL (Client for URLs) file transfer library

discussed, 389

`libcurl`, 389–392

`luacurl`, 392–395

curl.new, 395

curly braces, 202

currentline field, 312

Cygwin system, 2

D

data error, 220

data structure, 117

database

access to, with Lua application, 590–592

discussed, 449

MySQL, 458

MySQL, 458–465

relational, 449–458

SQL, 458, 464–468

datagram socket, 510

datatype, 40

date functions, 368–369

“dead” status, 278

debug library

for call stack, 216

discussed, 308–321, 370–373

functions in, 321, 371–372

hooks in, 315–321

running code in, 308–315

debug.debug function, 309, 371

debug.getenv function, 371

debug.gethook function, 321, 371

debug.getinfo function, 311, 314, 371

debug.getlocal function, 309, 310, 371

debug.getmetatable function, 371

debug.getregistry function, 371

debug.getupvalue function, 314, 371

debug.setenv function, 371

debug.sethook function, 319, 371

debug.setlocal function, 310, 372

debug.setmetatable function, 265–266, 372

debug.setupvalue function, 314, 372

debug.traceback function, 167, 217, 229, 310, 372

decimal-point notation, 28

degrees in radians, function for, 361

delimiters, 202

Demo subdirectory, 399, 566

Demo1 subdirectory, 567

Demo2 subdirectory, 567

Denom field, 255
digest newsgroup, 599
Dijkstra, Edsger, 76
discussion group, 566
dispatching, 280
division
 discussed, 24
 by zero, 27–28, 48
DLL (dynamic-linked library)
 configuration options for, 376–377
 discussed, 376–377
DLL (dynamic-linked library)
 external references of, 376
 in TCC, 16
DNS (Domain Name System), 507
dns namespace, 507
do block, 86
do statement, 63–65
doc files, 569
doc subdirectory, 8
document root, 474
documentation, for Plua application, 570–571
doFile function, 328, 330
Domain Name System (DNS), 507
domain names, 507–508
doskey utility, 27
dotted-decimal notation, 507
dotted-quad notation, 507
double quotes, 32–33
duplicate lines, 183–185
dynamic linked library. See DLL
dynamic Web calendar, 478–481
dynamic Web chart, 481–489
dynamic Web content
 for asynchronous calls to server, 476–478
 calendars, 478–481
 CGI scripts, problems with, 475–476
 charts, 481–489
 discussed, 468
 on embedded web server, 468
 on extended web server, 469
 run time, 469
 serving, 474–475

E

e-mail, LuaSocket for, 529–536
embedded web server, 468
embedding Lua, 414–415
empty string, 33
encoding, 525
end arguments, 511
end-of-file (EOF), 26
end-of-line character, 35
environment (variables)
 of command-line interface, 4–6
 discussed, 4
 on Unix-like systems, 4–5
 on Windows systems, 5–6
environment functions, 326
EOF (end-of-file), 26
ephemeral, 512
equality, of tables, 144–145
error(s)
 ambiguous syntax error, 215
 anticipating conditions for, 222

assert function for, 220
 assumptions of, 219–221
 in Boolean values, 38
 in call stack, 216–217
 code, 220
 compile-time error, 215
 containment of, functions for, 227–230
 data, 220
 default behavior of, 218–219
 defining conditions for, 221–222
 discussed, 213
 functions for, 220–221
 with global variables, 111
 handling, 218
 kinds of, 213–218
 locating, 230
 program termination and, 220
 return values and, 222–224
 runtime, 217–218
 stack overflow, 98–99
 stack tracebacks, 220
 structuring code and, 224–227
 syntax, 213–216
 unexpected symbol, 215
 unfinished string, 215
 in user-written scripts, 230
 with `vararg`, 215
error file, standard, 182
error function, 220–221, 334
error handler, 422
error-condition function, 333–334
error-containment function, 330–331
escape sequence, 36
etc subdirectory, 8
evaluated expression, 56
event handling, 287–297
event loop, 288–295
explicit nil value, 90
exponent function, 354–355
exponentiation, 24, 49
expression, 53–54
extended web server, 469
extending Lua, 414–415
extension function, 415
extension library
 of C programming language, 441–447
 layering, 441–447
external local variable. See upvalue
extracting tarballs, 6–7

F

Fact, 98–99, 102
factorial, 59
false, 43, 266
de Figueiredo, Luiz Henrique, 596
file handles, 366–367
File Stream library, 570
file transfer protocol (FTP), 510
filenaming, 241
files
 input/output for, 181–184
 mode setting for, 430–436
 naming of, 241
 writing and reading from, 181–184
filesystem function, 369

filter, 524–525

filtering, of data flow, 524–527

floating point representation, 360

floating-point rounding, 28

flow of control, 74–76

flush (filehandle), 366

for loop

- as block, 86
- discussed, 60–62
- local variables in, 85–86
- loop variable of, 111
- variable scope of, 84

formal argument

- discussed, 84
- function definitions with, 103

format code, 386

format placeholders, 175–179

formatting

- for credit card numbers, 186
- numbers, 174–179
- pattern matching for, 186–187
- strings, 174–179

formatting placeholder, 175–178

forms

- CGI for, 497
- JavaScript in, 498
- validation of, 498

forums, for Lua, 601

fractional integers, function for, 362

FTP (file transfer protocol), 510

“full” mode, 367

func field, 312

function(s)

- as assignments, 103–105
- built-in, 102–103
- in call stack, 96
- call stack for, 95–97
- calling, in C, 380–383, 421–423
- for calling functions, 95–102
- chunks as, 81–83
- closures, 108
- code inside, 71
- comparing, 103
- coroutines vs., 272
- for creating functions, 108–109
- in debug library, 371–372
- defining, 108–111
- discussed, 69–72
- for functions, 95–102, 108–109
- limits on, 105
- local, 105–106
- local variables in, 85–86
- for modules, 245–247
- multiple, 104
- obtaining, in C, 421
- printing, 103
- with private state, 110–111
- recursive, 97–98
- replacing, 102–103
- replacing, of built-in, 102–103
- for return values, 73–74
- for returning values, 72–80
- semicolons in, 107–108
- with side effects, 91–93
- side effects of, 91–95
- stack frame of, 109

stack overflow and, 98–99

tables and, 128–136, 147–148

tail calls, 99–102

upvalues, 108

as values, 102–106

variable scope and, 84–91, 111–113

whitespace and, 106–107

function argument error, 215

function call, 108

function expression, 103–104

function identifier, 30

function plot, 402–405

function type, 421

G

game programming, with SDL

for 2-D action game, 544

discussed, 543

installing programs for, 544–546

LuaCheia for, 544–546

reasons for, 543–544

sprites in, 551–561

writing programs for, 546–551

gaps, arrays with, 123

garbage collection

discussed, 303

mechanics of, 304–307

metamethods for, 267

usage of, 304

garbage-collection function, 332

GarbageTest, 304, 307

gd graphics library

building, 395

building, on Unix-like systems, 396

for charts, 481

defined, 395

discussed, 395–405

installing, on Windows systems, 396–397

lua-gd, 397–405

Get, 136

GET command (forms), 496–497, 527

GetEvents function, 551

getenv function, 326

getmetatable function, 326

global environment, 167

global variable(s)

assignments of, 244–245

avoiding, in namespaces, 244–245

defined, 89

errors with, 111

local variable definition and, 111

metatables for, 246–247

in namespaces, 244–245

shadowing, 89–90

in tables, 163–168

variable scope of, 91

Gmane, 599

GNU Readline History libraries, 27

Goldberg, David, 28

graphic generation, 582–583

graphical user interface (GUI)

modal programs vs., 287

shell access with, 3

graphics library. See gd graphics library

greedy characters, 197–198
gsub, 186. *See also* `string.gsub`
GUI (graphical user interface)
 modal programs vs., 287
 shell access with, 3
gui namespace, 589
gui.destroy, 589
gui.dialog, 589
gui.getstate, 589
gui.gettext, 589
gui.setstate, 589
gui.settext, 589
.gz extension, 6–7

H

handheld device, Lua on, 565
handle, 180
hash part, 307
hashing, 307
headers key, 528
helper routines (CGI), 489–498
hexadecimal color codes, 486
hexadecimal number, 49
highlighting (text editor), 21
history feature, 27
HISTORY file (tarball), 7
hook, 315–321
hook count, 320
HostFS, 577
HTML, 174, 176, 204, 467–468
HTML forms, 489, 491–497
http.request function, 527–529
hyperbolic angle, functions for, 351–353
hyperbolic functions, 351–353

I

identifier, 29–30
Ierusalimschy, Roberto, 596
if statement
 as block, 86
 discussed, 54–58
image tag (HTML), 481–482, 488
IMAP (Internet Message Access Protocol), 529
implementation, 233–234
implicit nil value, 90
Inc, 136
include directory, 14
indexed values
 in C programming language, 436–438
 retrieving, in C, 436–437
 setting, in C, 437–438
indexing metamethod, 258–265
inetd (sinetd, launchd), 536
inheritance, 264–265
initializing variables, 40
inner scope, variables from, 87–88
input file, standard, 182
input/output (I/O)
 discussed, 180
 functions for, 364–367
 mechanics of, 184–185
 for writing and reading from files, 181–184
input/output (I/O) library, 364–367

INSTALL file (tarball), 7
installation
 location of, 12
 of Lua, 1–3
 of LuaCheia, 544–546
 of LuaSocket, 503–506
 of `pack` library, 385
 of Palm OS emulator (POSE), 578
 of SDL, 544–546
instance (universe), 379
instantiated coroutine, 272
instantiated object, 134
integer (whole number), 26
integer exponent, function for, 360
integers, functions for, 358–359
interactive CGI application, *See* **CGI application**, *interactive interface*
 discussed, 233–234
 preservation of, 236–240
interfacing Lua, 413
interned string, 307
Internet Message Access Protocol (IMAP), 529
Internet Protocol (IP), 506
Internet Relay Chat (IRC), 601
interpreter
 defined, 1, 81
 interaction for arithmetical operations, 24–25
 quitting, 26
 shortcuts for, 26–27
interrupt signal, 26
intializing loops, 59
inverse cosine function, 349
inverse sine function, 348
inverse tangent function, 350
inverse trigonometric function, 348–350
I/O. *See* **input/output**
io.close function, 364
io.flush function, 364
io.input function, 364
io.lines function, 364
io.open function, 223, 226, 364
io.output function, 365
io.popen function, 366, 539, 540
io.read function, 365
io.stderr, 182
io.stdin, 182
io.stdout, 182
io.tmpfile function, 365
io.type function, 365
io.write function, 365
IP (Internet Protocol), 506
ipairs function, 161, 260, 334
IRC (Internet Relay Chat), 601
Iter function, 158
iterating
 of coroutines, 286–287
 of pattern-matching, 204–207
iterator, 158–159
iterator factory, 158–159
Ittner, Alexandre Erwin, 397

J

JavaScript, for form validation, 498
junk value, 30

K

Kepler, James, 498–499

Kepler project

- for CGI, 499
- defined, 1
- discussed, 498–499
- for Lua pages, 500
- mailing lists for, 603

keys, for tables, 119–120

KeyToNums, 264

key-value pair, 118

keyword argument, 143–144

keywords (reserved words), 29

Klein bottle, 275

L

lastlinedefined field, 312

launchd (inetd, inetd), 536

layering, of extension library, 441–447

left-associative operator, 49

length operator, 46–47

less command, 7

less filter (Linux), 239

lib directory, 14

libcurl

- building, 389–390
- building, on Unix-like systems, 390
- building, on Windows systems, 391–392
- discussed, 389

libkit subdirectory, 567

libraries

- building, from source code, 377–379
- core library, 325–335
- coroutine library, 336–337
- cURL file transfer library, 389–395

libraries

- debug library, 308–321
- debugging library, 370–373
- defined, 375
- discussed, 325, 375–376
- DLL (dynamic-linked libraries), 376–377
- gd graphics library, 395–405
- input/output (I/O), 364–367
- input/output (I/O) library, 364–367
- interaction of Lua with, 379–383
- math library, 345–364
- operating system library, 368–370
- pack library, 383–389
- package library, 338–340
- shared, 376
- from source code, 377–379
- SQLite database library, 405–411
- string library, 340–343
- table library, 344

Life subdirectory, 567

“line” mode, 367

linedefined field, 313

lines (filehandle), 366

linker, 7

lint-type program checker, 303

Linux systems

- compiling Lua on, 8–12
- compiling LuaSocket on, 503
- less filter in, 239

list(s). See array(s)

literal strings, 266

literal value, 54

load function, 328

loader function, 415

loadfile function, 328

loadlib function, 338

loadstring function

- description, 328
- function return by, 168
- for multiple return values, 83
- use of, 83

local function

- discussed, 105–106
- variable scope of, 89

local variable(s)

- discussed, 85–90
- in functions, 85–86
- global variable and, 111
- for global variable shadowing, 89–90
- initializing multiple, 90
- in for loop, 85–86
- numbering of, 310
- in stack, 109
- storage of, 109
- variable scope of, 85–90

locale, 39

logarithm function, 356–357

Logo programming language, 582

long string, 34–35

loop variable, 111

loop(s) and looping

- with custom-made loops, 158–163
- defined, 58
- initializing, 59
- for loop, 60–62
- repeat loop, 62–63
- through tables, 124–128, 158–163
- while loop, 58–59

low watermark, 288

Lua

- building, 2–3
- communicating between C and, 415–421
- compiling, 2–3, 7–18
- installation of, 1–3
- interfacing, with other languages, 413
- prebuilt, 3
- using, 543–545
- versions of, 2
- web site for, 596

Lua API stack, 96

Lua C interface, 1, 379

lua interpreter (lua.exe), 2, 24

- chunks in, 82–83
- defined, 81
- make command affecting, 9
- testing, 10

Lua mailing list

- accessing, with newsreaders, 599
- accessing, with web browsers, 599
- discussed, 597
- downloading archives of, 598–599
- newsreaders for accessing, 599
- posting messages to, 600

subscribing to, 599
 viewing and searching archives of, 597–598
 web browsers for accessing, 599

Lua pages, Kepler for, 500

Lua tarball, 7–8

Lua technical notes, 525

luac

bytecode with, 299–303
 make command affecting, 9
 mechanics of, 300–303

LuaCalc, 566

LuaCheia, 544–546

lua_close, 414

lua.out, 302

luacurl

building, on Unix-like systems, 392
 building, on Windows systems, 393
 discussed, 392
 mechanics of, 394–395
 using, 393–394

lua.exe. See lua interpreter; lua interpreter

LuaForge, 602–603

lua-gd

building, on Unix-like systems, 397–398
 building, on Windows systems, 398–399
 discussed, 397–405
 mechanics of, 401–402, 404–405
 using, 399–404

LuaSocket

addresses in, 507
 application protocols for, 524–536
 compiling, 503–505
 for connection handling, 518–524
 domain names in, 507–508
 for e-mail, 529–536
 for filtering flow of data, 524–527
 installing, 503–506
 Internet resources for, 508–509
 for network communication, 512–518
 networks and, 506
 routed packets in, 506–507
 select function in, 518–522
 sockets in, 510–512
 transport protocols, 508–509
 for web page access, 527–529
 Windows binary package for, 505–506

LuaSql, 458

lua-sqlite3

building, 407
 building, on Unix-like systems, 407–408
 building, on Windows systems, 408–409
 discussed, 407–411
 using, 409–411

lynx character-mode web browser, 7–8

M

mad.rad function, 361

magic character

for pattern matching, 198
 for pattern-matching, 188–189, 209
 punctuation characters, 189
 punctuation characters as, 189

mailing list

for Kepler project, 603
 for Lua. See Lua mailing list

main thread, of coroutine, 272

make utility, 9–10

makefile utility, 9

Man, Kei-Hong, 301

mantissa, function for, 360

manual, reference, 596

master object, 511

matching. See pattern matching

math library

adjustment functions in, 358–360
 angle conversion functions in, 361
 constants in, 363–364
 discussed, 345
 exponent functions in, 354–355
 floating point representation in, 360
 hyperbolic functions in, 351–353
 inverse trigonometric functions in, 348–350
 logarithm functions in, 356–357
 maximum functions in, 363
 minimum functions in, 363
 modulus functions in, 362–363
 pseudo-random number functions in, 362
 trigonometric functions in, 345–347

math.abs function, 358

math.acos function, 349

math.asin function, 348

math.atan function, 350

math.atan2 function, 350

math.ceil function, 359

math.cos function, 346

math.cosh function, 352

math.deg function, 361

math.exp function, 354

math.floor function, 360

math.fmod function, 362

math.frexp function, 360

math.huge function, 363–364

math.ldexp function, 360

MathLib, 566

mathlib.prc, 567

math.log function, 356

math.log10 function, 357

math.max function, 363

math.min function, 363

math.modf function, 362

math.pi function, 364

math.pow function, 355

math.random function, 362

math.randomseed function, 362

math.sin function, 345

math.sinh function, 351

math.sqrt function, 355

math.tan function, 347

math.tanh function, 353

maximum function, 363

memo record, 569

metamethod

applicability of, 268
 concatenation and, 249–258
 defined, 251
 discussed, 249
 for garbage collection, 267
 indexing, 258–265
 non-syntactical, 267
 nontables with, 265–267
 relational, 257–258
 syntactical, 265

metatable(s)

defined, 251
for global variables, 246–247
mechanics of, 251–253
for userdata, 423

metatable function, 326–328

method key, 528

Microsoft Visual C++. See also **C programming language**

building Lua with, 12–14
modules of, 247

Microsoft Visual C++ 6.0 SDK, 12

mime module, 526

MIME notation, 600

MinGW package

building Lua with, 16–18
for Windows-Unix compatibility, 2

minimum function, 363

mirror sites, for resources, 598

modal programs, 287

modal windows, 288

modularization technique, 244

module(s)

bookkeeping for, 240–241
bytecodes and, 241–242
with C programming, 247
defined, 233
directory for, 235–236
discussed, 233
function for, 245–247
implementations, 233–234
interface preservation, 236–240
interfaces, 233–234
namespaces and, 242–245
placement of, 235–236
require function and, 234–235
using, 234–235

module directory, 235–236

module function

in core library, 331
discussed, 245, 331
in math library, 362–363
mechanics of, 245–247

modulo operator, 47–48

Moen, Rick, 597

Monotone (revision control system), 22

MSVCRT.DLL, 12

MSYS, 16

multiple assignments, 31

multiple connections (LuaSocket)

on sever side, 522–523
timeout values for, 523–524

multiple-character string, 39

multiple-valued function, 78–79

multiplication, 24

multitasking

with coroutines, 273
coroutines for, 273–278
preemptive, 276

mutable tables, 144, 305

mutating, 144

MySQL, 458–465

namespace

creating, 242–243
discussed, 242–245
global variables in, 244–245
modules and, 242–245
reusing, 242–243

namewhat field, 313

natural exponent, function for, 354

nc (netcat) program, 539

Nehab, Diego, 503, 524–525

netcat (nc) program, 539

netiquette, 600

network communication (LuaSocket), 506, 512–518

Network News Transfer Protocol (NNTP), 524

networking

on client side, 538–541
on server side, 536–538

newsgroup, 599

newsreader, 599

next function, 334

nil value(s)

addition of, 97
discussed, 40–41
implicit/explicit, 90
from I/O function, 185
from and operator, 43
replacement function returning, 204

NNTP (Network News Transfer Protocol), 524

“no” mode, 367

nonalphanumeric character, 189

nongreedy character, 197–198

nonpunctuation character, 189

non-syntactical metamethod, 267

nontable, with metamethods, 265–267

“normal” status, 278

not unary operator (unary minus), 44–45

notation

for arithmetical operations, 25–26
RPN, 566
scientific, 25

notation, decimal-point, 28

null byte, 37

Number pattern, 511

numbers

comparing, with Boolean values, 37–38
comparing, with relational operators, 37–38
formatting, 174–179
rational, 255–256

Numer field, 255

NumToKeys, 264

nups field, 313

O

object-oriented programming, 133–136

octets, 507

OneMod function, 157

one-operand (unary) operator, 44

online documentation, 570–571

opaque handle, 282

operands

automatic conversion of, 48–49
defined, 42

operating system library

CPU timing in, 368
discussed, 368

N

Nakonechnyj, Alexandre, 399

name field, 313

filesystem functions in, 369
 other functions in, 370
 time and date functions in, 368–369

optimizing rings, 156–157

or operator

discussed, 43–44
 side effects and, 93

ordered table, 261–265

os.date function, 368

os.difftime function, 368

os.execute function, 370

os.exit function, 26, 370

os.getenv function, 370

os.remove function, 369

os.rename function, 369

os.setlocale function, 370

os.time function, 368–369

os.tmpname function, 369

outer scope, 87–88

output file, standard, 182

overflow, 27–28. See also stack overflow

overlapping, of pattern matching, 194

P

pack library

building, in Unix-type systems, 383–384
 building, in Windows systems, 384
 discussed, 383–389
 installing, 385
 mechanics of, 389
 testing, 384–385
 using, 385–389

package library functions, 331

package.cpath function, 339

package.loaded function, 339–340

package.loadlib function, 338

package.path function, 338–340

package.preload function, 338–340

package.sea11 function, 338

packaging, in columns, 441–442

pairs function, 260, 334

Pall, Mike, 280

Palm API, 570

Palm HotSync Manager, 567

Palm OS emulator (POSE)

compiling programs in, 580
 configuring, 578
 discussed, 577
 exiting, 580
 installing, 578
 obtaining, 577
 for Plua application, 577–580
 programming Plua in, 579–580
 running Plua in, 578–579

Palm OS simulator

discussed, 581
 obtaining, 581
 for Plua application, 581
 using, 581

Palm OS stream, 571–572

parameter (term), 84. See also Formal argument

parentheses

as delimiter, 202
 for literal strings, 266

within pattern, 199
 use of, 80

parser, 286

pattern(s)

anchor of, 188
 receive, 510–511

pattern (substring), 185

pattern item, 189

pattern matching

for any of several characters, 186–193
 of balanced delimiters, 202
 captures and, 198–201
 discussed, 185
 for formatting, 186–187
 iterating, 204–207
 magic characters for, 188–189, 198, 209
 overlapping of, 194
 searching and, 186
 for specific strings, 186
 with `string.find`, 203
 with `string.gsub`, 203–204
 with `string.match`, 203
 tricks for, 207–208
 for varying lengths, 193–198

pattern-based string function, 340–342

pcall function, 227–229, 330, 422

PDF (portable document format), 383

peers, 512

percent-escaped character, 189

pi, function for calculating, 364

plink, 539

Plot function, 405

Plua application

command-line compiler for, 576–577
 compiling applications in, 572–573
 compiling libraries in, 573–576
 contents of, 566–567
 database access with, 590–592
 discussed, 565
 features of, 567
 graphic generation in, 582–583
 on main computer, 576
 obtaining, 566
 online documentation for, 570–571
 Palm OS emulator for, 577–580
 Palm OS simulator for, 581
 in Palm OS streams, 571–572
 programming with, 581–592
 running, 567–568
 saving, 569–570
 user interface programming with, 583–590

Plua2, 565, 566

plua2help.prc, 566, 567

plua2.prc, 567

plua2rt.prc, 566

Pluto (persistence library), 149

PNG format, 484

pop, 382

POP (Post Office Protocol), 529, 532–536

PORT variable, 527

portable document format (PDF), 383

POSE. See Palm OS emulator

position capture, 200

POST method (forms), 497, 527

Post Office Protocol (POP), 529

posting messages

to Lua mailing list, 600

prebuilt binary packages. See **binary packages, prebuilt**
prebuilt Lua, 3
precedence, 49–51
preemptive multitasking, 276
print function
discussed, 103
for input/output, 180
quoting and, 32
private state, functions with, 110–111
privileges
in CGI scripts, 475–476
root, 11
program control, coroutines for, 271–272
program termination error, 220
program writing, 571–572
Programmer error, 220
programming, 581–592
Programming in Lua (Ierusalimsky), 596
progamp.lua2.prc, 566
programs, coroutines vs., 272–273
protected calls (C), 422–423
proxy key, 528
proxy table, 261
PROXY variable, 527
pseudo-index, 438
pseudo-random number function, 362
punctuation characters, 189
push, 382

Q

questions, phrasing, 597
Quoted-Printable encoding, 525
quoting strings
backslash escaping with, 35–37
discussed, 32
with double quotes, 32–33
with single quotes, 33
with square brackets, 33–35

R

radians, functions for calculating, 361
random numbers, function for, 362
rational numbers
defined, 255
defining, 253–255
mechanics of, 255–256
rawequal function, 326
rawget function, 326
rawset function, 326, 328
Raymond, Eric Steven, 597
read (filehandle), 366
read mode, 182
README file (tarball), 7
RealTbls, 264
receive array, 523
receive patterns, 510–511
recurse.lua, 215–216
recursion
coroutines and, 281
discussed, 97–98
recursive calls, 99
recursive descent, 286
recursive function, 97–98

redirect key, 528
redirection operators, 185
reference manual (Lua), 596
referencing value (C), 440–441
register (temporary storage), 301
registering variables, 379–380
registry, retaining values with (C), 438
regular assignment, 91
relational database, 449–458
relational metamethod, 257–258
relational operator
comparing numbers with, 37–38
comparing strings with, 38–40
discussed, 37
release directory, 581
repeat loop
as block, 86
discussed, 62–63
replaying, of captures, 201
Request for Comments (RFC), 524
require function
defined, 331
loadlib function vs., 338
modules and, 234–235, 241, 247
reserved words (keywords), 29
resolver library, 508
resources. See **community resources**
“restart” action, 332
retaining values, in C, 438–441
return statement
to alter control flow, 74–76
defining functions for, 73–74
errors and, 222–224
for flow of control, 74–76
functions for, 72–74
for multiple values, 77
with no value, 76–77
tail call as, 101
using functions for, 72–73
value lists for, 78–80
ReturnArgs function, 78–79
reverse argument, 144
reverse polish notation (RPN), 566
ReverseIpairs, 159
revision control system, 22
RFC (Request for Comments), 524
right-associative operator, 49
ring (data structure), 153–157
root privileges, 11
Roth, Michael, 407
rounding, 28
routed packet, 506–507
rows, 441–447
RPN (reverse polish notation), 566
run time, 469
“running” status, 278
runtime, 217–218
runtime error, 217–218

S

SampleWindow function, 296
sandboxing, 165
saving, 569–570
scientific notation, 25
scope, variable. See **variable scope**

- screen.heading**, 582
- screen.moveto**, 589
- screen.pos**, 589
- screen.turn**, 582
- screen.walk**, 582
- scripts**
 - CGI. See CGI scripting
 - for form validation, 498
 - JavaScript, 498
 - user-written, 230
 - as `vararg` functions, 140–143
- SDK. See software development kit**
- SDL. See Simple DirectMedia Layer library**
- search path (Windows)**, 5–6, 11
- searching**
 - pattern-matching and, 186
 - strings, 185–186
 - substrings, 185
- secure shell handling (SSH)**, 510
- secure sockets layer (SSL)**, 390
- security issues**
 - in Apache, 472
 - with asynchronous calls, 478
 - of CGI applications, 498
- seek (filehandle)**, 366
- select function**
 - described, 335
 - in `LuaSocket`, 518–522
- Selene Unicode Library**, 192
- semicolons, in functions**, 107–108
- send array**, 523
- server object**, 511
- server side**
 - multiple connections on, 522–523
 - networking on, 536–538
- Server Sockets Layer (SSL)**, 532
- set command**, 4
- setfenv function**, 164–167, 326
- SETGLOBAL instruction**, 244–245
- setmetatable function**, 326
- “setpause” action**, 332
- “setstepmul” action**, 332
- setvbuf (filehandle)**, 366
- shadowing, of global variables**, 89–90
- shallow copy**, 148–149
- shared library**, 376
- shebang (Unix)**, 470
- shell. See command-line interface**
- short-circuit evaluation**, 93–95
- short_src field**, 313
- show module**, 237–240
- side effects**
 - discussed, 91–95
 - of functions, 91–95
 - functions with, 91–93
 - ordering of, 91–93
 - short-circuit evaluation and, 93–95
- sieve.lua**, 281
- Simple DirectMedia Layer (SDL) library**
 - defined, 544
 - for game programming. See game programming, with SDL
 - installing, 544–546
- Simple Mail Transfer Protocol (SMTP)**, 529
- sine functions**, 345, 351
- sinetd (inetd, launchd)**, 536
- single quote (apostrophe character)**, 33
- single-character string**, 39
- sink**, 524–525
- sink key**, 528
- SMTP (Simple Mail Transfer Protocol)**, 529
- smtp module**, 529
- smtp.message function**, 529
- smtp.send function**, 529
- socket namespace**, 507
- socket.dns.gethostname function**, 507
- socket.dns.tohostname function**, 507
- socket.dns.toip function**, 507, 508
- sockets**
 - Berkeley sockets interface, 510
 - in `LuaSocket`, 510
 - TCP sockets, 511–512
 - types of, 510–511
 - in Windows system, 523
- socket.select**, 524
- socket.tcp**, 511
- software development kit (SDK)**
 - Microsoft Visual C++ 6.0 SDK, 12
 - need for, 1
 - for Windows, 12
- source**, 524–525
- source code**
 - building libraries from, 377–379
 - bytecodes and, 81, 302
- source field**, 313
- Spolsky, Joel**, 192
- sprite**
 - defined, 550
 - in game programming with SDL, 551–561
- SQL (structured query language)**, 458, 464–468
- SQLite 3. See SQLite database library**
- SQLite database library (SQLite 3)**
 - building, 405
 - building, on Unix-like systems, 405–406
 - building, on Windows systems, 406–407
 - discussed, 405–411
 - `lua-sqlite3`, 407–411
- square brackets**
 - as delimiter, 202
 - quoting strings with, 33–35
 - use of, 209
- square root, function for calculating**, 355
- squeeze function**, 193–198
- src subdirectory**, 8, 9
- SSH (secure shell handling)**, 510
- SSL (Server Sockets Layer)**, 532
- SSL (secure sockets layer)**, 390
- stack (call stack)**
 - for C function calls, 96
 - debug library for, 216
 - defined, 95
 - errors in, 216–217
 - for functions, 95–97
 - functions in, 96
- stack diagrams**, 416
- stack frame**, 109
- stack overflow**, 98–99
- stack overflow error**, 98–99
- stack traceback**
 - defined, 97
 - discussed, 220
- stack traceback function**, 218
- stack.look function**, 420
- standard error file**, 182
- standard input file**, 182

standard output file, 182

start arguments, 511

state, 282–287

stateful iterator, 159

stateless iterator, 159

statement(s)

abstractions and, 70

compound, 54–65

defined, 70

discussed, 53–54

“step” action, 332

step key, 528

“stop” action, 332

storage

in closures, 109

temporary, 31

stream programs, 570

stream (connection-oriented) socket, 510

stream-based server, 537–538

strict.lua module, 244

string(s)

of character codes, 173–174

of characters, 173–174

comparing, with Boolean values, 38–40

comparing, with relational operators, 38–40

conversion functions for, 171–173

conversion of, 173–174

defined, 32

discussed, 32, 171, 307

executed as chunk, 83

formatting, 174–179

implementation of, 307

interned, 307

length of, 173

multiple-character, 39

pattern-matching for, 185–209, 186

quoting, 32–35

searching, 185–186

string concatenation operator

discussed, 45–46, 49

metamethods and, 249–258

on tables, 249–257

string library

discussed, 340

pattern-based string functions in, 340–342

string-conversion functions in, 342–343

string.byte function, 173–174, 342

string.char function, 174, 342

string-conversion functions, 342–343

string.dump function, 185, 302, 342

string.find function

captures return from, 200

described, 340–341

pattern matching with, 196, 203

string.match function vs., 200

for zero-length match, 196

string.format, 174–179, 342–343

string.gmatch function, 204–206, 341

string.gsub function

bytecode and, 305

captures within, 201

defined, 186

described, 341

pattern matching with, 196–197, 203–204

of squeeze, 194

string.len function, 173, 343

string.lower function, 171–172, 343

string.match function

described, 341

features of, 199, 203

string.find vs., 200

string.pack function, 385, 386

string.rep function, 172, 343

string.reverse function, 172, 343

string.sub function, 172–173, 343

string.unpack function, 386

string.upper function, 172, 191, 343

structured query language. See SQL

structuring code, 224–227

subject (string), 185

substring, 185

subtraction, 24

sudo command, 10

super-server application, 536

surface, 551

“suspended” status, 278

SWIG, 416

syntactic sugar, 134

syntactical metamethods, 265

syntax, 134–135, 214, 303

syntax error, 213–216

system thread, 276

T

table(s)

altering contents of, 120–121

arithmetical operations in, 249–257

arrays and, 121–124

associative, 121

building data structures in, 152–158

case-insensitive, 259–261

changing content of, 144

content, changing, 144

contents of, altering, 120–121

copying, 148–152

defined, 117, 118

discussed, 117–119, 307

equality of, 144–145

functions and, 128–136, 147–148

global variables in, 163–168

implementation of, 307

keys for, 119–120

keyword arguments in, 143–144

looping through, 124–128, 158–163

mutable, 144, 305

object-oriented programming with, 133–136

ordered, 261–265

vararg functions in, 136–143

variables in, 145–147

table constructor, 122

table fields, 121

table library

discussed, 128

table.concat function, 131

table.maxn function, 132–133

table.remove function, 132

table.sort function, 128–131

table traversal functions, 334–335

table.concat function, 131, 344

table.insert function, 130, 132, 156, 344

table.maxn function, 132–133, 344

table.remove function, 132, 156, 344

table.sort function, 128–131, 224, 344

tail call

- defined, 100
- discussed, 99–102
- stack frame of, 109

tail calls, 99

tail recursive, 101

tail return, 320

tangent functions, 347, 353

.tar extension, 6–7

tar utility, 8

tarball

- contents of, 7
- discussed, 6
- extracting, 6–7

Tatham, Simon, 597

TCC. See Tiny C Compiler

TCP (Transmission Control Protocol), 509–510

TCP sockets, 511–512

technical notes, 525

termination, program, 220

Test function, 527

text editor, 21

.tgz extension, 6–7

this-var identifier, 30

thread, of coroutines, 272–273

time function, 368–369

time server

- creating, 474–475
- implementing, with AJAX, 476–478

timeout value, 523–524

TIMEOUT variable, 527

timing, 368

Tiny C Compiler (TCC)

- building Lua with, 14–16
- discussed, 3
- web site for, 14

TinyWeb, 472–473

token

- coroutine, 282–286
- defined, 204
- discussed, 204
- string.gmatch as, 204
- supported, 485

tonumber function, 333

tools

- revision control system, 22
- text editor, 21

ToString, 157

tostring, 174

tostring function, 333

total order, 258

Transform function, 527

Transmission Control Protocol (TCP), 509–510

transport protocol, 508–509

trigonometric function, 345–347

troubleshooting, bytecodes for, 302

“turtle” functions, 582

type and conversion function, 333

type function

- described, 333
- returning values from, 72–73

U

UDP (User Datagram Protocol), 509–510

unary minus (not unary) operator, 44–45

unary (one-operand) operator, 44

unexpected symbol error, 215

unfinished string error, 215

Unicode, 192

Unicode (UTF-8 format), 192

uninitializing variables, 40

universe (instance), 379

Unix/Unix-like system

- CGI scripts on, 470
- command-line interface on, 3
- compiling Lua on, 8–12
- compiling LuaSocket on, 503
- environment of, 4–5
- gd graphics library on, 396
- libcurl on, 390
- libraries on, building, 378
- luaacurl on, 392
- lua-gd on, 397–398
- lua-sqlite3 on, 407–408
- pack library in, 383–384
- prebuilt binary packages on, 19–20
- SQLite 3 on, 405–406
- Windows system and, 2

unpack function, 139, 335–336

until expression, 86

upvalue (external local variable)

- in C programming language, 439–440
- closures for, 440
- in functions, 108, 109

url key, 527

url.parse function, 508–509

User Datagram Protocol (UDP), 509–510

user interface programming, 583–590

USERAGENT variable, 527

userdata

- in C programming language, 423–436
- metatables for, 423

user-written script, errors in, 230

usr subdirectory, 14

UTF-8 format (Unicode), 192

V

validation, of credit card numbers, 187–193

value(s)

- in C, 436–438, 438–441
- defined, 118
- discussed, 51–52
- functions as, 102–106
- indexing, 436–438
- multiple, 77
- none, 76–77
- retaining, 438–441

value list

- adjusting, 78–80
- defined, 78
- discussed, 78
- multiple-valued functions in, 78–79
- for return values, 78–80

value-less function, 79, 80

vararg (...)

vararg (...)

- defining, 136–140
- discussed, 136
- errors with, 215
- for `loadstring`, 83
- as placeholder, 240
- scripts as, 140–143
- in tables, 136–143

vararg-related function, 335–336

variable(s)

- defined, 28
- discussed, 28, 51–52
- global, 91, 163–168, 244–245
- initializing, 40
- from inner scopes, 87–88
- local, 85–90
- in namespaces, 244–245
- from outer scopes, 87–88
- registering, 379–380
- on right side of assignments, 31
- in tables, 145–147
- uninitializing, 40
- variable scope, 85–91

variable scope

- arguments, 84
- in coroutines, 280
- determining, 111
- discussed, 84
- functions and, 84–91, 111–113
- global variables, 91
- of `local` function, 89
- local variables, 85–90
- of `for` loop, 84
- tricky situations involving, 111–113

velocity (velx), 559

VFS (virtual file system), 570

virtual file system (VFS), 570

virtual machine, 299

virtual stack, 415

Visual C. See C programming language

Visual C++. See Microsoft Visual C++

Visual C++ 6.0 SDK, 12

W

watermark, low, 288

waveform function, 405

weak key, 305

weak value, 306

Web applications

- CGI applications, interactive, 489–498
- CGI scripts, execution of, 469–471
- discussed, 467
- dynamic content for, 468–469, 474–489
- Kepler project, 498–500
- run time, 469

web browser

- for Lua mailing list access, 599
- `lynx` character-mode, 7–8

Web content, dynamic. See dynamic Web content

web page

- access to, with `LuaSocket`, 527–529
- charts in, 485

web server

- Apache, 471–472
- creating, 514–518

- discussed, 467–468
- embedded, 468
- extended, 469
- installing, 471–473
- testing, with static content, 474
- TinyWeb, 472–473

wget program, 8

what field, 313

while loop

- as block, 86
- discussed, 58–59

whitespace

- functions and, 106–107
- squeezing, 193–198

whole number (integer), 26

wiki pages, for Lua, 601–602

Win32 subdirectory, 20

Window.Close, 296

Windows binary package, 505–506

Windows system

- Apache on, 472
- CGI scripts on, 470–471
- character processing in, 430
- command-line interface on, 3
- compiling Lua on, 12–18
- compiling `LuaSocket` on, 503–504
- control code processing in, 365
- environment of, 5–6
- filenaming in, 241
- `gd` graphics library on, 396–397
- `libcurl` on, 391–392
- libraries on, building, 378–379
- `luacurl` on, 393
- `lua-gd` on, 398–399
- `lua-sqlite3` on, 408–409
- `pack` library in, 384
- prebuilt binary packages on, 20–21
- SDK for, 12
- search path mechanism of, 5–6
- settings recommended for, 6
- sockets in, 523
- SQLite 3 on, 406–407
- Unix-type systems and, 2

Window.Show, 296

Wirth, Niklaus, 214–215

workshops, for Lua, 603

World Wide Web, 468

wrapping

- of coroutines, 273
- of messages, 600

write (filehandle), 367

write mode, 182

X

X Window System, 288

XNextEvent function, 288

xpcall function, 229, 330

Z

zero, division by, 27–28, 48

zero-length match, 196

Zeus programmer's editor, 415