



Ruby Execution

As promised in Chapter 2, this appendix takes time to introduce the general execution environment of a Ruby script. It covers the various command line options available with the Ruby interpreter as well as the many pertinent variables and constants that make very short scripts possible.

Command Line Options

Perl users should feel right at home with many of the flags that can be passed to the Ruby interpreter, as the flags were modeled very much on those used by Perl. It is my opinion that Perl's biggest strength lies in its one-liners that, while often baffling to the newcomer, provide immense power to construct unbelievably dense code. This facility stems from two things: command line modes that do very common things for you and shorthand variables for accessing/setting pertinent data and behavior.

The fact that Ruby respects the success of Perl's approach in this area often leads to the charge that Ruby is too Perl-like. This is roughly equivalent to saying a human is too much like a turkey because they've both recognized the advantages of bipedal locomotion.

With that said, I can now present a list of command line options that is complete save for four: `--copyright`, `--version`, `--help (-h)`, and `--yydebug (-y)`. I'm going to assume that the first three hold no mystery. The fourth basically turns on a load of debugging for the interpreter itself and is of no use unless you plan to debug Ruby.

Octal Record Separator Specification: `-0[octal]`

Consists of the digit `0` optionally followed by an octal value for the character to use as the default global record separator `$/`, which is used by methods like `IO.each`. Without an explicit value, `NULL (000)` is assumed. Passed a value of zero (`-00`), Ruby goes into paragraph mode, where each record is considered to be separated by two default record separators (e.g., `"\n\n"`). Since `511` is not a valid character code, passing `-0777` will guarantee that the entire input file is read in as one record—a process somewhat indelicately referred to as *slurping*.

Auto Split Mode: `-a`

When used in conjunction with `-n` or `-p`, runs `$F = $_.split` at the beginning of each loop.

Change Directory: **-C path**

Causes the present working directory to be set to the specified path before the script is run.

Check Syntax Only: **-c**

Does exactly what you think it does (without running the script). Remember that the more sophisticated your scripts become, the less useful this flag will be, as it only invokes a simple syntax check. If all is well, you will receive the message `Syntax OK`. Otherwise, you will see one or more reports indicating a problem and the line it occurred on like this:

```
upload.rb:19: odd number list for Hash
upload.rb:28: syntax error
```

Debug Mode: **-d, --debug**

Sets `$DEBUG` to `true`, which is generally useful if you want a simple way to enable some extra tracing or differing behaviors during development. This flag also implicitly sets `$VERBOSE = true`.

Execute Line: **-e line**

Takes a line of Ruby and executes it (as you've already seen). This can be specified multiple times, in which case each instance is treated as a new separate line for bookkeeping purposes (try `ruby -e 'p __LINE__' -e 'p __LINE__'`).

Field Separator Specification: **-F pattern**

Sets the global field separator `$;`, which is used by methods like `String.split`. Note that this can be a simple string delimiter or a full-blown regular expression.

Include from Additional Directory: **-I path(s)**

Adds to the array of search paths for the inclusion of code via the `load` or `require` statement. This array is accessible in code as `$:` (`$LOAD_PATH`). Note that many paths can be specified per `-I` and that many `-Is` can be used.

In-place Editing: **-i[extension]**

For each file path specified as an argument to the interpreter (in `ARGV`), anything written to standard out while dealing with a particular file becomes the contents of that particular file. The original file is preserved with a new extension if and only if this is specified. Obviously, this flag is pretty meaningless without the use of `-n` or `-p`. Suppose I wanted to change all instances of my name in a set of header files to my secret squirrel name. I would write something like `ruby -p -i.bak -e 'gsub(/Andre/, "Nutkin")' *.h`.

KCode Encoding Choice: **-K encoding**

Specifies the encoding of the script to support character sets like SJIS as used by our Japanese colleagues. This is equivalent to setting the `$KCODE` variable. An encoding of `u` specifies UTF-8.

Line Ending Automated Handling: -l

Sets the output record separator (`$\`) to the same value as the input record separator (`$/`) and automatically performs a `chop!` on each input record (`$_`). In simpler terms, this flag causes each record to be presented to you without its line ending or other record separator, and then adds it back once you've finished.

Nested Retrieval Loop: -n

Places an implicit `while gets ... end` block around your code. Note that this allows you to read each line of not only a single file, but multiple files. It is well worth having a close look at `ri Kernel.gets` for the rather cute behavior conventions it provides. In particular, it sets `$_` to the value of the retrieved record (e.g., the next line).

Printed Nested Retrieval Loop: -p

Does the same as `-n`, but with an extra `print $_` at the end of each loop.

Require Library: -r library

Identical to having placed `require library` at the beginning of your code. This is particularly useful in `-n/-p` situations where you don't want to require the library through every loop (although this wouldn't be disastrous, as rerequiring a library involves only a single hash lookup).

Script Search: -S

Instead of assuming a given script is in the current directory, causes the interpreter to search in the location specified in the `PATH` environment variable. This is mostly to allow platforms that don't support the shebang-style script header (silly as they are) to emulate it.

Smart Variables: -s

Magically removes every command flag after the script name but before any file arguments from `ARGV` and turns them into global variables. So `ruby -s test.rb -col=blue file.c` would run the `test.rb` script with a single argument (`file.c`) and give the script the global variable `$col = "blue"`.

Taint Level: -T[level]

Turns on taint checking at the given `level` (or 1 if none is specified) and sets `$SAFE` to this value.

Verbose Mode: -v, --verbose

Activates all kinds of extra messages and warnings, sets `$VERBOSE = true`, and causes the version string to be printed before anything else is done (thus making a simple `ruby -v` do what you would expect).

Warnings Mode: -w

Does the same as -v, except that no version string is printed and the interpreter will expect input on standard in if no other code/scripts are specified.

Ignore Garbage: -x[path]

The interpreter will ignore everything it encounters until it sees a shebang line containing the string `ruby`, which it will interpret as the beginning of the script. It will then execute the script until it encounters an end-of-file or the keyword `__END__`. If a path is specified, the present working directory is changed to this path before execution commences.

Environment

Ruby provides access to its runtime environment through a set of variables that, again, should be familiar to any Perl user. First, there are the set of arguments to the script whereby the invocation

```
$ ruby count.rb 1 2 3
```

would give us

```
ARGV = ["1", "2", "3"]
```

inside `count.rb`. `ARGV` is just an `Array` and so supports all the methods you have/will come to love. Imagine we wanted our script to be able to take some command flags, one of which was a `-r` to remove something or other. One quick and dirty way to determine whether this had been passed is

```
remove_stuff = ARGV.include?("-r")
```

In practice, beyond the simplest command line options, a proper argument-parsing library tends to be a better idea.

Next on the list of most-used variables would have to be the environment itself—that dictionary of values you can see from the console by typing `env` (or something similar). Here's what part of mine looks like:

```
TERM_PROGRAM=Apple_Terminal
TERM=xterm-color
SHELL=/bin/bash
TERM_PROGRAM_VERSION=133
USER=andre
```

Perhaps unsurprisingly, this is accessed in Ruby using the `ENV` object (which acts like a `Hash` but isn't one). Values can be retrieved and set as you would expect:

```
terminal_type = ENV["TERM"]
ENV["USER"] = "Nutkin"
```

Beyond these two oft-used collections, a few global variables and a couple more constants are of interest.

Process Information

The following entities all relate to process-specific information such as the name of the command used to run the script and global variables for each of the command flags:

- `$0`: The command that originally ran the Ruby script (what C users would think of as the first argument of `ARGV`). On many platforms, this is writable so that how the process appears in `top/ps` can be changed by assigning something new to `$0`.
- `$$`: The process ID (PID) of the currently executing program.
- `$:`: The array of directories to check when including code via the `require` or `load` command. This array provides a programmatic alternative to using the `-I` flag on the command line to include new search paths. Append your chosen directory/directories as you would with any other array: `$: << "/tmp/foo"` or `$: += ["/tmp/foo", "/tmp/bar"]`.
- `$-<something>`: Almost all of the command flags have a corresponding variable of this form—for example, `$-a`, which corresponds to the value passed by the `-a` command flag, or `$-d`, which is true in debug mode.
- `__FILE__`: The name of the currently executing file.
- `__LINE__`: The number of the line that is currently executing.

File Handles

These variables specify the input and output streams used by your program:

- `STDIN`, `STDOUT`, `STDERR`: The actual standard file handles (of class `IO`) for the program and the initial values of `$stdin`, `$stdout`, and `$stderr`
- `$stdin`, `$stdout`, `$stderr`: The current `IO` objects for the standard file handles used by methods like `Kernel.puts`

Magic Processing Variables

This subsection lists global variables controlling some of the more “magical” field, file, and record processing behavior we unpacked in Chapter 7:

- `$/`: The input record separator used by methods like `Kernel.gets`.
- `$\`: The output record separator used by methods like `Kernel.puts`.
- `$;`: The input field separator (a simple string delimiter or a regular expression).
- `$.`: The output field separator used by methods like `Array.join` (defaults to `nil`).
- `$<`, `ARGF`: A special object that provides access to all of the lines of all of the files specified in `ARGV` (or the data from `$stdin` when no such files are specified). For example, `$<.file` returns an `IO` object for the current file being read, and `$FILENAME` is a synonym for `$<.filename`.

- `$_`: The last line read by `Kernel.gets` or `Kernel.readline`. Indeed, a number of the string manipulation functions in `Kernel` (such as `gsub`) operate on `$_`.
- `$F`: The array of fields as split under the `-a` command flag using `$;` to determine the field separator.

Other Sundries

Finally, here are some other variables (and a constant) that you might find useful in day-to-day scripting:

- `DATA`: The contents of the main program file past an `__END__` directive (if such exists)
- `$1` to `$9`: The up to nine matches from the last executed regular expression (local to the current scope)
- `$!`: The last exception object raised in the current context
- `$@`: The last exception object's backtrace

Improving Readability

Looking back over the preceding subsections, I see that I used to use a lot of these tersely named global variables (or their equivalents) when I was a Perl monkey. I'm pretty sure I've never employed most of these as a Ruby-holic, mostly because I found myself in a job where one-liners were never one-offs. Even trivial bits of code ended up getting reused, and the ease with which one can write things "properly" the first time in Ruby meant that beasts like `$<` never really had a chance.

If you absolutely insist on using these global variables in scripts, then you'd be well advised to check out the `English` library, which ships as part of the standard distribution. It maps many of the symbols to more intelligible variable names and can be bolted on to your script with a quick `require "English"`. Mappings include `$!` to `$ERROR_INFO`, `$;` to `$FIELD_SEPARATOR`, and `$/` to `$INPUT_RECORD_SEPARATOR`.

INDEX

■ Numbers and Symbols

- \$\$ command flag, 223
- \$@ variable, 224
- \$_ global variable, 224
- \$.split array, 101
- << (double chevron) construction, 62
- << method, 11
- \$0 command flag, 223
- 002_add_disk_usage_cache.rb file, 95
- \$1 variable, 224
- \$9 variable, 224

■ A

- a (auto split mode), 219
- a flag, 205
- abstracted locking, 61, 63
- abstracting safe file operations, 64, 65, 66
- abstraction, 3
- accept method, 144
- accessors, 8, 9
- Ackermann number, 212
- ActiveLDAP, 117, 121
- ActiveLdap::Base class, 118
- ActiveRecord module, 87, 89, 97
- ActiveRecord object, 201
- ActiveRecord subclass, 52
- ActiveRecord::Base class, 118
- ActiveResource::Base directory, 130
- acts_as-style method, 54
- acts_as_list macro, 54
- add method, 171
- add_graph method, 175
- add_object method, 113, 114
- aggregate analysis, 167, 168
- algorithmic optimization, 32, 33
- alias keyword, 62
- allow_anonymous key, 119

- analyzing data, 163–168
 - aggregate analysis, 167–168
 - filtering and assigning events, 165–166
 - marshalling data, 163–164
 - overview, 163
 - parsing events, 164–165
- analyzing performance, 27–32
 - Benchmark library, 27–29
 - Profiler library, 30–32
 - UNIX time command, 27
- append_features method, 50
- ARGF global variable, 223
- args variable, 61
- ARGV code, 101
- Array class, 10, 77
- Array method, 11
- Array subclass, 112
- Array.inspect property, 74
- Array.pop method, 213
- Array.push array, 113
- arrays, 213
- Array.shift method, 213
- Array.uniq, 33
- as key, 172
- asn1_type method, 157
- assert method, 199
- assert_in_delta statement, 201
- assigning events, 165–166
- atomicity, 83
- attr keyword, 215
- attr_accessor function, 9, 48
- attr_reader function, 9
- attr_writer function, 9
- author key, 189
- auto split mode (-a), 219
- automatic documentation, 203–205

- autorequire key, 189
- autorequire setting, 190
- B**
- bar_graph method, 174
- base key, 118
- BasicSocket class, 134
- Bean Scripting Framework (BSF), 212
- begin-end method, 138
- begin-ensure-end block, 137
- begin-rescue-end block, 135
- begin.ensure.end block, 61
- belongs_to macros, 120
- Benchmark library, 27, 29
- Benchmark module, 28
- Benchmark.bmbm file, 80
- Bignum class, 24
- bin directory, 179, 185
- bind_dn key, 119
- bindir key, 189
- block argument locality, 214
- blocking locks, 59–60
- blocks, 9–12
- blogging/aggregation software, 130
- bmbm method, 29
- BooksController.award file, 132
- BooksController.unreleased file, 132
- Boolean value, 33
- break keyword, 10
- BSF (Bean Scripting Framework), 212
- build verb, 190
- Builder commands, 67
- Builder library, 66, 71
- BWHost class, 151
- C**
- C applications, integration with, 39, 41
- C/C++ libraries, 41
- c (check syntax only), 220
- .c file, 197
- C path (change directory), 220
- cache_object.set file, 86
- call variant, 123
- call2 variant, 123
- cannon-status class, 144
- Cat class, 128
- CatteryService.rb file, 128
- CGI.escape class, 142
- change directory (-C path), 220
- charts, 169–174
 - CSS charts in Rails, 173–174
 - overview, 169–173
 - Scruffy Charts, 169–173
- check syntax only (-c), 220
- chmod u+x hello.rb command, 2
- chomp operation, 215
- chomping, 145
- :class parameter, 120
- cleanup verb, 182
- clear_dirs method, 149
- client functionality
 - Representational State Transfer (REST), 130
 - Simple Object Access Protocol (SOAP), 126
 - XML Remote Procedure Call (XML-RPC), 122–124
- clobber_rdoc task, 209
- cn (common name), 118
- collecting data, Secure Shell (SSH), 161, 162
- com.apple.desktop.plist folder, 109
- \$: command flag, 223
- command line options, 219–222
- comments, 16, 205–207
- common name (cn), 118
- Community key, 157
- :Community parameter, 159
- config directory, 89
- connect method, 88
- connection to agent, SNMP, 156
- connection to service, 118, 119
- consistency, 83
- Contactable module, 50, 54
- contacts file, 18
- contents command, 183
- control of devices, 148–153
 - packet monitoring, 150–153
 - SSH, 148–150
- :controller => :books: directory, 131
- cool_app statement, 197
- copyright command line option, 219

- coroutine support, 11
- count variable, 103
- create statement, 93
- created_at column, 91
- creation, retrieval, update, and destruction (CRUD) objects, 121, 122
- cron command, 96
- cron method, 19
- CRUD (creation, retrieval, update, and destruction) objects, 121, 122
- CSS charts in Rails, 173–174
- current? statement, 45
- customization
 - Marshal class, 77
 - YAML, 79

D

- d flag, 205
- data directory, 163, 167
- data structuring, 38–39
- data type, 115
- DATA variable, 224
- database administrator (DBA), 82
- database management systems (DBMSs), 83
- databases, 87–89
 - DBI, 87–89
 - SQL queries, 87–89
- database.yml file, 89
- data_files.each block, 167
- date key, 189
- date_string line, 168
- db/migrate/001_initial_schema.rb file, 90
- DBA (database administrator), 82
- dbh.disconnect method, 88
- DBMSs (database management systems), 83
- dc (domain component), 118
- \$DEBUG flag, 220
- debug mode (-d, --debug), 220
- :default directive, 95
- \$/ default global record separator, 219
- :default task, 194
- default_executable setting, 189, 190
- default_type key, 171
- #define statement, 43

- delete method, 86
- DELETE request, 131
- delimited values, 100–104
 - comma-separated values (CSVs), 103–104
 - creating numbered files, 102
 - explicit versions, 102–103
 - output behavior, 102
 - overview, 100–101
 - retrieving fields, 101–102
 - retrieving records (lines), 101
 - tab-separated values (TSVs), 103–104
- dependencies, 181
- dependency command, 183
- desc directive, 198
- describe method, 13
- description key, 189
- destroy method, 122
- /dev/blah file, 97
- Dir class, 69
- directory listings, 18–19
- Dir.exist? file class, 213
- Disk table, 94
- diskmon directory, 89
- diskmon project folder, 96
- diskmon_dev file, 89
- disks.create construction, 93
- dn_attribute directive, 120, 121
- DNSRecord class, 138
- docs directory, 204
- document object, 106
- documentation, 203–209
 - accessing via gem_server, 185
 - automatic, 203–205
 - comments, 205–207
 - headings, 207
 - links, 207
 - lists, 207–208
 - overview, 203
 - processing commands, 208
 - from rake, 208–209
 - separators, 207
- documenting tasks, 198
- domain component (dc), 118
- dot command line tool, 174

double chevron (`<<`) construction, 62
 Download link, 178
 drive method, 5
 duck typing, 12
`_dump` method, 77
 durability, 83

E

e flag, 1, 16, 20
 -e line (execute line), 220
 each method, 49, 121
 each_line, 213
 each_varbind method, 157
 each_with_index method, 69
 email key, 189
 Embedded Ruby (ERb), 70
 embezzle method, 37
 embezzled_total variable, 37
 Employee.initialize variable, 38
 encapsulation, 4
 English library, 224
 ensure method, 137
 enterprise data, 99–132

- network services, 116–132
 - Lightweight Directory Access Protocol (LDAP), 116–122
 - Representational State Transfer (REST), 128–132
 - Simple Object Access Protocol (SOAP), 125–128
 - XML Remote Procedure Call (XML-RPC), 122–125
- parsing data, 99–116
 - delimited values, 100–104
 - overview, 99–100
 - XML, 104–116

Enumerable module, 48, 69, 93, 121, 215–216
 Enumerable.count method, 216
 Enumerable.find method, 216
 Enumerable.find_index method, 216
 Enumerable.inject method, 152
 Enumerable.max method, 216
 Enumerable.min method, 216
 ENV object, 222
 environment command, 183

ERb (Embedded Ruby), 70
 erb script, 71
 Errno::ECONNREFUSED class, 135
 errors proxy object, 121
 /etc/motd configuration, 70
 /etc/passwd directory, 65
 /etc/rotate-logs file, 20
 ethereal method, 150
 Event class, 167
 events

- assigning, 165–166
- filtering, 165–166
- parsing, 164–165

events.each block, 168
 exclude method, 197
 exclusive lock, 59
 executables, 190
 executables key, 189
 execute line (-e line), 220
 expected_notice method, 70
 explicit versions

- delimited values, 102–103
- RubyGems, 184

export command, 179
 ext directory, 185
 extend method, 50
 extensions key, 189
 extra_rdoc_files key, 189

F

\$F array, 17, 101
 f file, 60
 -F pattern (field separator specification), 220
 \$F variable, 24, 224
 factorial method, 30
 false expression, 8
 FasterCSV gem, 104
 field separator specification (-F pattern), 220
 fields, retrieving, 101–102
 File class, 71
`__FILE__` command flag, 223
 file handles, 223
 \$<.file variable, 16
 File.directory? file class, 213

- FileList class, 196, 202
 - File::LOCK_EX argument, 60
 - filename/.namedfork/rsrc address, 25
 - \$<.filename variable, 16
 - File.open method, 60, 61
 - File.open_safely property, 75
 - files
 - creating, 57–72
 - overview, 57
 - program-driven file creation, 66–70
 - safety issues, 57–66
 - template-driven file creation, 70–71
 - locking, 59–63
 - abstracted locking, 61–63
 - blocking locks, 59–60
 - nonblocking locks, 60
 - overview, 59
 - process locking, 63
 - files key, 189
 - files object, 196
 - FileUtils library, 20
 - FileUtils module, 64, 65
 - FileUtils.move, 65
 - File.world_readable? file class, 213
 - File.world_writable? file class, 213
 - filtering events, 165–166
 - find command, 25
 - find_or_create_by_ construction, 97
 - fixtures, 201
 - flock method, 60
 - flocks, 63
 - /foot/ expression, 15
 - foreach directive, 10
 - :foreign_key parameter, 121
 - form submission, 141
 - fred.destroy object, 122
 - Futurama quote, 140
 - future developments, 211–217
 - execution environments, 211–212
 - language changes, 212–216
 - arrays, 213
 - block argument locality, 214
 - Enumerable module, 215–216
 - hashes, 213
 - I/O operations, 213–214
 - multisplating, 214–215
 - object tapping, 215
 - read-write attributes, 215
 - strings, 213
 - overview, 211
- G**
- gathering data, 155–163
- GD library, 67
- gem command, 179, 183, 190
- GEM_HOME environment variable, 178
- gems, 177–191
 - accessing documentation, 185
 - accessing documentation via
 - gem_server, 185
 - creating, 185–191
 - format, 185–187
 - gathering files, 187
 - Gem::Specification class, 188
 - publishing, 191
 - gem command, 179
 - installing, 180–181
 - listing installed, 181
 - overview, 177
 - removing, 182–183
 - searching for, 180
 - updating, 181–182
 - updating RubyGems, 180
 - using in code, 183–184
- gem_server directory, 187
- gem_server setting, 191
- gem_server statement, 185, 203
- gemspec, 186
- Gem::Specification class, 188
- GET method, 130
- get method, 157
- get_multi method, 85
- gets separator, 101
- \$.; global field separator, 220
- global variable, 221
- \$< global variable, 223
- \$, global variable, 223
- \$.; global variable, 223
- \$/ global variable, 223

- globbing patterns, 69, 167
 - gnuplot file, 169
 - Goat class, 39
 - Graph.render method, 171
 - graphs, 174–176
 - Graphviz format, 174
 - grep alternative, 15–16
 - Group method, 121
 - groups proxy, 120
 - Group.users method, 121
 - GSSAPI (Kerberos)-based authentication, 119
 - GZip library, 178
 - gzipped tarball, 186
- H**
- h (help) command line option, 219
 - hash array, 64
 - Hash class, 38, 47, 112
 - hashes, 84, 213
 - Hash.new(0) directory, 168
 - Hash.values_at method, 65
 - has_many class macros, 120
 - has_many directive, 92, 121
 - has_rdoc setting, 189, 190
 - headings, 207
 - help (-h) command line option, 219
 - HFS+ file system, 25
 - hh:mm string, 165
 - /home/user directory, 179
 - homepage key, 189
 - horizontal_bar_graph method, 174
 - host key, 118, 157
 - :Host parameter, 159
 - hosts hash, 152
 - href method, 67
 - html/images directory, 195
 - HTTP command, 129
 - http:-style link, 207
 - hyperlinks, 207
- I**
- i flag, 16
 - I/O operations, 213–214
 - _id tag, 120
 - idioms, 153
 - IETF (Internet Engineering Task Force), 156
 - if-elsif-else-end block, 70
 - ignore garbage, 222
 - ImageMagick package, 170
 - images, building, 67
 - imaginarycorp.com domain, 117
 - tag, 207
 - import_module method, 160
 - improving readability, 224
 - in-lining extension technique, 40
 - in-place editing, 220
 - include directive, 50, 54, 106
 - :include: filename comment, 208
 - include flag, 208
 - include from additional directory
 - (-I path(s)), 220
 - include? method, 49
 - include method, 159
 - include REXML statement, 115
 - include statement, 170
 - increment/decrement, 85
 - index.yaml file, 194
 - initialize method, 7, 36, 111
 - inject method, 93, 152
 - input record separator, 221
 - inspect property, 74, 75
 - inspection strings, 74–76
 - install verb, 182
 - installing
 - gems, 180–181
 - RubyGems, 178–179
 - instantiation, 36, 38
 - Integer class, 45
 - integration with C applications, 39–41
 - interclass relationships, 120–121
 - interface specification, 125
 - Internet Engineering Task Force (IETF), 156
 - IO-like objects, 146
 - IO.each method, 219
 - IO.readlines method, 69
 - IPSocket subclass, 134
 - irb shell, 3
 - isolation, 83
 - iso.org.dod.internet.mgmt.mib-2.system.sys
 - Location object, 157

J

join operation, 103
 JRuby, 212
 JSAN, 177

K

kcode encoding choice (-K encoding), 220
 \$KCODE variable, 220
 Kerberos (GSSAPI)-based authentication, 119
 Kernel module, 5, 6
 Kernel.eval property, 75
 Kernel.gets code, 101
 Kernel.puts code, 102
 Kernel.split code, 101
 key data, 115
 :key => value-style flexible method
 signature, 123

L

l flag, 20
 l option, 18
 label, 207
 last command, 165
 last_logins method, 162
 layers key, 171
 LDAP. *See* Lightweight Directory Access
 Protocol (LDAP)
 ldap_mapping directive, 119, 122
 lib directory, 185
 Libxml-Ruby, 116
 Lightweight Directory Access Protocol
 (LDAP), 116–122
 connection to service, 118–119
 creation, retrieval, update, and
 destruction (CRUD) objects, 121–122
 interclass relationships, 120–121
 mapped classes, 119–120
 schema, 117–118
 line ending automated handling, 221
 line numbers, 17
 __LINE__ command flag, 223
 \$<.lineno variable, 17
 lines (records), retrieving, 101
 linguistic optimization, 33–36

links, 207
 list verb, 181
 listener class, 159
 listing installed gems, 181
 lists, 207–208
 _load method, 77
 load statement, 220
 \$LOAD_PATH array, 190
 local disk storage, 73–80
 inspection strings, 74–76
 overview, 74–75
 pros and cons, 75–76
 retrieval, 75
 storage, 75
 Marshal class
 customization, 77
 overview, 76
 pros and cons, 77–78
 retrieval, 77
 storage, 76
 overview, 73–74
 performance comparison, 80
 YAML, 78–79
 locking, 59–63
 abstracted, 61–63
 blocking locks, 59–60
 considerations, 63
 nonblocking locks, 60
 overview, 59
 process, 63
 logger key, 119
 login_times data structure, 75
 Logo, 67

M

MacPorts package, 170
 make tool, 191
 Management Information Base (MIB)
 file, 156
 :many parameter, 120, 121
 map! method, 11
 mapped classes, 119, 120
 map.resources file, 131

- Marshal class, 76–78
 - Marshal library, 78
 - Marshal.dump method, 76
 - Marshal.dump(some_object) method, 76
 - marshalling data, 163, 164
 - Marshal.load method, 77
 - MathHandler method, 125
 - max_value key, 172
 - mb_used value, 94, 95
 - measure method, 28
 - MegaFishCannon class, 144
 - :member => :award => :post directory, 132
 - :member => directory, 132
 - memcache-client method, 84
 - memcache method, 86
 - memcacheclient library, 85
 - memcached, 83–87
 - connecting to service, 84–85
 - overview, 83–84
 - removing data, 86
 - MemCache.new method, 84
 - metadata file, 186
 - metadata.gz file, 188
 - method_missing method, 47, 67, 94
 - MIB (Management Information Base)
 - file, 156
 - MibDir key, 157
 - MibModules key, 157
 - min_value key, 172
 - mkdir_p method, 163
 - Module method, 48
 - module_function directive, 137
 - monitoring, 148–153
 - packet, 150–153
 - socket-based, 138–139
 - SSH, 148–150
 - multi-word label, 207
 - multisplatting, 214–215
 - :multithread parameter, 85
- N**
- n flag, 16, 20, 100
 - N flag, 205
 - n objects, 216
 - n wraps, 100
 - name key, 189
 - named task, 194
 - :name_prefix => :book_: directory, 132
 - :namespace parameter, 85
 - nested retrieval loop, 221
 - Net module, 139
 - net-ssh client library, 148
 - Net::HTTP-based class, 140
 - network, 174
 - network-aware storage, 81–97
 - ActiveRecord module, 89–97
 - databases, 87–89
 - DBI, 87–89
 - SQL queries, 87–89
 - general design principles, 81–83
 - memcached, 83–87
 - connecting to service, 84–85
 - overview, 83–84
 - removing data, 86
 - object-relational mapping (ORM), 89–97
 - overview, 81
 - network monitoring, 155–176
 - analyzing data, 163–168
 - aggregate analysis, 167–168
 - assigning events, 165–166
 - filtering events, 165–166
 - marshalling data, 163–164
 - overview, 163
 - parsing events, 164–165
 - gathering data, 155–163
 - overview, 155
 - Secure Shell (SSH), 160–163
 - Simple Network Management Protocol (SNMP), 155–160
 - overview, 155
 - presenting data, 168–176
 - charts, 169–174
 - graphs, 174–176
 - overview, 168–169

- network services, 116–132
 - Lightweight Directory Access Protocol (LDAP), 116–122
 - connection to service, 118–119
 - creation, retrieval, update, and destruction (CRUD) objects, 121–122
 - interclass relationships, 120–121
 - mapped classes, 119–120
 - schema, 117–118
 - Representational State Transfer (REST), 128–132
 - client functionality, 130
 - server functionality, 131–132
 - Simple Object Access Protocol (SOAP), 125–128
 - client functionality, 126
 - server functionality, 127–128
 - XML Remote Procedure Call (XML-RPC), 122–125
 - networking, 133–153
 - basic network services, 133–139
 - socket-based monitoring, 138–139
 - socket connections, 133–134
 - socket errors and exceptions, 135
 - timing out on purpose, 135–137
 - control/monitoring of devices, 148–153
 - packet monitoring, 150–153
 - SSH, 148–150
 - higher-level network services, 139–148
 - building web robots, 140–144
 - overview, 139
 - protocols, 139–140
 - writing servers, 144–148
 - overview, 133
 - new method, 5, 7, 122, 156
 - next keyword, 10
 - next_key value, 114
 - nil expression, 8
 - *nix-style machines, 148
 - :nodoc: comment, 208
 - nonblocking locks, 60
 - NULL (000) value, 219
 - numbered files, 102
 - numbers, 24–25
- 0**
- .o file, 196
 - o flag, 205
 - Object class, 47
 - object method, 5
 - object orientation (OO), 3–14
 - accessors, 8–9
 - blocks, 9–12
 - Ruby view of, 5–8
 - theory, 3–5
 - object-relational mapping (ORM), 89–97
 - object storage, 73–97
 - local disk storage, 73–80
 - inspection strings, 74–76
 - Marshal class, 76–78
 - overview, 73–74
 - performance comparison, 80
 - YAML, 78–79
 - network-aware storage, 81–97
 - ActiveRecord module, 89–97
 - databases, 87–89
 - general design principles, 81–83
 - memcached, 83–87
 - object-relational mapping (ORM), 89–97
 - overview, 81
 - overview, 73
 - object tapping, 215
 - object.commit operation, 73
 - object.is_a?(Array) method, 114
 - Object.method_missing method, 94
 - object.retrieve operation, 73
 - octal record separator specification
 - 0[octal], 219
 - o_files statement, 197
 - on-init method, 128
 - one-line scripts, 15–19
 - comments, 16
 - directory listings, 18–19
 - fields, 17–18
 - grep alternative, 15–16
 - line numbers, 17
 - record-oriented files, 18
 - watch utility, 19

- on_trap method, 159
 - on_trap_default method, 159
 - open command, 17, 21
 - open-like method, 62
 - open method, 133, 156
 - open_locked, 61
 - open_safely, 64
 - open_safely method, 66
 - /opt directory, 148
 - optimization, 32–41
 - algorithmic, 32–33
 - integration with C applications, 39–41
 - linguistic, 33–36
 - String operations, 35–36
 - symbols compared to strings, 34
 - minimizing extra operations, 36–39
 - data structuring, 38–39
 - instantiation, 36–38
 - or conditions, 33
 - OR operator, 60
 - organizational unit (ou), 118
 - ORM (object-relational mapping), 89–97
 - ou (organizational unit), 118
 - outdated command, 183
 - output behavior, 102
 - output method, 175
 - output record separator, 221
- P**
- p flag, 16, 100
 - p names property, 74
 - p <portnum> statement, 185
 - p scores property, 74
 - p some_object property, 74
 - packet monitoring, 150–153
 - page array, 67
 - parse_event method, 164
 - parsing
 - data, 99–116
 - events, 164–165
 - libraries
 - alternative, 116
 - REXML, 105–107
 - partial writes, 63
 - password key, 119
 - password_block Proc key, 119
 - path array, 64
 - PATH environment variable, 221
 - path method, 75
 - :path_prefix => :/authors/:author_id:
 - directory, 131
 - Pcaplet class, 150
 - performance, 23–41
 - analyzing, 27–32
 - Benchmark library, 27–29
 - Profiler library, 30–32
 - UNIX time command, 27
 - optimization, 32–41
 - algorithmic optimization, 32–33
 - integration with C applications, 39–41
 - linguistic optimization, 33–36
 - minimizing extra operations, 36–39
 - script speed, 23–26
 - numbers, 24–25
 - resource forks, 25–26
 - Perl, 1
 - PID (process ID), 223
 - pkg directory, 191
 - platform key, 189
 - PList (property list) format, 109–116
 - plist tag, 111
 - PListArray, 112
 - PListHash, 112
 - PListParser.add_object object, 113, 114
 - plug-in API: macros for adding macros, 54, 55
 - point_markers key, 171
 - policies, building/changing text files using, 68–70
 - polymorphism, 44
 - port key, 118, 157
 - :Port parameter, 159
 - POST method, 130
 - post method, 142
 - <prefix>/bin script, 179
 - prefix path, 178
 - presenting data, 168–176
 - :primary_key parameter, 121
 - print method, 134

- print \$_ variable, 16
 - printed nested retrieval loop, 221
 - printf format, 213
 - Proc object, 77
 - process ID (PID), 223
 - process locking, 63
 - processed directory, 167
 - processed folder, 167
 - Profiler library, 30–32
 - program-driven file creation, 66–70
 - property list (PList) format, 109–116
 - protocols, 139–140
 - proxy method, 123
 - proxy2 method, 123
 - public/images directory, 173
 - publishing gems, 191
 - PUT request, 131
 - puts command, 2, 134
- Q**
- query verb, 180
- R**
- r flag, 180
 - rails command, 181
 - rake, 193–198
 - documenting from, 208–209
 - documenting tasks, 198
 - ensuring existence of directories, 195
 - file tasks, 194–195
 - overview, 193–194
 - rules, 195–196
 - synthesizing tasks, 196–198
 - testing from, 202–203
 - rake index.yaml command line, 195
 - rake tool, 190
 - Rakefile file, 193
 - Rake::RDocTask documentation, 209
 - Rake::TestTask documentation, 202
 - rasterizer key, 171
 - RDoc-based documentation, 181
 - rdoc command, 203, 205
 - rdoc distribution, 203
 - rdoc --help command, 205
 - rdoc options, 208
 - rdoc_files task, 209
 - rdoc_options key, 189
 - read command, 134
 - read-write attributes, 215
 - readability, improving, 224
 - README file, 185, 187, 208
 - readonly parameter, 85
 - record-oriented files, 18
 - records (lines), retrieving, 101
 - RemoteHost class, 149, 160, 161
 - removing gems, 182–183
 - repetitive text files, 68
 - reporting object, 28
 - Representational State Transfer (REST), 128–132
 - request_timeout attribute, 85
 - require command, 183
 - require library, 221
 - require :rubygems: script, 183
 - require statement, 190, 220
 - required_ruby_version key, 189
 - requirements key, 189
 - require_paths key, 189
 - rerdoc task, 209
 - rescue keyword, 138
 - rescue method, 137
 - resource forks, 25–26
 - REST (Representational State Transfer), 128–132
 - Retries key, 157
 - retrieving
 - fields, 101–102
 - records (lines), 101
 - scalar data, SNMP, 157–158
 - return keyword, 139
 - REXML, 105, 107
 - REXML::Document constructor, 105
 - REXML::Document.parse_stream constructor, 108
 - REXML::StreamListener, 108
 - ri attr_accessor method, 48
 - ri command, 181
 - ri File.join method, 75
 - ri inject method, 93

- ri Marshal method, 78
 - ri module_function method, 137
 - ri Object method, 78
 - ri SomeClass.some_method, 2
 - ri String class, 2
 - ri Struct class, 39
 - ri Time.strptime method, 75
 - RMagick library, 170
 - routes.rb file, 131
 - rsrc flag, 26
 - Ruby, 1, 14
 - advantages, 1–3
 - basic example, 1
 - object orientation (OO), 3–14
 - accessors, 8–9
 - blocks, 9–12
 - Ruby view of, 5–8
 - theory, 3–5
 - types, 12–14
 - yield command, 9–12
 - ruby command, 194
 - ruby-core, 211
 - ruby-doc, 211
 - ruby-graphviz library, 174
 - Ruby/LDAP library, 117
 - ruby method, 149
 - Ruby scripts, 211
 - Ruby SNMP library, 158
 - Ruby SOAP library, 126, 128
 - ruby string, 222
 - ruby-talk, 211
 - RubyForge, 178
 - rubyforge_project key, 189
 - RubyGems
 - explicit versioning, 184
 - homepage, 177
 - installing, 178–179
 - requiring, 183
 - updating, 180
 - rubygems library, 188
 - RubyInline library, 40
 - rule statement, 197
 - runtime environment, 222–224
-
- S**
 - s flag, 205
 - safe file operations, 63–66
 - sasl_quiet parameter, 119
 - Schedule.current? statement, 45
 - Schwarzian transform, 19
 - script search, 221
 - scripts
 - executable Ruby script generator, 20–21
 - log rotating, 19–20
 - one-liners, 15–19
 - comments, 16
 - directory listings, 18–19
 - fields, 17–18
 - grep alternative, 15–16
 - line numbers, 17
 - record-oriented files, 18
 - watch utility, 19
 - speed, 23–26
 - Scruffy Charts, 169–173
 - scruffy library, 169
 - search verb, 181
 - searching for gems, 180
 - Secure Shell (SSH), 140, 148, 150, 160–163
 - select function, 146
 - select pattern, 146
 - self command, 96
 - self method, 11
 - self.each method, 11
 - separators, 207
 - server functionality
 - Representational State Transfer (REST), 131–132
 - Simple Object Access Protocol (SOAP), 127–128
 - XML Remote Procedure Call (XML-RPC), 124–125
 - server method, 54
 - servers, writing, 144, 148
 - server.start block, 128
 - set method, 158
 - SGMP (Simple Gateway Monitoring Protocol), 156
 - shared lock, 59

- shares method, 54
- shell command, 149
- shell.ruby invocation, 149
- shutdown method, 128
- Signal.trap(:INT:) command, 128
- signatures, flexible, 44–48
 - default values, 44–45
 - missing method dynamic dispatch, 47–48
- Simple Gateway Monitoring Protocol (SGMP), 156
- Simple Network Management Protocol (SNMP), 140, 155–160
 - connection to agent, 156
 - custom MIBs, 160
 - handling traps, 159–160
 - overview, 155–156
 - retrieving tabular data, 158
 - updating data, 158–159
- Simple Object Access Protocol (SOAP), 125–128
- :singular => :book: directory, 131
- size key, 172
- slurping, 219
- smart variables, 221
- smidump tool, 160
- SNMP. *See* Simple Network Management Protocol (SNMP)
- snmp gem, 156
- SNMP::Manager class, 156
- SNMP::MIB class, 160
- SNMP::TrapListener.new management class, 159
- SOAP (Simple Object Access Protocol), 125–128
- soap4r gem, 128
- SOAP::RPC::Driver class, 126
- SOAP::RPC::StandaloneServer class, 127
- SOAP::WSDLDriverFactory class, 126
- socket-based monitoring, 138–139
- socket connections, 133–134
- socket errors, 135
- socket exceptions, 135
- Socket operations, 136
- SomeClass::some_method, 2
- SomeClass#some_method, 2
- some_object.dump method, 76, 77
- some_object.inspect property, 74
- \$(something) command flag, 223
- something.o file, 196
- sort_by method, 19
- source flag, 191
- speak method, 13
- splatting trick, 215
- split command, 17
- square brackets, 208
- SSH (Secure Shell), 140, 148, 150, 160–163
- starfish gem, 180
- starfish verb, 183
- stats method, 87
- status commands, 141
- \$stderr variable, 223
- STDERR variable, 223
- \$stdin variable, 223
- STDIN variable, 223
- \$stdout variable, 223
- STDOUT variable, 223
- \$stdout.puts code, 102
- StorageBin class, 167
- store_password key, 119
- Streamer class, 108
- streaming classes, 107–109
- String class, 2, 6, 77
- String function, 35, 36
- String.bytes method, 213
- String.each construction, 97
- String.each method, 213
- String.lines method, 213
- strings, 34, 35–36, 213
- String.slice invocation, 161
- String.split code, 101
- String.split format, 103
- String.split method, 220
- String.tr, 103
- Struct class, 39
- subversion, 203, 212
- sudo mechanism, 150
- suites, 202
- summary key, 190

- super keyword, 13, 142
 - symbols, compared to strings, 34
 - system method, 19
 - system option, 181
- T**
- T option, 198
 - tab-separated values (TSVs), 103, 104, 169
 - tabular data, retrieving, 158
 - tag_end object, 114
 - tag_start object, 111
 - taint level, 221
 - tap method, 215
 - tarball (.tgz), 178
 - tc_statement, 202
 - tcpdump method, 150
 - TCPServer class, 144
 - TCPsocket class, 134, 138, 144
 - Tempfile class, 64
 - template-driven file creation, 70–71
 - templates, 70
 - ten-second script, 21
 - test harness, 201
 - test_statement, 201
 - test task, 194, 202
 - test utility, 26
 - test_arithmetic.rb file, 200
 - test_files key, 190
 - testing, 198–203
 - fixtures, 201
 - overview, 198–199
 - performing, 200–201
 - from rake, 202–203
 - test suites, 202
 - Test::Unit module, 199–200
 - test.rb script, 221
 - tests directory, 185
 - test_subtraction statement, 200
 - Test::Unit module, 199–200
 - Test::Unit::TestCase module, 199, 201
 - text files, building/changing using policies, 68–70
 - text method, 106, 114
 - .tgz (tarball), 178
 - theme key, 171, 172
 - threads << Thread.new(foo) { |bar| ... }
 - operation, 147
 - Time class, 164
 - time command, 27
 - Time method, 76
 - Timeout key, 157
 - timeout method, 136
 - Timeout module, 136
 - Timeout::Error module, 136
 - Timeout::timeout module, 136
 - Time.parse invocation, 165
 - timing out, 135–137
 - title key, 171
 - /tmp directory, 103, 148
 - to_s method, 28, 45, 106
 - to_yaml method, 78, 79
 - transaction command, 96
 - TrapPort key, 157
 - try_sasl parameter, 119
 - ts_statement, 202
 - TSVs (tab-separated values), 103, 104, 169
 - types, 9, 12, 14
- U**
- UDPTransport SNMP transport, 156
 - unary operator, 213
 - uninstall verb, 182
 - uniq method, 33
 - UNIX daemon, 63
 - UNIX-style systems, 63
 - UNIX time command, 27
 - UNIXSocket subclass, 134
 - update verb, 180
 - update_all method, 95
 - updating
 - gems, 181–182
 - RubyGems, 180
 - usage method, 95
 - usage value, 94
 - User class, 119, 204
 - user key, 119
 - User object, 52
 - User subclass, 52

User.authenticate method, 206
 User.groups method, 120
 user_info file, 17, 100
 User.new method, 122
 user.rb file, 203
 users array, 65
 users hash, 172
 users organizational unit, 119
 /usr/share/dict/words path, 16

V

value_formatter key, 171
 /var/run/myscript.pid file, 63
 VarBind objects, 160
 \$< variable, 16
 \$. variable, 17
 \$; variable, 100
 \$/ variable, 100, 101
 \$ variable, 102
 \$! variable, 224
 vendor/plugins directory, 173
 verbose mode, 221
 version command line option, 219
 version key, 157, 190
 virtual machine (VM), 212

W

w flag, 2
 warnings mode, 222
 watch utility, 19
 web robots, 140–144
 Web Services Description Language (WSDL)
 file, 126
 Webrick, 148
 WebSession class, 141, 187
 width key, 172

word element, 106
 :wrap parameter, 121
 wrapper method, 28
 WriteCommunity key, 157
 writing servers, 144–148
 WSDL (Web Services Description Language)
 file, 126
 wsdl2ruby.rb utility, 128
 wtmp files, 161–163

X

XML, 104–116
 alternative parsing libraries, 116
 building, 66–67
 overview, 104–105
 property list (PList) format, 109–116
 REXML, 105–107
 streaming classes, 107–109
 XML Remote Procedure Call (XML-RPC),
 122–125
 client functionality, 122–124
 server functionality, 124–125
 XMLRPC::Client class, 122
 XMLRPC::Server instance, 124

Y

-y (yydebug) command line option, 219
 YAML, 78–79, 178
 *.yaml globbing pattern, 167
 YAML.dump method, 78
 Yet Another Ruby Virtual Machine (YARV),
 211–212
 yield command, 9, 11, 12
 yydebug (-y) command line option, 219

Z

ZIP archive (.zip), 178