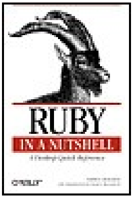
RUBY IN A NUTSHELL

A Desktop Quick Reference

O'REILLY®

Yukihiro Matsumoto
with translated text by David L. Reynolds, Jr.



Chapter 5. Ruby Tools

5.3 Ruby Application Archive

Do you want to access databases, such as PostgreSQL or MySQL from Ruby? Do you wish to use such nonstandard GUI toolkits as Qt, Gtk, FOX, etc.? You can with the Ruby Application Archive (RAA), which has a collection of Ruby programs, libraries, documentations, and binary packages compiled for specific platforms. You can access RAA at http://www.ruby-lang.org/en/raa.html. RAA is still far smaller than Perl's CPAN, but it's growing every day.

RAA contains the following elements:

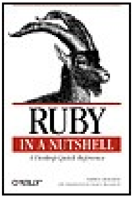
- The latest 10 items
- A list of Ruby applications
- A list of Ruby libraries

- A list of Ruby porting
- A list of Ruby documents

You can enter your program in RAA by clicking "add new entry" at the top of the RAA page, then following the instructions there. RAA itself is a fully automated web application written in Ruby. It uses eRuby and PStore as a backend.

Previous Next

Top



Chapter 5. Ruby Tools

5.2 Additional Tools

There are other useful tools that don't com
Ruby standard distribution. However, you
them yourself.

5.2.1 ri: Ruby Interactive Refe

`ri` is a online reference tool developed by
famous pragmatic programmer. When you
the behavior of a certain method, e.g., `IO#`
`ri IO#gets` to read the brief explanation o
get `ri` from
http://www.pragmaticprogrammer.com/ru

```
ri [ options ] [ name... ]
```

Here are the `ri` options:

`--version,`

`-v`

Displays version and exits.

`--line-length=n`

`-l n`

Sets the line length for the output (m
characters).

`--synopsis`

`-s`

Displays just a synopsis.

`--format=` *`name`*

`-f` *`name`*

Uses the *`name`* module (default is `Pla`
formatting. Here are the available mo

Tagged

Simple tagged output

Plain

Default plain output

`name` should be specified in any of th

- *`Class`*
- *`Class::method`*
- *`Class#method`*
- *`Class.method`*
- *`method`*

5.2.2 eRuby

eRuby stands for embedded Ruby; it's a to

fragments of Ruby code in other files sucl
Here's a sample eRuby file:

```
This is sample eRuby file<br>
The current time here is <%=Time.nc
<%[1,2,3].each{|x|print x,"<br>\n"}
```

Here's the output from this sample file:

```
This is sample eRuby file<br>
The current time here is Wed Aug 29
1
2
3
```

There are two eRuby implementations:

eruby

The original implementation of eRuk
from http://www.modruby.net/.

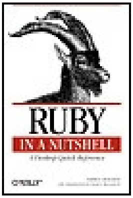
Erb

A pure Ruby (subset) implementatio

eRuby is available from
http://www2a.biglobe.ne.jp/~seki/ruby/er
version number may be changed in the fu
supporting page http://www2a.biglobe.ne
Japanese, but you can tell how to use it fr

Previous Next

Top



Ruby in a NutshellBy Yukihiro Matsumoto

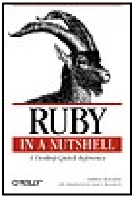
Chapter 6. Ruby Updates

Compared to most other languages, Ruby is rather young. As a result, it's still evolving fairly rapidly.

If you find a bug in Ruby, the first thing to do is to check the bug database and see if the problem has already been reported. The bug database can be found at http://www.ruby-lang.org/cgi-bin/ruby-bugs. You can either send the bug report directly from that page or send an email to ruby-bugs@ruby-lang.org. When you submit your bug, try to include all relevant information such as source code, operating system, the output from `ruby -v`, and what version/build of Ruby you are running. If you have compiled your own build of Ruby, you should also include the `rbconfig.rb`.

The current stable version of Ruby can always be found at http://www.ruby-lang.org/en/download.html. There are also several mirror sites available.

The current developmental release can be obtained from the CVS (Concurrent Version System) repository. See http://www.ruby-lang.org/en/cvsrepo.html for instructions. You can get CVS tools from http://www.cvshome.com/.



Ruby in a Nutshell

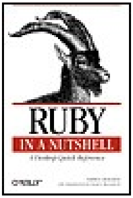
By Yukihiro Matsumoto

Publisher	: O'Reilly
Pub Date	: November 2001
ISBN	: 0-59600-214-9
Pages	: 218

Buy Print Version

- Table of Contents
- Reviews
- Reader Reviews
- Errata

Top



Chapter 5. Ruby Tools

5.3 Ruby Application Archive

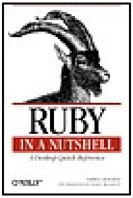
Do you want to access databases, such as PostgreSQL or MySQL from Ruby? Do you wish to use such nonstandard GUI toolkits as Qt, Gtk, FOX, etc.? You can with the Ruby Application Archive (RAA), which has a collection of Ruby programs, libraries, documentations, and binary packages compiled for specific platforms. You can access RAA at http://www.ruby-lang.org/en/raa.html. RAA is still far smaller than Perl's CPAN, but it's growing every day.

RAA contains the following elements:

- The latest 10 items
- A list of Ruby applications
- A list of Ruby libraries

- A list of Ruby porting
- A list of Ruby documents

You can enter your program in RAA by clicking "add new entry" at the top of the RAA page, then following the instructions there. RAA itself is a fully automated web application written in Ruby. It uses eRuby and PStore as a backend.


Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 5. Ruby Tools

5.1 Standard Tools

The standard Ruby distribution contains u
standard libraries: debugger, profiler, `irb`
for Emacs. These tools help you debug an

5.1.1 Debugger

It doesn't matter how easy a language is to
more than a few lines long. To help deal w
includes a debugger. In order to start the R
the command-line option `-r debug`. The d
executable code and asks for the input of

Here are the debugger commands:

`b[reak] [<`*`file`*`|`*`class`*`>:]<`*`line`*`|`*`meth`*

Sets breakpoints

`wat[ch]` *`expression`*

Sets watchpoints

`b[reak]`

Displays breakpoints and watchpoint

`del[ete] [ n]`

Deletes breakpoints

`disp[lay] expression`

Displays value of `expression`

`undisp[lay] [ n]`

Removes display of `n`

`c[ont]`

Continues execution

`s[tep] [ n]`

Executes next `n` lines stepping into m

`n[ext] [ n]`

Executes next *n* lines stepping over r

`w[here]`

Displays stack frame

`f[rame]`

Synonym for where

`l[ist][<-| n- m>]`

Displays source lines from *n* to *m*

`up [ n]`

Moves up *n* levels in the stack frame

`down [` *`n`*`]`

Moves down *n* levels in the stack fra

`fin[ish]`

Finishes execution of the current met

`tr[ace] [on|off]`

Toggles trace mode on and off

`q[uit]`

Exits debugger

`v[ar] g[lobal]`

Displays global variables

`v[ar] l[ocal]`

Displays local variables

`v[ar] i[instance]` *`object`*

Displays instance variables of *`objec`*

`v[ar] c[onst]` *`object`*

Displays constants of object

`m[ethod] i[instance]` *`object`*

Displays instance methods of *`object`*

`m[ethod]` *`class`*`|`*`module`*

Displays instance methods of the *`cla`*

`th[read] l[ist]`

Displays threads

`th[read] c[ur[rent]]`

Displays current thread

`th[read]` *n*

Stops specified thread

`th[read] stop` *n*`>`

Synonym for th[read] *n*

`th[read] c[ur[rent]]` *n*`>`

Synonym for th[read] *n*

`th[read] resume` *n*`>`

Resumes thread *n*

`p` *`expression`*

Evaluates the *`expression`*

`h[elp]`

Displays help message

`<` *`everything else`*`>`

Evaluates the expression

The following is a sample session that sho
the *Sieves of Eratosthenes* program (a fam
The interface is designed similarly to that

```
% ruby -r debug sieve.rb 100
Debug.rb
Emacs support available.

sieve.rb:2:max = Integer(ARGV.shift
(rdb:1) list
[-3, 6] in sieve.rb
   1
=> 2  max = Integer(ARGV.shift || 1
   3  sieve = []
   4  for i in 2 .. max
   5    sieve[i] = i
   6  end
(rdb:1) list
[7, 16] in sieve.rb
   7
   8  for i in 2 .. Math.sqrt(max)
   9    next unless sieve[i]
   10    (i*i).step(max, i) do |j|
   11      sieve[j] = nil
   12    end
   13  end
   14  puts sieve.compact.join ", "
 (rdb:1) b 8
Set breakpoint 1 at sieve.rb:8
(rdb:1) c
Breakpoint 1, toplevel at sieve.rb:
sieve.rb:8:for i in 2 .. Math.sqrt(
(rdb:1) p sieve
```

```
[nil, nil, 2, 3, 4, 5, 6, 7, 8, 9,
19, 20, 21, 22, 23, 24, 25, 26, 27,
37, 38, 39, 40, 41, 42, 43, 44, 45,
55, 56, 57, 58, 59, 60, 61, 62, 63,
73, 74, 75, 76, 77, 78, 79, 80, 81,
91, 92, 93, 94, 95, 96, 97, 98, 99,
(rdb:1) del 1
(rdb:1) b 14
Set breakpoint 2 at sieve.rb:14
(rdb:1) c
Breakpoint 2, toplevel at sieve.rb:
sieve.rb:14:puts sieve.compact.join
(rdb:1) p sieve
[nil, nil, 2, 3, nil, 5, nil, 7, ni
nil, 17, nil, 19, nil, nil, nil, 23
31, nil, nil, nil, nil, nil, 37, ni
nil, 47, nil, nil, nil, nil, nil, 5
61, nil, nil, nil, nil, nil, 67, ni
nil, nil, nil, 79, nil, nil, nil, 8
nil, nil, nil, nil, nil, nil, 97, n
(rdb:1) sieve.compact
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29
71, 73, 79, 83, 89, 97]
(rdb:1) c
```

```
2, 3, 5, 7, 11, 13, 17, 19, 23, 29,
71, 73, 79, 83, 89, 97
```

5.1.2 Profiler

In most cases, you can improve the perfor
bottleneck. The *profiler* is a tool that finds
your Ruby program, you need to first load
option `-r profile`. Here is the sample out
`Object#fact` method is a bottleneck.

```
% ruby -r profile sample/fact.rb 10
93326215443944152681699238856266700
32299156089414639761565182862536979
00000000000
  %   cumulative   self
 time   seconds   seconds    calls
 66.67     0.07      0.07        1
 16.67     0.08      0.02        1
  0.00     0.08      0.00        5
  0.00     0.08      0.00        2
  0.00     0.08      0.00        1
  0.00     0.08      0.00       95
  0.00     0.08      0.00        1
  0.00     0.08      0.00      101
  0.00     0.08      0.00        1
  0.00     0.08      0.00        1
  0.00     0.08      0.00        1
  0.00     0.08      0.00      100
  0.00     0.08      0.00        1
```

5.1.3 Tracer

When you want to trace the entrance and
you. In order to add method call/return tra
library using the command-line option `-r`

```
% ruby -r tracer fact.rb 2
#0:fact.rb:1::-: def fact(n)
#0:fact.rb:1:Module:>: def fact(n)
#0:fact.rb:1:Module:<: def fact(n)
#0:fact.rb:10::-: print fact(ARGV[0
#0:fact.rb:10:Array:>: print fact(A
#0:fact.rb:10:Array:<: print fact(A
#0:fact.rb:10:String:>: print fact(
#0:fact.rb:10:String:<: print fact(
#0:fact.rb:1:Object:>: def fact(n)
#0:fact.rb:2:Object:-:   return 1 i
#0:fact.rb:2:Fixnum:>:   return 1 i
#0:fact.rb:2:Fixnum:<:   return 1 i
#0:fact.rb:3:Object:-:   f = 1
#0:fact.rb:4:Object:-:   while n>0
#0:fact.rb:4:Fixnum:>:   while n>0
#0:fact.rb:4:Fixnum:<:   while n>0
#0:fact.rb:5:Object:-:     f *= n
#0:fact.rb:5:Fixnum:>:     f *= n
#0:fact.rb:5:Fixnum:<:     f *= n
#0:fact.rb:6:Object:-:     n -= 1
#0:fact.rb:6:Fixnum:>:     n -= 1
#0:fact.rb:6:Fixnum:<:     n -= 1
#0:fact.rb:6:Fixnum:>:     n -= 1
#0:fact.rb:6:Fixnum:<:     n -= 1
#0:fact.rb:5:Object:-:     f *= n
#0:fact.rb:5:Fixnum:>:     f *= n
#0:fact.rb:5:Fixnum:<:     f *= n
#0:fact.rb:6:Object:-:     n -= 1
#0:fact.rb:6:Fixnum:>:     n -= 1
#0:fact.rb:6:Fixnum:<:     n -= 1
#0:fact.rb:6:Fixnum:>:     n -= 1
```

```
#0:fact.rb:6:Fixnum:<:     n -= 1
#0:fact.rb:8:Object:-:   return f
#0:fact.rb:8:Object:<:   return f
#0:fact.rb:10:Kernel:>: print fact(
#0:fact.rb:10:IO:>: print fact(ARGV
#0:fact.rb:10:Fixnum:>: print fact(
#0:fact.rb:10:Fixnum:<: print fact(
2#0:fact.rb:10:IO:<: print fact(ARG
#0:fact.rb:10:IO:>: print fact(ARGV

#0:fact.rb:10:IO:<: print fact(ARGV
#0:fact.rb:10:Kernel:<: print fact(
```

You can turn on trace mode explicitly by i

```
Tracer.on
```

Turns on trace mode

```
Tracer.on {...}
```

Evaluates the block with trace mode

`Tracer.off`

Turns off trace mode

5.1.4 irb

`irb` (Interactive Ruby) was developed by
commands at the prompt and have the inte
program. `irb` is useful to experiment with

```
irb [ options ] [ programfile ] [ a
```

Here are the `irb` options:

`-f`

Suppresses loading of `~/.irbrc`.

`-m`

Math mode. Performs calculations us

`-d`

Debugger mode. Sets `$DEBUG` to `true`

`-r` *`lib`*

Uses `require` to load the library *`lib`*

`-v`

`--version`

Displays the version of `irb`.

`--inspect`

Inspect mode (default).

`--noinspect`

Noninspect mode (default for math m

`--readline`

Uses the `readline` library.

`--noreadline`

Suppresses use of the `readline` libra

`--prompt` *`mode`*

`--prompt-mode` *`mode`*

Sets the prompt mode. Predefined pr
`inf-ruby`.

`--inf-ruby-mode`

Sets the prompt mode to `inf-ruby` a

`--simple-prompt`

Sets the prompt mode to simple mod

`--noprompt`

Suppresses the prompt display.

`--tracer`

Displays a trace of method calls.

`--back-trace-limit` *`n`*

Sets the depth of backtrace informati

Here is a sample `irb` interaction:

```
irb
irb(main):001:0> a = 25
25
irb(main):002:0> a = 2
2
irb(main):003:0>
```

```
matz@ev[sample] irb
irb(main):001:0> a = 3
3
irb(main):002:0> a.times do |i|
irb(main):003:1* puts i
irb(main):004:1> end
0
1
2
3
irb(main):005:0> class Foo<Object
irb(main):006:1> def foo
irb(main):007:2> puts "foo"
irb(main):008:2> end
irb(main):009:1> end
nil
irb(main):010:0> Foo::new.foo
foo
nil
irb(main):011:0> exit
```

`irb` loads a startup file from either `~/.irb`
`Startup` file can contain an arbitrary Ruby

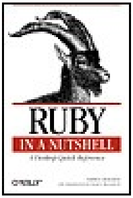
it, `irb` context object `IRB` is available.

`irb` works as if you fed the program line b
noninteractive interpreter executes the pro
example, in batch execution, the local var
treated as a local variable outside of `eval`.
a local variable or not statically. In non-`ir`
identifier is a local variable during compil
program first and then executes it, assignm
mode, `irb` normally executes inputs line b
compilation of the next line.

5.1.5 ruby-mode for Emacs

If you are an Emacs user, `ruby-mode` will
colorizing program text, etc. To use `ruby-`
included in your `load-path` variable, then

```
(autoload 'ruby-mode "ruby-mode")
(setq auto-mode-alist (append (list
       auto-mode-alist))
(setq interpreter-mode-alist (appen
       interpreter-mode-alist))
```



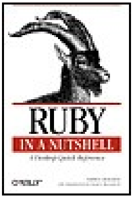
Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 6. Ruby Updates

6.1 Summary of Changes

Developmental releases of Ruby always have an odd minor revision number such as 1.5 or 1.7. Once a developmental release is stable and finalized, it's then "promoted" to a stable release. Stable releases always have an even minor revision number such as 2.0 or 3.2. Therefore, releases with even subversion numbers (1.4, 1.6, 1.8, etc.) are stable releases. Releases with odd subversion numbers (1.5, 1.7, etc.) are developmental versions and are available only from the CVS repository.

At of the writing of this book, the current stable release version is 1.6.5. The current developmental version is 1.7.1. The changes presented here are currently reflected in 1.7.1 and will probably remain relatively unchanged in the next stable release�ersion 1.8.



Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 1. Introduction

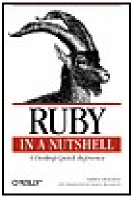
Ruby has been readily adopted by programmers in Japan and has had much documentation written for it in Japanese.

As programmers outside of Japan learn about the benefits of Ruby, there is a growing need for documentation in English. The first book I wrote for O'Reilly, *Ruby Pocket Reference*, was in Japanese. Since then Ruby has changed significantly. To meet the needs of non-Japanese programmers, we translated, updated, and expanded *Ruby Pocket Reference* into *Ruby in a Nutshell*.

Ruby is an object-oriented programming language that makes programming both enjoyable and fast. With the easy-to-use interpreter, familiar syntax, complete object-oriented functionality, and powerful class libraries, Ruby has become a language that can be applied to a broad range of fields from text processing and CGI scripts to professional, large-scale programs.

While Ruby is easy to learn, there are many details that you can't be expected to remember. This book presents those details in a clean and concise format. It is a reference to keep next to your desktop or laptop, designed to make Ruby even easier to use.

For those of you who are new to Ruby, there are several online tutorials available to get you started: Ruby's home page (http://www.ruby-lang.org/) is a good starting pointing as it offers Ruby tutorials and the Ruby Language FAQ.



Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 1. Introduction

1.1 Ruby's Elegance

Ruby is a genuine object-oriented scriptin
from the ground up to support the OOP m

Most modern languages incorporate aspec

programming. Because Ruby was designe
beginning to support OOP, most programm
elegant, easy to use, and a pleasure to prog
Ruby is an object; there's no exception.

While Ruby is object-oriented, you can al
procedural programming. But as you do, I
turning your nifty procedures into method
accessible object.

Throughout the development of the Ruby
focused my energies on making programm
easier. To do so, I developed what I call th
surprise. All features in Ruby, including o
features, are designed to work as ordinary
me) expect them to work. Here are some o

Interpretive programming

No compilation is needed; you can ed

program to the interpreter. The faster
helps you enjoy the programming pr

Dynamic programming

Almost everything in Ruby is done a
variables and expressions are determ
are class and method definitions. You
programs within programs and execu

Familiar syntax

If you've been programming in Java,
C/C++, or even Smalltalk, Ruby's sy
The following simple factorial functi
easily you can decipher its meaning:

```
def factorial(n)
  if n == 0
    return 1
  else
```

```
    return n * factorial(n-1)
  end
end
```

Iterators

The iterator feature for loop abstracti
language, which means a block of co
to a method call. The method can cal
from within its execution. For examp
`each` method to iterate over its conte
feature, you don't need to worry abou
or boundary condition.

```
ary = [1,2,3,4,5]
ary.each do |i|
  puts 1*2
end  # prints 2,3,4,8,10 for ea
```

A block is used not only for loops. It can b
purposes including the `select` method of

blocks to choose values that satisfy condit

```
ary = [1,2,3,4,5]
ary = ary.select do |i|
  i %2 == 0
end  # returns array of even number
```

Exceptions

Just as you'd expect in a modern OO
provides language-level support for e
For example, an attempt to open a fil
raises an exception, so that your prog
assuming an unmet precondition. Th
enhances the reliability of your progr
be caught explicitly using the `rescue`
statement:

```
begin
  f = open(path)
rescue
```

```
    puts "#{path} does not exist.
    exit 1
  end
```

Class libraries

Ruby comes with a strong set of bun
that cover a variety of domains, from
(strings, arrays, and hashes) to netwo
programming. The following progra
current time string from the local hos
socket connection:

```
require "socket"
print TCPSocket.open("localhost
```

In addition to bundled libraries, if yo
http://www.ruby-lang.org/en/raa.htm
many unbundled useful libraries alor
and documentation. Since Ruby is ra
number of libraries available is smal

for example, but new libraries are be
each day.

Portable

Ruby ports to many platforms, incluc
Windows, OS/2, etc. Ruby programs
platforms without modification.

Garbage collection

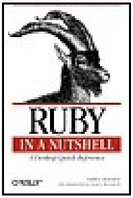
Object-oriented programming tends t
objects during execution. Ruby's garl
recycles unused object automatically

Built-in security check

Ruby's taint model provides safety w
untrusted data or programs.

Previous Next

Top



Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 1. Introduction

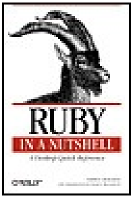
1.2 Ruby in Action

Like Python or Perl, Ruby is a scripting la
languages offer some great advantages ov
C++ and Java. They allow programmers t

programming concepts and principles in a
space. Ruby does this, while maintaining

```
# the "Hello World."
print "Hello World.\n"

# output file contents in reverse o
print File::readlines(path).reverse

# print lines that contains the wor
while line = gets(  )
  if /Ruby/ =~ line
    print line
  end
end



# class and methods
class Animal
  def legs
    puts 4
  end
end
```

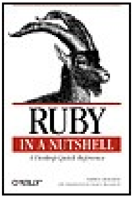
```
class Dog<Animal
  def bark
    puts "bow!"
  end
end

fred = Dog::new
fred.legs                       # prints 4
fred.bark                       # prints b

# exception handling
begin
  printf "size of %s is %d\n", path
rescue
  printf "error! probably %s does n
end

# rename all files to lowercase nam
ARGV.each {|path| File::rename(path

# network access
require 'socket'
print TCPSocket::open("localhost",
# Ruby/Tk
require 'tk'
TkButton.new(nil, 'text'=>'hello',
Tk.mainloop
```



Chapter 2. Language Basics

Ruby does what you'd expect it to do. It is highly consistent, and allows you to get down to work without having to worry about the language itself getting in your way.

Previous Next



Chapter 2. Language Basics

2.1 Command-Line Options

Like most scripting language interpreters,
run from the command line. The interpret
with the following options, which control

and behavior of the interpreter itself:

```
ruby [ options ] [� [ programfile ]
```

`-a`

Used with `-n` or `-p` to split each line.
stored in `$F`.

`-c`

Checks syntax only, without executi

`-C dir`

Changes directory before executing (

`-d`

Enables debug mode (equivalent to -
`$DEBUG` to `true`.

`-e` *`prog`*

Specifies *`prog`* as the program from t
Specify multiple `-e` options for multi

`-F` *`pat`*

Specifies *`pat`* as the default separator
by `split`.

`-h`

Displays an overview of command-li
(equivalent to `-help`).

`-i [` *`ext`*`]`

Overwrites the file contents with pro
original file is saved with the extensi
specified, the original file is deleted.

`-I` *`dir`*

Adds *`dir`* as the directory for loading

`-K [` *`kcode`*`]`

Specifies the multibyte character set
EUC (extended Unix code); `s` or `S` fc
`u` or `U` for UTF-8; and `a`, A, `n`, or `N` fo

`-l`

Enables automatic line-end processin
newline from input lines and append
output lines.

`-n`

Places code within an input loop (as
`... end`).

`-0[ octal]`

Sets default record separator (`$/`) as
to `\0` if *`octal`* not specified.

`-p`

Places code within an input loop. Wr
iteration.

`-r lib`

Uses `require` to load *`lib`* as a library

`-s`

Interprets any arguments between the
and filename arguments fitting the pa
switch and defines the corresponding

$xxx.-S

Searches for a program using the env
PATH.

-T [level]

Sets the level for tainting checks (1 i
specified). Sets the $SAFE variable.

-v

Displays version and enables verbos
to --verbose).

-w

Enables verbose mode. If programfi
reads from STDIN.

-x [*dir*]

Strips text before `#!ruby` line. Chang
before executing if *`dir`* is specified.

`-X` *`dir`*

Changes directory before executing (

`-y`

Enables parser debug mode (equival

`--copyright`

Displays copyright notice.

`--debug`

Enables debug mode (equivalent to `-`

`--help`

Displays an overview of command-li
(equivalent to `-h`).

`--version`

Displays version.

`--verbose`

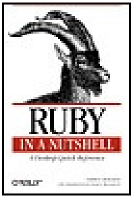
Enables verbose mode (equivalent to
`$VERBOSE` to `true`.

`--yydebug`

Enables parser debug mode (equival

Single character command-line
combined. The following two line
meaning:

```
                ruby -
ne 'print if /Ruby/' /usr/sh
                ruby -n -
e 'print if /Ruby/' /usr/sh
```



Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 2. Language Basics

2.2 Environment Variables

In addition to using arguments and options on the command line, the Ruby interpreter uses the following environment variables to control its behavior. The `ENV` object contains a list of current environment variables.

`DLN_LIBRARY_PATH`

Search path for dynamically loaded modules.

`HOME`

Directory moved to when no argument is passed to `Dir::chdir`. Also used by `File::expand_path` to expand "~".

`LOGDIR`

Directory moved to when no arguments are passed to

`Dir::chdir` and environment variable `HOME` isn't set.

`PATH`

Search path for executing subprocesses and searching for Ruby programs with the `-S` option. Separate each path with a colon (semicolon in DOS and Windows).

`RUBYLIB`

Search path for libraries. Separate each path with a colon (semicolon in DOS and Windows).

`RUBYLIB_PREFIX`

Used to modify the RUBYLIB

search path by replacing prefix of library *`path1`* with *`path2`* using the format *`path1`*;*`path2`* or *`path1path2`*. For example, if `RUBYLIB` is:

```
/usr/local/lib/ruby/site_ruby
```

and `RUBYLIB_PREFIX` is:

```
/usr/local/lib/ruby;f:/ruby
```

Ruby searches `f:/ruby/site_ruby`. Works only with DOS, Windows, and OS/2 versions.

`RUBYOPT`

Command-line options passed to Ruby interpreter. Ignored in taint mode (where `$SAFE` is greater than

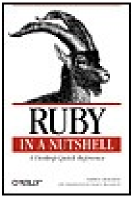
0).

`RUBYPATH`

With `-S` option, search path for Ruby programs. Takes precedence over `PATH`. Ignored in taint mode (where `$SAFE` is greater than 0).

`RUBYSHELL`

Specifies shell for spawned processes. If not set, `SHELL` or `COMSPEC` are checked.



Chapter 2. Language Basics

2.3 Lexical Conventions

Ruby programs are composed of elements
already familiar to most programmers:
lines, whitespace, comments, identifiers,

reserved words, literals, etc. Particularly f
those programmers coming from other
scripting languages such as Perl, Python o
tcl, you'll find Ruby's conventions familia
or at least straightforward enough not to
cause much trouble.

2.3.1 Whitespace

We'll leave the thorny questions like "Hov
much whitespace makes code more readal
and how much is distracting?" for anothe
day. If you haven't already caught onto thi
theme, the Ruby interpreter will do pretty
much what you expect with respect to
whitespace in your code.

Whitespace characters such as spaces and
tabs are generally ignored in Ruby code,

except when they appear in strings. Sometimes, however, they are used to interpret ambiguous statements. Interpretations of this sort produce warnir when the `-w` option is enabled.

`a + b`

> Interpreted as `a+b` (`a` is a local variable)

`a  +b`

> Interpreted as `a(+b)` (`a`, in this case, a method call)

2.3.2 Line Endings

Ruby interprets semicolons and newline characters as the ending of a statement.

However, if Ruby encounters operators, such as +, -, or backslash at the end of a line, they indicate the continuation of a statement.

2.3.3 Comments

Comments are lines of annotation within Ruby code that are ignored at runtime. Comments extend from # to the end of the line.

```
# This is a comment.
```

Ruby code can contain embedded documents too. Embedded documents extend from a line beginning with =begin the next line beginning with =end. =begin and =end must come at the beginning of a line.

```
=begin
This is an embedded document.
=end
```

2.3.4 Identifiers

Identifiers are names of variables, constar
and methods. Ruby distinguishes betweer
identifiers consisting of uppercase
characters and those of lowercase
characters. Identifier names may consist c
alphanumeric characters and the undersco
character `( _ )`. You can distinguish a
variable's type by the initial character of i
identifier.

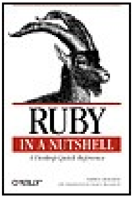
2.3.5 Reserved Words

The following list shows the reserved wo
in Ruby:

BEGIN	do	next	then
END	else	nil	true
alias	elsif	not	undef
and	end	or	unless
begin	ensure	redo	until
break	false	rescue	when
case	for	retry	while
class	if	return	yield

def	in	self	__FILE__
defined?	module	super	__LINE__

These reserved words may not be used as constant or local variable names. They ca however, be used as method names if a receiver is specified.



Chapter 2. Language Basics

2.4 Literals

I've often wondered why we programmer
I'm waiting for the day when a language c
"figuratives." In the interim, the rules Rub

intuitive, as you'll see the following sectio

2.4.1 Numbers

Strings and numbers are the bread and but
support for both integers and floating-poi
`Bignum`, and `Float`.

2.4.1.1 Integers

Integers are instances of class `Fixnum` or `B`

```
123                     # decimal
1_234                   # decimal wi
0377                    # octal
0xff                    # hexadecima
0b1011                  # binary
?a                      # character
12345678901234567890    # Bignum:  a
```

2.4.1.2 Floating-point numbers

Floating-point numbers are instances of cl

```
123.4                    # floating
1.0e6                    # scientif
4E20                     # dot not 
4e+20                    # sign bef
```

2.4.2 Strings

A string is an array of bytes (octets) and a

`"abc"`

Double-quoted strings allow substitu

`'abc'`

Single-quoted strings don't allow sub

notation only for `\\` and `\'`.

2.4.2.1 String concatenation

Adjacent strings are concatenated at the sa
program.

```
"foo" "bar"              # means "foob
```

2.4.2.2 Expression substitution

`#$var` and `#@var` are abbreviated forms of
value of expression in `#{...}` into a string

2.4.2.3 Backslash notation

In double-quoted strings, regular expressi
backslash notation can be represent unpri
Table 2-1.

Table 2-1. Backslash notat

Sequence	Character repres
`\n`	Newline (0x0a)
`\r`	Carriage return (0x0d)
`\f`	Formfeed (0x0c)
`\b`	Backspace (0x08)
`\a`	Bell (0x07)

`\e`	Escape (0x1b)
`\s`	Space (0x20)
`\nnn`	Octal notation (*n* being 0-7)
`\xnn`	Hexadecimal notation (*n* bein
`\cx, \C-x`	Control-*x*
`\M-x`	Meta-x (*c* | 0x80)
`\M-\C-x`	Meta-Control-*x*
`\x`	Character *x*

`` `command` ``

Converts command output to a string
backslash notation

2.4.2.4 General delimited strings

The delimiter `!` in expressions like this: `%(`
character. If the delimiter is any of the fol
becomes the corresponding closing delim
pairs.

`%!foo!`

`%Q!foo!`

Equivalent to double quoted string `"`

`%q!foo!`

Equivalent to single quoted string `'f`

```
%x!foo!
```

Equivalent to `` `foo` `` command outpu

2.4.2.5 here documents

Builds strings from multiple lines. Conter
the line that starts with the delimiter.

```
<<FOO

FOO
```

Using quoted delimiters after <<, you can
used for `String` literals. If a minus sign a
delimiter, you can indent the delimiter, as

```
puts <<FOO                    # String in
     hello world
     FOO
```

```
    puts <<"FOO"      # String in
    hello world
    FOO

    puts <<'FOO'       # String in
    hello world
    FOO

    puts <<`FOO`       # String in
    hello world
    FOO

    puts <<-FOO        # Delimiter
        hello world
        FOO
```

2.4.3 Symbols

A symbol is an object corresponding to an

```
:foo                    # symbol for
:$foo                   # symbol for
```

2.4.4 Arrays

An array is a container class that holds a c
an integer. Any kind of object may be stor
array can store a heterogeneous mix of ob
add elements. Arrays can be created using
array expression is a series of values betw

```
[]
```

An empty array (with no elements)

```
[1, 2, 3]
```

An array of three elements

```
[1, [2, 3]]
```

A nested array

2.4.4.1 General delimited string array

You can construct arrays of strings using t
whitespace characters and closing parenth
following notation:

```
%w(foo bar baz)        # ["foo", "b
```

2.4.5 Hashes

A hash is a collection of key-value pairs o
arbitrary types of objects.

A hash expression is a series of `key=>val`

```
{key1 => val1, key2 => val2}
```

2.4.6 Regular Expressions

Regular expressions are a minilanguage u
strings. A regular expression literal is a pa
arbitrary delimiters followed by `%r`:

```
/pattern/
/pattern/im          # option can b
%r!/usr/local!       # general deli
```

Regular expressions have their own powe
topic, see O'Reilly's *Mastering Regular E*

2.4.6.1 Regular-expression modifiers

Regular expression literals may include a
various aspects of matching. The modifie
slash character, as shown previously and r
these characters:

i

Case-insensitive

o

Substitutes only once

x

Ignores whitespace and allows comn

m

Matches multiple lines, recognizing

2.4.6.2 Regular-expression patterns

Except for control characters, (`+ ? . * ^ $`
match themselves. You can escape a contr

a backslash.

Regular characters that express repetition
strings, but when you follow such charact
invoke a nongreedy match that finishes at
`*`, etc.) followed by `?` (i.e., `+?`, `*?`, etc.).

`^`

Matches beginning of line.

`$`

Matches end of line.

`.`

Matches any single character except
to match newline as well.

`[...]`

Matches any single character in brac

`[^...]`

Matches any single character not in b

re`*`

Matches 0 or more occurrences of pr

re`+`

Matches 1 or more occurrences of pr

re`?`

Matches 0 or 1 occurrence of preced

re`{ `*n*`}`

Matches exactly *n* number of occurre

`re{ n,}`

Matches *n* or more occurrences of pr

`re{ n, m}`

Matches at least *n* and at most *m* occu

`a| b`

Matches either *a* or *b*.

`( re)`

Groups regular expressions and reme

`(?imx)`

Temporarily toggles on `i`, `m`, or `x` opt
If in parentheses, only that area is aff

`(?-imx)`

Temporarily toggles off `i`, `m`, or `x` opt
If in parentheses, only that area is aff

`(?: re)`

Groups regular expressions without r

`(?imx: re)`

Temporarily toggles on `i`, `m`, or `x` opt

`(?-imx: re)`

Temporarily toggles off `i`, `m`, or `x` opt

`(?#...)`

Comment.

`(?= re)`

Specifies position using a pattern. Do

`(?! re)`

Specifies position using pattern nega

`(?> re)`

Matches independent pattern withou

`\w`

Matches word characters.

`\W`

Matches nonword characters.

`\s`

Matches whitespace. Equivalent to [

`\S`

Matches nonwhitespace.

`\d`

Matches digits. Equivalent to [0-9].

`\D`

Matches nondigits.

`\A`

Matches beginning of string.

`\Z`

Matches end of string. If a newline e

newline.

`\z`

Matches end of string.

`\G`

Matches point where last match finis

`\b`

Matches word boundaries when outs
(0x08) when inside brackets.

`\B`

Matches nonword boundaries.

`\n`, `\t`, etc.

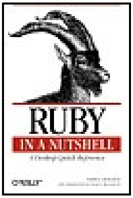
Matches newlines, carriage returns, t

`\1...\9`

Matches *n*th grouped subexpression.

`\10...`

Matches *n*th grouped subexpression
refers to the octal representation of a



Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 2. Language Basics

2.5 Variables

There are five types of variables in Ruby:
class, locals and constants. As you might
variables are accessible globally to the pro

variables belong to an object, class variab
constants are, well... constant. Ruby uses
to differentiate between the different kind
a glance, you can tell what kind of variabl

Global Variables

```
$foo
```

Global variables begin with `$`. Uninitializ
variables have the value `nil` (and produce
the `-w` option). Some global variables hav
behavior. See Section 3.1 in Chapter 3.

Instance Variables

```
@foo
```

Instance variables begin with `@`. Uninitiali
variables have the value `nil` (and produce
the `-w` option).

Class Variables

```
@@foo
```

Class variables begin with `@@` and must be
they can be used in method definitions. R
uninitialized class variable produces an er
variables are shared among descendants o
module in which the class variables are de
class variables produce warnings with the

Local Variables

```
foo
```

Local variables begin with a lowercase let
scope of a local variable ranges from `clas`
or `do` to the corresponding `end` or from a b
brace to its close brace `{}`. The scope intr
allows it to reference local variables outsi
scopes introduced by others don't. When a
local variable is referenced, it is interprete
method that has no arguments.

Constants

```
Foo
```

Constants begin with an uppercase letter.
within a class or module can be accessed
class or module, and those defined outsid
module can be accessed globally. Consta
defined within methods. Referencing an u
constant produces an error. Making an ass
constant that is already initialized produc
an error. You may feel it contradicts the n
but remember, this is listed under "variab

Pseudo-Variables

In addition to the variables discussed, the
pseudo-variables. Pseudo-variables have
local variables but behave like constants.
not be made to pseudo-variables.

`self`

The receiver object of the current me

`true`

Value representing `true`

`false`

Value representing `false`

`nil`

Value representing "undefined"; inter
in conditionals

`__FILE__`

The name of the current source file

`__LINE__`

The current line number in the sourc

Assignment

```
target = expr
```

The following elements may assign target

Global variables

Assignment to global variables alters
isn't recommended to use (or abuse)
They make programs cryptic.

Local variables

Assignment to uninitialized local var
as variable declaration. The variables

until the end of the current scope is r
lifetime of local variables is determi
parses the program.

Constants

Assignment to constants may not app
method body. In Ruby, re-assignmen
prohibited, but it does raise a warnin

Attributes

Attributes take the following form:

```
expr.attr
```

Assignment to attributes calls the `at`
result of `expr`.

Elements

Elements take the following form:

```
expr[arg...]
```

Assignment to elements calls the `[]=`
result of `expr`.

Parallel Assignment

```
target[, target...][, *target] = ex
[, *expr]
```

Targets on the left side receive assignmen
corresponding expressions on the right sic
side target is preceded by `*`, all remaining
are assigned to the target as an array. If th
expression is preceded by `*`, the array elen
expression are expanded in place before a

If there is no corresponding expression, n
the target. If there is no corresponding tar
right-side expression is just ignored.

Abbreviated Assignment

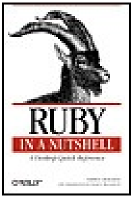
```
target op= expr
```

This is the abbreviated form of:

```
target = target op expr
```

The following operators can be used for a
assignment:

```
+=   -
=   *=   /=   %=   **=   <<=   >>=   &=   |
```



Chapter 2. Language Basics

2.6 Operators

Ruby supports a rich set of operators, as y
language. However, in keeping with Ruby
operators are in fact method calls. This fle

the semantics of these operators wherever

2.6.1 Operator Expressions

Most operators are actually method calls. interpreted as `a.+(b)`, where the `+` method variable `a` is called with `b` as its argument.

For each operator (`+ - * / % ** & | ^ << >>` form of abbreviated assignment operator (

Here are the operators shown in order of p

```
::
[]
**
+(unary) -(unary) ! ~
* / %
+ -
<< >>
&
```

`| ^`
`> >= < <=`
`<=> == === != =~ !~`
`&&`
`||`
`.. ...`
`?:`
`=` (and abbreviated assignment operators s
`not`
`and or`

2.6.1.1 Nonmethod operators

The following operators aren't methods ar

`...`
`!`
`not`
`&&`
`and`
`||`

`or`
`::`
`=`
`+=`, `-=`, (and other abbreviated assignment
`? :` (ternary operator)

2.6.1.2 Range operators

Range operators function differently depe
appear in conditionals, `if` expressions, an

In conditionals, they return `true` from the
left operand is `true:`

`expr1` `..` *`expr2`*

Evaluates *`expr2`* immediately after

`expr1` `...` *`expr2`*

Evaluates `expr2` on the iteration afte

In other contexts, they create a range obje

`expr1 .. expr2`

Includes both expressions (`expr1 <=`

`expr1 ... expr2`

Doesn't include the last expression (`e`

2.6.1.3 Logical operators

If the value of the entire expression can be
the left operand alone, the right operand is

`&& and`

Returns `true` if both operands are `tr`
returns the value of the left operand,
the right operand.

`|| or`

Returns `true` if either operand is `tru`
returns the value of the left operand,
the right operand.

The operators `and` and `or` have extre

2.6.1.4 Ternary operator

Ternary `?:` is the conditional operator. It's
statement.

```
a ? b : c
```

If `a` is `true`, evaluates `b`, otherwise ev
spaces before and after the operators
for the method `a?` and the second par

2.6.1.5 defined? operator

`defined?` is a special operator that takes t
determine whether or not the passed expre
description string of the expression, or `ni`

`defined?` *`variable`*

True if *`variable`* is initialized

```
foo = 42
defined? foo        # => "local-v
defined? $_         # => "global-
defined? bar        # => nil (und
```

`defined?` *`method_call`*

True if a method is defined (also che

```
defined? puts        # => "methoc
defined? puts(bar) # => nil (ba
defined? unpack      # => nil (nc
```

`defined? super`

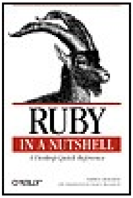
True if a method exists that can be ca

```
defined? super       # => "super'
defined? super       # => nil
```

`defined? yield`

True if a code block has been passed

```
defined? yield    # => "yield" (
defined? yield    # => nil     (
```



Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 2. Language Basics

2.7 Methods

Methods are the workhorses of Ruby; all
in methods on objects (and classes). In R
operation (e.g. "dump") and the code that

operation.

Strictly speaking, Ruby has no functions,
with any object. (In C++, this is what you
All code in Ruby is a method of some obj
of having some methods appear and work
even though behind the scenes they're stil

Normal Method Calls

```
obj.method([expr...[, *expr[, &expr
obj.method [expr...[, *expr[, &expr
obj::method([expr...[, *expr[, &exp
obj::method [expr...[, *expr[, &exp

method([expr...[, *expr[, &expr]]])
method [expr...[,
*expr[, &expr]]]
```

Calls a method. May take as arguments ar
and `&expr`. The last expression argument
braces. `*expr` expands the array value of t
method. `&expr` passes the `Proc` object valu
block. If it isn't ambiguous, arguments nee
`.` or `::` may be used to separate the object
Ruby code to use `::` as the separator for c

Calls a method of `self`. This is the only fo
called.

Within modules, module methods and priv
name and definition are referred to by the
kind of method group can be called in eith

```
Math.sin(1.0)
```

or:

```
include Math
```

```
sin(1.0)
```

You can append ! or ? to the nam
appended to a method that requi
same name without `!`. A questio
determines the state of a

Attempting to call a method wit
parentheses in a context in which
results in the method call being
variable, not a

2.7.1 Specifying Blocks with M

Methods may be called with blocks of coo
within the method.

```
method_call {[|[variable[, variable
method_call do [|[variable[, variab
```

Calls a method with blocks specified. The value is passed from the method to the blo block's argument) enclosed between `||`.

A block introduces its own scope for new appear first in the block are local to that b can refer local variables of outer scope; o `class`, `module` and `def` statement can't ref

The form `{...}` has a higher precedence t

```
identifier1 identifier2 {|varizable
```

actually means:

```
identifier1(identifier2 {|variable|
```

On the other hand:

```
identifier1 identifier2 do |variabl
```

actually means:

```
identifier1(identifier2) do |variab
```

def Statement

```
def method([arg..., arg=default...,
code
[rescue [exception_class[, exceptic
code]...
[else
code]
[ensure
code]
end
```

Defines a method. Arguments may includ

`arg`

Mandatory argument.

`arg= default`

Optional argument. If argument isn't
method, the `default` is assigned to `a`

`* arg`

If there are remaining actual argumen
optional arguments, they are assigned
remainder, empty array is assigned to

`& arg`

If the method is invoked with a block
assigned to `arg`. Otherwise, `nil` is as

Operators can also be specified as method

```
def +(other)
  return self.value + other.value
end
```

You should specify `+@` or `-@` for a single p
`begin` block, a method definition may end

2.7.2 Singleton Methods

In Ruby, methods can be defined that are
Such methods are called singleton method
`def` statements while specifying a receiver

Defines a singleton method associated wi
receiver. The *`receiver`* may be a constant
parentheses.

def Statement for Singleton Me

```
def
receiver.method([arg...,arg=default
code
[rescue [exception_class[, exceptio
code]...
[else
code]
[ensure
code]
end
```

`A period .` after *`receiver`* can
work the same way, but `::`

A restriction in the implementation of Rub
methods associated with instances of the

```
a = "foo"
def a.foo
  printf "%s(%d)\n", self, self.siz
end
a.foo     # "foo" is available for
```

2.7.3 Method Operations

Not only can you define new methods to
aliases to the methods and even remove th

alias Statement

```
alias new old
```

Creates an alias *`new`* for an existing metho
by *`old`*. This functionality is also available
making an alias of a method, it refers the

```
def foo
  puts "foo!"
  end
alias foo_orig foo
def foo
  puts "new foo!"
end
foo                    # => "new foo!"
foo_orig               # => "foo!"
```

undef Statement

`undef` *`method`*`...`

Makes method defined in the current class
defined in the superclass. This functionali
`Module#undef_method`.

```
class Foo
def foo
```

```
end
end
class Bar<Foo
# Bar inherits "foo"
undef foo
end
b = Bar.new
b.foo      # error!
```

2.7.4 Other Method-Related S

The following statements are to be used w
statement executes a block that is passed t
executes the overridden method of the sup

yield Statement

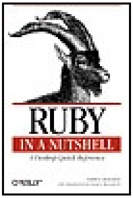
```
yield([expr...])
yield [expr...]
```

Executes the block passed to the method.
assigned to the block's arguments. Paralle
expressions are passed. The output of the
last expression in the block, is returned.

super Statement

```
super
super([expr...])
superexpr...
```

`super` executes the method of the same na
arguments nor parentheses are specified, t
directly to the superclass method. In other
no arguments to the superclass method, ha
`super`, where neither arguments nor paren



Chapter 2. Language Basics

2.8 Control Structures

Ruby offers control structures that are pre
also has a few unique ones.

if Statement

```
if conditional [then]
code
[elsif conditional [then]
code]...
[else
code]
end
```

Executes *`code`* if the *`conditional`* is `true`
`false` or `nil`. If the *`conditional`* isn't `tru`
executed. An `if` expression's *`conditiona`*
word `then`, a newline, or a semicolon. Th
statement modifier.

```
code if conditional
```

Executes `code` if `conditional` is `true`.

unless Statement

```
unless conditional [then]
code
[else
code]
end
```

Executes code if `conditional` is `false`. If
specified in the `else` clause is executed. L
statement modifier.

```
code unless conditional
```

Executes `code` unless `conditional` is `tru`

case Statement

```
case expression
[when expression[, expression...] |
code]...
[else
code]
end
```

Compares the *expression* specified by `ca`
`===` operator and executes the *code* of the
expression specified by the `when` clause i
`when` clauses match, `case` executes the *co*
expression is separated from *code* by the
semicolon.

while Statement

```
while conditional [do]
code
end
```

Executes *code* while *conditional* is true
separated from *code* by the reserved word
reserved word `while` can be used as stater

```
code while conditional
```

Executes*code* while *conditional* is `true`

```
begin code end while conditional
```

If a `while` modifier follows a `begin` stater
code is executed once before *conditiona*

until Statement

```
until conditional [do]
code
end
```

```
code untilconditional

begin
code
end until conditional
```

Executes *code* while *conditional* is fals
separated from code by the reserved word
`while`, `until` can be used as statement mo

Executes*code* while *conditional* is `fals`

If an `until` modifier follows a `begin` state
code is executed once before *conditiona*

for Statement

```
for variable[, variable...] in expr
code
```

```
end
```

Executes *`code`* once for each element in *`e`*
to:

```
expression.each do |variable[, vari
```

except that a `for` loop doesn't create a nev
`expression` is separated from *`code`* by the
semicolon.

break Statement

```
break
```

Terminates a `while`/`until` loop. Terminat
called within the block (with the method r

next Statement

```
next
```

Jumps to the point immediately before the
Terminates execution of a block if called
returning `nil`).

redo Statement

```
redo
```

Jumps to the point immediately after the e
Restarts `yield` or `call` if called within a b

retry Statement

```
retry
```

Repeats a call to a method with an associa
`rescue` clause.

Jumps to the top of a `begin`/`end` block if c

begin Statement

```
begin
code
[rescue [exception_class[, exceptio
code]...
[else
code]
[ensure
code]
end
```

The `begin` statement encloses *`code`* and pe
together with the `rescue` and `ensure` clau

When a `rescue` clause is specified, except
specified are caught, and the *`code`* is execu
enclosure is the value of its last line of `co`
the program is treated as if the `StandardE`
`variable` is specified, the exception objec
`exception_class` is separated from the re
`then`, a newline, or a semicolon. If no exc
executed if specified. If an `ensure` clause
before the `begin`/`end` block exits, even if t
before it can be completed.

rescue Statement

```
code rescue expression
```

Evaluates the `expression` if an exception
during the execution of the `code`. This is e

```
begin
  code
rescue StandardError
  expression
end
```

raise method

```
raise exception_class, message
raise exception_object
raisemessage
raise
```

Raises an exception. Assumes `RuntimeEr`
Calling `raise` without arguments in a `res`
so outside a rescue clause raises a messag

BEGIN Statement

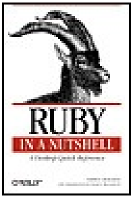
```
BEGIN {
code
}
```

Declares *code* to be called before the prog

END Statement

```
END {
code
}
```

Declares *code* to be called at the end of th



Chapter 2. Language Basics

2.9 Object-Oriented Programm

Phew, seems like a long time since I intro
oriented scripting language," eh? But now
to get the nitty-gritty details on how Ruby

After you've mastered a few concepts and
objects, you may never want to go back to
beware!

2.9.1 Classes and Instances

All Ruby data consists of objects that are
class itself is an object that is an instance
rule, new instances are created using the `r`
are some exceptions (such as the `Fixnum` c

```
a = Array::new
s = String::new
o = Object::new
```

class Statement

```
class class_name [< superclass]
code
```

```
end
```

Defines a class. A *`class_name`* must be a c
assigned to that constant. If a class of the
class and *`superclass`* must match, or the
specified, in order for the features of the n
to the existing class. `class` statements intr
variables.

2.9.2 Methods

Class methods are defined with the `def` st
adds a method to the innermost class or m
the `def` statement. A `def` statement outsid
(at the top level) adds a method to the `Obj`
method that can be referenced anywhere i

When a method is called, Ruby searches f
the following order:

1. **Among the methods defined in tha methods).**

- Among the methods defined by that obj
- Among the methods of the modules inc
- Among the methods of the superclass.
- Among the methods of the modules inc
- Repeats Steps 4 and 5 until the top-leve

2.9.3 Singleton Classes

Attribute definitions for a specific object definition construction. Uses for this form definition and a collection of singleton me

```
class << object
code

end
```

Creates a virtual class for a specific objec
(methods and constants) of the class using
construction.

2.9.4 Modules

A module is similar to a class except that
be instantiated. The `Module` class is the su

module Statement

```
module module_name
  code
```

```
end
```

A `module` statement defines a module. *moc*
The defined module is assigned to that co
name already exists, the features of the ne
to the existing module. `module` statements
local variables.

2.9.5 Mix-ins

Properties (methods and constants) define
a class or another module with the `incluc`
added to a specific object using the extenc
in [Section 3.4.9](#), and the `Object#extend` i

2.9.6 Method Visibility

There are three types of method visibility:

`Public`

Callable from anywhere

`Protected`

Callable only from instances of the s

`Private`

Callable only in functional form (i.e.
specified)

Method visibility is defined using the `pub`
methods in classes and modules.

`public(` *`[`*`symbol` *`...])`*

Makes the method specified by `symb`
have been previously defined. If no a
visibility of all subsequently defined

is made public.

`protected([ *symbol*...])`

Makes the method specified by `symb`
have been previously defined. If no a
visibility of all subsequently defined
is made protected.

`private([ *symbol*...])`

Makes the method specified by `symb`
have been previously defined. If no a
visibility of all subsequently defined
is made private.

2.9.7 Object Initialization

Objects are created using the `new` method

`new` object is created by the `new` method, t
is called with the arguments of the `new` me
associated with the `new` method are also p
For consistency, you should initialize obje
`initialize` method, rather than the `new` m
methods named `initialize` is automatica

2.9.8 Attributes

Attributes are methods that can be referen
as if they were variables. For example, the
`egid` can be manipulated in the following

```
Process.egid       # Reference
Process.egid=id    # Assignment
```

These are actually two methods, one that t
with a name ending with `=` that takes one a
such attributes are referred to as *accessor*

2.9.9 Hooks

Ruby notifies you when a certain event ha

Table 2-2. Events and their

Event	Hook
Defining an instance method	`method_add`
Defining a singleton method	`singleton_`
Make subclass	`inherited`

These methods are called *hooks*. Ruby cal
specific event occurs (at runtime). The de

is to do nothing. You have to override the
something on a certain event:

```
class Foo
  def Foo::inherited(sub)
    printf "you made subclass of Fo
  end
end
class Bar<Foo  # prints "you made s
end
```

There are other types of hook methods us
are called by `include` and `extend` to do th
in Table 2-3. You can use these as hooks,
when you override them.

Table 2-3. Mix-In ho

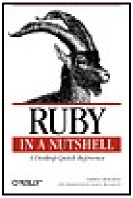
Event	Hook method	

Mixing in a module	`append_features`	M
Extending a object	`extend_object`	M

Ruby 1.7 and later provide more hooks. S
information on future versions.

Previous Next

Top



Chapter 2. Language Basics

2.10 Security

Ruby is portable and can easily use code o
This property gives you tremendous powe
introduces a commensurate burden: how d

without possibly causing damage?

Part of the answer lies in Ruby's security s
"lock down" the Ruby environment when
suspect. Ruby calls such data and code *ta*
mechanisms that allow you to decide how
"dangerous" data or code can be used insi

2.10.1 Restricted Execution

Ruby can execute programs with *security*
global variable `$SAFE` determines the leve
default safe level is 0, unless specified ex
option `-T`, or the Ruby script is run `setui`

`$SAFE` can be altered by assignment, but it
value of it:

```
$SAFE=1                    # upgrade th
$SAFE=4                    #  upgrade t
```

```
$SAFE=0                     # SecurityEr
```

`$SAFE` is thread local; in other words, the v
may be changed without affecting the valu
feature, threads can be sandboxed for untr

```
Thread::start {             # starting "
  $SAFE = 4                 # for this t
  ...                       # untrusted
}
```

Level 0

Level 0 is the default safe level. No check
data.

Any externally supplied string from `IO`, en
`ARGV` is automatically flagged as tainted.

The environment variable `PATH` is an exce
and tainted only if any directory in it is wr

Level 1

In this level, potentially dangerous operat
forbidden. This is a suitable level for prog
input, such as CGI.

- Environment variables `RUBYLIB` and
  startup.
- Current directory (.) isn't included in
- The command-line options `-e`, `-i`, `-I`
  prohibited.

- Process termination if the environme
- Invoking methods and class methods
  `FileTest` for tainted arguments is pr
- Invoking `test`, `eval`, `require`, `load`,
  argument is prohibited.

Level 2

In this level, potentially dangerous operat
forbidden, in addition to all restrictions in
operations are prohibited:

```
Dir::chdir
Dir::chroot
Dir::mkdir
Dir::rmdir
```

`File::chown`
`File::chmod`
`File::umask`
`File::truncate`
`File#lstat`
`File#chmod`
`File#chown`
`File#truncate`
`File#flock`
`IO#ioctl`
`IO#fctrl`
Methods defined in the `FileTest` module
`Process::fork`
`Process::setpgid`
`Process::setsid`
`Process::setpriority`
`Process::egid=`
`Process::kill`
`load` from a world-writable directory
`syscall`
`exit!`
`trap`

Level 3

In this level, all newly created objects are
addition to all restrictions in Level 2.

- All objects are created tainted.
- `Object#untaint` is prohibited.
- `Proc` objects retain current safe level
  methods are invoked.

Level 4

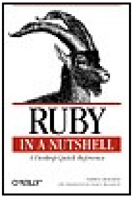
In this level, modification of global data i
restrictions in Level 3. `eval` is allowed ag
dangerous operations are blocked in this l

```
def safe_eval(str)
Thread::start {                # start s
  $SAFE = 4                    # upgrade
  eval(str)                    # eval in
}.value                        # retriev
end

eval('1 + 1')                  # => 2
eval('system "rm -rf /"') # Securit
```

The following operations are prohibited:

- `Object#taint`
- `autoload`, `load`, and `include`
- Modifying `Object` class

- Modifying untainted objects
- Modifying untainted classes or modu
- Retrieving meta information (e.g., va
- Manipulating instance variables
- Manipulating threads other than curr
- Accessing thread local data
- Terminating process (by `exit`, `abort`
- File input/output
- Modifying environment variables
- `srand`

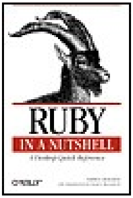

Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 3. Built-in Library Reference

We will now explore the core functionality that is built into the standard Ruby interpreter. You will find descriptions of more than 800 built-in methods in 42 classes and modules. Topics covered include predefined variables, predefined global constants, and built-in functions.

◀ Previous Next ▶

Top



Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 3. Built-in Library Reference

3.1 Predefined Variables

Ruby's predefined (built-in) variables affect the behavior of the entire program, so their use in libraries isn't recommended. The values in most predefined variables can be accessed by alternative means.

`$!`

> The last exception object raised. The exception object can also be accessed using `=>` in `rescue` clause.

`$@`

> The `stack backtrace` for the last exception raised. The `stack backtrace` information can retrieved by `Exception#backtrace` method of the last exception.

`$/`

The input record separator (newline by default). `gets`, `readline`, etc., take their input record separator as optional argument.

`$\`

The output record separator (`nil` by default).

`$,`

The output separator between the arguments to print and `Array#join` (`nil` by default). You can specify separator explicitly to `Array#join`.

`$;`

The default separator for `split`

(`nil` by default). You can specify separator explicitly for `String#split`.

`$.`

The number of the last line read from the current input file. Equivalent to `ARGF.lineno`.

`$<`

Synonym for `ARGF`.

`$>`

Synonym for `$defout`.

`$0`

The name of the current Ruby

program being executed.

`$$`

The `process.pid` of the current Ruby program being executed.

`$?`

The exit status of the last process terminated.

`$:`

Synonym for `$LOAD_PATH`.

`$DEBUG`

True if the `-d` or `--debug` command-line option is specified.

`$defout`

The destination output for `print` and `printf` (`$stdout` by default).

`$F`

The variable that receives the output from `split` when `-a` is specified. This variable is set if the `-a` command-line option is specified along with the `-p` or `-n` option.

`$FILENAME`

The name of the file currently being read from `ARGF`. Equivalent to `ARGF.filename`.

`$LOAD_PATH`

An array holding the directories to be searched when loading files with the load and require methods.

$SAFE

The security level. See Section 2.10.

0

No checks are performed on externally supplied (tainted) data. (default)

1

Potentially dangerous operations using tainted data are forbidden.

2

Potentially dangerous operations on processes and files are forbidden.

3

All newly created objects are considered tainted.

4

Modification of global data is forbidden.

`$stdin`

Standard input (`STDIN` by default).

`$stdout`

Standard output (`STDOUT` by default).

`$stderr`

Standard error (`STDERR` by default).

`$VERBOSE`

True if the `-v`, `-w`, or `--verbose` command-line option is specified.

`$-` *`x`*

The value of interpreter option `-`*`x`* (*`x`*=`0`, `a`, `d`, `F`, `i`, `K`, `l`, `p`, `v`).

The following are local variables:

`$_`

The last string read by `gets` or `readline` in the current scope.

`$~`

`MatchData` relating to the last match. `Regex#match` method returns the last match information.

The following variables hold values that change in accordance with the current value of `$~` and can't receive assignment:

`$`*`n`* `($1, $2, $3...)`

The string matched in the *n*th group of the last pattern match. Equivalent to *m*[*n*], where *m* is a `MatchData` object.

`$&`

The string matched in the last pattern match. Equivalent to `*m*[0]`, where *`m`* is a `MatchData` object.

``$` ``

The string preceding the match in the last pattern match. Equivalent to `*m*.pre_match`, where *`m`* is a `MatchData` object.

`$'`

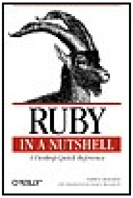
The string following the match in the last pattern match. Equivalent to `*m*.post_match`, where *`m`* is a `MatchData` object.

`$+`

The string corresponding to the last successfully matched group in the last pattern match.

Previous Next

Top



Chapter 3. Built-in Library Reference

3.2 Predefined Global Constants

`TRUE`, `FALSE`, and `NIL` are backward-compatible. It's preferable to use `true`, `false`, and `nil`.

`TRUE`

Synonym for `true`.

`FALSE`

Synonym for `false`.

`NIL`

Synonym for `nil`.

`ARGF`

An object providing access to virtual concatenation of files passed as command-line arguments or standard input if there are no command-line arguments. A synonym for `$<`.

`ARGV`

An array containing the command-line arguments passed to the program. A synonym for `$*`.

`DATA`

An input stream for reading the lines of code following the `__END__` directive. Not defined if `__END__` isn't present in code.

`ENV`

A hash-like object containing the program's environment variables. `ENV` can be handled as a hash.

`RUBY_PLATFORM`

A string indicating the platform of the Ruby interpreter, e.g., `i686-linux`.

`RUBY_RELEASE_DATE`

A string indicating the release date of the Ruby interpreter, e.g., `2001-09-19`.

`RUBY_VERSION`

A string indicating the version of the Ruby interpreter, e.g., `1.6.5`.

`STDERR`

Standard error output stream. Default value of `$stderr`.

`STDIN`

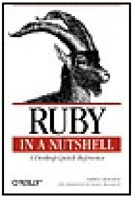
Standard input stream. Default value of `$stdin`.

`STDOUT`

Standard output stream. Default value of `$stdout`.

`TOPLEVEL_BINDING`

A `Binding` object at Ruby's top level.


Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 3. Built-in Library Reference

3.3 Built-in Functions

Since the `Kernel` module is included by `O`
everywhere in the Ruby program. They ca
form), therefore, they are often called *func*

`abort`

Terminates program. If an exception
is displayed.

`Array( obj)`

Returns *`obj`* after converting it to an

`at_exit {...}`

Registers a block for execution wher
(referenced in [Section 2.8](#)), but `END` s

`autoload( classname, file)`

Registers a class *`classname`* to be loa
`classname` may be a string or a symb

```
autoload :Foo, "foolib.rb".
```

`binding`

Returns the current variable and met
returned may be passed to the `eval` r

`block_given?`

Returns `true` if the method was calle

`callcc {|` *`c`*`|...}`

Passes a `Continuation` object *`c`* to th
be used for global exit or loop constr

```
def foo(c)
  puts "in foo"                #
  c.call                       # jum
  puts "out foo"               # thi
end
callcc{|c| foo(c)}             # pri
```

`caller([` *`n`*`])`

Returns the current execution stack i
file:*line*. If *n* is specified, returns s

`catch( tag) {...}`

Catches a nonlocal exit by a throw ca

```
def throwing(n)
  throw(:exit, n+2)
end

catch(:exit) {
  puts "before throwing"
  throwing(5)
  puts "after throwing"   # thi
} # returns 7
```

`chomp([ rs=$/])`

Returns the value of variable `$_` with
result back to `$_`. The value of the ne

```
    $_ = "foo\n"
    chomp                    # $_
    $_ = "foo"
    chomp                    # no
```

`chomp!([ rs=$/])`

Removes newline from `$_`, modifyin

`chop`

Returns the value of `$_` with its last c
result back to `$_`.

```
    $_ = "foo\n"
    chop                    # $_ =
    $_ = "foo"
    chop                    # $_ =
```

`chop!`

Removes the last character from `$_`,

`eval( str[, scope[, file, line]])`

Executes *`str`* as Ruby code. The bind
be specified with *`scope`*. The filenam
compiled may be specified using *`fil`*

`exec( cmd[, arg...])`

Replaces the current process by runn
are specified, the command is execut

```
exec "echo *"              # wild
exec "echo", "*"           # no wi
```

`exit([ result=0])`

Exits program, with *`result`* as the st

`exit!([ result=0])`

Kills the program bypassing exit han

`fail(...)`

See `raise(...)`

`Float( obj)`

Returns *obj* after converting it to a fl
`nil` is converted to 0.0; strings are co
rest are converted using *obj*`.to_f`.

```
Float(1)                      # =>
Float(nil)                    # =>
Float("1.5")                  # =>
Float("0xaa")                 # =>
```

`fork`

`fork {...}`

Creates a child process. `nil` is return

ID (integer) is returned in the parent
child process.

```
# traditional fork
if cpid = fork
  # parent process
else
  # child process
  exit!           # child proce
end

# fork using a block
fork {
  # child process
  # child terminates automatica
}
```

`format( fmt[, arg...])`

See `sprintf`.

`gets([ rs=$/])`

Reads the filename specified in the c
input. The record separator string ca

```
# easiest cat(1) imitation
while gets
  print $_              # gets u
end
```

`global_variables`

Returns an array of global variable n

`gsub( x, y)`

`gsub( x) {...}`

Replaces all strings matching *x* in `$_`
strings are replaced with the result of
to `$_`. See `String#gsub` in the next s

`gsub!( x, y)`

`gsub!( x) {...}`

Performs the same substitution as `gs`

`Integer( obj)`

Returns *`obj`* after converting it to an
directly; `nil` is converted to 0; string
prefix. The rest are converted using `c`

```
Integer(1.2)                  # => 1
Integer(1.9)                  # => 1
Integer(nil)                  # => 0
Integer("55")                 # => 55
Integer("0xaa")               # => 17
```

`lambda {| x|...}`

`proc {| x|...}`

`lambda`

`proc`

Converts a block into a `Proc` object.
with the calling method is converted

`load( file[, private=false])`

Loads a Ruby program from *`file`*. U
libraries. If *`private`* is `true`, the prog
thus protecting the namespace of the

`local_variables`

Returns an array of local variable na

`loop {...}`

Repeats a block of code.

`open( path[, mode="r"])`

`open( path[, mode="r"]) {| f|...}`

Opens a `file`. If a block is specified,
stream passed as an argument. The fi
exits. If `path` begins with a pipe `|`, th
the stream associated with that proce

`p( obj)`

Displays `obj` using its inspect metho

`print([ arg...])`

Prints `arg` to `$defout`. If no argumen

`printf( fmt[, arg...])`

Formats `arg` according to `fmt` using
formatting specifications, see `sprint`

`proc {| x|...}`

`proc`

See `lamda`.

`putc( c)`

Prints one character to the default ou

`puts([ str])`

Prints string to the default output (`$d`
newline, a newline is appended to th

```
puts "foo"                  # prints
puts "bar\n"                # prints
```

`raise(...)`

`fail(...)`

Raises an exception. Assumes Runti
Calling raise without arguments in
Doing so outside a rescue clause rais
obsolete name for raise. See "raise

rand([*max*=0])

Generates a pseudo-random number
If *max* is either not specified or is set
floating-point number greater than o
used to initialize pseudo-random stre

```
rand(10)          # => 8 (initiali
srand(42)         # initialize pse
rand              # => 0.744525000
rand              # => 0.342701478
srand(42)         # re-initialize 
rand              # => 0.744525000
rand              # => 0.342701478
```

readline([*rs*=$/])

Equivalent to `gets` except it raises ar

`readlines([ rs=$/])`

Returns an array of strings holding e
line arguments or the contents of sta

`require( lib)`

Loads the library (including extensio
`require` will not load the same libra
specified in *`lib`*, `require` tries to add

`scan( re)`

`scan( re) {|x|...}`

Equivalent to `$_.scan`. See `String#s`

`select( reads[, writes=nil[, excepts`

Checks for changes in the status of th
exceptions�hich are passed as array
that don't need checking. A three-ele
objects for which there were changes
timeout.

`set_trace_func( proc)`

Sets a handler for tracing. *proc* may
is used by the debugger and profiler.

`sleep([ sec])`

Suspends program execution for *sec*
is suspended forever.

```
sleep 1
sleep 1.5    # wait for 1.5 sec
```

`split([ sep[, max]])`

Equivalent to `$_.split`. See `String`

`sprintf( fmt[, arg...])`

`format( fmt[, arg...])`

Returns a string in which *`arg`* is form
specifications are essentially the sam
programming language. Conversion
specifier) in *`fmt`* are replaced by form

The following conversion specifiers,

`b`

Binary integer

`c`

Single character

`d,i`

Decimal integer

`e`

Exponential notation (e.g., 2.44e6)

`E`

Exponential notation (e.g., 2.44E6)

`f`

Floating-point number (e.g., 2.44)

`g`

use %e if exponent is less than -4, %

`G`

use %E if exponent is less than -4, %

o

Octal integer

s

String, or any object converted using

u

Unsigned decimal integer

x

Hexadecimal integer (e.g., 39ff)

X

Hexadecimal integer (e.g., 39FF)

Optional flags, width, and precision
field specifiers.

```
sprintf("%s\n", "abc")          #
sprintf("d=%d", 42)             #
sprintf("%04x", 255)            #
sprintf("%8s", "hello")         #
sprintf("%.2s", "hello")        #
```

`srand([ seed])`

Initializes an array of random numbe
performed using the time and other s
`rand`.

`String( obj)`

Returns *obj* after converting it to a st

```
String(1)                      #
String(Object)                 #
String("1.5")                  #
```

`syscall(` *`sys`*`[,` *`arg`*`...])`

Calls an operating system call functi
and meaning of *`sys`* is system-depen

`system(` *`cmd`*`[,` *`arg`*`...])`

Executes *`cmd`* as a call to the comma
the command is run directly with no
status is 0 (success).

```
system "echo *"                # w
system "echo", "*"             # r
```

`sub(` *`x`*`,` *`y`*`)`

`sub(` *`x`*`) {...}`

Replaces the first string matching *`x`* i

strings are replaced with the result of
to `$_`. See `String#sub` in Section 3.4

`sub!( x, y)`

`sub!( x) {...}`

Performs the same replacement as su

`test( test, f1[, f2])`

Performs one of the following file tes
to improve readability, you should us
`File::readable?`) rather than this fu
argument:

`?r`

Is *f1* readable by the effective `uid` of

`?w`

Is *`f1`* writable by the effective `uid` of

`?x`

Is *`f1`* executable by the effective `uid`

`?o`

Is *`f1`* owned by the effective `uid` of c

`?R`

Is *`f1`* readable by the real `uid` of calle

`?W`

Is *`f1`* writable by the real `uid` of calle

`?X`

Is *`f1`* executable by the real `uid` of ca

`?O`

Is *`f1`* owned by the real `uid` of caller'

`?e`

Does *`f1`* exist?

`?z`

Does *`f1`* have zero length?

`?s`

File size of *`f1`*(`nil` if `0`)

`?f`

Is *`f1`* a regular file?

`?d`

Is *`f1`* a directory?

`?l`

Is *`f1`* a symbolic link?

`?p`

Is *`f1`* a named pipe (`FIFO`)?

`?S`

Is *`f1`* a socket?

`?b`

Is *`f1`* a block device?

`?c`

Is `f1` a character device?

`?u`

Does `f1` have the `setuid` bit set?

`?g`

Does `f1` have the `setgid` bit set?

`?k`

Does `f1` have the sticky bit set?

`?M`

Last modification time for `f1`.

`?A`

Last access time for `f1`.

`?C`

Last `inode` change time for *`f1`*.

File tests with two arguments are as

`?=`

Are modification times of *`f1`* and *`f2`*

`?>`

Is the modification time of *`f1`* more

`?<`

Is the modification time of *`f1`* older

`?-`

Is *`f1`* a hard link to *`f2`* ?

`throw( tag[, value=nil])`

Jumps to the `catch` function waiting
return value to be used by `catch`.

`trace_var( var, cmd)`

`trace_var( var) {...}`

Sets tracing for a global variable. Th
may be a string or `Proc` object.

```
trace_var(:$foo) {|v|
  printf "$foo changed to %s\n'
}
$foo = 55                    # prints
```

`trap( sig, cmd)`

`trap( sig) {...}`

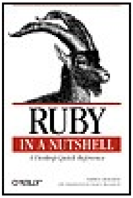
Sets a signal handler. *sig* may be a s
be omitted from signal name. Signal
is invoked just before process termin

cmd may be a string or `Proc` object. I
be ignored. If *cmd* is `DEFAULT` or `SIG_`
the operating system will be invoked

```
trap("USR1") {
  puts "receives SIGUSR1"
}
# prints message if SIGUSR1 is
```

`untrace_var(` *var*`[,` *cmd*`])`

Removes tracing for a global variabl
removed.



Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 3. Built-in Library Reference

3.4 Built-in Library

Ruby's built-in library provides you with
for your Ruby programs. There are classe
system services and abstractions (`IO`, `File`

etc.), and so on.

Using these basic building blocks, you ca
the next chapter, I lay out the Standard Li

3.4.1 Objects

Ruby couldn't lay claim to being an "obje
providing fundamental tools for OOP. Thi
`Object` class.

Object *Superclass of all classes*

`Object` is the parent class of all other clas
level, it becomes a private method of this
it were a function in other languages.

Included Modules

`Kernel`

Private Instance Methods

`initialize`

> Initializes an object. Any block and a
> passed directly to `initialize`. It's as
> subclasses for object initialization.

Kernel *Module containing built-*

`Kernel` is the module in which Ruby's bui
functions. Since it's included in `Object`, `K`

Private Instance Methods

Function-like methods are private method
fall into the same category, they are more
than function-like methods.

`remove_instance_variable( name)`

Removes instance variable specified

Instance Methods

`o == other`

Determines if the values are equal.

`o === other`

Comparison operator used by `case` s
membership).

`o =~ other`

Checks for pattern matches. The defi

`o.class`

`o.type`

Returns the class of the object *o*.

`o.clone`

Creates a copy of the object *o* (in as

`o.display([ out=$defout])`

Prints the object. The output is speci

`o.dup`

Creates a copy of the object (copying

`o.eql?( obj)`

Performs a hash comparison. In orde
both objects must have equal hash va

`o.equal?( obj )`

Returns `true` if the two objects are th

`o.extend( mod )`

Adds module features (instance meth

`o.freeze`

Freezes the object `o`, preventing furth

`o.frozen?`

Returns `true` if the object is frozen.

`o.hash`

Creates a hash value for the object *o*.
used as the key of a hash.

`o.id`

`o.__id__`

Returns the unique identifier value (i

`o.inspect`

Returns the human readable string re

`o.instance_eval( str)`

`o.instance_eval {...}`

Evaluates the string or block in the c
such as its instance variables, can be

`o.instance_of?( c)`

Returns `true` if *`o`* is an instance of th

`o``.instance_variables`

Returns an array of the object's insta

`o``.kind_of?(` *`mod`*`)`

`o``.is_a?(` *`mod`*`)`

Returns `true` if the object is an insta
`mod`.

`o``.method(` *`name`*`)`

Returns a `Method` object correspondi
corresponding method doesn't exist.

```
plus = 1.method(:+)
plus.call(2)     # => 3 (1+2)
```

`o.methods`

`o.public_methods`

Returns an array of the object's publi

`o.nil?`

Returns `true` if *o* is `nil`.

`o.private_methods`

Returns an array of the object's priva

`o.protected_methods`

Returns an array of the object's prote

`o.public_methods`

See `o.methods`.

`o.respond_to?( name)`

Returns `true` if method named `name`

`o.send( name[, arg...])`

`o.__send__( name[, arg...])`

Calls the method named `name` in the

`o.singleton_methods`

Returns an array of the object's singl

`o.taint`

Marks the object as tainted (unsafe).

`o.tainted?`

Returns `true` if the object `o` is tainte `o.to_a`

Returns an array representation of th
converted into an array, an array con

`o.to_s`

Returns a string representation of the

`o.type`

See `o.class`.

`o.untaint`

Removes the taint from the object.

3.4.2 Strings and Regular Exp

Death, taxes, and ... processing text. Yes,
programmer's life. In Ruby, I share your p

`MatchData` classes, Ruby provides sharp t
heart's content.

String *Character String class*

`String` is one of Ruby's basic datatypes, v
`String` can contain `\0`.

Included Module

`Enumerable`, `Comparable`

Class Method

`String::new(` *`str`*`)`

Creates a string.

Instance Methods

Methods of the `String` class ending in `!` r
modification took place, otherwise `nil`. M
the string.

`~ s`

Attempts to match pattern `s` against t

`s % arg`

An abbreviated form of `sprintf(s,`
using an array.

`s * n`

Returns a string consisting of `s` copie

`s + str`

Returns a string with `str` concatenat

`s << str`

Concatenates `str` to `s`.

`s =~ x`

Performs a regular expression match
object.

`s[ n]`

Returns the code of the character at p
offset from the end of the string.

`s[ n.. m]`

`s[ n, len]`

Returns a partial string.

```
"bar"[1..2]   # => "ar"
"bar"[1..-1]  # => "ar"
"bar"[-2..2]  # => "ar"
"bar"[-2..-1] # => "ar"
"bar"[1,2]    # => "ar"
"bar"[-1, 1]  # => "r"
```

`s[ n]= value`

Replaces the *n* th element in the strin
or string.

`s[ n.. m]= str`

`s[ n, len]= str`

Replaces a part of the string with *st*

`s.capitalize`

`s.capitalize!`

Returns a copy of `s` with the first cha
remainder to lowercase.

```
"fooBar".capitalize
```

`s.center( w)`

Returns a string of length `w` with `s` ce
if it has a length of less than `w`.

```
"foo".center(10)
"foo".center(2)
```

`s.chomp([ rs=$/])`

`s.chomp!([ rs=$/])`

Deletes the record separator from the
can be specified with `rs`.

```
    "foo\n".chomp
    "foo".chomp
    a = "foo\n"
    a.chomp!
    a
    a = "foo"
    a.chomp!
```

`s.chop`

`s.chop!`

Deletes the last character (byte) from

```
    "foo\n".chop
    "foo".chop
```

`s.concat( str)`

Concatenates *str* to the string.

`s.count( str...)`

Returns the number of occurrences o
of *str* if multiple *str* given) in *s*. *st*
c1-c2 means all characters between c

```
"123456789".count("2378")
"123456789".count("2-8", "^4-6'
```

`s.crypt( salt)`

Encrypts the string *s* using a one wa
for seed. See `crypt(3)`.

`s.delete( str...)`

`s.delete!( str...)`

Deletes the characters included in *st*
from *s*. Uses the same rules for build

```
"123456789".delete("2378")
"123456789".delete("2-8", "^4-6
```

`s.downcase`

`s.downcase!`

Replaces all uppercase characters in

`s.dump`

Returns version of string with all no
backslash notation.

`s.each([ rs=$/]) {|line| ...}`

`s.each_line([ rs=$/]) {|line| ...}`

Invokes the block for each line in *s*.
with *rs*.

`s.each_byte {| byte| ...}`

Invokes the block for each byte in *s*.

`s.empty?`

Returns `true` if *s* has a length of `0`.

`s.gsub( x, y)`

`s.gsub( x) {...}`

`s.gsub!( x, y)`

`s.gsub!( x) {...}`

Replaces all strings matching *x* in th
matched strings are replaced with the

```
"hello world".gsub(/[aeiou]/, '
"hello world".gsub(/[aeiou]/){|
```

`s.hex`

Treats *s* as a string of hexadecimal d

`s.include?( x[, pos=0])`

Returns `true` if *`str`* is present in `s`. *`x`*
code, a string, or a regular expressio
`pos`.

`s.index( x[, pos=0])`

Returns the index of *`x`* in string `s`, or
representing the character code, a str
started at offset *`pos`*.

`s.intern`

Returns the symbol corresponding to

`s.length`

See `s.size`.

`s.ljust( w)`

Returns a string of length *w* with *s* le
length of less than *w*.

`s.next`

`s.next!`

`s.succ`

`s.succ!`

Retrieves the next logical successor (

```
"aa".succ                    # => "a
"99".succ                    # => "1
"a9".succ                    # => "b
"Az".succ                    # => "B
"zz".succ                    # => "a
```

`s.oct`

Treats *s* as a string of octal digits anc
it's treated as a hexidecimal string; if
string.

`s.replace( str)`

Replaces contents of *s* with that of *s*

```
s = "abc"
s.replace("foobar")      # => '
s                        # => '
```

`s.reverse`

`s.reverse!`

Reverses the characters in the string

`s.rindex( x[, pos])`

Returns the index of last occurrence

string, or `nil` if *x* isn't present. *x* may
code, a string, or a pattern. If *pos* is g

`s.rjust( w)`

Returns a string of length *w* with *s* rig
a length of less than *w*.

```
"foo".rjust(10)              # =>
"foo".rjust(2)               # =>
```

`s.scan( re)`

`s.scan( re) {|x| ...}`

Attempts to match the regular expres
returns an array containing either arr
groups, or strings, which represent th
the expression. If a block is specified
in the array that would have been ret

```
    "foobarbaz".scan(/(ba)(.)/)  #
    "foobarbaz".scan(/(ba)(.)/) {|s
    # prints:
    #   ["ba", "r"]
    #   ["ba", "z"]
```

`s.size`

`s.length`

Returns the length of the string.

`s.slice( n)`

`s.slice( n.. m)`

`s.slice( n, len)`

Returns a partial string.

`s.slice!( n)`

`s.slice!( n.. m)`

`s.slice!( n, len)`

Deletes the partial string specified ar

```
a = "0123456789"
p a.slice!(1,2)          # "12"
p a                      # "0345
```

`s.split([ sep[, max]])`

Splits the contents of the string using
substrings as an array. If *sep* isn't spe
isn't `nil`) is used as the delimiter. If *m*
maximum of *max* elements.

```
"a b c".split              # => [
"a:b:c".split(/:/)         # => [
"a:b:c:::".split(/:/,4)    # => [
"a:b:c::".split(/:/,-1)    # => [
"abc".split(//)            # => [
s.squeeze([ str...])
s.squeeze!([ str...])
```

Reduces all running sequences of the
of `str` if multiple `str` given) to a sin
sequences of all characters are reduc

```
"112233445".squeeze        # =>
"112233445".squeeze("1-3") # =>
```

```
s.strip
s.strip!
```

Deletes leading and trailing whitespa

```
s.sub( x, y)
s.sub( x) {...}
s.sub!( x, y)
s.sub!( x) {...}
```

Replaces the first string matching *x* v
are replaced with the result of the blo

`s.succ`

See `s.next`.

`s.succ!`

See `s.next`.

`s.sum([ n=16])`

Returns an *n*-bit checksum of the stri

`s.swapcase`
`s.swapcase!`

Converts uppercase characters to low

`s.to_f`

Converts the string into a floating po
string. For more strict conversion, us

```
"1.5".to_f
"a".to_f
Float("a")
```

`s.to_i`

Converts the string into an integer. R
strict conversion, use `Integer( )`.

```
"1".to_i
"a".to_i
Integer("a")
```

`s.to_str`

Returns `s` itself. Every object that ha

`s.tr( str, r)`

`s.tr!( str, r)`

Replaces the characters in `str` with t

`s.tr_s`

`s.tr_s!`

After replacing characters as in `tr`, r
character in sections that were modif

```
"foo".tr_s("o", "f")
"foo".tr("o", "f").squeeze("f")
```

`s.succ`

See `s.next`.

`s.succ!`

See `s.next`.

`s.unpack( template)`

Unpacks `s` into arrays, decoding the
`Array#pack(template)`. `template` c
directives:

`a`

ASCII string

`A`

ASCII string (deletes trailing spaces

`b`

Bit string (ascending bit order)

`B`

Bit string (descending bit order)

c	Char
C	Unsigned char
d	Double (native format)
e	Little endian float (native format)
E	Little endian double (native format)
f	

Float (native format)

`g`

Big endian float (native format)

`G`

Big endian double (native format)

`h`

Hex string (low nibble first)

`H`

Hex string (high nibble first)

`i`

Integer

`I`	Unsigned integer
`l`	Long
`L`	Unsigned long
`m`	Base64 encoded string
`M`	Quoted printable string
`n`	

	Big-endian short (network byte orde
`N`	Big-endian long (network byte order
`p`	Pointer to a null-terminated string
`P`	Pointer to a structure (fixed-length st
`s`	Short
`S`	Unsigned short

`u`	UU-encoded string
`U`	UTF-8 string
`v`	Little-endian short (VAX byte order)
`V`	Little-endian long (VAX byte order)
`w`	BER-compressed integer
`x`	

Null byte

`X`

Backs up one byte

`Z`

ASCII string (deletes trailing null ch

`@`

Moves to absolute position

Each directive may be followed by a
elements to convert, or an asterisk, in
be converted. Directives may be sepa
followed by `_` use the native size for

```
"\001\002\003\004".unpack("CCCC
"\001\002\003\004".unpack("V")
```

```
    "\001\002\003\004".unpack("N")
```

`s.upcase`

`s.upcase!`

Replaces all lowercase characters in

`s.upto( max) {| x| ...}`

Returns *x* and continues to iterate to
method `s.next` is used to generate ea

```
"a".upto("ba") {|x|
  print x

}# prints a, b, c, ... z,aa, ..
```

Regexp *Regular expression clas*

`Regex` is object representation of regular e
language to describe patterns of strings. F
patterns," which is under Section 2.4.6 in

Class Methods

`Regexp::new(` *`str`*`[,` *`option`*`[,` *`code`*`]])`

`Regexp::compile(` *`str`*`[,` *`option`*`[,` *`code`*

Creates a `Regexp` object. *`option`* may
`Regexp::EXTENDED`, and `Regexp::MU`
multibyte character set code.

`Regexp::escape(` *`str`*`)`

`Regexp::quote(` *`str`*`)`

Returns a copy of *`str`* with all regula

Instance Methods

`~ r`

Performs a regular expression match
method is obsolete.

`r === str`

Synonym for `r =~ str` used in `case` s

`r =~ str`

Performs a regular expression match
or `nil` if the match failed.

`r.casefold?`

Returns `true` if the `Regexp` object is `r.match( str)`

Performs a regular expression match
a `MatchData` object, or `nil` if the mat

```
if m = /fo*b.r+/.match(str)
  puts m[0]          # print ma
end
```

`r.source`

Returns the original regular expressi

MatchData *Class for holding re data*

`MatchData` objects can be retrieved from t
`Regexp.match`.

Example

```
if m = pat.match(str)   # MatchData
  print "matched: ", m[0], "\n"
  print "pre: ", m.pre_match, "\n"
  print "post: ", m.post_match, "\n
end
```

Instance Methods

m[*n*]

Returns the match corresponding to t
`0`, the entire matched string is returne

m.begin(*n*)

Returns the offset of the start of the n
regular expression. If *n* is `0`, the offse
returned.

`m.end( n)`

Returns the offset of the end of the m
regular expression. If *n* is `0`, the offse
returned.

`m.length`

See `m.size`

`m.offset( n)`

Returns a two-element array containi
string corresponding to the nth grou

`m.post_match`

Returns the part of the original strin

`m.pre_match`

Returns the part of the original string

`m.size`

`m.length`

Returns the number of groups in the

`m.string`

Returns the original string used for th

`m.to_a`

Returns an array of the matches (i.e.,

3.4.3 Arrays and Hashes

One of the cornerstones of scripting langu
mechanisms for manipulating program da
provide intuitive and rich capabilities for

Array *Array class*

`Array` is a class for an ordered collection
object may be stored in an `Array`. `Arrays`

Included Module

`Enumerable`

Class Methods

`Array[` *`x...`*`]`

Creates an array.

`Array::new([` *`size`*`=0[,` *`fill`*`=nil]])`

Creates an array. Its *size* and initial

```
Array::new(4, "foo")        # =>
```

Instance Methods

Methods of the `Array` class ending in `!` m
modification took place, otherwise `nil`. M
the array.

arr & *array*

Returns an array of elements commo

```
[1,3,5]|[1,2,3]             # =>
[1,3,5]|[2,4,6]             # =>
```

arr | *array*

Returns an array combining element

```
    [1,3,5]|[2,4,6]            # =>
```

arr * *n*

If *n* is an integer, returns a copy of ar
n is a string, the equivalent of `arr.jo`

```
    [5] * 3                    # =>
    ["foo", "bar"] * "-"       # =>
```

arr + *array*

Returns a copy of `arr` with *array* co

arr - *array*

Returns a new array that is a copy of

```
    [1, 2, 3, 4] - [2, 3]      # =>
```

arr << item

Appends `item` to `arr`.

`arr[ n]`

References the nth element of `arr`. If
from the end of `arr`.

`arr[ n.. m]`

`arr[ n, len]`

Returns a partial string.

`arr[ n]= item`

`arr[r.. m]= array`

`arr[r, len]= array`

Assigns item or `arr` to the specified

```
    arr = [0, 1, 2, 3, 4, 5]
    arr[0..2] = ["a", "b"]     # ar
    arr[1, 0] = ["c"]          # ar
```

`arr.assoc( key)`

Searches through an array of arrays, element matching `key`.

```
    a = [[1,2],[2,4],[3,6]]
    a.assoc(2)                  # =
```

`arr.at( n)`

Returns the nth element of `arr`.

`arr.clear`

Removes all elements from `arr`.

`arr.collect {| x| ...}`

`arr.collect! {| x| ...}`

`arr.map {| x| ...}`

`arr.map! {| x| ...}`

Invokes the block on each element re

```
[1,2,3].collect{|x|x*2}    # =>
```

`arr.compact`

`arr.compact!`

Removes all `nil` elements from *`arr`*.

`arr.concat( array)`

Appends the elements of array to *`ar`*

`arr.delete( item)`

`arr.delete( item) {| item| ...}`

Deletes all elements matching `item` u
the block if no elements were deletec

`arr.delete_at( n)`

Deletes the nth element of `arr`.

`arr.delete_if {| x| ...}`

Deletes elements where the value of

`arr.each {| x| ...}`

Invokes the block on each element o

`arr.each_index {| i| ...}`

Invokes the block on each element, p

from `0` to `arr.length - 1`.

`arr.empty?`

Returns `true` if the array length is `0`.

`arr.fill( value[, beg[, len]])`

`arr.fill( value, n.. m)`

Sets the specified element (or range

`arr.first`

Returns the first element of `arr`. Equ

`arr.flatten`

`arr.flatten!`

Returns a flattened, one-dimensional

subelements of *arr* into the new arra

```
[1, [2, 3, [4], 5]].flatten    #
```

arr.include?(*item*)

arr.member?(*item*)

Returns `true` if *arr* contains item as

arr.index(*item*)

Returns the index number of the first
first index number), or `nil` if item isn

arr.indexes([*index...*])

arr.indices([*index...*])

Returns an array of elements from th

arr.join([*s*=$,])

Returns a string by joining together a
with *s*.

```
["foo", "bar].join           # =
["hello", "world].join(" ") # =
```

arr.last

Returns the last element of *arr*. Equi

arr.length

See *arr*.size

arr.map {| *x*| ...}

See *arr*.collect {|*x*|...}

arr.map! {| *x*| ...}

See `arr.collect {|x|...}`

`arr.member?( item)`

See `arr.include?(item)`

`arr.nitems`

Returns the number of elements with

`arr.pack( template)`

Packs the elements of an array into a `template`. `template` may consist of a

`a`

ASCII string (null padded)

`A`

	ASCII string (space padded)
b	Bit string (ascending bit order)
B	Bit string (descending bit order)
c	Char
C	Unsigned char
d	Double (native format)

`e`

Little endian float (native format)

`E`

Little endian double (native format)

`f`

Float (native format)

`g`

Big endian float (native format)

`G`

Big endian double (native format)

`h`

Hex string (low nibble first)

`H`

Hex string (high nibble first)

`i`

Integer

`I`

Unsigned integer

`l`

Long

`L`

Unsigned long

`m`

Base64-encoded string

`M`

Quoted printable string

`n`

Big-endian short (network byte orde

`N`

Big-endian long (network byte order

`p`

Pointer to a null-terminated string

`P`

Pointer to a structure (fixed-length st

s

Short

S

Unsigned short

u

UU-encoded string

U

UTF-8 string

v

Little-endian short (VAX byte order)

`V`

Little-endian long (VAX byte order)

`w`

BER-compressed integer

`x`

Null byte

`X`

Backs up one byte

`Z`

ASCII string (space padded)

`@`

Moves to absolute position

Each directive may be followed by e
number of elements to convert, or an
elements should be converted. Direct
Directives `sSiIlL` followed by `_` use
platform.

```
[1, 2, 3, 4].pack("CCCC")     #
[1234].pack("V")              #
[1234].pack("N")              #
```

`arr.pop`

Removes the last element from *`arr`* a

`arr.push( obj... )`

Appends *`obj`* to *`arr`*.

`arr.rassoc( value)`

Searches through an array of arrays,
element matching `value`.

```
[[1,2],[2,4],[3,6]].rassoc(2) #
```

`arr.reject {| x| ...}`

`arr.reject! {| x| ...}`

Deletes elements where the value of

`arr.replace( array)`

Replaces the contents of `arr` with tha

`arr.reverse`

`arr.reverse!`

Puts the elements of the array in reve

`arr.reverse_each {| x| ...}`

Invokes the block on each element o

`arr.rindex( item)`

Returns the index of the last object i

```
a = [1, 2, 3, 1, 3, 4]
a.rindex(3)                          #=
a.rindex(9)                          #=
```

`arr.shift`

Removes the first element from *arr*

```
a = [1, 2, 3, 1, 3, 4]
a.shift                              #=
a                                    #=
```

`arr.size`

arr.length

Returns the number of elements in *a*

arr.slice(*n*)

arr.slice(*n*.. *m*)

arr.slice(*n*, *len*)

Deletes the partial string specified ar

```
a = "0123456789"
a.slice!(1,2)     # => "12"
a                 # => "03456789
```

arr.slice!(*n*)

arr.slice!(*n*.. *m*)

arr.slice!(*n*, *len*)

Deletes the partial string specified ar

```
a = [0,1,2,3,4]
a.slice!(4)         # => 4
a                   # => [0,1,2,3
a.slice!(1..2)      # => [1,2]
a                   # => [0,3]
```

`arr.sort`

`arr.sort!`

Sorts the array.

`arr.sort {| a, b| ...}`

`arr.sort! {| a, b| ...}`

Arrays can be sorted by specifying th
block. The block must compare `a` an
number when `a < b`, and a positive nu

`arr.uniq`

`arr.uniq!`

Deletes duplicate elements from *`arr`*

`arr.unshift( item)`

Prepends *`item`* to *`arr`*.

```
a = [1,2,3]
a.unshift(0)          #=> [0,1,2,3]
```

Hash *Hash class*

`Hash` is a class for collection of key-value
by arbitrary type of objects, which define

Included Module

`Enumerable`

Class Methods

`Hash[key, value ...]`

Creates a `Hash`.

```
Hash[1,2,2,4]  # => {1=>2, 2=>4
```

`Hash::new([default=nil])`

Creates a `Hash`. A default value may

```
h = Hash::new(15)   # => {}
h[44]               # => 15 (no
```

Instance Methods

Methods of the `Hash` class ending in a pip
modification took place, otherwise `nil`. M

the hash.

`h[ key]`

Returns the `value` associated with `ke`

`h[ key]= value`

Associates `value` with `key`.

`h.clear`

Deletes all key-value pairs from `h`.

```
h = {1=>2, 2=>4}
h.clear
h                    # => {}
h = {1=>2, 2=>4}
h.delete_if{|k,v| k % 2 == 0}
h          # => {1=>2}
```

`h.default`

Returns the default value for a key th
isn't copied, so that modifying the de
thereafter.

`h.default= value`

Sets the default value.

`h.delete( key)`

Deletes a key-value pair with a key e

`h.delete_if {| key, value| ...}`

Deletes key-value pairs where the ev

`h.each {| key, value| ...}`

`h.each_pair {| key, value| ...}`

Executes the block once for each key

`h.each_key {| key| ...}`

Executes the block once for each key

`h.each_value {| value| ...}`

Executes the block once for each val

`h.empty?`

Returns `true` if the hash is empty.

`h.fetch( key[, ifnone=nil])`

`h.fetch( key) {| key| ...}`

Returns the value associated with `ke`
block is returned. If no block is speci

`h.has_value?( value)`

See `h.value?(value)`

`h.index( value)`

Returns the key for `value`, or `nil` if i

```
h = {1=>2, 2=>4}
h.index(4)        # => 2
h.index(6)        # => nil
```

`h.indexes([ key...])`

`h.indices([ key...])`

Returns an array of values associatec

`h.invert`

Returns a hash containing *h*'s values

one keys have same value, arbitrary

```
h = {"y" => 365, "m" => 31, "d'
p h.invert   # => {60=>"h", 365
```

`h.key?( key)`

`h.has_key?( key)`

`h.include?( key)`

`h.member?( key)`

Returns `true` if key is present in *h*.

`h.keys`

Returns an array of all keys.

`h.rehash`

Rebuilds the hash. If a hash isn't rebu
changed, that key will no longer be a

```
a = [1,2]          # array as key
h = {a=>3}
h[a]               # => 3
a[0] = 2           # modify key
h[a]               # => nil (canno
h.rehash
h[a]               # => 3
```

`h.reject {| key, value| ...}`

`h.reject! {| key, value| ...}`

Deletes key-value pairs where the va

`h.replace( hash)`

Replaces the contents of *h* with that o

`h.shift`

Removes a key-value pair from `h` an

`h.size`

`h.length`

Returns the number of key-value pai

`h.sort`

`h.sort {| a, b| ...}`

Produces an array using `h.to_a` and

`h.store( key, value)`

Synonym for `h[key]=value`.

`h.to_a`

Returns an array containing the array

```
h = {"y" => 365, "m" => 31, "d'
h.to_a        # => [["m", 31],
```

`h.to_hash`

Returns *h* itself. Every object that ha by `h.replace` and `h.update`.

`h.update( hash)`

Updates *h* with the contents of the sp associated value of *hash* takes precec

```
h1 = { "a" => 100, "b" => 200 }
h2 = { "b" => 300, "c" => 400 }
h1.update(h2)   #=> {"a"=>100,
```

`h.value?( value)`

`h.has_value?( value)`

Returns `true` if value is present in *h*.

h.values

Returns an array of all values.

```
h = {"y" => 365, "m" => 31, "d'
p h.values          # => [31, 24,
```

Enumerable *Enumerable mix-ir*

The `Enumerable` module assumes that the
add the following methods to a class that

Instance Methods

e.collect {| *x*| ...}

e.map {| *x*| *...*}

Returns an array containing the resul

e.detect {| *x*| *...*}

See *e*.find {|*x*|*...*}

e.each_with_index {| *x*, *i*| *...*}

Executes the block once for each iter
the block.

```
["foo","bar","baz"].each_with_i
  printf "%d: %s\n", i, x
}
# prints:
#  0: foo
#  1: bar
#  2: baz.
```

e.entries

`e.to_a`

Returns an array containing the items

`e.find {| x| ...}`

`e.detect {| x| ...}`

Returns the first item for which the b

```
["foo","bar","baz"].detect {|s|
```

`e.find_all {| x| ...}`

`e.select {| x| ...}`

Returns an array of all items for whic

```
["foo","bar","baz"].select {|s|
```

`e.grep( re)`

`e.grep( re) {| x| ...}`

Returns an array containing all items
specified, it's run on each matching i

```
["foo","bar","baz"].grep(/^b/)
[1,"bar",4.5].grep(Numeric)
[1,"bar",4.5].grep(Numeric) {|x
   puts x+1
}
# prints:
#  2
#  5.5
```

`e.include?( item)`

`e.member?( item)`

Returns `true` if an item equal to `ite`

`e.map {| x| ...}`

See `e.collect {|x|...}`

`e.max`

Returns the item in `e` with the maxim
possible between the items.

```
[1,5,3,2].max          # => 5
```

`e.member?( item)`

See `e.include?(item)`

`e.min`

Returns the item in `e` with the minim
possible between the items.

```
[1,5,3,2].min          # => 1
```

`e.reject {|x| ...}`

Returns an array of all items for whic

```
["foo","bar","baz"].reject {|s|
```

`e.select {| x| ...}`

See `e.find_all {|x|...}`

`e.sort`

`e.sort {| a, b| ...}`

Returns an array of sorted items from
comparison. Like `<=>`, the block mus
number (`a> b`), 0(`a == b`), or a negati

`e.to_a`

See `e.entries`

3.4.4 Numbers

As you'd expect, Ruby provides a suitably
numeric data, through the classes `Numeric`
addition, further tools are available in the
manipulating numeric data.

Numeric *Superclass of all conc*

`Numeric` provides common behavior of nu
should not instantiate this class.

Included Module

`Comparable`

Instance Methods

`+` *n*

Returns *n*.

`-` *n*

Returns *n* negated.

n `+ num`

n `- num`

n `* num`

n `/ num`

Performs arithmetic operations: addi

`n % num`

Returns the modulus of `n`.

`n ** num`

Exponentiation.

`n.abs`

Returns the absolute value of `n.`

`n.ceil`

Returns the smallest integer greater t

`n.coerce( num)`

Returns an array containing `num` and
allows them to be operated on mutua
numeric operators.

`n.divmod( num)`

Returns an array containing the quoti

`n.floor`

Returns the largest integer less than o

```
1.2.floor                    #=> 1
2.1.floor                    #=> 2
(-1.2).floor                 #=> -2
(-2.1).floor                 #=> -3
```

`n.integer?`

Returns `true` if *n* is an integer.

`n.modulo( num)`

Returns the modulus obtained by div
`floor`. Equivalent to `n.divmod(num)`

`n.nonzero?`

Returns *n* if it isn't zero, otherwise `n:`

`n.remainder( num)`

Returns the remainder obtained by di
the quotient. The result and n always

```
(13.modulo(4))          #=>  1
(13.modulo(-4))         #=> -3
((-13).modulo(4))       #=>  3
((-13).modulo(-4))      #=> -1

(13.remainder(4))       #=>  1
(13.remainder(-4))      #=>  1
((-13).remainder(4))    #=> -1
(-13).remainder(-4))    #=> -1
```

`n.round`

Returns *n* rounded to the nearest inte

```
    1.2.round                    #=> 1
    2.5.round                    #=> 3
    (-1.2).round                 #=> -1
    (-2.5).round                 #=> -3
```

`n.truncate`

Returns *n* as an integer with decimal

```
    1.2.truncate                 #=> 1
    2.1.truncate                 #=> 2
    (-1.2).truncate              #=> -1
    (-2.1).truncate              #=> -2
```

`n.zero?`

Returns zero if *n* is 0.

Integer *Integer class*

`Integer` provides common behavior of in
abstract class, so you should not instantiat

Inherited Class

`Numeric`

Included Module

`Precision`

Class Method

`Integer::induced_from(numeric)`

Returns the result of converting num

Instance Methods

`~` *`i`*

Bitwise operations: `AND`, `OR`, `XOR`, and

`i & int`

`i | int`

`i ^ int`

`i << int`

`i >> int`

Bitwise left shift and right shift.

`i[ n]`

Returns the value of the *n*th bit from

```
5[0]      # => 1
5[1]      # => 0
5[2]      # => 1.
```

i.chr

Returns a string containing the chara

```
65.chr      # => "A"
?a.chr      # => "a"
```

i.downto(*min*) {| *i*| ...}

Invokes the block, decrementing eac

```
3.downto(1) {|i|
  puts i
}
# prints:
#  3
#  2
#  1
```

i.next

i.succ

Returns the next integer following *i*.

`i.size`

Returns the number of bytes in the m

`i.step( upto, step) {| i| ...}`

Iterates the block from *i* to *upto*, inc

```
10.step(5, -2) {|i|
  puts i
}
# prints:
#  10
#  8
#  6
```

`i.succ`

See `i.next`

`i.times {| i| ...}`

Iterates the block *i* times.

```
3.times {|i|
  puts i
}
# prints:
#  0
#  1
#  2  .
```

`i.to_f`

Converts *i* into a floating point numl
information.

```
1234567891234567.to_f   # => 1.
```

`i.to_int`

Returns *i* itself. Every object that ha `i.upto( max) {| i| ...}`

Invokes the block, incrementing each

```
1.upto(3) {|i|
  puts i
}
# prints:
#  1
#  2

#  3
```

Fixnum *Fixed-length number c*

`Fixnum` objects are fixed-length numbers w
an operation exceeds this range, it's autom

Inherited Class

`Integer`

Bignum *Infinite-length integer c*

`Bignum` objects are infinite-length integers
memory can hold. Conversions between `F`
automatically.

Inherited Class

`Integer`

Float *Floating-point number cla*

`Float` objects represent floating-point nur
point numbers as internel representation c

Inherited Class

`Numeric`

Included Module

`Precision`

Class Method

`Float::induced_from( *num*)`

Returns the result of converting *num*

Instance Methods

`*f*.finite?`

Returns `true` if *f* isn't infinite and *f*.

`*f*.infinite?`

Returns `1` if *`f`* is positive infinity, `-1`

`f``.nan?`

Returns `true` if *`f`* isn't a valid IEEE f

Precision *Precision conversion*

`Precision` is a module to provide a conve

Instance Methods

`prec(` *`c`*`)`

Returns the result of converted self t
the Precision module actually return

`prec_f`

Equivalent to `prec(Float)`.

`prec_i`

Equivalent to `prec(Integer)`.

Comparable *Comparable mix-i*

The `Comparable` module assumes that the
The `<=>` method compares two objects an
is greater, `0` if it's equal to the right opera
can add the following methods to a class t
module.

Instance Methods

`c` `<` *`other`*

Returns `true` if `c` is less than *`other`* (

`c` `<=` *`other`*

Returns `true` if `c` is less than or equa
negative number or `0`).

`c` `>` *`other`*

Returns `true` if *`c`* is greater than *`othe`*
number).

`c` `>=` *`other`*

Returns `true` if *`c`* is greater than or e
a positive number or `0`).

`c` `== other`

Returns `true` if the objects are equal

`c.between?( min, max)`

Returns `true` if `c` is between `min` and

Math *Module of math functions*

The `Math` module provides a collection of
private instance methods and module met
definition.

Module Functions

`atan2( x, y)`

Calculates the arc tangent.

`cos( x)`

Calculates the cosine of *x*.

`exp( x )`

Calculates an exponential function (

`frexp( x )`

Returns a two-element array containi
x.

`ldexp( x, exp )`

Returns the value of *x* times 2 to the

`log( x )`

Calculates the natural logarithm of *x*

`log10( x )`

Calculates the base 10 logarithm of *x*

`sin( x )`

Calculates the sine of *x*.

`sqrt( x )`

Returns the square root of *x*. *x* must

`tan( x )`

Calculates the tangent of *x*.

Constants

`E`

`e`, the base of natural logarithms

π

pi; the Ludolphian number

3.4.5 Operating System Servic

Ruby's portability necessitates some level
the underlying operating system. Abstract
provided through the Ruby built-in classe
`Process`.

IO *I/O class*

`IO` is object-oriented representation of `std`
classes, such as `File`, `BasicSocket`, etc.

Included Module

`Enumerable`

Class Methods

`IO::foreach( path) {| x| ...}`

Opens the file and executes the block
block exits.

```
n = 1
IO::foreach(path) {|line|
  print n, ":", lib
  n+=1
}
```

`IO::new( fd[, mode="r"])`

Returns a new `IO` stream for the spec

`IO::pipe`

Creates a pair of `IO` streams connect
`([readIO, writeIO]).`

`IO::popen( cmd[, mode="r"])`

`IO::popen( cmd[, mode="r"]) {| io| ..`

Executes the command specified by
stream connected to it. If *`cmd`* is `-`, a r
subprocess with an `IO` object returne
process. If a block is specified, it's ru
stream is closed when the block exits

`IO::readlines( path)`

Returns the contents of a file as an ar

`IO::select( reads[, writes=nil[, exc`

Checks for changes in the status of th
exceptions, which are passed as array
that don't need checking. A three-ele
for which there were changes in statu

```
    IO::select([STDIN], nil, nil, 1
```

Instance Methods

io << *str*

Prints *str* to *IO*.

io.binmode

Enables binary mode (for use on DO
mode, it can't be reset to non-binary

io.close

Closes the *io*.

io.close_read

Closes the read-only end of a duplex

`io.close_write`

Closes the write-only end of a duplex

`io.closed?`

Returns `true` if `io` is closed.

`io.each {| x| ...}`

`io.each_line {| x| ...}`

Reads in the contents of `io` one line a

```
f = open(path)
n = 1
f.each_line {|line|
  print n, ":", lib
  n+=1
}.
```

`io.each_byte {| x| ...}`

Reads in the contents of `io` one byte

`io.eof`

`io.eof?`

Returns `true` if `EOF` has been reachec

`io.fcntl( req[, arg])`

Calls `fcntl(2)` system call. Argumer
implemented on all platforms.

`io.fileno`

`io.to_i`

Returns the file descriptor number fo

`io.flush`

Flushes output buffers.

`io.getc`

Reads one character (8-bit byte) fron
`nil` on `EOF`.

`io.gets([ rs=$/])`

Reads one line from `io`. Returns `nil`

`io.ioctl( req[, arg])`

Calls `ioctl(2)` system call. Argume
implemented on all platforms.

`io.isatty`

See `io.tty?`

`io.lineno`

Returns the current line number in *io*

`io.lineno=n`

Sets the current line number in *io*.

`io.pid`

Returns the process ID associated wi

`io.pos`

`io.tell`

Returns the current position of the fil

`io.pos= offset`

Sets the position of the file pointer.

`io.print( arg...)`

Writes the specified arguments to *io*.

`io.printf( fmt[, arg...])`

Writes the specified arguments to *io* specifiers, see `sprintf` in Section 3.3

`io.putc( c)`

Writes one character to *io*.

`io.puts( str)`

Writes *str* to *io*, appending `newline`

```
io.puts("foo")      # prints "fo
io.puts("bar\n")    # prints "ba
```

`io.read([ len])`

Reads only the specified number of b

file is read.

`io.readchar`

Reads one character (8-bit byte) fron

`io.readline([ rs=$/])`

Reads one line from *io*. Raises an ex

`io.readlines([ rs=$/])`

Reads all lines in *io* and returns then

`io.reopen( f)`

Resets *io* to a copy of *f*. The class of

`io.rewind`

Moves the file pointer to the beginni

io.seek(*pos*[, *whence*=IO::SEEK_SET])

Moves the file pointer. The starting p
(beginning of stream), `IO::SEEK_CUR`
stream).

io.stat

Calls `fstat(2)` system call and retur

io.sync

Returns `true` if sync mode is enable
flushed after each write.

io.sync= *mode*

Sets the sync mode for output to `tru`

io.sysread(*len*)

Reads *len* bytes from *io* using `read(`
with other reading `IO` methods.

`io.syswrite( str)`

Writes *str* to *io* using `write(2)` syst
other writing `IO` methods, or you ma

`io.tell`

See `io.pos`

`io.to_i`

See `io.fileno`

`io.to_io`

Returns *io* itself. Every object that h
`IO::select` and `io.reopen`.

`io.tty?`

`io.isatty`

Returns `true` if *`io`* is connected to `tt`

`io.ungetc( c)`

Pushes one character back onto *`io`*.

`io.write( str)`

Writes *`str`* to *`io`*. Every object that h
`$defout`, the default output destinati

File *File class*

A `File` represents an `stdio` object that co instance of this class for regular files.

Inherited Class

`IO`

Class Methods

`File::atime(` *`path`*`)`

Returns the last access time for *`path`*

`File::basename(` *`path`*`[,` *`suffix`*`])`

Returns the filename at the end of *`pa`*
the end of the filename.

```
File.basename("/home/matz/bin/r
File.basename("/home/matz/bin/r
```

`File::blockdev?(` *`path`*`)`

Returns `true` if *`path`* is a block devic

`File::chardev?( path)`

Returns `true` if *`path`* is a character de

`File::chmod( mode, path...)`

Changes the permission mode of the

`File::chown( owner, group, path...)`

Changes the owner and group of the

`File::ctime( path)`

Returns the last `inode` change time fo

`File::delete( path...)`

`File::unlink( path...)`

Deletes the specified files.

`File::directory?( path)`

Returns `true` if *path* is a directory.

`File::dirname( path)`

Returns the directory portion of *path*

`File::executable?( path)`

Returns `true` if *path* is executable.

`File::executable_real?( path)`

Returns `true` if *path* is executable w

`File::exist?(` *`path`*`)`

Returns `true` if *`path`* exists.

`File::expand_path(` *`path`*`[,` *`dir`*`])`

Returns the absolute path of *`path`*, ex
directory, and *`~user`* to the *`user`*'s ho
from the directory specified by *`dir`*,
omitted.

`File::file?(` *`path`*`)`

Returns `true` if *`path`* is a regular file

`File::ftype(` *`path`*`)`

Returns one of the following strings

`file`

Regular file

`directory`

Directory

`characterSpecial`

Character special file

`blockSpecial`

Block special file

`fifo`

Named pipe (FIFO)

`link`

Symbolic link

`socket`

Socket

`unknown`

Unknown file type

`File::grpowned?( path)`

Returns `true` if *`path`* is owned by the

`File::join( item...)`

Returns a string consisting of the spe
`File::Separator` separating each ite

```
File::join("", "home", "matz",
```

`File::link( old, new)`

Creates a hard link to file `old`.

`File::lstat( path)`

Same as stat, except that it returns in
the files they point to.

`File::mtime( path)`

Returns the last modification time fo

`File::new( path[, mode="r"])`

`File::open( path[, mode="r"])`

`File::open( path[, mode="r"]) {|f| .`

Opens a file. If a block is specified, t
as an argument. The file is closed au
methods differ from `Kernel#open` in

string isn't run as a command.

`File::owned?( path)`

Returns `true` if *`path`* is owned by the

`File::pipe?( path)`

Returns `true` if *`path`* is a pipe.

`File::readable?( path)`

Returns `true` if *`path`* is readable.

`File::readable_real?( path)`

Returns `true` if *`path`* is readable with

`File::readlink( path)`

Returns the file pointed to by *`path`*.

`File::rename(` *`old`*`,` *`new`*`)`

Changes the filename from *`old`* to *`ne`*

`File::setgid?(` *`path`*`)`

Returns `true` if *`path`*'s set-group-id p

`File::setuid?(` *`path`*`)`

Returns `true` if *`path`*'s set-user-id pe

`File::size(` *`path`*`)`

Returns the file size of *`path`*.

`File::size?(` *`path`*`)`

Returns the file size of *path*, or nil i

File::socket?(*path*)

Returns true if *path* is a socket.

File::split(*path*)

Returns an array containing the conte
and File::basename(*path*).

File::stat(*path*)

Returns a File::Stat object with in

File::sticky?(*path*)

Returns true if *path*'s sticky bit is se

File::symlink(*old*, *new*)

Creates a symbolic link to file `old`.

`File::symlink?( path)`

Returns `true` if `path` is a symbolic li

`File::truncate( path, len)`

Truncates the specified file to `len` by

`File::unlink( path...)`

See `File::delete(path...)`

`File::umask([ mask])`

Returns the current umask for this pr
argument is specified, the umask is s

`File::utime( atime, mtime, path...)`

Changes the access and modification

`File::writable?( path)`

Returns `true` if *`path`* is writable.

`File::writable_real?( path)`

Returns `true` if *`path`* is writable with

`File::zero?( path)`

Returns `true` if the file size of *`path`* i

Instance Methods

`f``.atime`

Returns the last access time for *`f`*.

`f``.chmode( mode)`

Changes the permission mode of `f`.

`f.chown( owner, group)`

Changes the owner and group of `f`.

`f.ctime`

Returns the last `inode` change time f

`f.flock( op)`

Calls `flock(2)`. *op* may be `0` or a log
`LOCK_NB`, `LOCK_SH`, and `LOCK_UN`.

`f.lstat`

Same as `stat`, except that it returns i
the files they point to.

`f.mtime`

Returns the last modification time fo

`f.path`

Returns the pathname used to create

`f.reopen( path[, mode="r"])`

Reopens the file.

`f.truncate( len)`

Truncates `f` to len bytes.

Constants

Constants in the `File` class are also define
they may be included separately if necess

open constants

`RDONLY`

Read-only mode

`WRONLY`

Write-only mode

`RDWR`

Read and write mode

`APPEND`

Append mode

`CREAT`

Create file

`EXCL`

Exclusive open

ioctl constants

`NONBLOCK`

Nonblocking mode

`TRUNC`

Truncate to 0 bytes

`NOCTTY`

Don't allow a terminal device to beco

`BINARY`

Binary mode

`SYNC`

Sync mode

flock constants

`LOCK_EX`

Exclusive lock

`LOCK_NB`

Don't block when locking

`LOCK_SH`

Shared lock

`LOCK_UN`

Unlock

File::Stat *File status class*

`File::Stat` contains file status informatic
methods.

Included Module

`Comparable`

Instance Methods

`s` `<=>` *`stat`*

Compares the modification times of

`s``.atime`

Returns the last access time for *`s`*.

`s``.blksize`

Returns the block size of *s*'s file syst

`s.blockdev?`

Returns `true` if *s* is a block device.

`s.blocks`

Returns the number of blocks allocat

`s.chardev?`

Returns `true` if *s* is a character devic

`s.ctime`

Returns the last `inode` change time f

`s.dev`

Returns an integer representing the d

`s.directory?`

Returns `true` if *s* is a directory.

`s.executable?`

Returns `true` if *s* is executable.

`s.executable_real?`

Returns `true` if *s* is executable with

`s.file?`

Returns `true` if *s* is a regular file.

`s.ftype`

Returns one of the following strings

`file`

Regular file

`directory`

Directory

`characterSpecial`

Character special file

`blockSpecial`

Block special file

`fifo`

Named pipe (FIFO)

`link`

Symbolic link

`socket`

Socket

`unknown`

Unknown file type

`s.gid`

Returns the group ID.

`s.grpowned?`

Returns `true` if `s` is owned by the us

`s.ino`

Returns the `inode` number for `s`.

`s.mode`

Returns the access permission mode

`s.mtime`

Returns the modification time for *s*.

`s.nlink`

Returns the number of hard links to

`s.owned?`

Returns `true` if *s* is owned by the eff

`s.pipe?`

Returns `true` if *s* is a pipe.

`s.rdev`

Returns an integer representing the d

`s.readable?`

Returns `true` if *s* is readable.

`s.readable_real?`

Returns `true` if *s* is readable with rea

`s.setgid?`

Returns `true` if *s*'s set-group-id perm

`s.setuid?`

Returns `true` if *s*'s set-user-id permis

`s.size`

Returns the file size of *s*

`s.size?`

Returns the file size of `s`, or `nil` if it'

`s.socket?`

Returns `true` if `s` is a socket.

`s.sticky?`

Returns `true` if `s`'s sticky bit is set.

`s.symlink?`

Returns `true` if `s` is a symbolic link.

`s.uid`

Returns the user ID.

`s.writable?`

Returns `true` if `s` is writable.

`s.writable_real?`

Returns `true` if `s` is writable with rea

`s.zero?`

Returns `true` if the file size of `s` is 0.

FileTest *File testing module*

The `FileTest` module contains methods f
are also provided as class methods of the

Module Functions

`blockdev?( path)`

Returns `true` if `path` is a block devic `chardev?( path)`

Returns `true` if *`path`* is a character d

`directory?( path)`

Returns `true` if *`path`* is a directory.

`executable?( path)`

Returns `true` if *`path`* is executable.

`executable_real?( path)`

Returns `true` if *`path`* is executable w

`exist?( path)`

Returns `true` if *`path`* exists.

`file?( path)`

Returns `true` if *`path`* is a regular file

`grpowned?( path)`

Returns `true` if *`path`* is owned by the

`owned?( path)`

Returns `true` if *`path`* is owned by the

`pipe?( path)`

Returns `true` if *`path`* is a pipe.

`readable?( path)`

Returns `true` if *`path`* is readable.

`readable_real?( path)`

Returns `true` if *`path`* is readable with

`setgid?( path)`

Returns `true` if *`path`*'s set-group-id p

`setuid?( path)`

Returns `true` if *`path`*'s set-user-id per

`size( path)`

Returns the file size of *`path`*.

`size?( path)`

Returns the file size of *`path`* or `nil` if `socket?( path)`

Returns `true` if *`path`* is a socket.

`sticky?( path)`

Returns `true` if *`path`*'s sticky bit is se

`symlink?( path)`

Returns `true` if *`path`* is a symbolic li

`writable?( path)`

Returns `true` if *`path`* is writable.

`writable_real?( path)`

Returns `true` if *`path`* is writable with `zero?( path)`

Returns `true` if the file size of `path` i

Dir *Directory class*

A `Dir` is a class to represent a directory st
the operating system. `Dir` class also holds
card filename matching, changing current

Included Module

`Enumerable`

Class Methods

`Dir[pat]`

`Dir::glob(` *`pat`*`)`

Returns an array of filenames matchi

`*`

Matches any string including the nul

`**`

Matches any string recursively

`?`

Matches any single character

`[...]`

Matches any one of enclosed charact

`{a,b...}`

Matches any one of strings

```
Dir["foo.*"]     # matches "foo
Dir["foo.?"]     # matches "foo
Dir["*.[ch]"]    # matches "mai
Dir["*.{rb,c}"]  # matches "mai
Dir["**/*.c"]    # recursively
```

`Dir::chdir( path)`

Changes the current directory.

`Dir::chroot( path)`

Changes the root directory (only allo
platforms.

`Dir::delete( path)`

See `Dir::rmdir(path)`.

`Dir::entries( *path*)`

Returns an array of filenames in dire

`Dir::foreach( *path*) {| *f*| *...*}`

Executes the block once for each file

`Dir::getwd`

`Dir::pwd`

Returns the current directory.

`Dir::glob( *pat*)`

See `Dir[pat]`.

`Dir::mkdir( *path*[, *mode*=0777])`

Creates the directory specified by *pa*

value of `File::umask` and is ignored

`Dir::new( *path* )`

`Dir::open( *path* )`

`Dir::open( *path* ) {| *dir*| *...*}`

Returns a new directory object for *pa*
object is passed to the block, which c

`Dir::pwd`

See `Dir::getwd`.

`Dir::rmdir( *path* )`

`Dir::unlink( *path* )`

`Dir::delete( *path* )`

Deletes the directory specified by *pa*

Instance Methods

`d.close`

Closes the directory stream.

`d.each {| f| ...}`

Executes the block once for each ent

`d.pos`

`d.tell`

Returns the current position in *d*.

`d.pos= offset`

Sets the position in the directory stre

`d.pos= pos`

`d.seek(po s)`

Moves to a position in *d*. *pos* must b

`d.read`

Returns the next entry from *d*.

`d.rewind`

Moves position in *d* to the first entry

`d.seek(po s)`

See `d.pos=pos`.

`d.tell`

See `d.pos`.

Process *Process handling modu*

`The Process` module provides methods to
platform dependent.

Module Methods

`Process.egid`

Returns the effective group ID of this

`Process.egid=` *`gid`*

Sets the effective group ID of this pr

`Process.euid`

Returns the effective user ID of this

`Process.euid=` *`uid`*

Sets the effective user ID of this proc

`Process.gid`

Returns the group ID of this process.

`Process.gid=` *`gid`*

Sets the group ID of this process.

`Process.pid`

Returns the process ID of this proces

`Process.ppid`

Returns the process ID of the parent

`Process.uid`

Returns the user ID of this process.

`Process.uid= uid`

Sets the user ID of this process.

Module Functions

`exit!([ result=0])`

Kills the program bypassing exit han

`fork`

`fork {...}`

Creates a child process. `nil` is return
ID (`Integer`) is returned in the paren
in the child process.

`getpgid(` *`pid`*`)`

Returns the process group ID for pro

`getpgrp([` *`pid`*`=$$])`

Returns the process group ID for this

`getpriority(` *`which, who`*`)`

Returns the current priority.

`kill(` *`sig, pid...`*`)`

Sends signal to a process. *`sig`* is spec

`setpgid(` *`pid`*`)`

Sets the process group ID for process

`setpgrp`

Equivalent to `setpgid(0,0)`.

`setpriority(` *`which, who, prio`* `)`

Sets process priority.

`setsid`

Establishes this process as a new ses

`wait`

Waits for a child process to exit and r

`wait2`

Waits for a child process to exit and r
array.

`waitpid(` *`pid`*`[,` *`flags`*`])`

Waits for child process *pid* to exit ar
process if *pid*=0 is specified. Flags m
WNOHANG and WUNTRACE.

`waitpid2( pid[, flags])`

Waits for child process *pid* to exit ar
array.

Constants

`PRIO_PROCESS`

Process priority. Specified in a logica
method.

`PRIO_PGRP`

Process group priority. Specified in a
setpriority method.

`PRIO_USER`

User priority. Specified in a logical o
method.

`WNOHANG`

Terminate immediately without block
logical or as the second argument of

`WUNTRACED`

Terminate any stopped children whos
logical or as the second argument of

3.4.6 Threads

Threads are a powerful tool for creating a
implementations, for making your softwa
benefit is the one emphasized�leaner coo

Microthreads are in-process threads simul
interpreter itself. Hence, Ruby's `Thread` cl
library or operating systems, making Ruby

Thread *Thread class*

The class for user-level threads. When the
killed, and the interpreter quits.

Class Methods

`Thread::abort_on_exception`

Returns `true` if thread is set to abort

`Thread::abort_on_exception=` *`bool`*

Sets whether or not to abort on an ex

displays an error message for excepti
program.

`Thread::critical`

Returns `true` when scheduling of exi

`Thread::critical= bool`

Sets the status of thread-scheduling p

`Thread::current`

Returns the current thread.

`Thread::exit`

Terminates the current thread.

`Thread::fork([ arg...]) {| x...| ...`

See `Thread::start([`*`arg...`*`]) {|`*`x`*

`Thread::kill(` *`th`*`)`

Terminates the specified thread.

`Thread::list`

Returns an array of all threads.

`Thread::main`

Returns the main thread.

`Thread::new([` *`arg...`*`]) {|` *`x...`*`|` *`...`*`}`

See `Thread::start([`*`arg...`*`]) {|`*`x`*

`Thread::pass`

Passes execution to another thread.

`Thread::start([` *`arg...`*`]) {|` *`x...`*`|` *`..`*

`Thread::fork([` *`arg...`*`]) {|` *`x...`*`|` *`...`*

`Thread::new([` *`arg...`*`]) {|` *`x...`*`|` *`...`*`}`

Creates a new thread and executes th
to the block.

`Thread::stop`

Stops the current thread.

Instance Methods

`t``[` *`name`*`]`

Retrieves the value of a thread-local
either a string or a symbol.

`t[ name]= value`

Sets the `value` of a thread-local varia

`t.abort_on_exception`

Returns `true` if thread is set to abort

`t.abort_on_exception= bool`

Sets whether or not this thread will a
exception, displays an error message
the program.

`t.alive?`

Returns `true` if the thread is alive (sl

`t.exit`

See `t.kill`.

`t.join`

Waits for the thread to terminate. If t
that exception is raised again.

`t.key?( name)`

Returns `true` if a thread-local variab

`t.kill`

`t.exit`

Terminates the thread.

`t.raise( exc[, mesg])`

Raises an exception from the thread.

`t.run`

Makes the thread eligible for schedul

`t.safe_level`

Returns the value of `$SAFE`, the threa

`t.status`

Returns the status of thread (`true` if a
terminated with an exception).

`t.stop?`

Returns `true` if the thread is stopped

`t.value`

Waits for the thread to terminate and
evaluated. If the thread is terminated
again.

`t.wakeup`

Marks the thread as eligible for sche

ThreadGroup *Thread group cla*

A thread can belong to only one thread gr
specified, a newly created thread belongs
originally created it.

Class Method

`ThreadGroup::new`

Creates a new thread group.

Instance Methods

`tg.add( th)`

Adds `th` to the thread group. A threa

`tg.list`

Returns an array of threads belongin

Constants

`Default`

The default thread group.

3.4.7 Exceptions

Ruby's exception handling class, `Excepti`
the notion that the code discovering some
code that can handle that error condition.

Exception *Superclass for excep*

Instance Methods

`e.backtrace`

Returns backtrace information (from
strings.

`e.exception`

Returns clone of the exception objec

`e.message`

Returns exception message.

Errno *System call exceptions m*

`Errno::ENOENT` and other errors are defin

3.4.8 Built-in Exceptions

`Exception` and the classes derived from it
blocks for handling error conditions in yo
you know and love from OOP, you can ea
you see fit.

The following are abstract `Exception` clas

`Exception`

Superclass of all exceptions

`ScriptError`

Error originating from program mista

`StandardError`

Superclass of standard error exceptic

The following are subclasses of `Standard`

`ArgumentError`

Argument error (incorrect number of

`EOFError`

End of file reached

`FloatDomainError`

Float calculation error

`IndexError`

Error related to index.

`IOError`

Error related to input or output.

`LocalJumpError`

Error related to break, next, redo, ret

`NoMemoryError`

Insufficient memory.

`RangeError`

Error produced when range exceedec

`RegexpError`

Regular expression error

`RuntimeError`

General runtime error

`SecurityError`

Error related to security

`SystemCallError`

Superclass of system call exceptions

`SystemStackError`

Insufficient stack area

`TypeError`

Error produced when types don't mat

`ZeroDivisionError`

Error produced when attempting to d

The following are two subclasses of `Syst`

`Errno::ENOENT`

File or directory doesn't exist

`Errno::EPERM`

Insufficient access rights

The following are subclasses of `ScriptEr`

`LoadError`

Error occurring during the loading of

`NameError`

Name error caused by accessing und

`NotImplementedError`

Function not supported by interprete

`SyntaxError`

Error related to syntax

The following are subclasses of `Exceptio`

`Fatal`

Fatal error that can't ever be caught

`Interrupt`

Interrupt (SIGINT) received

`SystemExit`

`exit` called

3.4.9 Classes and Modules

Support for OOP in Ruby can be found in
objects are of class `Class`, and the `Module`
mix-ins.

Module *Module class*

A `Module` is similar to a class, except that

Class Methods

`Module::class_variables`

Returns an array of class variable na

`Module::constants`

Returns an array of constant names.

`Module::nesting`

Returns an array of classes and modu

`Module::new`

Creates a new anonymous module.

Instance Methods

`m` `<` *`mod`*

Returns `true` if *`m`* is a descendant of

`m` `<=` *`mod`*

Returns `true` if *`m`* is a descendant of

`m` `<=>` *`mod`*

Returns +1 if *`m`* is an ancestor of *`mod`*,
descendant of *`mod`*.

`m === obj`

Returns `true` if `obj` is an instance of

`m > mod`

Returns `true` if `m` is an ancestor of `mc`

`m >= mod`

Returns `true` if `m` is an ancestor of or

`m.ancestors`

Returns an array of ancestors, includ

`m.const_defined?( name)`

Returns `true` if the constant specifie

`m.const_get( name)`

Returns the value of the specified co

m.`const_set(` *name*, *value*`)`

Sets the *value* of a constant.

m.`constants`

Returns an array of constant names.

m.`included_modules`

Returns an array of names of include

m.`instance_method(` *name*`)`

Returns a `UnboundMethod` object cor
the corresponding method doesn't ex
invocation.

```
    unbound_plus = Fixnum.instance_
    plus = unbound_plus.bind(1)
    p plus.call(2)                  # => 3
```

`m.instance_methods([ all=false])`

Returns an array of instance method
superclasses are also returned.

`m.method_defined?( name)`

Returns `true` if the method specified

`m.module_eval( str)`

`m.module_eval {...}`

Evaluates *str* or block in the context
added to *m*.

`m.name`

Returns the module's name.

`m.private_class_method( name...)`

Sets visibility of class methods to `pr`

`m.private_instance_methods([ all=fa`

Returns an array of instance methods
instance methods from superclasses a

`m.protected_instance_methods([ all=`

Returns an array of instance methods
instance methods from superclasses a

`m.public_class_method( name...)`

Sets visibility of class methods to `pu` *m*.public_instance_methods([*all*=fals

Returns an array of instance methods
instance methods from superclasses a

Private Instance Methods

alias_method(*new*, *old*)

Creates an alias for a method. Equiva
name is specified with a symbol or st

append_features(*mod*)

Adds module definitions (methods ar
This is the callback method used by
processing during the inclusion of m

attr(*name*[, *flag*=false])

Defines a named attribute, creating a
variable `@name`. If `flag` is `true`, also
the attribute.

`attr_accessor( name...)`

Defines read accessor (`name`) and wri
`@name`.

`attr_reader( name...)`

Defines read accessor (`name`) for eacl

`attr_writer( name...)`

Defines write accessor (`name=`) for ea

`extend_object( obj)`

Adds the current module's methods a

method used by `Object#extend`. Use

`include( *mod...* )`

Includes the methods and constants o

`method_added( *name* )`

Method called by the interpreter ever
statement. The standard definition do

`module_function( *name...* )`

Copies the definition of each of the i
method and converts it to a module f

`private([ *name...* ])`

Sets the visibility of each instance m
with no arguments, sets the visibility

`protected([ name...])`

Sets the visibility of each instance m
used with no arguments, sets the visi
protected.

`public([ name...])`

Sets the visibility of each instance m
with no arguments, sets the visibility

`remove_const( name)`

Removes the definition of constant,

`remove_method( name)`

Removes method (*name*) from the cu
defined in a superclass, it becomes v

```
class Foo
  def foo
    puts "Foo"
  end
end

class Bar<Foo
  def foo
    puts "Bar"
  end
end

b = Bar.new
b.foo
class Bar
  remove_method :foo
end
b.foo
```

undef_method(*name*)

Turns method (*name*) into an undefin
name is defined in a superclass, it be

```
    class Foo
          def foo
          end
        end

        class Bar<Foo
          undef_method :foo
        end

        b = Bar.new
        b.foo
```

Class *Class class*

A class named `Class` is a class for every c
class objects in Ruby. `Class` can be create
unnamed classes can be created by `Class`

Inherited Class

`Module`

Inherited Class

`Object`

Class Methods

`Class::inherited(` *`c`*`)`

Called when a subclass is defined. U

`Class::new([` *`superclass`*`=Object])`

Creates a new class.

Instance Methods

`Class` class doesn't inherit the `module_fu`

`c``.class_eval`

Alias for `c.module_eval`.

`c.name`

Returns the class name.

`c.new([arg ...])`

Creates an instance of the class. Any
the initialize method of the object cre

`c.superclass`

Returns the class's superclass.

3.4.10 Proc Objects and Bindi

The `Proc` class provides support for conve
them just like other objects in Ruby. The
can recreate its execution environment wh

you with a tool for packaging up an execu
`Binding` class.

Proc *Procedure object class*

`Proc` is an objectified block that is given t
calling the `proc` method or by using the bl

```
p1 = proc{|a| a + 1}    # Proc from
p2 = proc               # Proc from

def foo(&proc)          # Proc from
  proc.call(42)         # invoke Pr
end
```

`Proc::new`

`Proc::new {|` *`x`*`| ...}`

Converts the block into a `Proc` objec
with the calling method is converted
functions `lambda` and `proc`.

Instance Methods

`p[ arg...]`

`p.call([ arg...])`

Calls a Proc object.

`p.arity`

Returns the number of arguments acc
variable number of arguments, returr
arguments. Notice `{|a|}` gives `-1`, si
arguments are passed.

```
Proc.new{||}.arity          #=> 0
Proc.new{|a|}.arity         #=> -
```

```
    Proc.new{|a,b|}.arity      #=> 2
    Proc.new{|a,b,c|}.arity    #=> 3
    Proc.new{|*a|}.arity       #=> -
    Proc.new{|a,*b|}.arity     #=> -
```

Method *Method object class*

The method of an object that has been ma
using the method `obj.method(name)`.

Instance Methods

`m[ arg...]`

`m.arity`

Returns the number of arguments ac
number of arguments, returns `-n-1`,
arguments.

m.call([*arg...*])

Calls a `method` object.

m.to_proc

Converts *m* into a `Proc` object.

m.unbind

Returns an `UnboundMethod` object co

UnboundMethod *Method witho*

The method definition without a receiver
`UnboundMethod`. You have to bind `Unboun`
Created using the method `Module#instan`

Inherited Class

`Method`

Instance Method

`um.bind( obj)`

Returns callable `Method` object bound
from which *`UnboundMethod`* retrieved

```
unbound_plus = String.instance_
plus = unbound_plus.bind("a")
p plus.call("b")
unbound_plus.bind(1)
```

Binding *Encapsulated executio*

An object encapsulating the execution co

some place in the code. Created using the
second argument of the built-in function `e`

Continuation *Continuation clas*

Allows a return to (continuation of) execu
using the built-in function `callcc`. See `ca`

Instance Method

`c.call([ arg...])`

> Continues execution from the end of
> Continuation. `callcc` returns *`arg`*`...`

3.4.11 Miscellaneous Classes a

Of course, there's a whole lot of other stuf

program: things like garbage collection (G
FalseClass), the ability to poke around a
ObjectSpace), and so on. There's nothing
Ruby's philosophy of transparency, so div

GC *GC module*

GC module is a collection of garbage colle

Module Methods

disable

Disables GC

enable

Enables GC

`start`

Starts GC

Instance Method

`g.garbage_collect`

Starts GC

ObjectSpace *ObjectSpace modu*

`ObjectSpace` module provides manipulati

Module Functions

`_id2ref( id)`

Obtains object from *`id`*. Do not use t especially in finalizers. *id* is already

`define_finalizer( obj, proc)`

`define_finalizer( obj) {| id| ...}`

Creates a finalizer for *obj*. *obj* shoul
from the finalizers.

```
class Foo
def Foo::finalizer(io)     # ty
io.close
end
def initialize(path)
@io = open(path)
ObjectSpace.define_finalizer(se
end
```

`each_object([ c]) {| x| ...}`

Calls the block once for all objects. V
for all objects that match *c* or are sub

`garbage_collect`

Starts GC. Alias for `GC::start`.

`undefine_finalizer(` *`obj`*`)`

Removes all finalizers for *`obj`*.

NilClass *Nil class*

The only instance of `NilClass` is `nil`. `Nil`

TrueClass *True class*

The only instance of `TrueClass` is `true`. `T` which evaluate both operands before exec

Instance Methods

`true & *other*`

Logical AND, without short circuit b

`true | *other*`

Logical OR, without short circuit be

`true ^ *other*`

Logical exclusive Or (XOR)

FalseClass *False class*

The only instance of `FalseClass` is `false`
operations, which do evaluate both operar

Instance Methods

`false & other`

Logical AND, without short circuit b

`false | other`

Logical OR, without short circuit bel

`false ^ other`

Exclusive Or (XOR)

Data *C data wrapper class*

`Data` is an external language data wrappe
methods of its own.

Marshal *Object storage module*

`Marshal` is a module for dumping objects

Module Functions

`dump(` *`obj`*`[,` *`port`*`][,` *`level`*`])`

> Dumps an object. Dumps to port if a
> specified, *`obj`* is returned as a dumpe
> to that depth are dumped.

`load(` *`from`*`)`

`restore(` *`from`*`)`

Restores a dumped object. The string

Range *Range class*

`Range` is a class for interval. Ranges can b
the `Range::new` method.

Included Module

`Enumerable`

Class Method

`Range::new(` *`first`*`,` *`last`*`[,` *`excl`*`=false`

Creates a `Range` object. Does not inc
`last` should be comparable using `<=`

Instance Methods

r `===` *`other`*

Returns `true` if *`other`* is within the ra

r`.begin`

r`.first`

Returns the first object in the range.

r`.each {|` *`x`*`|` *`...`*`}`

Executes the block for each object w

```
(1..5).each {|x|
puts x          # prints 1 to 5
}
(1...5).each {|x|
puts x           # prints 1 to 4
}
```

`r.end`

`r.last`

Returns the last object in the range.

`r.size`

`r.length`

Returns the number of objects in the
other than an integer, the number of

Struct *Structure class*

`Stuct` is a abstract class that collects nam
to generate your own `Struct` class (subcl
returns new `Struct` class.

Example

```
S = Struct::new(:foo, :bar)
s = S::new(1,2)
s.foo                  # => 1
s.bar = 5              # update the mem
s.bar                  # => 5
s                      # => #<S foo=1,
```

Included Module

`Enumerable`

Class Method

`Struct::new([ name,] mem...)`

> Creates a new structure class contain
> is given, the structure class is bound
> `Struct::Passwd`. Note that `Struct:`
> but rather a class that is used as a ten

Structure Class Methods

`S::members`

Returns an array of member names.

`S::new( value...)`

Creates a new structure object. *value*
member and must match the number
created.

Instance Methods

`s[ mem]`

Returns the value of member *mem* wh
integer, the value of the *mem*th memb

`s[ mem]= value`

Sets the value of member *mem*. *mem* m

`s.each {| x|...}`

Calls block once for each member.

`s.members`

Returns an array of member names.

`s.values`

Returns an array containing the value

Time *Time class*

Time is an object corresponding to a certa
seconds since the *epoch*, 00:00:00, Januar

system's local time and UTC at the same t

Included Module

`Comparable`

Class Methods

`Time::at( *time*[, *usec*=0])`

Creates a `Time` object. `time` may be a
number of seconds elapsed since the

`Time::gm( *year*[, *month*=1[, *day*=1[, *ho*`

see `Time::utc(year[,month=1[,day`
`[,usec=]]]]]])`

`Time::local( *year*[, *month*=1[, *day*=1[`
`*usec*=0]]]]]])`

`Time::mktime(` *`year`*`[,` *`month`*`=1[,` *`day`*`=1`
`usec``=0]]]]]])`

Creates a `Time` object interpreted in t

`Time::new`

`Time::now`

Creates a `Time` object expressing the

`Time::times`

Returns a `Tms` structure containing us
times system call. Here are the `Tms` s

`utime`

User CPU time

`stime`

System CPU time

`cutime`

CPU time elapsed for user child proc

`cstime`

CPU time elapsed for system child p

`Time::utc( *year*[, *month*=1[, *day*=1[, h`

`Time::gm( *year*[, *month*=1[, *day*=1[, ho`

Creates a `Time` object interpreted as U
known as GMT).

Instance Methods

`t` `+` *`n`*

Returns a `Time` object with *n* number

`t - x`

If *x* is another `Time` object, the time c
x is a number, a `Time` object with *x* n

`t <=> other`

`t > other`

`t >= other`

`t < other`

`t <= other`

Time comparisons.

`t.asctime`

`t.ctime`

Returns `t` as a string.

`t.day`

`t.mday`

Returns the day of the month (1-31)

`t.gmtime`

See `t.utc`

`t.gmtime?`

See `t.utc?`

`t.hour`

Returns the hour of the day (0-23) fo

`t.isdst`

Returns `true` if `t` occurs during dayli

`t.localtime`

Turns on representation mode of `t` to

`t.min`

Returns the minute of the hour (1-59

`t.mon`

`t.month`

Returns the month of the year (1-12)

`t.sec`

Returns the second of the minute (1-
minute due to leap second.

`t.strftime( format)`

Formats *t* according to formatting di

%A	Full weekday name (Sunday
%a	Abbreviated weekday name
%B	Full month name (January,
%b	Abbreviated month name (J
%c	Date and time
%d	Day of the month in decima
%H	Hour, 24-hour clock (00-23)

%I	Hour, 12-hour clock (01-12
%j	Day of the year (001-366)
%M	Minutes (00-59)
%m	Month in decimal (01-12)
%p	Meridian indicator (A.M. o
%S	Seconds (00-60)
%U	Week number, with the first 53)
%W	Week number, with the first (00-53)

%w	Day of the week, Sunday be
%X	Time only
%x	Date only
%Y	Year with century
%y	Year without century (00-9
%Z	Time zone
%%	Literal % character

t.to_f

Returns the value of *t* as a Float of
microseconds.

`t.to_i`

`t.tv_sec`

Returns the value of *t* as an integer r

`t.tv_usec`

`t.usec`

Returns just the number of microsec

`t.utc`

`t.gmtime`

Converts *t* to UTC, modifying the re

`t.utc?`

`t.gmt?`

Returns `true` if *t* represents a time ir

`t.wday`

Returns the day of the week (0-6, Su

`t.yday`

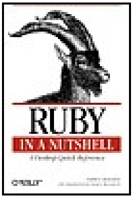
Returns the day of the year (1-366) f

`t.year`

Returns the year for *t*.

`t.zone`

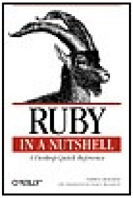
Returns the local time zone for *t*.



Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 4. Standard Library Reference

We will now explore the useful libraries that come with the standard Ruby distribution, from network access via HTTP and CGI programming to data persistence using the DBM library.


Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 4. Standard Library Reference

4.1 Standard Library

The Ruby standard library extends the fou
classes and abstractions for a variety of pr
programming, operating-system services,

flexible capabilities at a high level of abst
powerful Ruby scripts useful in a variety

Many common tasks are performed by Ru
these tasks include network access such a
access, controlling processes with threads
classes, and manipulating dates. These are
with all standard distributions of Ruby; w
from your programs, they will be availabl
these libraries yourself? Probably. Would
exhaustively tested, optimized, and debug
great time saver. And as Ruby grows and
everyone's benefit.

Although not every library section will co
each section is as follows:

- Required library

- Example
- Inherited class
- Class methods
- Instance methods

4.1.1 Network

Use Ruby's network classes to let your scr
UDP as a client, a server, or both. These li
Internet protocols and classes that make a
crawl up the protocol stack and find suppo
IMAP, and so on. All have an intuitive, tra
This is the largest group of libraries and o

Oh, and don't worry. There's support for d
`CGI::Cookie` and `CGI::Session` classes.

BasicSocket *Socket-related supe*

`BasicSocket` is an abstract base class for
provides common behavior among `Socke`

Required Library

require 'socket'

Inherited Class

`IO`

Class Methods

`BasicSocket::do_not_reverse_lookup`

Returns `true` if a query returns nume

`BasicSocket::do_not_reverse_lookup=`

Sets `reverse_lookup` status

Instance Methods

`s.getpeername`

Returns information on this connecti
into a string.

`s.getsockname`

Returns information on *`s`* as a `struct`

`s.getsockopt( lev, optname)`

Gets the specified socket option.

`s.setsockopt( lev, optname, value)`

Sets the specified socket option.

`s.shutdown([ how=2])`

Shuts down the socket connection. `0`

`s.recv( len[, flags])`

Receives data from `s`, and returns it a

`s.send( mesg, flags[, to])`

Sends data over the socket `s`, returnin
`struct sockaddr` packed into a strin

IPSocket *IP socket class*

`IPSocket` class is a base class of `TCPSock`

common behavior among Internet Protoc
support IPv6, if the native platform suppo

Required Library

require 'socket'

Inherited Class

`BasicSocket`

Class Method

`IPSocket::getaddress( *host*)`

Returns the IP address of the specifie
such as `127.10.0.1` (IPv4) or `::1` (II

Instance Methods

`*s*.addr`

Returns an array containing informat
hostname, and IP address)

```
s = TCPSocket.open("www.ruby-la
s.addr# => ["AF_INET", 4030, "c
     "192.168.1.198"]
```

`s.peeraddr`

Returns an array containing informat
`s.addr`

```
s = TCPSocket.open("www.ruby-la
s.recvfrom(255)
# => ["Wed Aug  1 00:30:54 2001
      "210.251.121.214"]]
```

`s.recvfrom( len[, flags])`

Receives data and returns it in an arr
sender's socket in the same format as

UDPSocket *UDP socket class*

`UDPSocket` is a class for User Datagram P
unreliable protocol.

Required Library

require 'socket'

Inherited Class

`IPSocket`

Class Methods

`UDPSocket::new([` *`socktype`*`=Socket::AI`

`UDPSocket::open([` *`socktype`*`=Socket::A`

Creates a UDP datagram socket

Instance Methods

`s.bind( host, port)`

Binds the socket to *port* on *host*. *ho*
`INADDR_ANY` or `<broadcast>` for `INAI`

`s.connect( host, port)`

Connects the socket to *port* on *host*
`INADDR_ANY` or `<broadcast>` for `INAI`

`s.send( mesg, flags[, to])`

`s.send( mesg, flags[, host, port])`

Sends data on a socket *s*, returning th
arguments are specified, the destinati

connection. Otherwise, it may be spe
the method with three arguments or b
four arguments.

TCPSocket *TCP/IP socket class*

`TCPSocket` is a class for Transmission Co
oriented, reliable protocol.

Required Library

require 'socket'

Example

```
require 'socket'

host=(if ARGV.length == 2; ARGV.shi
print("Trying ", host, " ...")
```

```
STDOUT.flush
s = TCPsocket.open(host, ARGV.shift
print(" done\n")
print("addr: ", s.addr.join(":"), "
print("peer: ", s.peeraddr.join(":"
while gets(  )
  s.write($_)
  print(s.readline)
end
s.close
```

Inherited Class

`IPSocket`

Class Methods

`TCPSocket::new(` *`host`*`,` *`service`*`)`

`TCPSocket::open(` *`host`*`,` *`service`*`)`

Opens a TCP connection to *`host`* for

TCPServer *TCP/IP server socke*

TCPServer is a class for server-side TCP s
connection by the accept method, then re
client.

Required Library

require 'socket'

Example

```
require 'socket'

gs = TCPserver.open(0)
addr = gs.addr
addr.shift            # removes "AF
printf("server is on %s\n", addr.jo
while true
   Thread.start(gs.accept) do |s|
     print(s, " is accepted\n")
     while s.gets
       s.write($_)
     end
     print(s, " is gone\n")
     s.close
   end
 end
```

Inherited Class

```
TCPSocket
```

Class Methods

TCPServer::new([*host*="localhost",]

TCPServer::open([*host*="localhost",

Creates a server socket

Instance Method

`s.accept`

Waits for a connection and returns a

UNIXSocket *Unix domain sock*

`UNIXSocket` is a class for the Unix domai

Required Library

require 'socket'

Inherited Class

`BasicSocket`

Class Methods

`UNIXSocket::new( *path*)`

`UNIXSocket::open( *path*)`

Creates a Unix domain socket

Instance Methods

s`.addr`

Returns an array containing informat

s`.path`

Returns the path of the Unix domain

s`.peeraddr`

Returns an array containing informat

`s.addr`

`s.recvfrom( len[, flag=0])`

Receives data and returns it in an arr
sender's socket in the same format as

UNIXServer Unix domain serve

`UNIXServer` is a class for server-side Unix
client connection by the `accept` method, t
the client.

Required Library

require 'socket'

Inherited Class

`UNIXSocket`

Class Methods

`UNIXServer::new( path)`

`UNIXServer::open( path)`

Creates a server socket

Instance Method

`s.accept`

Waits for a connection and returns a

Socket *General socket class*

The `Socket` class is necessary to gain acce interfaces. Interface structures can be crea

Required Library

require 'socket'

Inherited Class

`BasicSocket`

Class Methods

`Socket::for_fd( fd)`

Creates a socket object correspondin

`Socket::getaddrinfo( host, port[, fa`

Returns an array containing socket a
number, hostname, host IP address, p

```
Socket::getaddrinfo("www.ruby-
lang.org", "echo", Socket::AF_I
# => [["AF_INET", 7, "www", "21
```

`Socket::gethostbyaddr(` *`addr`*`[,` *`type`*`=S`

Returns an array containing socket ad
number, hostname, host IP address, p

```
Socket::getaddrinfo("www.ruby-
lang.org", "echo", Socket::AF_I
# => [["AF_INET", 7, "www", "21
```

`Socket::gethostbyname(` *`name`*`)`

Returns an array containing host info

```
Socket.gethostbyaddr(([127,0,0,
# => ["ev", ["localhost", "ev.n
```

`Socket::gethostname`

Returns the current hostname.

`Socket::getnameinfo(` *`addr`*`[,` *`flags`*`])`

Returns an array containing the name
specified socket address information
a string or an array (address family, p

```
sockaddr = [Socket::AF_INET, 80
Socket::getnameinfo(sockaddr)

Socket::getnameinfo(["AF_INET",
```

`Socket::getservbyname(` *`service`*`[,` *`pro`*

Returns the port number for *`service`*

```
Socket::getservbyname("http") #
```

`Socket::new(domain, type, proto)`

`Socket::open(domain, type, proto)`

Creates a socket.

`Socket::socketpair( domain, type, pr`

`Socket::pair( domain, type, proto)`

Returns an array containing a pair of

Instance Methods

`s.accept`

Waits for a connection and, once one
array that also includes a `struct soc`

`s.addr`

Synonym for `s.getsockname`. Returr

`s.bind( addr)`

Binds *s* to `addr`, a `sockaddr` structur

`s.connect( addr)`

Connects *s* to `addr`, a `sockaddr` struc

`s.listen( backlog)`

Specifies the size of the *backlog* que

`s.recvfrom( len[, flags])`

Receives data and returns it in an arr
sender's socket in the form of a `sock`

`s.peeraddr`

Synonym for `s.getpeername`. Retur

Constants

The following constants are defined for us

```
AF_INET
AF_UNIX
MSG_OOB
MSG_PEEK
SOCK_DGRAM
SOCK_STREAM
SOL_SOCKET
SO_KEEPALIVE
SO_LINGER
SO_SNDBUF
...
```

These constants are also defined in the mo
including them in your code.

Net::FTP *FTP connection class*

`Net::FTP` is a class for File Transfer Proto

Required Library

require 'net/ftp'

Example

```
require 'net/ftp'

ftp = Net::FTP::new("ftp.ruby-lang.
ftp.login("anonymous", "matz@ruby-l
ftp.chdir("/pub/ruby")
tgz = ftp.list("ruby-*.tar.gz").sor
print "the latest version is ", tgz
ftp.getbinaryfile(tgz, tgz)
ftp.close
```

Class Methods

`Net::FTP::new([` *host*`[,` *user*`[,` *passwd*

`Net::FTP::open(` *host*`[,` *user*`[,` *passwd*

Creates a `Net::FTP` object

Instance Methods

`f.abort`

Aborts the previous command.

`f.acct( acct)`

Sets the account.

`f.chdir( path)`

Changes the current directory.

`f.close`

Closes the connection.

`f.closed?`

Returns `true` if the connection is clo

`f.connect( host[, port=21])`

Connects to host.

`f.debug_mode`

Returns the debug mode status.

`f.debug_mode= bool`

Sets the debug mode status.

`f.delete( file)`

Deletes a file.

`f.getbinaryfile( remote, local[, blo`

`f.getbinaryfile( remote, local[, blo`

`f.gettextfile( remote, local[, callb`

`f.gettextfile( remote, local) {| data`

Retrieves a remote file from the serv
executed with the retrieved data. `get`

`f.help([ arg])`

Displays help.

`f.lastresp`

Returns the server's last response.

`f.list( path...)`

`f.dir( path...)`

`f.ls( path...)`

Returns an array of file information i
iterates through the listing.

```
f.list("/pub/ruby") # =>
   ["drwxr-xr-x   2 matz     us
```

`f.login([ user="anonymous"[, passwd[`

Logs into the server.

`f.mkdir( path)`

Creates a directory.

`f.mtime( file[, local=false])`

Returns the last modification time of
time, otherwise as Coordinated Univ

`f.nlst([ dir])`

Returns an array of filenames in the

```
f.nlst("/pub/ruby") # => ["/pub
```

`f.putbinaryfile( remote, local[, blo`

`f.putbinaryfile( remote, local[, blo`

`f.puttextfile( remote, local[, callba`

`f.puttextfile( remote, local) {| data`

Transfers a file. If callback or a bloc
run. `puttextfile` performs newline

`f.pwd`

`f.getdir`

Returns the current directory.

`f.passive`

Returns `true` if passive mode is enab

`f.passive= bool`

Sets passive mode on or off.

`f.quit`

Exits the FTP session.

`f.rename( old, new)`

Renames filename *`old`* to *`new`*.

`f.rmdir( path)`

Removes the directory specified by

`f.resume`

Returns `true` if resumption of file tra

`f.resume= bool`

Sets file transfer resumption on or of

`f.return_code`

Returns the newline code of the curr

`f.return_code= ret`

Sets the newline code of the current

`f.size( file)`

Returns the size of file.

`f.status`

Returns the status.

`f.system`

Returns system information.

`f.welcome`

Returns the server's welcome messag

Net::HTTP *HTTP connection c*

`Net::HTTP` is a class for Hypertext Transf

Required Library

require 'net/http'

Example

```
require 'net/http'
```

```
h = Net::HTTP::new("www.ruby-lang.c
resp, data = h.get("/en/index.html"
print data
```

Class Methods

`Net::HTTP::new([ *host*="localhost"[,`

`Net::HTTP::start([ *host*="localhost"`

`Net::HTTP::start([ *host*="localhost"`
`*http*| *...*}`

> Creates a `Net::HTTP` connection obje
> executed with the `Net::HTTP` object
> closed automatically when the block

Instance Methods

`*h*.finish`

Closes the HTTP session.

`h.get( path[, header[, dest]])`

`h.get( path[, header]) {| str| ...}`

Retrieves data from *path* using a GE
HTTPResponse object and the data. *h*
and values. *dest* may be a string to v
specified, the retrieved data is passed

`h.head( path[, header])`

Sends a HEAD request for *path*, and re

`h.post( path, data[, header[, dest]])`

`h.post( path, data[, header]) {| str|`

Sends *data* to *path* using a POST requ

`HTTPResponse` object and the reply b
request type is different, the block ar
handled in the same way as `h.get`.

`h.start`

`h.start {|http| ...}`

Starts an `HTTP` session. If a block is s
block exits.

Net::IMAP *IMAP access class*

`Net::IMAP` is a class for Internet Message
side connection. IMAP4 allows you to sto

Required Library

require "net/imap"

Example

```
require "net/imap"
imap = Net::IMAP::new("imap.ruby-la
  imap.login("matz", "skwkgjv;")
  imap.select("inbox")
  fetch_result = imap.fetch(1..-1,
  search_result = imap.search(["BOD
  imap.disconnect
```

Class Methods

`Net::IMAP::add_authenticator(` *`auth_`*

Adds an authenticator for `Net::IMAP`

`Net::IMAP::debug`

Returns `true` if in the debug mode.

`Net::IMAP::debug=` *`bool`*

Sets the debug mode.

`Net::IMAP::new(` *`host`*`[,` *`port`*`=143])`

Creates a new `Net::IMAP` object and
named *`host`*.

Instance Methods

`imap``.append(` *`mailbox`*`,` *`message`*`[,` *`flag`*

Appends the *`message`* to the end of th

```
imap.append("inbox", <<EOF.gsub
Subject: hello
From: shugo@ruby-lang.org
To: shugo@ruby-lang.org

hello world
```

```
    EOF
```

imap.authenticate(*auth_type*, *arg...*

Authenticates the client. The *auth_t*
authentication mechanism to be used
"CRAM-MD5" for the *auth_type*.

```
imap.authenticate('CRAM-MD5", '
```

imap.capability

Returns an array of capabilities that t

```
imap.capability  # => ["IMAP4",
```

imap.check

Requests a checkpoint of the current

imap.close

Closes the current mailbox. Also per
messages that have the `\Deleted` fla

`imap.copy( mesgs, mailbox)`

Copies *`mesgs`* in the current mailbox
an array of message sequence numbe

`imap.create( mailbox)`

Creates a new *`mailbox`*.

`imap.delete( mailbox)`

Removes the *`mailbox`*.

`imap.disconnect`

Disconnects from the server.

`imap.examine(mailbox)`

Selects a *`mailbox`* as a current mailbo
accessed. The selected mailbox is ide

`imap``.expunge`

Removes from the current mailbox a

`imap``.fetch(` *`mesgs`*`,` *`attr`*`)`

Fetches data associated with a messa
message sequence numbers or an `Ra`
`Net::IMAP::FetchData`.

```
data = imap.uid_fetch(98, ["RFC
data.seqno                       #
data.attr["RFC822.SIZE"]         #
data.attr["INTERNALDATE"]        #
data.attr["UID"]                 #
```

`imap``.greeting`

Returns an initial greeting response f

imap.list(*dir*, *pattern*)

Returns an array of mailbox informa
value is an array of `Net::IMAP::Mai`
(which matches any characters) and `%`
delimiter).

```
imap.list("foo", "*")# matches
imap.list("foo", "f%")
                    # matches
```

imap.login(*user*, *password*)

Logs into the server.

imap.logout

Logs out from the server.

`imap.lsub( refname, mailbox)`

Returns an array of subscribed mailb
return value is an array of `Net::IMAF`
wildcards `*` (which matches any char
except delimiter).

`imap.noop`

Sends a NOOP command to the serv

`imap.rename( mailbox, newname)`

Renames the *`mailbox`* to *`newname`*.

`imap.responses`

Returns recorded untagged responses

```
imap.select("inbox")
imap.responses["EXISTS"][-1]
```

```
    imap.responses["UIDVALIDITY"][-
```

imap.search(*keys*[, *charset*])

Searches the mailbox for messages th
returns an array of message sequence

```
    imap.search(["SUBJECT", "hello"
    imap.search('SUBJECT "hello"')
```

imap.select(*mailbox*)

Selects a *mailbox* as a current mailbo
accessed.

imap.sort(*sort_keys*, *search_keys*, *c*

Returns an array of message sequenc
according to the *sort_keys*.

```
    imap.sort(["FROM"], ["ALL"], "U
    imap.sort(["DATE"], ["SUBJECT",
```

```
imap.status( mailbox, attr)
```

Returns the status of the *mailbox*. Th

```
    imap.status("inbox", ["MESSAGES
         {"RECENT"=>0, "MESSAGES"=>
```

```
imap.store( mesgs, attr, flags)
```

Stores data associated with a messag
message sequence numbers or a Rang

```
    # add \Deleted to FLAGS attribu
    imap.store(6..8, "+FLAGS", [:De
```

```
imap.subscribe( mailbox)
```

Appends the specified *mailbox* to th

`imap.unsubscribe( mailbox)`

Removes the specified `mailbox` from

`imap.uid_copy( mesg, mailbox)`

Copies `mesgs` in the current mailbox
an array of unique message identifie

`imap.uid_fetch( mesgs, attr)`

Fetches data associated with a messa
of unique message identifiers or an R
`Net::IMAP::FetchData`.

`imap.uid_search( keys[, charset])`

Searches the mailbox for messages t
returns an array of unique identifiers

imap.uid_sort(*sort_keys*, *search_key*

Returns an array of unique message i
according to the *sort_keys*.

imap.uid_store(*mesgs*, *attr*, *flags*)

Stores data associated with a messag
message identifiers or a `Range` object
`Net::IMAP::FetchData`.

Net::POP3 *POP3 connection cl*

`Net::POP3` is a class for Post Office Proto
POP3 is a simple protocol that retrieves in

Required Library

require 'net/pop'

Example

```
require 'net/pop'

pop = Net::POP3::new("pop.ruby-lang
# authenticate just for SMTP before
pop.start("matz", "skwkgjv;") {
  mails = pop.mails            # arra
}
```

Class Methods

`Net::POP3::new([` *addr*`="localhost"[,`

Creates a new `Net::POP3` object.

`Net::POP3::start([` *addr*`="localhost"`

`Net::POP3::start([` *addr*`="localhost"`

Equivalent to `Net::POP3::new(`*`addr`*
`Net::POP3` object is passed to the blo
terminated when the block exits.

Instance Methods

`p``.each {|mail| `*`...`*`}`

Synonym for *`p`*`.mails.each`.

`p``.finish`

Closes the POP3 session.

`p``.mails`

Returns an array of `Net::POPMail` ob

`p``.start( `*`acct`*`, `*`passwd`*`)`

`p``.start( `*`acct`*`, `*`passwd`*`) {|pop| `*`...`*`}`

Starts a POP3 session. If a block is s
block exits.

Net::APOP *APOP connection c

The `Net::APOP` class has the same interfa
method of authentication.

Required Library

require 'net/pop'

Inherited Class

`Net::POP3`

Net::POPMail *POP mail class*

The `Net::POPMail` class is used by classe
individual message objects.

Required Library

require 'net/pop'

Instance Methods

m`.all([` *dest*`])`

m`.mail([` *dest*`])`

m`.pop([` *dest*`])`

Retrieves the contents of mail messa
appended to it using the `<<` method. I
of each message as a string and run c

`m.delete`

Deletes the message.

`m.deleted?`

Returns `true` if the message has beer

`m.header([ dest])`

Returns the message header.

`m.size`

Returns the message size in bytes.

`m.top( lineno[, dest])`

Returns the message header and *line*

Net::SMTP *SMTP connection c*

`Net::SMTP` is a class for Simple Mail Tran
SMTP is a protocol to talk to Mail Transfe

Required Library

require 'net/smtp'

Example

```
require 'net/smtp'

user = "you@your-domain.com"
from = "matz@ruby-lang.org"
server = "localhost"
smtp = Net::SMTP::new(server)
smtp.start
smtp.sendmail(<<BODY, from, user)
From: matz@ruby-lang.org
```

```
Subject: this is a test mail.

this is body
BODY
smtp.finish
```

Class Methods

`Net::SMTP::new([ `*addr*`="localhost"[,`

Creates a new `Net::SMTP` object.

`Net::SMTP::start([ `*addr*`="localhost"`

`Net::SMTP::start([ad`*d*`r="localhost"`

Equivalent to `Net::SMTP::new(`*addr*
`Net::SMTP` object is passed to the blo
terminated when the block exits.

Instance Methods

`s.finish`

Closes an SMTP session.

`s.ready( from, to) {| adapter| ...}`

Sends a message, passing an `adapte`
calling the adapter's `write` method.

`s.start([ domain[, account[, passwor`

`s.start([ domain[, account[, passwor`

Starts an SMTP session. An `Net::SM`
The session is terminated when the b

`s.send_mail( mailsrc, from, to)`

`s.sendmail( mailsrc, from, to)`

Sends mail. *`to`* may be either a string

Net::Telnet *Telnet connection cl*

`Net::Telnet` is a class for a Telnet conne
client but also a useful tool to interact wit

When a block is specified with class and i
it's passed status output strings from the s

Required Library

require 'net/telnet'

Class Method

`Net::Telnet::new(options)`

Creates a `Net::Telnet` object. *`options`* m
following options:

Key	Function
`Binmode`	Binary mode
`Host`	Telnet server
`Output_log`	Output log
`Dump_log`	Dump log
`Port`	Port to connect to
`Prompt`	Pattern matching the server's

`Telnetmode`	Telnet mode
`Timeout`	Timeout
`Waittime`	Wait time
`Proxy`	Proxy

Instance Methods

Besides the following methods, the `Net::` `Socket` object, so that methods provided b also available for `Net::Telnet`.

`*t*.binmode`

Returns `true` if binary mode is enabl

`*t*.binmode= *bool*`

Sets binary mode on or off.

`t.cmd( options)`

Sends a command to the server. *opti*
the server or a hash specifying one o

Key	Function	Default v
`String`	String to be sent	(Required)
`Match`	Pattern to match	Value of `Promp`
`Timeout`	Timeout	Value of `Timeo`

`t.login( options)`

`t.login( user[, passwd])`

Logs in to the server. The following

Key	**Function**
`Name`	Username
`Password`	Password

`t.print( str)`

Sends `str` to the server, performing T

`t.telnetmode`

Returns `true` if Telnet mode is enabl

`t.telnetmode= bool`

Sets Telnet mode on or off.

`t.waitfor( options)`

Waits for a response from the server.
`t.cmd`.

`t.write( str)`

Sends `str` to the server without perf

CGI CGI support class

`CGI` provides useful features to implemen
programs, such as retrieving CGI data fro
generating the HTTP header and the HTM

Example

```
require 'cgi'

     cgi = CGI::new("html3")

     input, = cgi["input"]
     if input
       input = CGI::unescape(input)
     end
     p input

     begin
       value = Thread::new{
         $SAFE=4
         eval input
       }.value.inspect
     rescue SecurityError
       value = "Sorry, you can't do
     end

     cgi.out {
       cgi.html{
         cgi.head{cgi.title{"Walter
         cgi.body{
           cgi.form("post", "/cgi-b
             "input your favorite e
             cgi.text_field("input"
             cgi.br +
             "the result of you inp
             CGI::escapeHTML(value)
             cgi.br +
             cgi.submit
           }
         }
       }
     }
```

Required Library

require 'cgi'

Class Methods

`CGI::new([ level="query"])`

Creates a CGI object. *`level`* may be
HTML levels is specified, the follow

conforming to that level:

`query`

No HTML output generated

`html3`

HTML3.2

`html4`

HTML4.0 Strict

`html4Tr`

HTML4.0 Transitional

`html4Fr`

HTML4.0 Frameset

`CGI::escape( str)`

Escapes an unsafe string using URL-

`CGI::unescape( str)`

Expands a string that has been escap

`CGI::escapeHTML( str)`

Escapes HTML special characters, in

`CGI::unescapeHTML( str)`

Expands escaped HTML special char

`CGI::escapeElement( str[, element...`

Escapes HTML special characters in

`CGI::unescapeElement(` *`str`*`,` *`element`*`[,`

Expands escaped HTML special char

`CGI::parse(` *`query`*`)`

Parses the query and returns a hash c

`CGI::pretty(` *`string`*`[,` *`leader`*`=" "])`

Returns a neatly formatted version of
written at the beginning of each line.

`CGI::rfc1123_date(` *`time`*`)`

Formats the data and time according
00:00:00 GMT).

Instance Methods

`c[ name]`

Returns an array containing the valu

`c.checkbox( name[, value[, check=fal`

`c.checkbox( options)`

Returns an HTML string defining a c
specified in a hash passed as an argu

`c.checkbox_group( name, value...)`

`c.checkbox_group( options)`

Returns an HTML string defining a c
specified in a hash passed as an argu

`c.file_field( name[, size=20[, max]]`

`c.file_field( options)`

Returns an HTML string defining a f

`c.form([ method="post"[, url]]) { ..`

`c.form( options)`

Returns an HTML string defining a f
produced by its output creates the co
specified in a hash passed as an argu

`c.cookies`

Returns a hash containing a `CGI::Co`
a cookie.

`c.header([ header])`

Returns a CGI header containing the

its key-value pairs are used to create

`c.hidden( name[, value])`

`c.hidden( options)`

Returns an HTML string defining a H
in a hash passed as an argument.

`c.image_button( url[, name[, alt]])`

`c.image_button( options)`

Returns an HTML string defining an
specified in a hash passed as an argu

`c.keys`

Returns an array containing the field

`c.key?( name)`

`c.has_key?( name)`

`c.include?( name)`

Returns `true` if the form contains the

`c.multipart_form([ url[, encode]]) {`

`c.multipart_form( options) { ...}`

Returns an HTML string defining a r
string produced by its output creates
be specified in a hash passed as an ar

`c.out([ header]) { ...}`

Generates HTML output. Uses the st
the body of the page.

`c.params`

Returns a hash containing field name

`c.params= hash`

Sets field names and values in the fo

`c.password_field( name[, value[, siz`

`c.password_field( options)`

Returns an HTML string defining a p
specified in a hash passed as an argu

`c.popup_menu( name, value...)`

`c.popup_menu( options)`

`c.scrolling_list( name, value...)`

c.scrolling_list(*options*)

Returns an HTML string defining a p
in a hash passed as an argument.

c.radio_button(*name*[, *value*[, *checke*

c.radio_button(*options*)

Returns an HTML string defining a r
in a hash passed as an argument.

c.radio_group(*name*, *value*...)

c.radio_group(*options*)

Returns an HTML string defining a r
specified in a hash passed as an argu

c.reset(*name*[, *value*])

`c.reset( options)`

Returns an HTML string defining a r
a hash passed as an argument.

`c.text_field( name[, value[, size=40`

`c.text_field( options)`

Returns an HTML string defining a t
hash passed as an argument.

`c.textarea( name[, cols=70[, rows=10`

`c.textarea( options) {...}`

Returns an HTML string defining a t
produced by its output creates the co
specified in a hash passed as an argu

HTML Generation Methods

In addition to the previous instance metho
methods, which generate HTML tag strin
when the `CGI` object was created. These m
adding any specified tags to a body create
attributes may be specified in a hash that i

Here are the tags common to `html3`, `html`

`a`	`address`	`area`	`b`	`base`
`big`	`blockquote`	`body`	`br`	`captio`
`cite`	`code`	`dd`	`dfn`	`div`
`dl`	`doctype`	`dt`	`em`	`form`
`h1`	`h2`	`h3`	`h4`	`h5`

h6	head	hr	html	i
img	input	kbd	li	link
map	meta	ol	option	p
param	pre	samp	script	select
small	strong	style	sub	submi
sup	table	td	th	title
tr	tt	ul	var	

Here are the `html3` tags:

applet	basefont	center	dir	fo

isindex	listing	menu	plaintext	st
u	xmp			

Here are the `html4` tags:

abbr	acronym	bdo	button
colgroup	del	fieldset	ins
legend	noscript	object	optgroup
span	tbody	tfoot	thead

Here are the `html4Tr` tags:

abbr	acronym	applet	basefont	bo

`button`	`center`	`col`	`colgroup`	`de`
`dir`	`fieldset`	`font`	`iframe`	`in`
`isindex`	`label`	`legend`	`map`	`me`
`noframes`	`noscript`	`object`	`optgroup`	`q`
`s`	`span`	`strike`	`tbody`	`tf`
`thead`	`u`			

Here are the `htmlFr` tags:

`abbr`	`acronym`	`applet`	`basefont`	`bo`
`button`	`center`	`col`	`colgroup`	`de`
`dir`	`fieldset`	`font`	`frame`	`fr`

iframe	ins	isindex	label	le
menu	noframes	noscript	object	op
q	s	span	strike	tb
tfoot	thead	u		

Object Attributes

The CGI class has the following accessors

accept	Acceptable MIME t
accept_charset	Acceptable characte
accept_encoding	Acceptable encodin

`accept_language`	Acceptable languag
`auth_type`	Authentication type
`raw_cookie`	Cookie data (raw st
`content_length`	Content length
`content_type`	Content type
`From`	Client email addres
`gateway_interface`	CGI version string
`path_info`	Extra path
`path_translated`	Converted extra pat

`Query_string`	Query string
`referer`	Previously accessed
`remote_addr`	Client host address
`remote_host`	Client hostname
`remote_ident`	Client name
`remote_user`	Authenticated user
`request_method`	Request method (`GE`
`script_name`	Program name
`server_name`	Server name

`server_port`	Server port
`server_protocol`	Server protocol
`server_software`	Server software
`user_agent`	User agent

CGI::Cookie *HTTP cookie clas*

`CGI::Cookie` represents the HTTP cookie
sessions.

Required Library

require 'cgi'

Object Attributes

The CGI::Cookie class has the following

`c.name`	Cookie name
`c.value`	An array of cookie values
`c.path`	The cookie's path
`c.domain`	The domain
`c.expires`	The expiration time (as a `Tim`
`c.secure`	True if secure cookie

CGI::Session *CGI session class*

`CGI::Session` maintains a persistent sessi
information is represented by string to stri
stored via the user-defined database class.

Required Library

require 'cgi/session'

Example

```
request 'cgi/session'

cgi = CGI::new("html3")
s = CGI::Session(cgi)

if s["last_modified"]
  # previously saved data
  t = s["last_modified"].to_i
else
  t = Time.now.to_i
  # save data to session database
  s["last_modified"] = t.to_s
end
  # ... continues ...
```

Class Methods

`CGI::Session::new(` *`cgi`*`[,` *`option`*`])`

Starts a new CGI session and returns
`option` may be an option hash specif

Key	Functi
`session_key`	Key name holding th
`session_id`	Unique session ID
`new_session`	If `true`, a new sessio

`database_manager`	Database manager cl session data

An option hash can specify options when
default database manager class (`CGI::Ses`
options:

Key	Function	Def
`tmpdir`	Directory for temporary files	`/tm`
`prefix`	Prefix for temporary files	Nor

Methods for Database Manager

Database manager object should have foll `initialize( *session*[, *options*])`

Initializes the database. *`session`* is a
hash that passed to `CGI::Session::r`

`restore`

Returns the hash that contains sessio

`update`

Updates the hash returned by `restor`

`close`

Closes the database

`delete`

Removes the session-specific data fr

Instance Methods

`s[key]`

Returns the value for the specified se

`s[ key]= value`

Sets the value for the specified sessi

`s.delete`

Deletes the session

`s.update`

Writes session data to the database, c
manager object

4.1.2 Operating System Servic

A mixed bag of OS services are provided
curses, filesystem searching and file hand
others.

If you're coming from another scripting la
interfaces you'll find familiar and straight
here.

Curses *Character-based interfa*

The `Curses` module provides an interface
`curses`.

Required Library

require 'curses'

Module Functions

`addch(` *`ch`*`)`

Outputs one character to the screen

`addstr(` *`str`*`)`

Outputs *`str`* to the screen

`beep`

Beeps the bell

`cbreak`

Turns on `cbreak` mode

`nocbreak`

Turns off `cbreak` mode

`clear`

Clears the screen

`close_screen`

Finalizes the `curses` system

`cols`

Returns the screen width

`crmode`

Alias to the `cbreak`

`nocrmode`

Alias to the `nocbreak`

`delch`

Deletes a character at the cursor posi

`deleteln`

Deletes a line at the cursor position

`doupdate`

Updates the screen by queued chang

`echo`

Turns on echo mode

`noecho`

Turns off echo mode

`flash`

Flashes the screen

`getch`

Reads one character from the keyboa

`getstr`

Reads a line of string from the keybo

`inch`

Reads a character at the cursor positi

`init_screen`

Initializes the `curses` system

`insch(` *`ch`*`)`

Outputs one character before the curs

`lines`

Returns the screen height

`nl`

Turns on newline mode, which transl

`nonl`

Turns off newline mode

`raw`

Turns on raw mode

`noraw`

Turns off raw mode

`refresh`

Refreshes the screen

`setpos(` *`y`*`,` *`x`*`)`

Moves the cursor to the (*y*, *x*) positic

`standout`

Turns on `standout` (highlighting) mo

`standend`

Turn off `standout` mode

`stdscr`

Returns the reference to the standard

`ungetch(` *`ch`*`)`

Pushes *`ch`* back to input buffer

Curses::Window *Character-bas*

`Curses::Window` is a class for character-b
library.

Required Library

`require "curses"`

Class Method

`Curses::Window::new(` *`h`*`,` *`w`*`,` *`y`*`,` *`x`*`)`

Creates a new `curses` window of siz

Instance Methods

`w` `<<` *`str`*

`w``.addstr(` *`str`*`)`

Outputs *`str`* to the screen.

`w.addch( ch)`

Outputs one character to the screen.

`w.begx`

Returns the window's beginning *x* po

`w.begy`

Returns the window's beginning *y* po

`w.box( v, h)`

Draws a box around the window. *v* is
character that draws a horizontal side

`w.clear`

Clears the window.

`w.close`

Closes the window.

`w.curx`

Returns *x* position of the window's c

`w.cury`

Returns *y* position of the window's c

`w.delch`

Deletes a character at the window's c

`w.deleteln`

Deletes a line at the window's cursor

`w.getch`

Reads one character from the keyboa

`w.getstr`

Reads a line of string from the keybo

`w.inch`

Reads a character at the window's cu

`w.insch( ch)`

Outputs one character before the win

`w.maxx`

Returns the window's *x* size.

`w.maxy`

Returns the window's *y* size.

`w.move( y, x)`

Moves the window to the position (*y*

`w.refresh`

Refreshes the window.

`w.setpos( y, x)`

Moves the window's cursor to the po

`w.standend`

Turns on `standout` (highlighting) mo

`w.standout`

Turns off `standout` mode in the win

`w.subwin( h, w, y, x)`

Creates a new `curses` subwindow of

***Etc** Module for /etc directory da*

The `Etc` module provides functions to retr
under */etc* directory. This module is Unix-

Required Library

require 'etc'

Example

```
require 'etc'

print "you must be ", Etc.getlogin,
```

Module Functions

getlogin

Returns login name of the user. If thi

getpwnam(*name*)

Searches in /etc/passwd file (or equ
for the user *name*. See getpwnam(3) f
structure, which includes the followi

name	Username(string)
passwd	Encrypted password(string)
uid	User ID(integer)
gid	Group ID(integer)

`gecos`	Gecos field(string)
`dir`	Home directory(string)
`shell`	Login shell(string)
`change`	Password change time(integer)
`quota`	Quota value(integer)
`age`	Password age(integer)
`class`	User access class(string)
`comment`	Comment(string)
`expire`	Account expiration time(integer

`getpwuid([ uid])`

Returns `passwd` entry for the specifie
`getuid`. See `getpwuid(3)` for details

`getgrgid( gid)`

Searches in `/etc/group` file (or equiv
the *`gid`*. See `getgrgid(3)` for detail.
includes the following members:

`name`	Group name(string)
`passwd`	Group password(string)
`gid`	Group ID(integer)
`mem`	Array of the group member name

`getgrnam(` *`name`*`)`

Returns the group entry for the speci
structure. See `getgrnam(3)` for detai

`group`

Iterates over all `group` entries.

`passwd`

Iterates over all `passwd` entries.

Fcntl *Fcntl constant module*

The `Fcntl` module provides constant defi

Required Library

require 'fcntl'

Constants

`F_DUPFD`	Duplicates file descriptor
`F_GETFD`	Reads the close-on-exec flag
`F_SETFD`	Sets the close-on-exec flags
`F_GETFL`	Reads the descriptor's flags
`F_SETFL`	Gets the descriptor's flags (`O`
`F_GETLK`	Gets the flock structure
`F_SETLK`	Gets lock according to the lo

F_SETLKW	Sets lock like F_SETLK (bloc
F_RDLCK	Reads lock flag for flock stru
F_WRLCK	Writes lock flag for flock str
F_UNLCK	Unlocks flag for flock struct
FD_CLOEXEC	Close-on-exec flag
O_CREAT	Creates file if it doesn't exist
O_EXCL	File shouldn't exist before cr
O_TRUNC	Truncates to *length* 0
O_APPEND	Appends mode

`O_NONBLOCK`	Nonblocking mode
`O_NDELAY`	Nonblocking mode
`O_RDONLY`	Read-only mode
`O_RDWR`	Read-write mode
`O_WRONLY`	Write-only mode

Find *Directory tree traversal m*

The `Find` module provides a depth-first di

Required Library

require 'etc'

Example

```
require 'find'

# prints all files with ".c" extens
Find.find(".") {|f|
  puts f if /\.c$/ =~ f
}
```

Module Functions

`find( path...) {| f| ...}`

Traverses directory tree giving each

`prune`

Terminates traversal down from the

ftools *File utility library*

`ftools` is a library that enhances file hand

Required Library

require 'ftools'

Class Methods

`File::chmod(` *`mode`*`,` *`files...`*`[,` *`verbos`*

`ftools` enhances `File::chmod` to tak
`true`, prints log to `stderr`.

`File::cmp(` *`path1`*`,` *`path2`*`[,` *`verbose`*`=fa`

`File::compare(` *`path1`*`,` *`path2`*`[,` *`verbos`*

Compares two files and returns `true`
`true`, prints log to `stderr`.

`File::cp(` *`path1`*`,` *`path2`*`[,` *`verbose`*`=fal`

`File::copy(` *`path1`*`,` *`path2`*`[,` *`verbose`*`=f`

Copies a file at *`path1`* to *`path2`*. If *`ve`*

`File::install(` *`path1`*`,` *`path2`*`[,` *`mode`* `[,`

Copies a file at *`path1`* to *`path2`*. If *`mo`*
`mode`. If file at *`path2`* exists, it's remo
operation log to `stderr`.

`File::makedirs(` *`path...`*`[,` *`verbose`*`=fa`

`File::mkpath(` *`path...`*`[,` *`verbose`*`=fals`

Creates the specified directories. If a
creates them as well. If the last argu

`File::move(` *`path1`*`,` *`path2`*`[,` *`verbose`*`=f`

`File::mv( path1, path2[, verbose=fal`

Moves file from `path1` to `path2`. If th
to `stderr`.

`File::rm_f( path...[, verbose=false]`

`File::safe_unlink( path...[, verbose`

Removes files regardless of file-perm
prints operation log to `stderr`.

`File::syscopy( path1, path2)`

Copies a file from `path1` to `path2` us
copies permissions of the file as well

GetoptLong *Command line opti*

The GetoptLong class parses command-li
getoptlong library.

Required Library

require 'gettextfile'

Example

```
require 'getoptlong'

 opt = GetoptLong.new(
     ['--max-size', '-m', GetoptLon
     ['--quiet',    '-q', GetoptLon
     ['--help',           GetoptLon
     ['--version',        GetoptLon
 opt.each_option do |name,arg|
    case name
    when '--max-size'
  printf "max-size is %d\n", arg
    when '--quiet'
  print "be quiet!\n"
    when '--help'
  print "help message here\n"
  exit
    when '--version'
  print "version 0.1\n"
  exit
    end
  end
```

Inherited Class

`Object`

Class Method

`GetoptLong::new(` *`option...`* `)`

Creates and returns a `GetoptLong` ob
the `set_options` method.

Instance Methods

`opt``.each {|` *`optname`*`,` *`optarg`*`| ...}`

`opt.each_option {| optname, optarg|`

Iterates over each command-line opt
block.

`opt.get`

`opt.get_option`

Retrieves an option from command-l
pair of option.

`opt.error`

`opt.error?`

Returns type of the current error or `n`

`opt.error_message`

Returns an error message of the curr

`opt.ordering= ordering`

Sets option ordering. `ordering` is an
`RETURN_IN_ORDER`.

`opt.ordering`

Returns current ordering.

`opt.quiet= bool`

Sets status of quiet mode. In quiet m
messages to `stdout` on errors. The d

`opt.quiet`

`opt.quiet?`

Returns current status of quiet mode.

`opt.set_options( option...)`

Sets command-line options that your
option names and option type consta

Option type is any of `NO_ARGUMENT`, `R`
You have to call `set_options` before
`each_option`.

`*opt*.terminate`

Terminates option processing. Raises
before termination.

`*opt*.terminated?`

Returns `true` if option processing is
returns `false`.

Constants

Ordering specifiers

`PERMUTE`

`REQUIRE_ORDER`

`RETURN_IN_ORDER`

Argument type specifiers

`NO_ARGUMENT`

`REQUIRED_ARGUMENT`

`OPTIONAL_ARGUMENT`

PTY *Pseudo TTY access module*

The `PTY` module executes commands as if

Required Library

require "pty"

Module Functions

`getpty( *command*)`

`spawn( *command*)`

Reserves a `PTY`, executes *`command`* ov
elements (reading I/O, writing I/O, a
block, the array is passed to the bloc
while *`command`* is running.

`protect_signal { ...}`

Protects block execution from `SIGCH`
invoke other subprocesses while usi

`reset_signal`

Disables to handle `SIGCHLD` while `PT`

Readline *GNU readline library*

The `Readline` module provides a interfac
`readline`.

Required Library

require 'readline'

Example

```
require 'readline'
include Readline
line = readline("Prompt> ", true)
```

Module Function

`readline( prompt, add_history)`

Reads one line with line editing. If th
history.

Module Methods

`Readline::completion_proc= proc`

Specifies `Proc` object to determine co
returns completion candidates.

`Readline::completion_proc`

Returns the completion `Proc` object.

`Readline::completion_case_fold=bo o`

Sets whether or not to ignore case on

`Readline::completion_case_fold`

Returns `true` if completion ignores c

`Readline::completion_append_charact`

Specifies a character to be appended
specified, nothing is appended.

`Readline::completion_append_charact`

Returns a string containing a charact
space.

`Readline::vi_editing_mode`

Specifies *vi* editing mode.

`Readline::emacs_editing_mode`

Specifies Emacs editing mode.

Constant

`HISTORY`

The history buffer; it behaves just lik

Tempfile *Temporary file class*

Temporary files are always deleted when
terminates.

Required Library

require 'tempfile'

Example

```
require 'tempfile'
f = Tempfile.new("foo")
f.print("foo\n")
f.close
```

```
f.open
p f.gets      # => "foo\n"
f.close(true) # f will be automatic
```

Class Method

`Tempfile::new(` *basename*`[,` *tmpdir*`="/t`

Opens a temporary file that includes

Instance Methods

t`.open`

Reopens the temporary file, allowing
the file.

t`.close([` *permanently*`=false])`

Closes the temporary file. If *permane*

```
t.path
```

Returns the path of the temporary fil

In addition to the previous methods, objec
methods of class `File`.

Win32API *Microsoft Windows A*

Win32API represents functions in Windo

Required Library

require 'Win32API'

Example

```
require 'Win32API'
```

```
getch = Win32API.new("crtdll", "_ge
puts getch.Call.chr
```

Class Method

`Win32API::new(` *`dll`*`,` *`proc`*`,` *`import`*`,` *`exp`*

Returns the object representing the `W` *`dll`*, which has the signature specifie of strings denoting types. *`export`* is a the following:

`"n"`

Number

`"l"`

Number

`"i"`

Integer

`"p"`

Pointer

`"v"`

Void (export only)

Type strings are case-insensitive.

Instance Methods

`call([ *arg...*])`

`Call([ *arg...*])`

Invokes the `Win32API` function. Argu
by `Win32API::new`.

4.1.3 Threads

Threading classes in the Ruby standard lil
support for parallel programming with su
mutexes, queues and a handy-dandy threa

ConditionVariable *Synchroniza*

This class represents condition variables f

Required Library

require 'thread'

Class Method

`ConditionVariable::new`

Creates a `ConditionVariable` object

Instance Methods

`c.broadcast`

Wakes up all waiting queued threads

`c.signal`

Wakes up the next thread in the queu

`c.wait( mutex)`

Waits on `condition variable`

Monitor *Exclusive monitor secti*

This class represents exclusive sections be

Required Library

require 'monitor'

Included Module

`MonitorMixin`

Class Method

`Monitor::new`

Creates a `Monitor` object

Instance Methods

`*m*.enter`

Enters exclusive section.

`*m*.exit`

Leaves exclusive section.

`m.owner`

Returns the thread that owns the mon

`m.synchronize{ ...}`

Enters exclusive section and execute
automatically when the block exits.

`m.try_enter`

Attempts to enter exclusive section.

MonitorMixin *Exclusive monito*

Adds monitor functionality to an arbitrary

Required Library

require 'monitor'

Instance Methods

m.mon_enter

Enters exclusive section.

m.mon_exit

Leaves exclusive section.

m.mon_owner

Returns the thread that owns the mo

m.mon_synchronize{ *...*}

Enters exclusive section and execute

automatically when the block exits.

`m.try_mon_enter`

Attempts to enter exclusive section.

Mutex *Mutual exclusion class*

This class represents mutually exclusive l

Required Library

require 'thread'

Class Method

`Mutex::new`

Creates a `Mutex` object

Instance Methods

`m.lock`

Locks the `Mutex` object *m*.

`m.locked?`

Returns `true` if *m* is locked.

`m.synchronize {...}`

Locks *m* and runs the block, then rele

`m.try_lock`

Attempts to lock *m*. Returns `false` if

`m.unlock`

Releases lock on *m*.

Queue *Message queue class*

This class provides the way to communica

Required Library

require 'thread'

Class Method

`Queue::new`

Creates a `queue` object

Instance Methods

`*q*.empty?`

Returns `true` if the queue is empty.

`q.num_waiting`

Returns the number of threads waitin

`q.pop([ non_block=false])`

Retrieves data from the queue. If the
suspended until data is pushed onto t
isn't suspended, and an exception is r

`q.push( obj)`

`q.enq( obj)`

Pushes *obj* to the queue.

`q.size`

`q.length`

Returns the length of the queue.

SizedQueue *Fixed-length queue*

This class represents queues of specified s
blocked if the capacity is full.

Required Library

require 'thread'

Inherited Class

`Queue`

Class Method

`SizedQueue::new(` *`max`*`)`

Creates a fixed-length queue with a m

Instance Methods

`q.max`

Returns the maximum size of the que

`q.max= n`

Sets the maximum length of the queu

ThreadsWait *Thread terminatio*

This class watches termination of multiple

Required Library

require 'thwait'

Class Methods

`ThreadsWait::all_waits( th,...)`

`ThreadsWait::all_waits( th...) { ...`

Waits until all specified threads are t
method, evaluates it for each thread t

`ThreadsWait.new( th...)`

Creates a `ThreadsWait` object, specif

Instance Methods

`th.threads`

Lists threads to be synchronized

`th.empty?`

Returns `true` if there is no thread to

`th.finished?`

Returns `true` if there is any terminat

`th.join( th...)`

Waits for specified threads.

`th.join_nowait( th...)`

Specifies threads to wait; non-blocki

`th.next_wait`

Waits until any specified thread is te

`th.all_waits`

`th.all_waits{ ...}`

Waits until all specified threads are t
method, evaluates it for each thread t

4.1.4 Data Persistence

These libraries provide interfaces or hook
(OS, GNU, and public domain).

Ruby lets you store and retrieve "live" dat
you're probably used through the `DBM`, `GDE`

DBM *DBM class*

`DBM` implements a database with the same
limited to strings. Uses `ndbm` library inclu

Required Library

require 'dbm'

Included Module

`Enumerable`

Class Methods

`DBM::open( *path*[, *mode*=0666])`

`DBM::new( *path*[, *mode*=0666])`

Opens a new `DBM` database. Access ri
an integer.

Instance Methods

The DBM class has all the methods of the `dup`, and `rehash`. `DBM` also has the `close` m

`d.close`

Closes `DBM` database

GDBM *GDBM class*

GNU implementation of `DBM`. Has the sam

Required Library

require 'gdbm'

Instance Methods

In addition to methods from the `DBM` class,

`d.reorganize`

Reconfigures the database; shouldn't t

SDBM *SDBM class*

Public domain implementation of `DBM`. Ha
anywhere but has inferior performance an
`DBM`s.

Required Library

require 'sdbm'

PStore *Simple object-oriented d*

`PStore` is a simple object-oriented databa
persistence (using `Marshal`) and transacti

Required Library

require 'pstore'

Class Method

`PStore::new( *path*)`

Creates a database object. Data is stc

Instance Methods

`*p*.transaction {| *ps*| *...*}`

Starts a transaction (a series of datab
database can be achieved only throu

`*p*[ *name*]`

Retrieves an object stored in the data

`p[ name]= obj`

Stores `obj` in the database under the
all objects accessed reflexively by `ob`
file.

`p.root?( name)`

Returns `true` if the key name exists i

`p.commit`

Completes the transaction. When this
transaction method is executed, and
database file.

`p.abort`

Aborts the transaction. When this me
passed to the transaction method is te

objects during the transaction aren't v

4.1.5 Numbers

These libraries let you handle numeric cal
`Complex`, `Rational`, and `Matrix`.

Complex *Complex number class*

When this library is loaded with `require`,
handle complex numbers.

Required Library

require 'complex'

Inherited Class

`Numeric`

Class Methods

`Complex( *r* [, *i*=0])`

`Complex::new( *r* [, *i*=0])`

Creates a complex number object. Th

Instance Methods

`*c*.abs`

Returns the absolute value of the con

`*c*.abs2`

Returns the square of the absolute va

`*c*.arg`

Returns the argument of the complex

`c.conjugate`

Returns the `conjugate` of the comple

`c.image`

Returns the imaginary part of the co
image method to the `Numeric` class.

`c.polar`

Returns the array `arr[c.abs`, `c.arg]`

`c.real`

Returns the real part of the complex
method to the `Numeric` class.

Rational *Rational number class*

When this library is loaded with `require`,
can handle rational numbers, and the follo
class:

`to_r`

Converts a number to a rational num

`lcm`

Returns the least common multiple

`gcd`

Returns the greatest common divisor

Required Library

require 'rational'

Inherited Class

`Numeric`

Class Methods

`Rational(` *`a`*`, `*`b`*`)`

`Rational::new(` *`a`*`, `*`b`*`)`

Creates a rational number object. Th

Matrix *Matrix class*

Required Library

require 'matrix'

Class Methods

`Matrix::[ `*`row...`*`]`

Creates a matrix where *`row`* indicates

```
Matrix[[11, 12], [21, 22]]
```

`Matrix::identity( `*`n`*`)`

`Matrix::unit( `*`n`*`)`

`Matrix::I( `*`n`*`)`

Creates an `n`-by-*`n`* unit matrix.

`Matrix::columns( `*`columns`*`)`

Creates a new matrix using *`columns`*

```
Matrix::columns([[11, 12], [21,
```

`Matrix::column_vector(` *`column`*`)`

Creates a 1-by-*n* matrix such that col

`Matrix::diagonal(` *`value...`*`)`

Creates a matrix where diagonal com

```
Matrix.diagonal(11, 22, 33)  #
      [0, 22, 0], [0, 0, 33]]
```

`Matrix::rows(` *`rows`*`[,` *`copy`*`=true])`

Creates a matrix where *`rows`* is an ar
If the optional argument *`copy`* is `fals`
structure of the matrix without copyi

```
Matrix::rows([[11, 12], [21, 22
```

`Matrix::row_vector(` *`row`*`)`

Creates an 1-by-*n* matrix such that th

`Matrix::scalar(` *`n, value`*`)`

Creates an *n*-by-*n* diagonal matrix su
value.

```
Matrix::scalar(3,81)    # => Ma

p ParseDate::parsedate("Fri Aug
# => [2001, 8, 3, 17, 16, 57, '
p ParseDate::parsedate("1993-02
# => [1993, 2, 24, nil, nil, ni
```

`Matrix::zero(` *`n`*`)`

Creates an *n*-by-*n* zero matrix.

Instance Methods

`m[ i, j]`

Returns (`i,j`) component.

`m * mtx`

Multiplication.

`m + mtx`

Addition.

`m- mtx`

Subtraction.

`m / mtx`

Returns `m * mtx.inv`.

`m ** n`

Power of *n* over matrix.

`m.collect{ ...}`

`m.map{ ...}`

Creates a matrix that is the result of i
components of the matrix *m*.

`m.column( j)`

Returns the *j*-th column vector of th
method, the block is iterated over all

`m.column_size`

Returns the number of columns.

`m.column_vectors`

Returns array of column vectors of th

`m.determinant`

`m.det`

Returns the determinant of the matrix

`m.inverse`

`m.inv`

Returns an inversed matrix of the ma

`m.minor( from_row, row_size, from_co`

`m.minor( from_row..to_row, from_col`

Returns submatrix of the matrix *m*.

`m.rank`

Returns the rank of the matrix `m`.

`m.row( i)`

`m.row( i) {...}`

Returns the `i`-th row vector of the ma
method, the block is iterated over all

`m.row_size`

Returns the number of rows.

`m.row_vectors`

Returns an array of row vectors of th

`m.regular?`

Returns `true` if `m` is a regular matrix.

`m.singular?`

Returns `true` if *m* is a singular (i.e., n

`m.square?`

Returns `true` if *m* is a square matrix.

`m.trace`

`m.tr`

Returns the trace of the matrix *m*.

`m.transpose`

`m.t`

Returns the transpose of the matrix *m*

4.1.6 Design Patterns

Design patterns are a terrific way to get y
Ruby provides support in the standard lib
design patterns. This group of libraries pr
programming techniques for delegators, f

Delegator *Delegator pattern su*

`Delegator` is an abstract class for the Del
achieved by creating a subclass of the `Del`

Required Library

require 'delegate'

Class Method

`Delegator::new(` *`obj`*`)`

Creates a delegate object to which m

Instance Method

`_ _getobj_ _`

Returns the object to which methods
subclass.

SimpleDelegator *Simple concre*

This class allows for easy implementation

Required Library

require 'delegate'

Inherited Class

`Delegator`

Class Method

`SimpleDelegator::new(` *`obj`*`)`

Creates an object that forwards meth

Instance Method

`_ _setobj_ _`

Sets the object to which methods are

DelegatorClass *Class creation f*

This function dynamically creates a class

Required Library

require 'delegate'

Function

`DelegateClass( c)`

Creates a new class to which the met

Method of Generated Class

`D::new( obj)`

Creates a delegate object with `obj` as

Forwardable *Module to add sel class*

The `Forwardable` module provides more
method name and destination object expli

Required Library

require "forwardable"

Example

```
class Foo
  extend Forwardable
  # ...
  def_delegators("@out", "printf",
  def_delegators(:@in, :gets)
  def_delegator(:@contents, :[], "c
end
f = Foo.new
f.printf("hello world\n")   # forwa
f.gets                      # forwa
f.content_at(1)             # forwa
```

Instance Methods

`f.def_delegator( accessor, method[, a`

`f.def_instance_delegator( accessor,`

Defines delegation from `method` to `a`
called instead of `method`.

`f.def_delegators( accessor, method..`

`f.def_instance_delegators( accessor`

Defines delegation to `accessor` for e

SingleForwardable *Selective de*

The `SingleForwardable` module provides
specific object.

Required Library

require 'forwardable'

Example

```
require 'forwardable'

# ...
g = Goo.new
g.extend SingleForwardable
g.def_delegator("@out", :puts)
g.puts("hello world")             #
```

Instance Methods

f.def_singleton_delegator(*accessor*

f.def_delegator(*accessor*, *method*[,

Defines delegation from *method* to *a*

called instead of *method*.

f.def_singleton_delegators(*accesso*

f.def_delegators(*accessor*, *method..*

Defines delegation to *accessor* for e

Singleton *Singleton pattern mod*

The Singleton module allows the implen
including the module, you can ensure that

Required Library

require 'singleton'

Class Method

`instance`

Returns the only instance of the class
reused. `instance` is a class method a
module.

Observable *Observable pattern*

The `Observable` module allows the imple
Classes that include this module can notif
object can become an observer as long as

Required Library

```
require 'observer'
```

Instance Methods

`o.add_observer( obj)`

Adds observer `obj` as an observer of

`o.count_observers`

Returns the number of observers of `c`

`o.changed([ state=true])`

Sets the changed state of `o`.

`o.changed?`

Returns `true` if `o` has been changed.

`o.delete_observer( obj)`

Removes observer `obj` as an observe

`o.delete_observers`

Removes all observers of *o*.

`o.notify_observers([ arg...])`

If *o*'s changed state is `true`, invokes t
it the specified arguments.

4.1.7 Miscellaneous Libraries

It almost goes without saying, but there's
into any category. Ruby's standard library
includes anything that isn't in one of the p

In Ruby's standard library, you'll find clas
manipulation, timeouts on long operation

Date *Date class*

Date is a class to represent the calendar da
which is the number of days since midday

Currently we use the Gregorian calendar,
time (before 1752 in England, for exampl
country. Date class can handle both calend

There's no relation between Julian day nu
coincidence.

Required Library

```
require 'date'
```

Example

```
require 'date'

# 3000 days after Ruby was born
puts Date::new(1993,2,24)+3000, "\n
```

Included Module

`Comparable`

Class Methods

`Date::exist?( year, month, day[, sta`

`Date::exist3?( year, month, day[, st`

Returns the Julian day number corres *day* of year, if they are correct. If the

`Date::exist2?( year, yday[, start])`

Returns the Julian day number corres year, if they are correct. If they aren't

`Date::existw?( year, week, wday[, st`

Returns the Julian day number corres
based *year*, calendar *week*, and calen
correct, returns nil.

`Date::new( year, month, day[, start]`

`Date::new3( year, month, day[, start`

Creates a Date object corresponding
month.

`Date::new1( jd[, start])`

Creates a Date object corresponding

`Date::new2( year, yday[, start])`

Creates a Date object corresponding

`Date::neww( year, week, wday[, start`

Creates a `Date` object corresponding
calendar *`week`*, and calendar weekday

`Date::today([ `*`start`*`])`

Creates a `Date` object corresponding

Instance Methods

d << *n*

Returns a `Date` object that is *n* month

d >> *n*

Returns a `Date` object that is *n* month

d <=> *x*

Compares dates. *x* may be a `Date` ob

`*d* + *n*`

Returns a `Date` object that is *n* days l

`*d* - *x*`

Returns the difference in terms of da
integer, returns a `Date` object that is *x*

`*d*.cwday`

Returns the calendar weekday (1-7, M

`*d*.cweek`

Returns the calendar week (1-53) for

`*d*.cwyear`

Returns the calendar week-based yea

`d.day`

`d.mday`

Returns the day of the month (1-31)

`d.downto( min) {| date| ...}`

Runs block on dates ranging from *d*
`{|date|...}`.

`d.jd`

Returns the Julian day number for *d*.

`d.leap?`

Returns `true` if *d* is a leap year.

`d.mjd`

Returns the modified Julian day num
number of days since midnight Nove

`d.mon`

`d.month`

Returns the month (1-12) for `d`.

`d.newsg([ start])`

Copies `d` to a new `Date` object and re
`start`.

`d.next`

`d.succ`

Returns a new `Date` object one day la

`d.sg`

Returns the Julian day number of the

d`.step(` *`limit, step`*`) {|` *`date`*`| ...}`

Runs block on `Date` objects from *d* t
each time.

d`.upto(` *`max`*`) {|` *`date`*`| ...}`

Runs block on dates ranging from *d*
`{|date|...}`.

d`.wday`

Returns the day of the week for *d* (0-

d`.yday`

Returns the day of the year for *d* (1-3

d`.year`

Returns the year for *d*.

Constants

`MONTHNAMES`

An array of the names of the months

`DAYNAMES`

An array of the names of the days of

`ITALY`

Gregorian calendar start day number

`ENGLAND`

Gregorian calendar start day number

`JULIAN`

Start specifier for Julian calendar

`GREGORIAN`

Start specifier for Gregorian calenda

ParseDate *Date representation*

The `ParseDate` module parses strings that

Required Library

require 'parsedate'

Module Function

`parsedate( *str*[, *cyear*=false])`

Parses a date and/or time expression
(year, month, day, hour, minute, seco
array. Sunday is represented as 0 in t
for elements that can't be parsed or h
`cyear` is `true`, years with a value of
2000s and years ranging from 69 to
summary, beware of the Y2K69 prob

timeout *Time out a lengthy proc*

Times out a lengthy procedure or those th

Required Library

require 'timeout'

Function

`timeout( sec) { ...}`

Executes the block and returns `true`
prior to elapsing of the timeout perio
execution of the block and raises a `T`

```
require 'timeout'
status = timeout(5) {
  # something that may take tim
}
```

MD5 *MD5 message digest class*

The `MD5` class provides a one-way hash fu
algorithm described in RFC-1321

Example

```
requires 'md5'
```

```
md5 = MD5::new("matz")
puts md5.hexdigest # prints: 3eb50a
```

Class Methods

`MD5::new([ *str*])`

`MD5::md5([ *str*])`

Creates a new `MD5` object. If a string

Instance Methods

md`.clone`

Copies the `MD5` object.

md`.digest`

Returns the `MD5` hash of the added st

`md.hexdigest`

Returns the `MD5` hash of the added str

`md``.update(` *`str`*`)`

`md` `<<` *`str`*

Updates the `MD5` object with the strin
single call with the concatenation of
`m.update(b)` is equivalent to `m.upda`
`<< a+b`.

SHA1 *SHA1 message digest cla*

The `SHA1` class provides a one-way hash f

Class Methods

`SHA1::new([ str])`

`SHA1::sha1([ str])`

Creates a new `SHA1` object. If a string

Instance Methods

`sh.clone`

Copies the `SHA1` object.

`sh.digest`

Returns the `SHA1` hash of the added s

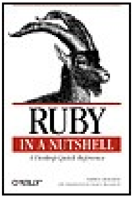
`sh.hexdigest`

Returns the `SHA1` hash of the added s

`sh.update( str)`

`sh << str`

Updates the `SHA1` object with the stri
single call with the concatenation of
`m.update(b`) is equivalent to `m.upda`
`<< a+b`.


Ruby in a NutshellBy Yukihiro Matsumoto

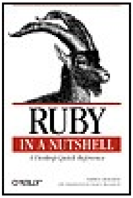
Chapter 5. Ruby Tools

As a matter of course in Ruby, you edit your Ruby program and then feed it to the interpreter. Theoretically, the editor and interpreter are all you need to program Ruby. But you can get help from other tools. In this chapter, you will find descriptions of tools to help Ruby programmers.

Previous Next

Top

Previous Next



Ruby in a NutshellBy Yukihiro Matsumoto

Chapter 6. Ruby Updates

6.2 Changes from 1.6.5 to 1.7.1

The following information details the changes that are occurring in developmen
versions 1.7.1 and 1.8 (though 1.8 will ha

additional changes as well):

- Multiple assignment behavior is clarified.
- Syntax enhanced to interpret argume parentheses to allow `p ("xx"*2).to_i`
- `break` and `next` extended to take an optional expression, which is used as return value of the iterating method and `yield`, respectively.
- The following new methods (or modifications to methods) have been added:

```
Array#fetch
Array#insert
```

```
Enumerable#all?
Enumerable#any?
Enumerable#inject
Enumerable#sort_by
File#fnmatch
MatchData#to_ary
Method#==
Module#include?
Module#included
Module#method_removed
Module#method_undefined
Object#singleton_method_remove
Object#singleton_method_undefi
Proc#==
Proc#yield
Range#to_ary
Range#step
Regexp#options
String#casecmp
String#insert
Symbol#intern
Symbol::all_symbols
SystemExit#status
File::lchmod
```

```
File::lchown
IO::for_fd
IO::read
Math::acos
Math::asin
Math::atan
Math::cosh
Math::hypot
Math::sinh
Math::tanh
Process::times
Process::waitall
SystemCallError::===
```

- `String#eql?` is now always case-sensitive.

- `Dir::chdir` extended to take a block

- `NoMethodError` raised for undefined method.

- `Interrupt` is a subclass of `SignalException` (it was a subclass Exception in 1.6 and prior).

- `$?` now gives `Process::Status` alon with `Process::wait2`, `Process::waitpid2`.

- `Regexp.last_match(n)` extended to take an optional argument.

- `The Digest` module has been added a replacement for the `md5` and `sha1` modules.

- Line-range operation is now obsolet except when used in a one-liner (e.g. `ruby -e ...`).

- Comparison of exception classes in a

rescue clause now uses `Module#===`.

- `TCPSocket.new` and `TCPSocket.oper` extended to take an address and a po number for the local side in optional third and fourth arguments.

- `Time` extended to accept a negative `time_t` (only if the platform support it).

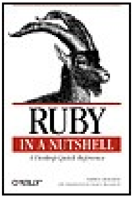
- Objects that have `to_str` now behav more like strings.

- The `Signal` module has been added.

- Generational garbage collection has been added.

◀ Previous Next ▶

Top



Chapter 6. Ruby Updates

6.3 The Future of Ruby

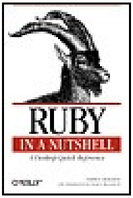
As Ruby is now used by so many programmers worldwide, I don't see making any radical changes in the near future. But I'd like to keep Ruby competitive with other scripting languages.

I don't have a concrete plan for future versions, even 2.0, but I do have plans to fix some of the remaining drawbacks in the Ruby implementation. For example, Ruby's internals are too complex to maintain and can be slower than other languages. I'm going to reimplement the interpreter as a bytecode engine to simplify interpreter core and boost performance. Also, recently an intriguing but still vague possibility of a joint backend among Perl, Python, and Ruby has surfaced.

I'd also like to support M17N

(Multilingualization) in Ruby. M17N offers the ability to handle various human languages along with the necessary encodings. We already implemented a prototype that can handle ASCII, UTF-8, and several Japanese encodings.

The future is unknown, and my imagination is limited. But you can certainly contribute to the evolution of Ruby via the process called RCR (or Ruby Change Requests) explained in the next section. We look forward to your contributions.



Chapter 6. Ruby Updates

6.4 Participate in Ruby

Programmers often get ideas on how they'd like to improve Ruby. These ideas are sometimes useful and interesting, sometimes not. Since the language needs to stay consistent, I often need to choose which fixes or ideas to add and which to reject. To make this process easier, we have instituted Ruby Change Requests (RCRs).

When you want to propose a new feature for Ruby, you have to submit your proposal to http://www.rubygarden.org/?topic=RCR. The more concrete and detailed the proposal, the greater chance of success you have of getting it accepted. The proposal should preferably be consistent, backward-compatible, and follow the principle of least surprise.

The RCR page offers a discussion forum

and web-based voting box. Once you submit your proposal, discussion is held on it. If it's decided (with the help of the community) that your proposal is indeed useful, it will be added to future versions of Ruby.

◀ Previous Next ▶

Top

Table of Contents