Table of Contents

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

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.

4.1. Standard Library

The Ruby standard library extends the foundation of the Ruby built-in library with classes and abstractions for a variety of programming needs, including network programming, operating-system services, threads, and more. These classes provide flexible capabilities at a high level of abstraction, giving you the ability to create powerful Ruby scripts useful in a variety of problem domains.

Many common tasks are performed by Ruby programmers all over the world. Some of these tasks include network access such as TCP/IP and CGI, OS access, database access, controlling processes with threads, numeric calculations, implementing design classes, and manipulating dates. These are used so frequently that they are included with all standard distributions of Ruby; when you access these classes and methods from your programs, they will be available from the Standard Library. Could you write these libraries yourself? Probably. Would you feel confident they have been exhaustively tested, optimized, and debugged? Usually not. The Standard Library is a great time saver. And as Ruby grows and evolves, so will its Standard Library, to everyone's benefit.

Although not every library section will contain all these entries, the basic format for each section is as follows:

- Required library
- Example
- Inherited class
- Class methods
- Instance methods

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

4.1.1. Network

Use Ruby's network classes to let your scripts speak basic protocols such as TCP and UDP as a client, a server, or both. These libraries provide socket access to a variety of Internet protocols and classes that make access to those protocols easier. You can even crawl up the protocol stack and find support for higher-level protocols like FTP, HTTP, IMAP, and so on. All have an intuitive, transparent interface that won't get in your way. This is the largest group of libraries and one of the most frequently used.

Oh, and don't worry. There's support for doing web programming through the CGI, `CGI::Cookie` and `CGI::Session` classes.

BasicSocket — *Socket-related superclass*

`BasicSocket` is an abstract base class for network socket-related classes. This class provides common behavior among `Socket` classes.

Required Library

require 'socket'

Inherited Class

`IO`

Class Methods

`BasicSocket::do_not_reverse_lookup`

Returns `true` if a query returns numeric address, not hostname

`BasicSocket::do_not_reverse_lookup=` *`bool`*

Sets `reverse_lookup` status

Instance Methods

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`s.getpeername`

Returns information on this connection's peer socket as a `struct sockaddr` packed into a string.

`s.getsockname`

Returns information on *s* as a `struct sockaddr` packed into a string.

`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` shuts down receiving, `1` sending, and `2` both.

`s.recv( len[, flags])`

Receives data from *s*, and returns it as a string.

`s.send( mesg, flags[, to])`

Sends data over the socket *s*, returning the length of the data sent. *to* may be a `struct sockaddr` packed into a string indicating the recipient address.

IPSocket — *IP socket class*

`IPSocket` class is a base class of `TCPSocket` and `UDPSocket`. `IPSocket` class provides common behavior among Internet Protocol (IP) sockets. Sockets classes in Ruby support IPv6, if the native platform supports it.

Required Library

require 'socket'

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Inherited Class

`BasicSocket`

Class Method

`IPSocket::getaddress(`*`host`*`)`

Returns the IP address of the specified *`host`*. The IP address is returned as a string such as `127.10.0.1` (IPv4) or `::1` (IPv6).

Instance Methods

`s``.addr`

Returns an array containing information on the socket connection (`AF_INET`, port, hostname, and IP address)

```
s = TCPSocket.open("www.ruby-lang.org", "http")
s.addr# => ["AF_INET", 4030, "dhcp198.priv.netlab.jp",
     "192.168.1.198"]
```

`s``.peeraddr`

Returns an array containing information on the peer socket in the same format as *`s`*`.addr`

```
s = TCPSocket.open("www.ruby-lang.org", "daytime")
s.recvfrom(255)
# => ["Wed Aug  1 00:30:54 2001\r\n", ["AF_INET", 13, "www",
      "210.251.121.214"]]
```

`s``.recvfrom(`*`len`*`[,`*`flags`*`])`

Receives data and returns it in an array that also includes information on the sender's socket in the same format as *`s`*`.addr`

UDPSocket *UDP socket class*

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`UDPSocket` is a class for User Datagram Protocol (UDP), which is a connectionless, unreliable protocol.

Required Library

require 'socket'

Inherited Class

`IPSocket`

Class Methods

`UDPSocket::new([ socktype=Socket::AF_INET])`

`UDPSocket::open([ socktype=Socket::AF_INET])`

Creates a UDP datagram socket

Instance Methods

`s.bind( host, port)`

Binds the socket to *port* on *host*. *host* may be an empty string ("") for `INADDR_ANY` or `<broadcast>` for `INADDR_BROADCAST`.

`s.connect( host, port)`

Connects the socket to *port* on *host*. *host* may be an empty string ("") for `INADDR_ANY` or `<broadcast>` for `INADDR_BROADCAST`.

`s.send( mesg, flags[, to])`

`s.send( mesg, flags[, host, port])`

Sends data on a socket *s*, returning the length of the data sent. If only two arguments are specified, the destination is assumed to be the port of the existing connection. Otherwise, it may be specified using a `struct sockaddr` when calling the method with three arguments or by indicating host and port when specifying four arguments.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

TCPSocket — *TCP/IP socket class*

`TCPSocket` is a class for Transmission Control Protocol (TCP), which is connection-oriented, reliable protocol.

Required Library

require 'socket'

Example

```
require 'socket'

host=(if ARGV.length == 2; ARGV.shift; else "localhost"; end)
print("Trying ", host, " ...")
STDOUT.flush
s = TCPsocket.open(host, ARGV.shift)
print(" done\n")
print("addr: ", s.addr.join(":"), "\n")
print("peer: ", s.peeraddr.join(":"), "\n")
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 *`service`*, which may also be a port number

TCPServer — *TCP/IP server socket class*

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`TCPServer` is a class for server-side TCP sockets. A `TCPServer` waits for client connection by the `accept` method, then returns a `TCPSocket` object connected to the client.

Required Library

require 'socket'

Example

```
require 'socket'

gs = TCPserver.open(0)
addr = gs.addr
addr.shift            # removes "AF_INET"
printf("server is on %s\n", addr.join(":"))

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",] service)`

`TCPServer::open([ host="localhost",] service)`

Creates a server socket

Instance Method

`s.accept`

Waits for a connection and returns a new `TCPSocket` object once one is accepted

UNIXSocket *Unix domain socket class*

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`UNIXSocket` is a class for the Unix domain, which can be specified by the path.

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 information on the socket (`AF_UNIX` and the path)

`s``.path`

Returns the path of the Unix domain socket

`s``.peeraddr`

Returns an array containing information on the peer socket in the same format as *`s`*`.addr`

`s``.recvfrom(`*`len`*`[,`*`flag`*`=0])`

Receives data and returns it in an array that also includes information on the sender's socket in the same format as *`s`*`.addr`

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

UNIXServer — *Unix domain server socket class*

`UNIXServer` is a class for server-side Unix domain sockets. A `UNIXServer` waits for client connection by the `accept` method, then returns a `UNIXSocket` object connected to 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 new `UNIXSocket` object once one is accepted

Socket — *General socket class*

The `Socket` class is necessary to gain access to all the operating system's socket interfaces. Interface structures can be created using `String#pack`.

Required Library

require 'socket'

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Inherited Class

`BasicSocket`

Class Methods

`Socket::for_fd( fd)`

Creates a socket object corresponding to the file descriptor *`fd`* (an integer).

`Socket::getaddrinfo( host, port[, family[, type[, proto[, flags]]]])`

Returns an array containing socket address information (address family, port number, hostname, host IP address, protocol family, socket type, and protocol).

```
Socket::getaddrinfo("www.ruby-lang.org", "echo", Socket::AF_INET, Socket::SOCK_DGRAM)
# => [["AF_INET", 7, "www", "210.251.121.214", 2, 2, 17]]
```

`Socket::gethostbyaddr( addr[, type=Socket::AF_INET)`

Returns an array containing socket address information (address family, port number, hostname, host IP address, protocol family, socket type, and protocol).

```
Socket::getaddrinfo("www.ruby-lang.org", "echo", Socket::AF_INET, Socket::SOCK_DGRAM)
# => [["AF_INET", 7, "www", "210.251.121.214", 2, 2, 17]]
```

`Socket::gethostbyname( name)`

Returns an array containing host information retrieved from a host *`name`*.

```
Socket.gethostbyaddr(([127,0,0,1].pack("CCCC")))
# => ["ev", ["localhost", "ev.netlab.jp"], 2, "\177\000\000\001"]
```

`Socket::gethostname`

Returns the current hostname.

`Socket::getnameinfo( addr[, flags])`

Returns an array containing the name of the host and service retrieved from the specified socket address information. `addr` may be a `struct sockaddr` packed into a string or an array (address family, port, and hostname).

```
sockaddr = [Socket::AF_INET, 80, 127,0,0,1,""].pack("snCCCCa8")
Socket::getnameinfo(sockaddr)                    # => ["ev","www"]
```


Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

```
Socket::getnameinfo(["AF_INET",80,"localhost"]) # => ["ev","www"]
```

`Socket::getservbyname(` *`service`*`[,` *`proto`*`="tcp"])`

Returns the port number for *`service`* and *`proto`* specified.

```
Socket::getservbyname("http") # => 80
```

`Socket::new(domain, type, proto)`

`Socket::open(domain, type, proto)`

Creates a socket.

`Socket::socketpair(` *`domain`*`,` *`type`*`,` *`proto`*`)`

`Socket::pair(` *`domain`*`,` *`type`*`,` *`proto`*`)`

Returns an array containing a pair of connected sockets.

Instance Methods

`s.accept`

Waits for a connection and, once one is accepted, returns a new socket object in an array that also includes a `struct sockaddr` packed into a string.

`s``.addr`

Synonym for *`s`*`.getsockname`. Returns `struct socaddr` packed in a string.

`s``.bind(` *`addr`*`)`

Binds *`s`* to `addr`, a `sockaddr` structure packed into a string.

`s``.connect(` *`addr`*`)`

Connects *`s`* to `addr`, a `sockaddr` structure packed into a string.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

s.listen(*backlog*)

Specifies the size of the *backlog* queue.

s.recvfrom(*len*[, *flags*])

Receives data and returns it in an array that also includes information on the sender's socket in the form of a sockaddr structure packed into a string.

s.peeraddr

Synonym for *s*.getpeername. Returns struct socaddr packed in a string.

Constants

The following constants are defined for use in socket specifications:

```
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 module Socket::Constants and are used by including them in your code.

Net::FTP *FTP connection class*

Net::FTP is a class for File Transfer Protocol (FTP) client-side connection.

Required Library

require 'net/ftp'

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Example

```
require 'net/ftp'

ftp = Net::FTP::new("ftp.ruby-lang.org")
ftp.login("anonymous", "matz@ruby-lang.org")
ftp.chdir("/pub/ruby")
tgz = ftp.list("ruby-*.tar.gz").sort.last
print "the latest version is ", tgz, "\n"
ftp.getbinaryfile(tgz, tgz)
ftp.close
```

Class Methods

`Net::FTP::new([ host[, user[, passwd[, acct]]]])`

`Net::FTP::open( host[, user[, passwd[, acct]]])`

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 closed.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

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*[, *blocksize*=4096[, *callback*]])

f.getbinaryfile(*remote*, *local*[, *blocksize*=4096]) {| *data*| ...}

f.gettextfile(*remote*, *local*[, *callback*])

f.gettextfile(*remote*, *local*) {| *data*| ...}

Retrieves a remote file from the server. If callback or a block is specified, it's executed with the retrieved data. `gettextfile` performs newline code conversion.

f.help([*arg*])

Displays help.

f.lastresp

Returns the server's last response.

f.list(*path*...)

f.dir(*path*...)

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

f.ls(*path...*)

Returns an array of file information in the directory. If a block is specified, it iterates through the listing.

```
f.list("/pub/ruby") # =>
   ["drwxr-xr-x   2 matz     users        4096 Jul 17  1998 1.0",...]
```

f.login([*user*="anonymous"[, *passwd*[, *acct*]]])

Logs into the server.

f.mkdir(*path*)

Creates a directory.

f.mtime(*file*[, *local*=false])

Returns the last modification time of *file*. If *local* is true, it's returned as a local time, otherwise as Coordinated Universal Time (UTC) time.

f.nlst([*dir*])

Returns an array of filenames in the directory.

```
f.nlst("/pub/ruby") # => ["/pub/ruby/1.0",...]
```

f.putbinaryfile(*local*, *remote*[, *blocksize*=4096[, *callback*]])

f.putbinaryfile(*local*, *remote*[, *blocksize*=4096]) {| *data*| ...}

f.puttextfile(*local*, *remote*[, *callback*])

f.puttextfile(*local*, *remote*) {| *data*| ...}

Transfers a file. If callback or a block is specified, the data is passed to it and is run. puttextfile performs newline code conversion.

f.pwd

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

f.getdir

Returns the current directory.

f.passive

Returns `true` if passive mode is enabled.

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 *path*.

f.resume

Returns `true` if resumption of file transfers is enabled.

f.resume= *bool*

Sets file transfer resumption on or off.

f.return_code

Returns the newline code of the current session.

f.return_code= *ret*

Sets the newline code of the current session.

f.size(*file*)

Returns the size of file.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`f.status`

Returns the status.

`f.system`

Returns system information.

`f.welcome`

Returns the server's welcome message.

Net::HTTP — *HTTP connection class*

`Net::HTTP` is a class for Hypertext Transfer Protocol (HTTP) client-side connection.

Required Library

require 'net/http'

Example

```
require 'net/http'

h = Net::HTTP::new("www.ruby-lang.org")
resp, data = h.get("/en/index.html")
print data
```

Class Methods

`Net::HTTP::new([ host="localhost"[, port=80[, proxy[, proxy_port]]]])`

`Net::HTTP::start([ host="localhost"[, port=80[, proxy[, proxy_port]]]])`

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

```
Net::HTTP::start([ host="localhost"[, port=80[, proxy[,
proxy_port]]]]) {| http| ...}
```

Creates a `Net::HTTP` connection object. If a block is specified, the block is executed with the `Net::HTTP` object passed as an parameter. The connection is closed automatically when the block exits.

Instance Methods

```
h.finish
```

Closes the HTTP session.

```
h.get( path[, header[, dest]])
```

```
h.get( path[, header]) {| str| ...}
```

Retrieves data from *path* using a `GET` request, and returns an array containing an `HTTPResponse` object and the data. *header* may be a hash indicating header names and values. *dest* may be a string to which the data is appended. If a block is specified, the retrieved data is passed to it.

```
h.head( path[, header])
```

Sends a `HEAD` request for *path*, and returns the response.

```
h.post( path, data[, header[, dest]])
```

```
h.post( path, data[, header]) {| str| ...}
```

Sends *data* to *path* using a `POST` request, and returns an array containing an `HTTPResponse` object and the reply body. Although the post method's HTTP request type is different, the block and arguments, such as *header* and *dest*, are handled in the same way as `h.get`.

```
h.start
```


Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`h.start {| http| ...}`

Starts an `HTTP` session. If a block is specified, the session is terminated when the block exits.

Net::IMAP — *IMAP access class*

`Net::IMAP` is a class for Internet Message Access Protocol Version 4 (IMAP4) client-side connection. IMAP4 allows you to store and manage messages in the server side.

Required Library

require "net/imap"

Example

```
require "net/imap"
imap = Net::IMAP::new("imap.ruby-lang.org")
  imap.login("matz", "skwkgjv;")
  imap.select("inbox")
  fetch_result = imap.fetch(1..-1, "UID")
  search_result = imap.search(["BODY", "hello"])
  imap.disconnect
```

Class Methods

`Net::IMAP::add_authenticator(`*auth_type*`,` *authenticator*`)`

Adds an authenticator for `Net::IMAP#authenticate`.

`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 connects it to the specified *port* on the named *host*.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Instance Methods

imap.append(*mailbox*, *message*[, *flags* [, *date_time*]])

Appends the *message* to the end of the *mailbox*.

```
imap.append("inbox", <<EOF.gsub(/\n/, "\r\n"), [:Seen], Time.now)
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_type* parameter is a string that represents the authentication mechanism to be used. Currently `Net::IMAP` supports `"LOGIN"` and `"CRAM-MD5"` for the *auth_type*.

```
imap.authenticate('CRAM-MD5", "matz", "crampass")
```

imap.capability

Returns an array of capabilities that the server supports.

```
imap.capability  # => ["IMAP4", "IMAP4REV1", "NAMESPACE", ...]
```

imap.check

Requests a checkpoint of the current mailbox.

imap.close

Closes the current mailbox. Also permanently removes from the mailbox all messages that have the `\Deleted` flag set.

imap.copy(*mesgs*, *mailbox*)

Copies *mesgs* in the current mailbox to the end of the specified *mailbox*. *mesgs* is an array of message sequence numbers or a `Range` object.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

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 mailbox so that messages in the mailbox can be accessed. The selected mailbox is identified as read-only.

imap.expunge

Removes from the current mailbox all messages that have `\Deleted` flag set.

imap.fetch(*mesgs*, *attr*)

Fetches data associated with a message in the mailbox. *mesgs* is an array of message sequence numbers or an `Range` object. The `return_value` is an array of `Net::IMAP::FetchData`.

```
data = imap.uid_fetch(98, ["RFC822.SIZE", "INTERNALDATE"])[0]
data.seqno                    #=> 6
data.attr["RFC822.SIZE"]      #=> 611
data.attr["INTERNALDATE"]     #=> "12-Oct-2000 22:40:59 +0900"
data.attr["UID"]              #=> 98
```

imap.greeting

Returns an initial greeting response from the server.

imap.list(*dir*, *pattern*)

Returns an array of mailbox information in *dir* matching *pattern*. The return value is an array of `Net::IMAP::MailboxList`. *pattern* may contain wildcards `*` (which matches any characters) and `%` (which matches any characters except delimiter).

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

```
imap.list("foo", "*")# matches any mailbox under foo recursively
imap.list("foo", "f%")
                     # matches any mailbox start with "f" under "foo"
```

imap.login(*user*, *password*)

Logs into the server.

imap.logout

Logs out from the server.

imap.lsub(*refname*, *mailbox*)

Returns an array of subscribed mailbox information in *dir* matching *pattern*. The return value is an array of Net::IMAP::MailboxList. *pattern* may contain wildcards * (which matches any characters) and % (which matches any characters except delimiter).

imap.noop

Sends a NOOP command to the server. It does nothing.

imap.rename(*mailbox*, *newname*)

Renames the *mailbox* to *newname*.

imap.responses

Returns recorded untagged responses.

```
imap.select("inbox")
imap.responses["EXISTS"][-1]          #=> 2
imap.responses["UIDVALIDITY"][-1]     #=> 968263756
```

imap.search(*keys*[, *charset*])

Searches the mailbox for messages that match the given searching criteria, and returns an array of message sequence numbers.

```
imap.search(["SUBJECT", "hello"])     #=> [1, 6, 7, 8]
imap.search('SUBJECT "hello"')        #=> [1, 6, 7, 8]
```


Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

imap.select(*mailbox*)

Selects a *mailbox* as a current mailbox so that messages in the mailbox can be accessed.

imap.sort(*sort_keys*, *search_keys*, *charset*)

Returns an array of message sequence numbers that matches *search_keys*_sorted according to the *sort_keys*.

```
imap.sort(["FROM"], ["ALL"], "US-ASCII")
                                    #=> [1, 2, 3, 5, 6, 7, 8, 4, 9]
imap.sort(["DATE"], ["SUBJECT", "hello"], "US-ASCII")
                                    #=> [6, 7, 8, 1]
```

imap.status(*mailbox*, *attr*)

Returns the status of the *mailbox*. The return value is a hash of attributes.

```
imap.status("inbox", ["MESSAGES", "RECENT"]) #=>
     {"RECENT"=>0, "MESSAGES"=>44}
```

imap.store(*mesgs*, *attr*, flags)

Stores data associated with a message in the mailbox. *mesgs* is an array of message sequence numbers or a Range object.

```
# add \Deleted to FLAGS attribute to mails No.6,7,8.
imap.store(6..8, "+FLAGS", [:Deleted])
```

imap.subscribe(*mailbox*)

Appends the specified *mailbox* to the list of active or subscribed mailboxes.

imap.unsubscribe(*mailbox*)

Removes the specified *mailbox* from the list of active or subscribed mailboxes.

imap.uid_copy(*mesg*, *mailbox*)

Copies *mesgs* in the current mailbox to the end of the specified *mailbox*. *mesgs* is an array of unique message identifiers or a Range_object.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

imap.uid_fetch(*mesgs*, *attr*)

Fetches data associated with a message in the current mailbox. *mesgs* is an array of unique message identifiers or an Range object. The return value is an array of Net::IMAP::FetchData.

imap.uid_search(*keys*[, *charset*])

Searches the mailbox for messages that match the given search criteria, and returns an array of unique identifiers.

imap.uid_sort(*sort_keys*, *search_keys*, *charset*)

Returns an array of unique message identifiers that matches *search_keys* sorted according to the *sort_keys*.

imap.uid_store(*mesgs*, *attr*, *flags*)

Stores data associated with a message in the mailbox. *mesgs* is an array of unique message identifiers or a Range object. The return value is an array of Net::IMAP::FetchData.

Net::POP3 — *POP3 connection class*

Net::POP3 is a class for Post Office Protocol Version 3 (POP3) client-side connection. POP3 is a simple protocol that retrieves incoming mail from the server.

Required Library

require 'net/pop'

Example

```
require 'net/pop'

pop = Net::POP3::new("pop.ruby-lang.org")
# authenticate just for SMTP before POP
pop.start("matz", "skwkgjv;") {
  mails = pop.mails          # array of Net::POPMail
}
```

Class Methods

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

```
Net::POP3::new([addr="localhost"[, port=80]])
```

Creates a new `Net::POP3` object.

```
Net::POP3::start([addr="localhost"[, port=80[, ...]]])
```

```
Net::POP3::start([addr="localhost"[, port=80[, ...]]]) {|
pop| ...}
```

Equivalent to `Net::POP3::new(addr, port).start(...)`. A newly created `Net::POP3` object is passed to the block, if specified. The POP3 session is 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` objects.

```
p.start(acct, passwd)
```

```
p.start(acct, passwd) {|pop| ...}
```

Starts a POP3 session. If a block is specified, the session is terminated when the block exits.

Net::APOP *APOP connection class*

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

The `Net::APOP` class has the same interface as Net::POP3. They differ only in their method of authentication.

Required Library
require 'net/pop'

Inherited Class

`Net::POP3`

Net::POPMail *POP mail class*

The `Net::POPMail` class is used by classes `Net::POP3` and `Net::APOP` to return individual message objects.

Required Library
require 'net/pop'

Instance Methods

m`.all([` *dest*`])`

m`.mail([` *dest*`])`

m`.pop([` *dest*`])`

Retrieves the contents of mail messages. If *dest* is specified, each message is appended to it using the << method. If a block is specified, it's passed the contents of each message as a string and run once for each line in the message.

m`.delete`

Deletes the message.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`m.deleted?`

Returns `true` if the message has been deleted.

`m.header([ dest])`

Returns the message header.

`m.size`

Returns the message size in bytes.

`m.top( lineno[, dest])`

Returns the message header and *`lineno`* number of lines of the body.

Net::SMTP *SMTP connection class*

`Net::SMTP` is a class for Simple Mail Transfer Protocol (SMTP) client-side connection. SMTP is a protocol to talk to Mail Transfer Agent (MTA).

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

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

```
Net::SMTP::new([ addr="localhost"[, port=25]])
```

Creates a new `Net::SMTP` object.

```
Net::SMTP::start([ addr="localhost"[, port=25[, ...]]])
```

```
Net::SMTP::start([ad dr="localhost"[, port=25[, ...]]]) {|
smtp| ...}
```

Equivalent to `Net::SMTP::new(addr, port).start(...)`. A newly created `Net::SMTP` object is passed to the block, if specified. The SMTP session is terminated when the block exits.

Instance Methods

```
s.finish
```

Closes an SMTP session.

```
s.ready( from, to) {| adapter| ...}
```

Sends a message, passing an *adapter* object to the block. The message is sent by calling the adapter's `write` method.

```
s.start([ domain[, account[, password[, authtype]]]])
```

```
s.start([ domain[, account[, password[, authtype]]]]) {| smtp| ...}
```

Starts an SMTP session. An `Net::SMTP` object is passed to the block, if specified. The session is terminated when the block exits.

```
s.send_mail( mailsrc, from, to)
```

```
s.sendmail( mailsrc, from, to)
```

Sends mail. *to* may be either a string or an array of strings.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Net::Telnet — *Telnet connection class*

`Net::Telnet` is a class for a Telnet connection. This class isn't only a Telnet protocol client but also a useful tool to interact with interactive services.

When a block is specified with class and instance methods of the `Net::Telnet` class, it's passed status output strings from the server as they are received by the method.

Required Library

require 'net/telnet'

Class Method

```
Net::Telnet::new(options)
```

Creates a `Net::Telnet` object. *`options`* may be a hash specifying zero or more of the following options:

Key	Function	Default
`Binmode`	Binary mode	`false`
`Host`	Telnet server	`"localhost"`
`Output_log`	Output log	`nil (no output)`
`Dump_log`	Dump log	`nil (no output)`
`Port`	Port to connect to	`23`
`Prompt`	Pattern matching the server's prompt	`/[$%#>/ \z/n`
`Telnetmode`	Telnet mode	`true`
`Timeout`	Timeout	`10`
`Waittime`	Wait time	`0`
`Proxy`	Proxy	`nil`

Instance Methods

Besides the following methods, the `Net::Telnet` object delegates its methods to `Socket` object, so that methods provided by the `Socket` class (and its parent classes) are also available for `Net::Telnet`.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

t.binmode

Returns `true` if binary mode is enabled.

t.binmode= *bool*

Sets binary mode on or off.

t.cmd(*options*)

Sends a command to the server. *options* may be the command string to be sent to the server or a hash specifying one or more of the following options:

Key	Function	Default value
`String`	String to be sent	(Required)
`Match`	Pattern to match	Value of `Prompt` option
`Timeout`	Timeout	Value of `Timeout` option

t.login(*options*)

t.login(*user*[, *passwd*])

Logs in to the server. The following hash options may be specified.:

Key	Function
`Name`	Username
`Password`	Password

t.print(*str*)

Sends *str* to the server, performing Telnet protocol translation.

t.telnetmode

Returns `true` if Telnet mode is enabled.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

t.telnetmode= *bool*

Sets Telnet mode on or off.

t.waitfor(*options*)

Waits for a response from the server. The same hash options may specified as with *t*.cmd.

t.write(*str*)

Sends *str* to the server without performing Telnet protocol translation.

CGI — *CGI support class*

CGI provides useful features to implement Common Gateway Interface (CGI) programs, such as retrieving CGI data from server, manipulating cookies, and generating the HTTP header and the HTML body.

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 this"
     end

     cgi.out {
       cgi.html{
         cgi.head{cgi.title{"Walter's Web Arithmetic Page"}} +
         cgi.body{
           cgi.form("post", "/cgi-bin/arith.rb") {
             "input your favorite expression: " +
             cgi.text_field("input", input) +
             cgi.br +
             "the result of you input: " +
             CGI::escapeHTML(value) +
             cgi.br +
             cgi.submit
```


Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

```
            }
          }
        }
      }
```

Required Library

require 'cgi'

Class Methods

`CGI::new([ level="query"])`

Creates a CGI object. *level* may be one of the following options. If one of the HTML levels is specified, the following methods are defined for generating output 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-encoding.

`CGI::unescape( str)`

Expands a string that has been escaped using URL-encoding.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`CGI::escapeHTML(str)`

Escapes HTML special characters, including: & < >.

`CGI::unescapeHTML(str)`

Expands escaped HTML special characters, including: & < >.

`CGI::escapeElement(str[, element...])`

Escapes HTML special characters in the specified HTML elements.

`CGI::unescapeElement(str, element[, element...])`

Expands escaped HTML special characters in the specified HTML elements.

`CGI::parse(query)`

Parses the query and returns a hash containing its key-value pairs.

`CGI::pretty(string[, leader=" "])`

Returns a neatly formatted version of the HTML string. If *leader* is specified, it's written at the beginning of each line. The default value for *leader* is two spaces.

`CGI::rfc1123_date(time)`

Formats the data and time according to RFC-1123 (for example, Sat, 1 Jan 2000 00:00:00 GMT).

Instance Methods

`c[name]`

Returns an array containing the value of the field name corresponding to *name*.

`c.checkbox(name[, value[, check=false]])`

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

c.checkbox(*options*)

Returns an HTML string defining a checkbox field. Tag attributes may be specified in a hash passed as an argument.

c.checkbox_group(*name*, *value*...)

c.checkbox_group(*options*)

Returns an HTML string defining a checkbox group. Tag attributes may be specified in a hash passed as an argument.

c.file_field(*name*[, *size*=20[, *max*]])

c.file_field(*options*)

Returns an HTML string defining a file field.

c.form([*method*="post"[, *url*]]) { ...}

c.form(*options*)

Returns an HTML string defining a form. If a block is specified, the string produced by its output creates the contents of the form. Tag attributes may be specified in a hash passed as an argument.

c.cookies

Returns a hash containing a CGI::Cookie object containing keys and values from a cookie.

c.header([*header*])

Returns a CGI header containing the information in *header*. If *header* is a hash, its key-value pairs are used to create the header.

c.hidden(*name*[, *value*])

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`c.hidden(options)`

Returns an HTML string defining a `HIDDEN` field. Tag attributes may be specified in a hash passed as an argument.

`c.image_button(url[, name[, alt]])`

`c.image_button(options)`

Returns an HTML string defining an image button. Tag attributes may be specified in a hash passed as an argument.

`c.keys`

Returns an array containing the field names from the form.

`c.key?(name)`

`c.has_key?(name)`

`c.include?(name)`

Returns `true` if the form contains the specified field name.

`c.multipart_form([url[, encode]]) { ...}`

`c.multipart_form(options) { ...}`

Returns an HTML string defining a multipart form. If a block is specified, the string produced by its output creates the contents of the form. Tag attributes may be specified in a hash passed as an argument.

`c.out([header]) { ...}`

Generates HTML output. Uses the string produced by the block's output to create the body of the page.

`c.params`

Returns a hash containing field names and values from the form.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

c.params= *hash*

Sets field names and values in the form using a hash.

c.password_field(*name*[, *value*[, *size*=40[, *max*]]])

c.password_field(*options*)

Returns an HTML string defining a password field. Tag attributes may be specified in a hash passed as an argument.

c.popup_menu(*name*, *value*...)

c.popup_menu(*options*)

c.scrolling_list(*name*, *value*...)

c.scrolling_list(*options*)

Returns an HTML string defining a pop-up menu. Tag attributes may be specified in a hash passed as an argument.

c.radio_button(*name*[, *value*[, *checked*=false]])

c.radio_button(*options*)

Returns an HTML string defining a radio button. Tag attributes may be specified in a hash passed as an argument.

c.radio_group(*name*, *value*...)

c.radio_group(*options*)

Returns an HTML string defining a radio button group. Tag attributes may be specified in a hash passed as an argument.

c.reset(*name*[, *value*])

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

c.reset(*options*)

Returns an HTML string defining a reset button. Tag attributes may be specified in a hash passed as an argument.

c.text_field(*name*[, *value*[, *size*=40[, *max*]]])

c.text_field(*options*)

Returns an HTML string defining a text field. Tag attributes may be specified in a hash passed as an argument.

c.textarea(*name*[, *cols*=70[, *rows*=10]]) { ...}

c.textarea(*options*) { ...}

Returns an HTML string defining a text area. If a block is specified, the string produced by its output creates the contents of the text area. Tag attributes may be specified in a hash passed as an argument.

HTML Generation Methods

In addition to the previous instance methods, each CGI object provides the following methods, which generate HTML tag strings corresponding to the HTML level specified when the CGI object was created. These methods return a string that is produced by adding any specified tags to a body created from the string output of the block. Tag attributes may be specified in a hash that is passed as an argument to each method.

Here are the tags common to html3, html4, html4Tr, and html4Fr:

a	address	area	b	base
big	blockquote	body	br	caption
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

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

param	pre	samp	script	select
small	strong	style	sub	submit
sup	table	td	th	title
tr	tt	ul	var	

Here are the `html3` tags:

applet	basefont	center	dir	font
isindex	listing	menu	plaintext	strike
u	xmp			

Here are the `html4` tags:

abbr	acronym	bdo	button	col
colgroup	del	fieldset	ins	label
legend	noscript	object	optgroup	q
span	tbody	tfoot	thead	

Here are the `html4Tr` tags:

abbr	acronym	applet	basefont	bdo
button	center	col	colgroup	del
dir	fieldset	font	iframe	ins
isindex	label	legend	map	menu
noframes	noscript	object	optgroup	q
s	span	strike	tbody	tfoot
thead	u			

Here are the `htmlFr` tags:

abbr	acronym	applet	basefont	bdo
button	center	col	colgroup	del
dir	fieldset	font	frame	frameset
iframe	ins	isindex	label	legend
menu	noframes	noscript	object	optgroup
q	s	span	strike	tbody

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

tfoot	thead	u		

Object Attributes

The CGI class has the following accessors:

accept	Acceptable MIME type
accept_charset	Acceptable character set
accept_encoding	Acceptable encoding
accept_language	Acceptable language
auth_type	Authentication type
raw_cookie	Cookie data (raw string)
content_length	Content length
content_type	Content type
From	Client email address
gateway_interface	CGI version string
path_info	Extra path
path_translated	Converted extra path
Query_string	Query string
referer	Previously accessed URL
remote_addr	Client host address
remote_host	Client hostname
remote_ident	Client name
remote_user	Authenticated user
request_method	Request method (`GET`, `POST`, etc.)
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 class*

`CGI::Cookie` represents the HTTP cookie that carries information between HTTP sessions.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Required Library

require 'cgi'

Object Attributes

The CGI::Cookie class has the following accessors:

`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 `Time` object)
`c.secure`	True if secure cookie

CGI::Session *CGI session class*

`CGI::Session` maintains a persistent session between HTTP accesses. Session information is represented by string to string mapping. Session information can be 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

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

CGI::Session::new(*cgi*[, *option*])

Starts a new CGI session and returns the corresponding `CGI::Session` object. *option* may be an option hash specifying one or more of the following:

Key	Function	Default value
`session_key`	Key name holding the session ID	`_session_id`
`session_id`	Unique session ID	Generated automatically
`new_session`	If `true`, a new session is created	`false`
`database_manager`	Database manager class for storing session data	`CGI::Session::FileStore`

An option hash can specify options when creating the database manager object. The default database manager class (`CGI::Session::FileStore`) recognizes the following options:

Key	Function	Default value
`tmpdir`	Directory for temporary files	`/tmp`
`prefix`	Prefix for temporary files	None

Methods for Database Manager

Database manager object should have following methods:

initialize(*session*[, *options*])

Initializes the database. *session* is a `CGI::Session` object. *options* is an option hash that passed to `CGI::Session::new`

restore

Returns the hash that contains session-specific data from the database

update

Updates the hash returned by `restore`

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`close`

Closes the database

`delete`

Removes the session-specific data from the database

Instance Methods

`s[key]`

Returns the value for the specified session *`key`*

`s[ key]= value`

Sets the value for the specified session *`key`*

`s.delete`

Deletes the session

`s.update`

Writes session data to the database, calling the update method of the database manager object

4.1.2. Operating System Services

A mixed bag of OS services are provided in the Ruby standard library, including curses, filesystem searching and file handling, command-line argument processing, and others.

If you're coming from another scripting language background, these classes will have interfaces you'll find familiar and straightforward access to Unix services. No surprises, here.

Curses	*Character-based interface module*

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

The `Curses` module provides an interface to the character-based interface library called `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

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`crmode`

Alias to the `cbreak`

`nocrmode`

Alias to the `nocbreak`

`delch`

Deletes a character at the cursor position

`deleteln`

Deletes a line at the cursor position

`doupdate`

Updates the screen by queued changes

`echo`

Turns on echo mode

`noecho`

Turns off echo mode

`flash`

Flashes the screen

`getch`

Reads one character from the keyboard

`getstr`

Reads a line of string from the keyboard

`inch`

Reads a character at the cursor position

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`init_screen`

Initializes the `curses` system

`insch(` *ch*`)`

Outputs one character before the cursor

`lines`

Returns the screen height

`nl`

Turns on newline mode, which translates the return key into newline (`\n`)

`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*) position

`standout`

Turns on `standout` (highlighting) mode

`standend`

Turn off `standout` mode

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`stdscr`

Returns the reference to the standard `curses` screen object

`ungetch(` *`ch`*`)`

Pushes *`ch`* back to input buffer

Curses::Window — *Character-based window class*

`Curses::Window` is a class for character-based windows implemented by the `curses` library.

Required Library

```
require "curses"
```

Class Method

`Curses::Window::new(` *`h`*`, `*`w`*`, `*`y`*`, `*`x`*`)`

Creates a new `curses` window of size (*`h`*, *`w`*) at position (*`y`*, *`x`*).

Instance Methods

`w``<<` *`str`*

`w``.addstr(` *`str`*`)`

Outputs *`str`* to the screen.

`w``.addch(` *`ch`*`)`

Outputs one character to the screen.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`w.begx`

Returns the window's beginning *x* position.

`w.begy`

Returns the window's beginning *y* position.

`w.box( v, h)`

Draws a box around the window. *v* is a character that draws a vertical side. *h* is a 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 cursor.

`w.cury`

Returns *y* position of the window's cursor.

`w.delch`

Deletes a character at the window's cursor position.

`w.deleteln`

Deletes a line at the window's cursor position.

`w.getch`

Reads one character from the keyboard.

`w.getstr`

Reads a line of string from the keyboard.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`w.inch`

Reads a character at the window's cursor position.

`w.insch( ch)`

Outputs one character before the window's cursor.

`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*, *x*).

`w.refresh`

Refreshes the window.

`w.setpos( y, x)`

Moves the window's cursor to the position (*y*, *x*).

`w.standend`

Turns on `standout` (highlighting) mode in the window.

`w.standout`

Turns off `standout` mode in the window.

`w.subwin( h, w, y, x)`

Creates a new `curses` subwindow of size (*h*, *w*) in the window at position (*y*, *x*).

Etc *Module for /etc directory data retrieval*

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

The `Etc` module provides functions to retrieve user account-related data from files under */etc* directory. This module is Unix-dependent.

Required Library

require 'etc'

Example

```
require 'etc'

print "you must be ", Etc.getlogin, ".\n"
```

Module Functions

`getlogin`

Returns login name of the user. If this fails, try `getpwuid`.

`getpwnam(`*`name`*`)`

Searches in `/etc/passwd` file (or equivalent database), and returns password entry for the user *`name`*. See `getpwnam(3)` for details. The return value is a `passwd` structure, which includes the following members:

`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)

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`getpwuid([ uid])`

Returns `passwd` entry for the specified *`uid`*. If *`uid`* is omitted, uses the value from `getuid`. See `getpwuid(3)` for details.

`getgrgid( gid)`

Searches in `/etc/group` file (or equivalent database), and returns group entry for the *`gid`*. See `getgrgid(3)` for detail. The return value is a group structure, which includes the following members:

`name`	Group name(string)
`passwd`	Group password(string)
`gid`	Group ID(integer)
`mem`	Array of the group member names

`getgrnam( name)`

Returns the group entry for the specified *`name`*. The return value is the group structure. See `getgrnam(3)` for details.

`group`

Iterates over all `group` entries.

`passwd`

Iterates over all `passwd` entries.

Fcntl *Fcntl constant module*

The `Fcntl` module provides constant definitions for `IO#fcntl`.

Required Library

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

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_APPEND, O_NONBLOCK, or O_ASYNC)
F_GETLK	Gets the flock structure
F_SETLK	Gets lock according to the lock structure (nonblocking)
F_SETLKW	Sets lock like F_SETLK (blocking)
F_RDLCK	Reads lock flag for flock structure
F_WRLCK	Writes lock flag for flock structure
F_UNLCK	Unlocks flag for flock structure
FD_CLOEXEC	Close-on-exec flag
O_CREAT	Creates file if it doesn't exist
O_EXCL	File shouldn't exist before creation
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 module*

The Find module provides a depth-first directory traversal.

Required Library

require 'etc'

Example

```
require 'find'

# prints all files with ".c" extension.
Find.find(".") {|f|
  puts f if /\.c$/ =~ f
}
```


Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Module Functions

`find(path...) {| f| ...}`

Traverses directory tree giving each filename to the block

`prune`

Terminates traversal down from the current directory

ftools — *File utility library*

`ftools` is a library that enhances file handling utility class methods of the `File` class.

Required Library

require 'ftools'

Class Methods

`File::chmod(mode, files...[, verbose=false])`

`ftools` enhances `File::chmod` to take verbose arguments. If the last argument is `true`, prints log to `stderr`.

`File::cmp(path1, path2[, verbose=false])`

`File::compare(path1, path2[, verbose=false])`

Compares two files and returns `true` if they have identical contents. If *`verbose`* is `true`, prints log to `stderr`.

`File::cp(path1, path2[, verbose=false])`

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`File::copy(` *`path1`*`, `*`path2`*`[, `*`verbose`*`=false])`

Copies a file at *`path1`* to *`path2`*. If *`verbose`* is `true`, prints operation log to `stderr`.

`File::install(` *`path1`*`, `*`path2`*`[, `*`mode`*` [, `*`verbose`*`=false]])`

Copies a file at *`path1`* to *`path2`*. If *`mode`* is supplied, its file permission is set to *`mode`*. If file at *`path2`* exists, it's removed before copying. If *`verbose`* is `true`, prints operation log to `stderr`.

`File::makedirs(` *`path...`*`[, `*`verbose`*`=false])`

`File::mkpath(` *`path...`*`[, `*`verbose`*`=false])`

Creates the specified directories. If any parent directories in *`path`* don't exist, it creates them as well. If the last argument is `true`, prints operation log to `stderr`.

`File::move(` *`path1`*`, `*`path2`*`[, `*`verbose`*`=false])`

`File::mv(` *`path1`*`, `*`path2`*`[, `*`verbose`*`=false])`

Moves file from *`path1`* to *`path2`*. If the last argument is `true`, prints operation log to `stderr`.

`File::rm_f(` *`path...`*`[, `*`verbose`*`=false])`

`File::safe_unlink(` *`path...`*`[, `*`verbose`*`=false])`

Removes files regardless of file-permission mode. If the last argument is `true`, prints operation log to `stderr`.

`File::syscopy(` *`path1`*`, `*`path2`*`)`

Copies a file from *`path1`* to *`path2`* using `IO#sysread` and `IO#syswrite`. *`syscopy`* copies permissions of the file as well.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

GetoptLong — *Command line option parser*

The `GetoptLong` class parses command-line option arguments in a way similar to GNU `getoptlong` library.

Required Library

require 'gettextfile'

Example

```
require 'getoptlong'

 opt = GetoptLong.new(
     ['--max-size', '-m', GetoptLong::REQUIRED_ARGUMENT],
     ['--quiet',    '-q', GetoptLong::NO_ARGUMENT],
     ['--help',           GetoptLong::NO_ARGUMENT],
     ['--version',        GetoptLong::NO_ARGUMENT])
 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` object. If *`options`* are given, they are passed to the `set_options` method.

Instance Methods

`opt``.each {| `*`optname`*`, `*`optarg`*`| ...}`

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

opt.each_option {| *optname*, *optarg*| ...}

Iterates over each command-line option. Option name and value are passed to the block.

opt.get

opt.get_option

Retrieves an option from command-line arguments, and returns the name-value pair of option.

opt.error

opt.error?

Returns type of the current error or `nil` if no error occurs.

opt.error_message

Returns an error message of the current error or `nil` if no error occurs.

opt.ordering= *ordering*

Sets option ordering. *ordering* is any of `PERMUTE`, `REQUIRE_ORDER`, or `RETURN_IN_ORDER`.

opt.ordering

Returns current ordering.

opt.quiet= *bool*

Sets status of quiet mode. In quiet mode, option parser doesn't output error messages to `stdout` on errors. The default value is `false`.

opt.quiet

opt.quiet?

Returns current status of quiet mode.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

opt.set_options(*option...*)

Sets command-line options that your program accepts, specified by arrays of option names and option type constants.

Option type is any of NO_ARGUMENT, REQUIRED_ARGUMENT, or OPTIONAL_ARGUMENT. You have to call set_options before invoking get, get_option, each, or each_option.

opt.terminate

Terminates option processing. Raises RuntimeError exception if any errors occur before termination.

opt.terminated?

Returns true if option processing is finished without causing errors, otherwise 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*

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

The PTY module executes commands as if their standard I/O is connected to *ttys*.

Required Library

require "pty"

Module Functions

`getpty(` *command*`)`

`spawn(` *command*`)`

Reserves a PTY, executes *command* over the PTY, and returns an array of three elements (reading I/O, writing I/O, and the PID of the child process). With a block, the array is passed to the block as block parameters. SIGCHLD is captured while *command* is running.

`protect_signal { ...}`

Protects block execution from SIGCHLD signal exception. This is required to invoke other subprocesses while using any PTY.

`reset_signal`

Disables to handle SIGCHLD while PTY subprocess is active.

Readline — *GNU readline library interface*

The Readline module provides a interface to the GNU line editing library named readline.

Required Library

require 'readline'

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Example

```
require 'readline'
include Readline
line = readline("Prompt> ", true)
```

Module Function

`readline(`*prompt*`,` *add_history*`)`

Reads one line with line editing. If the add is `true`, the line is also added to the history.

Module Methods

`Readline::completion_proc=`*proc*

Specifies `Proc` object to determine completion behavior. Takes input string, and returns completion candidates.

`Readline::completion_proc`

Returns the completion `Proc` object.

`Readline::completion_case_fold=bo` *ol*

Sets whether or not to ignore case on completion.

`Readline::completion_case_fold`

Returns `true` if completion ignores case.

`Readline::completion_append_character=` *char*

Specifies a character to be appended on completion. If empty string ("") or nil is specified, nothing is appended.

`Readline::completion_append_character`

Returns a string containing a character to be appended on completion. Default is a space.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`Readline::vi_editing_mode`

Specifies *vi* editing mode.

`Readline::emacs_editing_mode`

Specifies Emacs editing mode.

Constant

`HISTORY`

The history buffer; it behaves just like an array.

Tempfile — *Temporary file class*

Temporary files are always deleted when garbage collection is activated, and Ruby 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 automatically removed
```

Class Method

`Tempfile::new(`*`basename`*`[, `*`tmpdir`*`="/tmp"])`

Opens a temporary file that includes *`basename`* as part of the filename in `w+` mode.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Instance Methods

`t.open`

Reopens the temporary file, allowing its contents to be read from the beginning of the file.

`t.close([ permanently=false])`

Closes the temporary file. If *permanently* is `true`, the file is also deleted.

`t.path`

Returns the path of the temporary file.

In addition to the previous methods, objects of class `Tempfile` also possess all instance methods of class `File`.

Win32API — *Microsoft Windows API access class*

Win32API represents functions in Windows DLLs.

Required Library

require 'Win32API'

Example

```
require 'Win32API'

getch = Win32API.new("crtdll", "_getch", [], 'L')
puts getch.Call.chr
```

Class Method

`Win32API::new( dll, proc, import, export)`

Returns the object representing the `Win32API` function specified by *proc* name in *dll*, which has the signature specified by *import* and *export*. *import* is an array of strings denoting types. *export* is a type specifying string. Type string is any of 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. Arguments must conform the signature specified by `Win32API::new`.

4.1.3. Threads

Threading classes in the Ruby standard library extend and enhance the built-in library support for parallel programming with support for condition variables, monitors and mutexes, queues and a handy-dandy thread termination watcher class.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

ConditionVariable — *Synchronization condition variable class*

This class represents condition variables for synchronization between threads.

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 queue

c`.wait(`*mutex*`)`

Waits on `condition variable`

Monitor — *Exclusive monitor section class*

This class represents exclusive sections between threads.

Required Library

require 'monitor'

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

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 monitor.

m`.synchronize{ ...}`

Enters exclusive section and executes the block. Leaves the exclusive section automatically when the block exits.

m`.try_enter`

Attempts to enter exclusive section. Returns `false` if lock fails.

MonitorMixin — *Exclusive monitor section mix-in module*

Adds monitor functionality to an arbitrary object by mixing the modules with `include`.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

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 monitor.

m.mon_synchronize{ ...}

Enters exclusive section and executes the block. Leaves the exclusive section automatically when the block exits.

m.try_mon_enter

Attempts to enter exclusive section. Returns `false` if lock fails.

Mutex — *Mutual exclusion class*

This class represents mutually exclusive locks.

Required Library

require 'thread'

Class Method

Mutex::new

Creates a `Mutex` object

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

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 releases the lock when the block exits.

`m.try_lock`

Attempts to lock *m*. Returns `false` if lock fails.

`m.unlock`

Releases lock on *m*.

Queue — *Message queue class*

This class provides the way to communicate data between threads.

Required Library

require 'thread'

Class Method

`Queue::new`

Creates a `queue` object

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Instance Methods

`q.empty?`

Returns `true` if the queue is empty.

`q.num_waiting`

Returns the number of threads waiting on the queue.

`q.pop([ non_block=false])`

Retrieves data from the queue. If the queue is empty, the calling thread is suspended until data is pushed onto the queue. If *non_block* is `true`, the thread isn't suspended, and an exception is raised.

`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 class*

This class represents queues of specified size capacity. The `push` operation may be blocked if the capacity is full.

Required Library

require 'thread'

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Inherited Class

Queue

Class Method

`SizedQueue::new(`*max*`)`

Creates a fixed-length queue with a maximum size of *max*

Instance Methods

q`.max`

Returns the maximum size of the queue

q`.max=`*n*

Sets the maximum length of the queue

ThreadsWait *Thread termination watcher class*

This class watches termination of multiple threads.

Required Library

require 'thwait'

Class Methods

`ThreadsWait::all_waits(`*th*`,...)`

`ThreadsWait::all_waits(`*th*`...) { ...}`

Waits until all specified threads are terminated. If a block is supplied for the method, evaluates it for each thread termination.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`ThreadsWait.new( th...)`

Creates a `ThreadsWait` object, specifying threads to wait.

Instance Methods

`th.threads`

Lists threads to be synchronized

`th.empty?`

Returns `true` if there is no thread to be synchronized.

`th.finished?`

Returns `true` if there is any terminated thread.

`th.join( th...)`

Waits for specified threads.

`th.join_nowait( th...)`

Specifies threads to wait; non-blocking.

`th.next_wait`

Waits until any specified thread is terminated.

`th.all_waits`

`th.all_waits{ ...}`

Waits until all specified threads are terminated. If a block is supplied for the method, evaluates it for each thread termination.

4.1.4. Data Persistence

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

These libraries provide interfaces or hooks into databases via various implementations (OS, GNU, and public domain).

Ruby lets you store and retrieve "live" data and objects in the filesystem with tools you're probably used through the `DBM`, `GDBM`, `SDBM`, and `PStore` classes.

DBM — *DBM class*

`DBM` implements a database with the same interface as a hash. Keys and values are limited to strings. Uses `ndbm` library included in operating systems.

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 rights to the database are specified in *`mode`* as an integer.

Instance Methods

The DBM class has all the methods of the Hash class except for `default`, `default=`, `dup`, and `rehash`. `DBM` also has the `close` method, which isn't in `Hash`.

`d``.close`

Closes `DBM` database

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

GDBM — *GDBM class*

GNU implementation of `DBM`. Has the same interface as `DBM`.

Required Library

require 'gdbm'

Instance Methods

In addition to methods from the `DBM` class, the `GDBM` class has the `reorganize` method.

d.reorganize

Reconfigures the database; shouldn't be used with great frequency

SDBM — *SDBM class*

Public domain implementation of `DBM`. Has the same interface as `DBM`. Runs almost anywhere but has inferior performance and data-size limitations compared to other `DBM`s.

Required Library

require 'sdbm'

PStore — *Simple object-oriented database class*

`PStore` is a simple object-oriented database class that provides almost arbitrary data persistence (using `Marshal`) and transaction.

Required Library

require 'pstore'

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Class Method

`PStore::new(path)`

Creates a database object. Data is stored in a file specified by *path*.

Instance Methods

`p.transaction {|ps| ...}`

Starts a transaction (a series of database operations). Access to the contents of the database can be achieved only through this transaction method.

`p[name]`

Retrieves an object stored in the database under the key name.

`p[name]=obj`

Stores *obj* in the database under the key name. When the transaction is completed, all objects accessed reflexively by *obj* (see `Marshal` in Section 3.4) are saved in a file.

`p.root?(name)`

Returns `true` if the key name exists in the database.

`p.commit`

Completes the transaction. When this method is called, the block passed to the transaction method is executed, and changes to the database are written to the database file.

`p.abort`

Aborts the transaction. When this method is called, the execution of the block passed to the transaction method is terminated, and changes made to database objects during the transaction aren't written to the database file.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

4.1.5. Numbers

These libraries let you handle numeric calculations using advanced numbers such as `Complex`, `Rational`, and `Matrix`.

Complex	*Complex number class*

When this library is loaded with `require`, the ability of the `Math` module is expanded to 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. The former is recommended.

Instance Methods

c`.abs`

Returns the absolute value of the complex number *c*.

c`.abs2`

Returns the square of the absolute value of the complex number *c*.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`c.arg`

Returns the argument of the complex number *c*.

`c.conjugate`

Returns the `conjugate` of the complex number *c*.

`c.image`

Returns the imaginary part of the complex number *c*. The `Complex` library adds the 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 number *c*. The `Complex` library adds the real method to the `Numeric` class.

Rational — *Rational number class*

When this library is loaded with `require`, the `**` operator method of the `Integer` class can handle rational numbers, and the following methods are added to the `Integer` class:

`to_r`

Converts a number to a rational number

`lcm`

Returns the least common multiple

`gcd`

Returns the greatest common divisor

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Required Library

require 'rational'

Inherited Class

`Numeric`

Class Methods

`Rational(`*a*`,` *b*`)`

`Rational::new(`*a*`,` *b*`)`

Creates a rational number object. The former, `Rational(`*a*`,`*b*`)`, is recommended.

Matrix — *Matrix class*

Required Library

require 'matrix'

Class Methods

`Matrix::[` *row...*`]`

Creates a matrix where *row* indicates each row of the matrix.

```
Matrix[[11, 12], [21, 22]]
```

`Matrix::identity(` *n*`)`

`Matrix::unit(` *n*`)`

`Matrix::I(` *n*`)`

Creates an `n`-by-*n* unit matrix.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Matrix::columns(*columns*)

Creates a new matrix using *columns* as sets of column vectors.

```
Matrix::columns([[11, 12], [21, 22]]) # => Matrix[[11, 21], [12, 22]]
```

Matrix::column_vector(*column*)

Creates a 1-by-*n* matrix such that column vector is *column*.

Matrix::diagonal(*value...*)

Creates a matrix where diagonal components are specified by *value*.

```
Matrix.diagonal(11, 22, 33)  # => Matrix[[11, 0, 0],
      [0, 22, 0], [0, 0, 33]]
```

Matrix::rows(*rows*[, *copy*=true])

Creates a matrix where *rows* is an array of arrays that indicates rows of the matrix. If the optional argument *copy* is `false`, use the given arrays as the internal structure of the matrix without copying.

```
Matrix::rows([[11, 12], [21, 22]])
```

Matrix::row_vector(*row*)

Creates an 1-by-*n* matrix such that the row vector is *row*.

Matrix::scalar(*n, value*)

Creates an *n*-by-*n* diagonal matrix such that the diagonal components are given by *value*.

```
Matrix::scalar(3,81)    # => Matrix[[81,0,0],[0,81,0],[0,0,81]]


p ParseDate::parsedate("Fri Aug  3 17:16:57 JST 2001")
# => [2001, 8, 3, 17, 16, 57, "JST", 5]
p ParseDate::parsedate("1993-02-24")
# => [1993, 2, 24, nil, nil, nil, nil, nil]
```

Matrix::zero(*n*)

Creates an *n*-by-*n* zero matrix.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

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 iteration of the given block over all components of the matrix `m`.

`m.column( j)`

Returns the `j`-th column vector of the matrix `m`. When the block is supplied for the method, the block is iterated over all column vectors.

`m.column_size`

Returns the number of columns.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

m.column_vectors

Returns array of column vectors of the matrix *m*.

m.determinant

m.det

Returns the determinant of the matrix *m*.

m.inverse

m.inv

Returns an inversed matrix of the matrix *m*.

m.minor(*from_row, row_size, from_col, col_size*)

m.minor(*from_row..to_row, from_col..to_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 matrix *m*. When the block is supplied for the method, the block is iterated over all row vectors.

m.row_size

Returns the number of rows.

m.row_vectors

Returns an array of row vectors of the matrix *m*.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

m.regular?

Returns `true` if *m* is a regular matrix.

m.singular?

Returns `true` if *m* is a singular (i.e., nonregular) matrix.

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 your job done without reinventing the wheel. Ruby provides support in the standard library for a small number of commonly used design patterns. This group of libraries provides advanced object-oriented programming techniques for delegators, forwardables, singletons, and observers.

Delegator *Delegator pattern superclass*

`Delegator` is an abstract class for the Delegator design pattern. Delegation is actually achieved by creating a subclass of the `Delegator` class.

Required Library

require 'delegate'

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Class Method

`Delegator::new(`*obj*`)`

Creates a delegate object to which methods of *obj* are forwarded.

Instance Method

`__getobj__`

Returns the object to which methods are forwarded. Needs to be redefined in the subclass.

SimpleDelegator — *Simple concrete Delegator pattern class*

This class allows for easy implementation of the Delegator design pattern.

Required Library

require 'delegate'

Inherited Class

`Delegator`

Class Method

`SimpleDelegator::new(`*obj*`)`

Creates an object that forwards methods to *obj*

Instance Method

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`__setobj__`

Sets the object to which methods are forwarded

DelegatorClass — *Class creation function for Delegator patterns*

This function dynamically creates a class that delegates to other fixed classes.

Required Library
require 'delegate'

Function

`DelegateClass(c)`

Creates a new class to which the methods of class `c` are forwarded

Method of Generated Class

`D::new(obj)`

Creates a delegate object with `obj` as the object to which methods are forwarded

Forwardable — *Module to add selected method delegations to a class*

The `Forwardable` module provides more explicit method delegation. You can specify method name and destination object explicitly.

Required Library
require "forwardable"

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Example

```
class Foo
  extend Forwardable
  # ...
  def_delegators("@out", "printf", "print")
  def_delegators(:@in, :gets)
  def_delegator(:@contents, :[], "content_at")
end
f = Foo.new
f.printf("hello world\n")   # forward to @out.printf
f.gets                     # forward to @in.gets
f.content_at(1)            # forward to @contents.[]
```

Instance Methods

f.def_delegator(*accessor*, *method*[, *alt*= *method*])

f.def_instance_delegator(*accessor*, *method*[, alt= *method*])

Defines delegation from *method* to *accessor*. If *alt* is specified, *alt* method is called instead of *method*.

f.def_delegators(*accessor*, *method...*)

f.def_instance_delegators(*accessor*, *method...*)

Defines delegation to *accessor* for each *method*.

SingleForwardable *Selective delegation module*

The SingleForwardable module provides more explicit method delegation for a specific object.

Required Library

require 'forwardable'

Example

```
require 'forwardable'

# ...
g = Goo.new
g.extend SingleForwardable
```


Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

```
g.def_delegator("@out", :puts)
g.puts("hello world")            # forward to @out.puts
```

Instance Methods

f.def_singleton_delegator(*accessor*, *method*[, *alt*= *method*])

f.def_delegator(*accessor*, *method*[, *alt*= *method*])

Defines delegation from *method* to *accessor*. If *alt* is specified, *alt* method is called instead of *method*.

f.def_singleton_delegators(*accessor*, *method...*)

f.def_delegators(*accessor*, *method...*)

Defines delegation to *accessor* for each *method*.

Singleton *Singleton pattern module*

The Singleton module allows the implementation of the Singleton design pattern. By including the module, you can ensure that only one instance of a class is created.

Required Library

require 'singleton'

Class Method

instance

Returns the only instance of the class. If an instance has already been created, it's reused. instance is a class method added to classes that include the Singleton module.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Observable — *Observable pattern module*

The `Observable` module allows the implementation of the Observer design pattern. Classes that include this module can notify multiple observers of changes in self. Any object can become an observer as long as it has the update method.

Required Library

```
require 'observer'
```

Instance Methods

`o.add_observer( obj)`

Adds observer *obj* as an observer of *o*.

`o.count_observers`

Returns the number of observers of *o*.

`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 observer of *o*.

`o.delete_observers`

Removes all observers of *o*.

`o.notify_observers([ arg...])`

If *o*'s changed state is `true`, invokes the `update` method of each observer, passing it the specified arguments.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

4.1.7. Miscellaneous Libraries

It almost goes without saying, but there's always a bunch of stuff that doesn't quite fit into any category. Ruby's standard library is no exception. This group of libraries includes anything that isn't in one of the preceding groups.

In Ruby's standard library, you'll find classes providing abstractions for date manipulation, timeouts on long operations, and `MD5` and `SHA1` message digests.

Date — *Date class*

Date is a class to represent the calendar date. Date is based on the Julian day number, which is the number of days since midday, January 1st 4713 BC.

Currently we use the Gregorian calendar, but the Julian calendar was used prior to that time (before 1752 in England, for example). The calendar shift date is different in each country. Date class can handle both calendars and arbitrary shift dates.

There's no relation between Julian day number and Julian calendar; it's just coincidence.

Required Library

```
require 'date'
```

Example

```
require 'date'

# 3000 days after Ruby was born
puts Date::new(1993,2,24)+3000, "\n"  # 2001-05-13
```

Included Module

```
Comparable
```

Class Methods

```
Date::exist?(year, month, day[, start])
```


Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`Date::exist3?(`*`year, month, day`*`[,`*`start`*`])`

Returns the Julian day number corresponding to the specified *`year`*, *`month`*, and *`day`* of year, if they are correct. If they aren't correct, returns `nil`.

`Date::exist2?(`*`year, yday`*`[,`*`start`*`])`

Returns the Julian day number corresponding to the specified *`year`* and *`day`* of year, if they are correct. If they aren't correct, returns `nil`.

`Date::existw?(`*`year, week, wday`*`[,`*`start`*`])`

Returns the Julian day number corresponding to the specified calendar week-based *`year`*, calendar *`week`*, and calendar *`weekday`*, if they are correct. If they aren't correct, returns `nil`.

`Date::new(`*`year, month, day`*`[,`*`start`*`])`

`Date::new3(`*`year, month, day`*`[,`*`start`*`])`

Creates a `Date` object corresponding to the specified *`year`*, *`month`*, and *`day`* of the month.

`Date::new1(`*`jd`*`[,`*`start`*`])`

Creates a `Date` object corresponding to the specified Julian day number.

`Date::new2(`*`year, yday`*`[,`*`start`*`])`

Creates a Date object corresponding to the specified *`year`* and day of the year.

`Date::neww(`*`year, week, wday`*`[,`*`start`*`])`

Creates a `Date` object corresponding to the specified calendar week-based *`year`*, calendar *`week`*, and calendar weekday.

`Date::today([`*`start`*`])`

Creates a `Date` object corresponding to today's date.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Instance Methods

`d << n`

Returns a `Date` object that is `n` months earlier than `d`.

`d >> n`

Returns a `Date` object that is `n` months later than `d`.

`d <=> x`

Compares dates. `x` may be a `Date` object or an integer (Julian day number).

`d + n`

Returns a `Date` object that is `n` days later than `d`.

`d - x`

Returns the difference in terms of days if `x` is another `Date` object. If `x` is an integer, returns a `Date` object that is `x` days earlier than `d`.

`d.cwday`

Returns the calendar weekday (1-7, Monday being 1) for `d`.

`d.cweek`

Returns the calendar week (1-53) for `d`.

`d.cwyear`

Returns the calendar week-based year for `d`.

`d.day`

`d.mday`

Returns the day of the month (1-31) for `d`.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

d.downto(*min*) {|*date*| ...}

Runs block on dates ranging from *d* down to *min*. Equivalent to *d*.step(*min*), -1) {|*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 number for *d*. Modified Julian day number is the number of days since midnight November 17, 1858.

d.mon

d.month

Returns the month (1-12) for *d*.

d.newsg([*start*])

Copies *d* to a new Date object and returns it after converting its cutover date to *start*.

d.next

d.succ

Returns a new Date object one day later than *d*.

d.sg

Returns the Julian day number of the start of Gregorian dates for *d*.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

d.step(*limit*, *step*) {| *date*| ...}

Runs block on Date objects from *d* to *limit* incrementing *step* number of days each time.

d.upto(*max*) {| *date*| ...}

Runs block on dates ranging from *d* up to *max*. Equivalent to *d*.step(*max*, 1) {| *date*|...}.

d.wday

Returns the day of the week for *d* (0-6, Sunday being 0).

d.yday

Returns the day of the year for *d* (1-366).

d.year

Returns the year for *d*.

Constants

MONTHNAMES

An array of the names of the months of the year

DAYNAMES

An array of the names of the days of the week (Sunday being the first element)

ITALY

Gregorian calendar start day number in Italy

ENGLAND

Gregorian calendar start day number in England

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

`JULIAN`

Start specifier for Julian calendar

`GREGORIAN`

Start specifier for Gregorian calendar

ParseDate — *Date representation parser module*

The `ParseDate` module parses strings that represent calendar dates in various formats.

Required Library

require 'parsedate'

Module Function

`parsedate( str[, cyear=false])`

Parses a date and/or time expression within *`str`* and returns the parsed elements (year, month, day, hour, minute, second, time zone, and day of the week) as an array. Sunday is represented as 0 in the day-of-the-week element. `nil` is returned for elements that can't be parsed or have no corresponding string representation. If *`cyear`* is `true`, years with a value of 68 or less are interpreted as being in the 2000s and years ranging from 69 to 99 are interpreted as being in the 1900s. In summary, beware of the Y2K69 problem!

timeout — *Time out a lengthy procedure*

Times out a lengthy procedure or those that continue execution beyond a set duration.

Required Library

require 'timeout'

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

Function

`timeout(sec) { ...}`

Executes the block and returns `true` if the block execution terminates successfully prior to elapsing of the timeout period, otherwise immediately terminates execution of the block and raises a `TimeoutError` exception.

```
require 'timeout'
status = timeout(5) {
  # something that may take time
}
```

MD5 — *MD5 message digest class*

The `MD5` class provides a one-way hash function from arbitrary text data by using the algorithm described in RFC-1321

Example

```
requires 'md5'

md5 = MD5::new("matz")
puts md5.hexdigest # prints: 3eb50a8d683006fdf941b9860798f9aa
```

Class Methods

`MD5::new([ str])`

`MD5::md5([ str])`

Creates a new `MD5` object. If a string argument is given, it's added to the object.

Instance Methods

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

md.clone

Copies the MD5 object.

md.digest

Returns the MD5 hash of the added strings as a string of 16 bytes.

md.hexdigest

Returns the MD5 hash of the added strings as a string of 32 hexadecimal digits.

md.update(*str*)

md << *str*

Updates the MD5 object with the string *str*. Repeated calls are equivalent to a single call with the concatenation of all the arguments, i.e., m.update(a); m.update(b) is equivalent to m.update(a+b), and m << a << b is equivalent to m << a+b.

SHA1 — *SHA1 message digest class*

The SHA1 class provides a one-way hash function from arbitrary text data.

Class Methods

SHA1::new([*str*])

SHA1::sha1([*str*])

Creates a new SHA1 object. If a string argument is given, it's added to the object.

Instance Methods

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01

sh.clone

Copies the SHA1 object.

sh.digest

Returns the SHA1 hash of the added strings as a string of 16 bytes.

sh.hexdigest

Returns the SHA1 hash of the added strings as a string of 32 hexadecimal digits.

sh.update(*str*)

sh << *str*

Updates the SHA1 object with the string *str*. Repeated calls are equivalent to a single call with the concatenation of all the arguments, i.e., m.update(a); m.update(b) is equivalent to m.update(a+b), and m << a << b is equivalent to m << a+b.

Ruby in a Nutshell By Yukihiro Matsumoto ISBN: 0-59600-214-9 Publisher: O'Reilly
Print Publication Date: 2001/11/01