RUBY FILE CLASS AND METHODS

http://www.tutorialspoint.com/ruby/ruby_file_methods.htm Copyright © tutorialspoint.com

A *File* represents an *stdio* object that connected to a regular file. open returns an instance of this class for regular files.

Class Methods:

SN	Methods with Description
1	**File::atime*path*** Returns the last access time for *path*.
2	**File::basename*path*[, *suffix*]** Returns the filename at the end of *path*. If *suffix* is specified, it's deleted from the end of the filename. e.g. File.basename " */home/users/bin/ruby.exe* " #=> "ruby.exe"
3	**File::blockdev?*path*** Returns true if path is a block device.
4	**File::chardev?*path*** Returns true if path is a character device.
5	**File::chmod*mode, path. . .*** Changes the permission mode of the specified files.
6	**File::chown*owner, group, path. . .*** Changes the owner and group of the specified files.
7	**File::ctime*path*** Returns the last inode change time for path.
8	**File::delete*path. . .*** **File::unlink*path. . .*** Deletes the specified files.
9	**File::directory?*path*** Returns true if path is a directory.
10	**File::dirname*path*** Returns the directory portion of path, without the final filename.
11	**File::executable?*path***

	Returns true if path is executable.
12	**File::executable_real?*path*** Returns true if path is executable with real user permissions.
13	**File::exist?*path*** Returns true if path exists.
1	**File::expand_path*path*[, *dir*]** Returns the absolute path of path, expanding ~ to the process owner's home directory, and ~user to the user's home directory. Relative paths are resolved from the directory specified by dir, or the current working directory if dir is omitted.
14	**File::file?*path*** Returns true if path is a regular file.
15	**File::ftype*path*** Returns one of the following strings representing a file type: • **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
16	**File::grpowned?*path*** Returns true if path is owned by the user's group.
17	**File::join*item. . .*** Returns a string consisting of the specified items joined together with File::Separator separating each item. e.g File::join "" , " *home* " , " *usrs* " , " *bin* " # => "/home/usrs/bin"
18	**File::link*old, new*** Creates a hard link to file old.
19	**File::lstat*path*** Same as stat, except that it returns information on symbolic links themselves, not the files they point to.

20	**File::mtime*path*** Returns the last modification time for path.
21	**File::new*path*[, *mode* =" *r* "]** **File::open*path*[, *mode* =" *r* "]** **File::open*path*[, *mode* =" *r* "] {|f| ...}** Opens a file. If a block is specified, the block is executed with the new file passed as an argument. The file is closed automatically when the block exits. These methods differ from Kernel.open in that even if path begins with |, the following string isn't run as a command.
22	**File::owned?*path*** Returns true if path is owned by the effective user.
23	**File::pipe?*path*** Returns true if path is a pipe.
24	**File::readable?*path*** Returns true if path is readable.
25	**File::readable_real?*path*** Returns true if path is readable with real user permissions.
25	**File::readlink*path*** Returns the file pointed to by path.
26	**File::rename*old, new*** Changes the filename from old to new.
27	**File::setgid?*path*** Returns true if path's set-group-id permission bit is set.
28	**File::setuid?*path*** Returns true if path's set-user-id permission bit is set.
29	**File::size*path*** Returns the file size of path.
30	**File::size?*path*** Returns the file size of path, or nil if it's 0.
31	**File::socket?*path*** Returns true if path is a socket.

32	**File::split***path* Returns an array containing the contents of path split into File::dirname*path* and File::basename*path*.
33	**File::stat***path* Returns a File::Stat object with information on path.
34	**File::sticky?***path* Returns true if path's sticky bit is set.
35	**File::symlink***old, new* Creates a symbolic link to file old.
36	**File::symlink?***path* Returns true if path is a symbolic link.
37	**File::truncate***path, len* Truncates the specified file to len bytes.
38	**File::unlink***path. . .* Delete a file given at the path
39	**File::umask[***mask***]** Returns the current umask for this process if no argument is specified. If an argument is specified, the umask is set, and the old umask is returned.
40	**File::utime***atime, mtime, path. . .* Changes the access and modification times of the specified files
41	**File::writable?***path* Returns true if path is writable.
42	**File::writable_real?***path* Returns true if path is writable with real user permissions.
43	**File::zero?***path* Returns true if the file size of path is 0.

Instance Methods:

Assuming **f** is an instance of **File** class:

SN	Methods with Description
1	**f.atime** Returns the last access time for f.
2	**f.chmode***mode* Changes the permission mode of f.
3	**f.chown***owner, group* Changes the owner and group of f.
4	**f.ctime** Returns the last inode change time for f.
5	**f.flock***op* Calls flock2. op may be 0 or a logical or of the File class constants LOCK_EX, LOCK_NB, LOCK_SH, and LOCK_UN.
6	**f.lstat** Same as stat, except that it returns information on symbolic links themselves, not the files they point to.
7	**f.mtime** Returns the last modification time for f.
8	**f.path** Returns the pathname used to create f.
9	**f.reopen***path*[, *mode* =" *r* "] Reopens the file.
10	**f.truncate***len* Truncates f to len bytes.

Processing math: 100%