Ruby 文件的输入与输出 Ruby Dir 类和方法

Ruby File 类和方法

File 表示一个连接到普通文件的 *stdio* 对象。open 为普通文件返回该类的一个实例。

类方法

序号	方法 & 描述
1	**File::atime(path)** 返回 *path* 的最后访问时间。
2	**File::basename(path[, suffix])** 返回 *path* 末尾的文件名。如果指定了 *suffix*，则它会从文件名末尾被删除。 例如：File.basename("/home/users/bin/ruby.exe") #=> "ruby.exe"
3	**File::blockdev?(path)** 如果 path 是一个块设备，则返回 true。
4	**File::chardev?(path)** 如果 path 是一个字符设备，则返回 true。
5	**File::chmod(mode, path...)** 改变指定文件的权限模式。
6	**File::chown(owner, group, path...)** 改变指定文件的所有者和所属组。
7	**File::ctime(path)** 返回 path 的最后一个 inode 更改时间。
8	**File::delete(path...)** **File::unlink(path...)** 删除指定的文件。
9	**File::directory?(path)** 如果 path 是一个目录，则返回 true。
10	**File::dirname(path)** 返回 path 的目录部分，不包括最后的文件名。
11	**File::executable?(path)** 如果 path 是可执行的，则返回 true。

12	**File::executable_real?(path)** 如果 path 通过真正的用户权限是可执行的，则返回 true。
13	**File::exist?(path)** 如果 path 存在，则返回 true。
1	**File::expand_path(path[, dir])** 返回 path 的绝对路径，扩展 ~ 为进程所有者的主目录，~user 为用户的主目录。相对路径是相对于 dir 指定的目录，如果 dir 被省略则相对于当前工作目录。
14	**File::file?(path)** 如果 path 是一个普通文件，则返回 true。
15	**File::ftype(path)** 返回下列其中一个字符串，表示文件类型： • **file** - 普通文件 • **directory** - 目录 • **characterSpecial** - 字符特殊文件 • **blockSpecial** - 块特殊文件 • **fifo** - 命名管道（FIFO） • **link** - 符号链接 • **socket** - Socket • **unknown** - 未知的文件类型
16	**File::grpowned?(path)** 如果 path 由用户的所属组所有，则返回 true。
17	**File::join(item...)** 返回一个字符串，由指定的项连接在一起，并使用 File::Separator 进行分隔。 例如：File::join("", "home", "usrs", "bin") # => "/home/usrs/bin"
18	**File::link(old, new)** 创建一个到文件 old 的硬链接。
19	**File::lstat(path)** 与 stat 相同，但是它返回自身符号链接上的信息，而不是所指向的文件。
20	**File::mtime(path)**

	返回 path 的最后一次修改时间。
21	**File::new(path[, mode="r"])** **File::open(path[, mode="r"])** **File::open(path[, mode="r"]) {|f| ...}** 打开文件。如果指定了块，则通过传递新文件作为参数来执行块。当块退出时，文件会自动关闭。这些方法有别于 Kernel.open，即使 path 是以 | 开头，后续的字符串也不会作为命令运行。
22	**File::owned?(path)** 如果 path 由有效的用户所有，则返回 true。
23	**File::pipe?(path)** 如果 path 是一个管道，则返回 true。
24	**File::readable?(path)** 如果 path 是可读的，则返回 true。
25	**File::readable_real?(path)** 如果 path 通过真正的用户权限是可读的，则返回 true。
25	**File::readlink(path)** 返回 path 所指向的文件。
26	**File::rename(old, new)** 改变文件名 old 为 new。
27	**File::setgid?(path)** 如果设置了 path 的 set-group-id 权限位，则返回 true。
28	**File::setuid?(path)** 如果设置了 path 的 set-user-id 权限位，则返回 true。
29	**File::size(path)** 返回 path 的文件大小。
30	**File::size?(path)** 返回 path 的文件大小，如果为 0 则返回 nil。
31	**File::socket?(path)** 如果 path 是一个 socket，则返回 true。
32	**File::split(path)** 返回一个数组，包含 path 的内容，path 被分成 File::dirname(path) 和 File::basename(path)。

33	**File::stat(path)** 返回 path 上带有信息的 File::Stat 对象。
34	**File::sticky?(path)** 如果设置了 path 的 sticky 位，则返回 true。
35	**File::symlink(old, new)** 创建一个指向文件 old 的符号链接。
36	**File::symlink?(path)** 如果 path 是一个符号链接，则返回 true。
37	**File::truncate(path, len)** 截断指定的文件为 len 字节。
38	**File::unlink(path...)** 删除 path 给定的文件。
39	**File::umask([mask])** 如果未指定参数，则为该进程返回当前的 umask。如果指定了一个参数，则设置了 umask，并返回旧的 umask。
40	**File::utime(atime, mtime, path...)** 改变指定文件的访问和修改时间。
41	**File::writable?(path)** 如果 path 是可写的，则返回 true。
42	**File::writable_real?(path)** 如果 path 通过真正的用户权限是可写的，则返回 true。
43	**File::zero?(path)** 如果 path 的文件大小是 0，则返回 true。

实例方法

假设 **f** 是 **File** 类的一个实例：

序号	方法 & 描述
1	**f.atime** 返回 f 的最后访问时间。
2	**f.chmode(mode)** 改变 f 的权限模式。

3	**f.chown(owner, group)** 改变 f 的所有者和所属组。
4	**f.ctime** 返回 f 的最后一个 inode 更改时间。
5	**f.flock(op)** 调用 flock(2)。op 可以是 0 或一个逻辑值或 File 类常量 LOCK_EX、LOCK_NB、LOCK_SH 和 LOCK_UN。
6	**f.lstat** 与 stat 相同，但是它返回自身符号链接上的信息，而不是所指向的文件。
7	**f.mtime** 返回 f 的最后修改时间。
8	**f.path** 返回用于创建 f 的路径名。
9	**f.reopen(path[, mode="r"])** 重新打开文件。
10	**f.truncate(len)** 截断 f 为 len 字节。

← Ruby 文件的输入与输出 Ruby Dir 类和方法 →