class File::Stat

クラス・モジュールの継承リスト: File::Stat < **Comparable** < **Object** < **Kernel** < **BasicObject** [edit]

要約

ファイルの情報を格納したオブジェクトのクラス。

FileTest に同名のモジュール関数がある場合はそれと同じ働きをします。ただ、ファイル名を引数に取るかわりに Stat 自身について判定する点が違います。

```
p File::Stat.new($0).directory? #=> false
p FileTest.directory?($0) #=> false
```

1.8 以降では、属性メソッドがシステムでサポートされていない場合 nil が返ります。なお、1.7 以前では 0 が返っていました。

```
dev         デバイス番号(ファイルシステム)
dev_major   dev の major 番号部
dev_minor   dev の minor 番号部
ino         i-node 番号
mode        ファイルモード
nlink       ハードリンクの数
uid         オーナーのユーザID
gid         オーナーのグループID
rdev        デバイスタイプ(スペシャルファイルのみ)
rdev_major  rdev の major 番号部
rdev_minor  rdev の minor 番号部
size        ファイルサイズ(バイト単位)
blksize     望ましいI/Oのブロックサイズ
blocks      割り当てられているブロック数
atime       最終アクセス時刻
mtime       最終更新時刻
ctime       最終状態変更時刻(状態の変更とは chmod などによるもので、Unix では i-node の変更を意味します)
```

目次

特異メソッド

インスタンスメソッド

継承しているメソッド

Comparableから継承しているメソッド

<
<=
==
>
>=
between?
clamp

特異メソッド

`new(path) -> File::Stat`

[permalink][rdoc][edit]

path に関する File::Stat オブジェクトを生成して返します。 **File.stat** と同じです。

[PARAM] path:
ファイルのパスを指定します。

[EXCEPTION] Errno::ENOENT:
pathに該当するファイルが存在しない場合発生します。

```
p $:[0]
#=> 例
# "C:/Program Files/ruby-1.8/lib/ruby/site_ruby/1.8"
p File::Stat.new($:[0])
#=> 例
#<File::Stat dev=0x2, ino=0, mode=040755, nlink=1, uid=0, gid=0, rdev=0x2, size=0, blksize=nil, blocks=nil, atime=Sun Sep 02 14:15:20 +0900 2
```

インスタンスメソッド

`self <=> o -> Integer | nil`

[permalink][rdoc][edit]

ファイルの最終更新時刻を比較します。self が other よりも新しければ正の数を、等しければ 0 を古ければ負の数を返します。比較できない場合は nil を返します。

[PARAM] o:
File::Stat のインスタンスを指定します。

```
require 'tempfile' # for Tempfile

fp1 = Tempfile.open("first")
fp1.print "古い方\n"
sleep(1)
fp2 = Tempfile.open("second")
fp2.print "新しい方\n"

p File::Stat.new(fp1.path) <=> File::Stat.new(fp2.path) #=> -1
p File::Stat.new(fp2.path) <=> File::Stat.new(fp1.path) #=>  1
p File::Stat.new(fp1.path) <=> fp2.path #=> nil
```

`atime -> Time`

[permalink][rdoc][edit]

最終アクセス時刻を返します。

```
fs = File::Stat.new($0)
#例
p fs.atime.to_a  #=> [45, 5, 21, 5, 9, 2007, 3, 248, false, "\223\214\213\236 (\225W\217\200\216\236) "]
```

[SEE_ALSO] **Time**

`birthtime -> Time`

[permalink][rdoc][edit]

作成された時刻を返します。

[EXCEPTION] NotImplementedError:
Windows のような birthtime のない環境で発生します。

```
File.write("testfile", "foo")
sleep 10
File.write("testfile", "bar")
sleep 10
File.chmod(0644, "testfile")
sleep 10
File.read("testfile")
File.stat("testfile").birthtime   #=> 2014-02-24 11:19:17 +0900
File.stat("testfile").mtime       #=> 2014-02-24 11:19:27 +0900
File.stat("testfile").ctime       #=> 2014-02-24 11:19:37 +0900
File.stat("testfile").atime       #=> 2014-02-24 11:19:47 +0900
```

`blksize -> Integer`

[permalink][rdoc][edit]

望ましいI/Oのブロックサイズを返します。

```
fs = File::Stat.new($0)
#例
p fs.blksize #=> nil
```

`blockdev? -> bool`

[permalink][rdoc][edit]

ブロックスペシャルファイルの時に真を返します。

```
Dir.glob("/dev/*") {|bd|
  if File::Stat.new(bd).blockdev?
    puts bd
  end
}
#例
#...
#=> /dev/hda1
#=> /dev/hda3
#...
```

`blocks -> Integer`

[permalink][rdoc][edit]

割り当てられているブロック数を返します。

```
fs = File::Stat.new($0)
#例
p fs.blocks  #=> nil
```

`chardev? -> bool`

[permalink][rdoc][edit]

キャラクタスペシャルファイルの時に真を返します。

```
Dir.glob("/dev/*") {|bd|
  if File::Stat.new(bd).chardev?
    puts bd
  end
}
#例
#...
#=> /dev/tty1
#=> /dev/stderr
#...
```

`ctime -> Time`

[permalink][rdoc][edit]

最終状態変更時刻を返します。 (状態の変更とは chmod などによるもので、Unix では i-node の変更を意味します)

```
fs = File::Stat.new($0)
#例
p fs.ctime.to_f   #=> 1188719843.0
```

[SEE_ALSO] **Time**

`dev -> String`

[permalink][rdoc][edit]

デバイス番号(ファイルシステム)を返します。

```
fs = File::Stat.new($0)
p fs.dev
#例
#=> 2
```

`dev_major -> Integer`

[permalink][rdoc][edit]

dev の major 番号部を返します。

```
fs = File::Stat.new($0)
p fs.dev_major
#例
#=> nil #この場合ではシステムでサポートされていないため
```

`dev_minor -> Integer`

[permalink][rdoc][edit]

dev の minor 番号部を返します。

```
fs = File::Stat.new($0)
p fs.dev_minor
#例
#=> nil
```

`directory? -> bool`

[permalink][rdoc][edit]

ディレクトリの時に真を返します。

```
p File::Stat.new($0).directory? #=> false
```

[SEE_ALSO] **FileTest.#directory?**

`executable? -> bool`

[permalink][rdoc][edit]

実効ユーザ/グループIDで実行できる時に真を返します。

```
p File::Stat.new($0).executable?
# 例
#=> true
```

`executable_real? -> bool`

[permalink][rdoc][edit]

実ユーザ/グループIDで実行できる時に真を返します。

```
p File::Stat.new($0).executable_real?
#例
#=> true
```

`file? -> bool`

[permalink][rdoc][edit]

通常ファイルの時に真を返します。

```
p File::Stat.new($0).file? #=> true
```

`ftype -> String`

[permalink][rdoc][edit]

ファイルのタイプを表す文字列を返します。

文字列は以下のうちのいずれかです。

```
"file"
"directory"
"characterSpecial"
"blockSpecial"
"fifo"
"link"
"socket"

"unknown"
```

例

```
fs = File::Stat.new($0)
p fs.ftype #=> "file"
p File::Stat.new($:[0]).ftype #=> "directory"
```

1.8 以降では、属性メソッドがシステムでサポートされていない場合 nil が返ります。なお、1.7 以前では 0 が返っていました。

`gid -> Integer`

[permalink][rdoc][edit]

オーナーのグループIDを返します。

```
fs = File::Stat.new($0)
#例
p fs.gid      #=> 0
```

`grpowned? -> bool`

[permalink][rdoc][edit]

グループIDが実効グループIDと等しい時に真を返します。

補助グループIDは考慮されません。

```
printf "%s %s\n", $:[0], File::Stat.new($:[0]).grpowned?
#例
#=> /usr/local/lib/site_ruby/1.8 false
printf "%s %s\n", $0, File::Stat.new($0).grpowned?
#例
#=> filestat.rb true
```

`ino -> Integer`

[permalink][rdoc][edit]

i-node 番号を返します。

```
fs = File::Stat.new($0)
#例
p fs.ino      #=> 0
```

`mode -> Integer`

[permalink][rdoc][edit]

ファイルモードを返します。

```
fs = File::Stat.new($0)
printf "%o\n", fs.mode
#例
#=> 100644
```

`mtime -> Time`

[permalink][rdoc][edit]

最終更新時刻を返します。

```
fs = File::Stat.new($0)
#例
p fs.mtime   #=> Wed Sep 05 20:42:18 +0900 2007
```

[SEE_ALSO] **Time**

`nlink -> Integer`

[permalink][rdoc][edit]

ハードリンクの数を返します。

```
fs = File::Stat.new($0)
#例
p fs.nlink    #=> 1
```

`owned? -> bool`

[permalink][rdoc][edit]

自分のものである時に真を返します。

```
printf "%s %s\n", $:[0], File::Stat.new($:[0]).owned?
#例
#=> /usr/local/lib/site_ruby/1.8 false
```

`pipe? -> bool`

[permalink][rdoc][edit]

無名パイプおよび名前つきパイプ(FIFO)の時に真を返します。

```
system("mkfifo /tmp/pipetest")
p File::Stat.new("/tmp/pipetest").pipe? #=> true
```

`rdev -> Integer`

[permalink][rdoc][edit]

デバイスタイプ(スペシャルファイルのみ)を返します。

```
fs = File::Stat.new($0)
#例
p fs.rdev     #=> 2
```

`rdev_major -> Integer`

[permalink][rdoc][edit]

rdev の major 番号部を返します。

```
fs = File::Stat.new($0)
#例
p fs.rdev_major #=> nil
```

`rdev_minor -> Integer`

[permalink][rdoc][edit]

rdev の minor 番号部を返します。

```
fs = File::Stat.new($0)
#例
p fs.rdev_minor #=> nil
```

`readable? -> bool`

[permalink][rdoc][edit]

読み込み可能な時に真を返します。

```
p File::Stat.new($0).readable? #=> true
```

`readable_real? -> bool`

[permalink][rdoc][edit]

実ユーザ/実グループによって読み込み可能な時に真を返します。

```
p File::Stat.new($0).readable_real? #=> true
```

`setgid? -> bool`

[permalink][rdoc][edit]

setgidされている時に真を返します。

```
Dir.glob("/usr/sbin/*") {|bd|
  if File::Stat.new(bd).setgid?
    puts bd
  end
}
#例
#...
#=> /usr/sbin/postqueue
#...
```

`setuid? -> bool`

[permalink][rdoc][edit]

setuidされている時に真を返します。

```
Dir.glob("/bin/*") {|bd|
  if File::Stat.new(bd).setuid?
    puts bd
  end
}
#例
#...
#=> /bin/ping
#=> /bin/su
#...
```

`size -> Integer`

[permalink][rdoc][edit]

ファイルサイズ(バイト単位)を返します。

```
fs = File::Stat.new($0)
#例
p fs.size    #=> 1548
```

`size? -> Integer | nil`

[permalink][rdoc][edit]

サイズが0の時にはnil、それ以外の場合はファイルサイズを返します。

```
require 'tempfile'

fp = Tempfile.new("temp")
p fp.size #=> 0
p File::Stat.new(fp.path).size? #=> nil
fp.print "not 0 "
fp.close
p FileTest.exist?(fp.path) #=> true
p File::Stat.new(fp.path).size? #=> 6
```

`socket? -> bool`

[permalink][rdoc][edit]

ソケットの時に真を返します。

```
Dir.glob("/tmp/*"){|file|
  if File::Stat.new(file).socket?
    printf "%s\n", file
  end
}
#例
#=> /tmp/uimhelper-hogehoge
#...
```

`sticky? -> bool`

[permalink][rdoc][edit]

stickyビットが立っている時に真を返します。

```
Dir.glob("/usr/bin/*") {|bd|
  begin
    if File::Stat.new(bd).sticky?
      puts bd
    end
  rescue
  end
}
#例
#...
#=> /usr/bin/emacs-21.4
#...
```

`symlink? -> false`

[permalink][rdoc][edit]

シンボリックリンクである時に真を返します。ただし、File::Statは自動的にシンボリックリンクをたどっていくので常にfalseを返します。

```
require 'fileutils'
outfile = $0 + ".ln"
FileUtils.ln_s($0, outfile)
p File::Stat.new(outfile).symlink? #=> false
p File.lstat(outfile).symlink?     #=> true
p FileTest.symlink?(outfile)       #=> true
```

[SEE_ALSO] **File.lstat**

`uid -> Integer`

[permalink][rdoc][edit]

オーナーのユーザIDを返します。

```
fs = File::Stat.new($0)
#例
p fs.uid    #=> 0
```

`world_readable? -> Integer | nil`

[permalink][rdoc][edit]

全てのユーザから読めるならば、そのファイルのパーミッションを表す整数を返します。そうでない場合は nil を返します。

整数の意味はプラットフォームに依存します。

```
m = File.stat("/etc/passwd").world_readable?  # => 420
sprintf("%o", m)                              # => "644"
```

`world_writable? -> Integer | nil`

[permalink][rdoc][edit]

全てのユーザから書き込めるならば、そのファイルのパーミッションを表す整数を返します。そうでない場合は nil を返します。

整数の意味はプラットフォームに依存します。

```
m = File.stat("/tmp").world_writable?         # => 511
sprintf("%o", m)                              # => "777"
```

`writable? -> bool`

[permalink][rdoc][edit]

書き込み可能な時に真を返します。

```
p File::Stat.new($0).writable? #=> true
```

`writable_real? -> bool`

[permalink][rdoc][edit]

実ユーザ/実グループによって書き込み可能な時に真を返します。

```
p File::Stat.new($0).writable_real? #=> true
```

`zero? -> bool`

[permalink][rdoc][edit]

サイズが0である時に真を返します。

```
p File::Stat.new($0).zero? #=> false
```