

17 IO类

作者 bluetea

网站 <https://github.com/bluetea>

17章中 IO类是File类的父类

18.1.1 变更文件名

```
[12] pry(main)> File.rename("line.txt", "after.txt")
=> 0
[13] pry(main)> File.rename("after.txt", "~/aaa.txt")
Errno::ENOENT: No such file or directory - (after.txt, ~/aaa.txt)
from (pry):5:in `rename'
[14] pry(main)> File.rename("after.txt", "../aaa.txt")
=> 0
[15] pry(main)> File.rename("after.txt", "../aaa.txt")
Errno::ENOENT: No such file or directory - (after.txt, ../aaa.txt)
from (pry):7:in `rename'
[16] pry(main)>
```

注意：File.rename方法无法跨越文件系统或者驱动器，也就windows无法跨越盘符
如果不存在会抛出 Errno:ENOENT类异常文件不存在异常

18.1.2 复制文件

Ruby没有预定义的函数可以赋值文件，所以要自己实现

```
1 def copy(from, to)
2   File.open(from) do |input|
3     File.open(to, "w") do |output|
4       puts "#{input.size} word have copie
5       output.write(input.read)
6     end
7   end
8 end

Line 8, Column 4      Spaces: 2
[33] pry(main)> copy("try1.txt", "try2.txt")
4 word have copied
=> 4
```

这种文件操作Ruby有个fileutils，里面有FileUtils.cp 和FileUtils.mv 文件移动等方法来操作文件

```
1 require "fileutils"
2 a = FileUtils.cp("try1.txt", "try2.txt")
3 p a
4 b = FileUtils.mv("try2.txt", "trytest.txt")
5 p b

1. bash
ruby      bash
wangmjcdeMacBook-Pro:ruby wangmjc$ ruby file.rb
nil
0
```

File.rename 不能实现文件跨文件系统，但是FileUtils.mv 可以轻易实现

18.1.3 删除文件

```
[36] pry(main)> File.delete("trytest.txt")
=> 1
[37] pry(main)> File.delete("trytest.tsxt")
Errno::ENOENT: No such file or directory - trytest.tsxt
from (pry):40:in `delete'
```

18.2 目录的操作

Dir类中实现

在window的命令符中，目录分隔符用的是\，由于使用\后使字符串很难读，而且不能直接在unix中使用，所以建议使用/

Dir.pwd

Dir.chdir(dir)

可以获取运行时所在的目录信息，即当前目录(current directory)

Dir.pwd 获取当前文件夹

Dir.chdir("/etc") change dir

```
[3] pry(main)> Dir.pwd
=> "/Users/wangmjc/Desktop/ruby"
[4] pry(main)> Dir.chdir("/etc")
=> 0
```

当前目录下的文件，可以通过指定文件名直接打开，但如果变更了当前目录，则还需要指定目录名

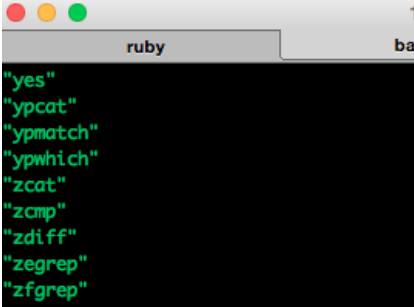
```
[1] pry(main)> Dir.pwd
=> "/Users/wangmjc/Desktop/ruby"
[2] pry(main)> io = File.open("line1.txt")
=> #<File:line1.txt>
[3] pry(main)> io.close
=> nil
[4] pry(main)> Dir.chdir("../..")
=> 0
[5] pry(main)> Dir.pwd
=> "/Users/wangmjc"
[6] pry(main)> io = File.open("/Desktop/ruby/line1.txt")
Errno::ENOENT: No such file or directory - /Desktop/ruby/line1.txt
from (pry):6:in `initialize'
[7] pry(main)> io = File.open("Desktop/ruby/line1.txt")
Errno::ENOENT: No such file or directory - Desktop/ruby/line1.txt
from (pry):7:in `initialize'
[8] pry(main)> io = File.open("Desktop/ruby/line1.txt")
=> #<File:Desktop/ruby/line1.txt>
[9] pry(main)> io.close
=> nil
```

18.2.1 目录内容的读取

Dir.open(path)

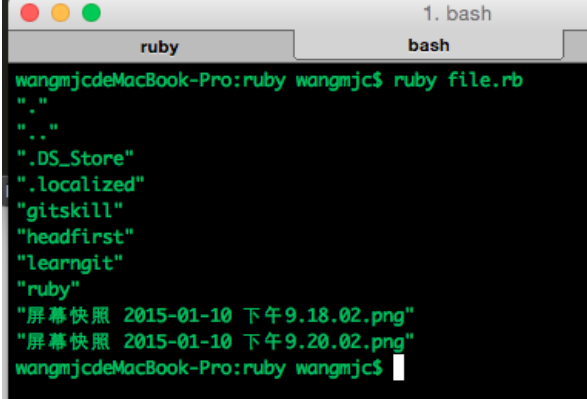
Dir.close

```
1 dir = Dir.open("/usr/bin")
2 while name = dir.read
3   p name
4 end
5 dir.close
```



或者用块的方式

```
1 Dir.open("/Users/wangmjc/Desktop") do |dir|
2   dir.each do |name|
3     p name
4   end
5 end
```



dir.read

与File类一样，Dir类也有read方法

Dir#read执行后，程序会遍历最先打开的目录下的内容，这里读取的内容可以分为以下4类：

- 1.表示当前目录的.
- 2.表示上级目录的..
- 3.其它目录名
- 4.文件名

请注意，在Dir.open里面 /usr/bin 和 /usr/bin/ 表示同一个目录

例如如下程序：输出指定目录下所有文件名，执行结果只在控制台显示

```
file.rb
1 def traverse(path)
2   if File.directory?(path) #如果是目录
3     dir = Dir.open(path) #打开目录
4     while name = dir.read
5       next if name == "."
6       next if name == ".."
7       traverse(path + "/" + name) #递归查询，再进入下一级目录
8     end
9     dir.close
10  else
11    process_file(path)
12  end
13 end
14
15 def process_file(path)
16   p path
17 end
18
19 traverse(ARGV[0])
```

1. bash

wangmjcdeMacBook-Pro:ruby wangmjc\$ ruby file.rb /Users/wangmjc/Desktop

Dir.glob 可以想shell那样用*或者?等通配符(wildcard character)来取得文件名，Dir.glob方法会将匹配到的文件名(目录名)以数组的形式返回

```
[5] pry(main)> Dir.glob("file.*")
=> ["file.rb"]
[6] pry(main)> Dir.glob("*.rb")
=> ["error.rb",
"extend_sample.rb",
"file.rb",
"io.rb",
"iterator.rb",
"lambda.rb",
"random.rb",
"reg.rb",
"string.rb",
"test.rb",
"wc.rb",
"yunsuan.rb"]
```

取得子目录下扩展名为.html或者.htm的文件名

```
[7] pry(main)> Dir.glob(["*/*.html", "*/*.htm"])
=> []
```

取得文件名为foo.c, foo.h, foo.o的文件

```
[10] pry(main)> Dir.glob("foo.[cho]")
=> ["foo.c", "foo.h", "foo.o"]
```

获得当前目录及其子目录中所有的文件名，递归查找目录

```
[12] pry(main)> Dir.glob("**/*")
```

获取目录foo及其子目录中所有扩展名为.html的文件名，递归查找目录

任意文件夹

```
[11] pry(main)> Dir.glob("/Users/wangmjc/Desktop/**/*")
=> ["/Users/wangmjc/Desktop/gitkill",
"/Users/wangmjc/Desktop/headfirst",
"/Users/wangmjc/Desktop/learnig",
"/Users/wangmjc/Desktop/ruby"]
```

任意文件夹里面的任意文件和目录

```
[10] pry(main)> Dir.glob("/Users/wangmjc/Desktop/**/*")
=> ["/Users/wangmjc/Desktop/gitkill",
"/Users/wangmjc/Desktop/gitkill/aa.aa",
"/Users/wangmjc/Desktop/gitkill/README.md",
```

任意文件夹内的隐藏文件

```
[12] pry(main)> Dir.glob("/Users/wangmjc/Desktop/**/*.")
=> ["/Users/wangmjc/Desktop/.DS_Store",
"/Users/wangmjc/Desktop/.localized",
"/Users/wangmjc/Desktop/gitkill/.git",
"/Users/wangmjc/Desktop/learnig/.git",
"/Users/wangmjc/Desktop/ruby/.git"]
```

任意文件夹内以f打头的内容

所以改造上面的递归内容，把所有文件内容都输出出来

```
file.rb
1 def traverse(path)
2   Dir.glob("#{path}/**/*", "#{path}/**/*.").each do |name|
3     "#{path}/**/*" 表示任意文件夹内的任意内容
4     "#{path}/**/*.*" 表示任意文件夹内的隐藏文件
5     unless File.directory?(name)
6       process_file(name)
7     end
8   end
9 end
10
11 def process_file(name)
12   p name
13 end
14
15 traverse(ARGV[0])
16
```

18.2.2 目录的创建与删除

```
[15] pry(main)> Dir.mkdir("temp")
=> 0
[16] pry(main)> Dir.rmdir("temp")
=> 0
```

18.3 文件与目录的属性

文件与目录的所有者，最后更新时间，等，
File.stat(path)返回的是File::Stat类的实例，实例方法如下：

dev 文件系统编号

ino i-node编号

mode 文件的属性

nlink 链接数

uid 文件所有者的用户uid

gid 文件所属组的组ID

rdev 文件系统的块设备种类

size 文件大小

blksize 文件系统的块大小

blocks 文件占用的块数量

atime 文件的最后访问时间

mtime 文件的最后修改时间

ctime 文件状态的最后更改时间

```
[18] pry(main)> File.stat("ruby")
=> #<File::Stat
  dev=0x1000004,
  ino=1154184,
  mode=040755 (directory rwxr-xr-x),
  nlink=30,
  uid=501 (wangmjc),
  gid=20 (staff),
  rdev=0x0 (0, 0),
  size=1020,
  blksize=4096,
  blocks=0,
  atime=2015-01-10 22:06:52 +0800 (1420898812),
  mtime=2015-01-10 22:07:56 +0800 (1420898876),
  ctime=2015-01-10 22:07:56 +0800 (1420898876)>
[22] pry(main)> File.stat("file.rb")
=> #<File::Stat
  dev=0x1000004,
  ino=1573014,
  mode=0100644 (file rw-r--r--),
  nlink=1,
  uid=501 (wangmjc),
  gid=20 (staff),
  rdev=0x0 (0, 0),
  size=333,
  blksize=4096,
  blocks=8,
  atime=2015-01-10 22:06:36 +0800 (1420898796),
  mtime=2015-01-10 22:07:56 +0800 (1420898876),
  ctime=2015-01-10 16:21:40 +0800 (1420878100)>
```

通过UID和GID获得相对应的用户ID和组ID时，需要Etc模块 通过require etc来加载

```

1 require "etc"
2 st = File.stat("/Users/wangmjc/Desktop")
3 pw = Etc.getpwuid("st.uid") #用Etc.getpwuid功能根据uid读取信息
4 p pw.name
5 gr = Etc.getgrgid("st.gid") #用Etc.getpwuid功能根据gid读取信息
6 p gr.name

```

```

[2] pry(main)> st = File.stat("/Users/wangmjc/Desktop")
=> #<File::Stat
  dev=0x10000004,
  ino=401031,
  mode=040700 (directory rwx-----),
  nlink=8,
  uid=501 (wangmjc),
  gid=20 (staff),
  rdev=0x0 (0, 0),
  size=272,
  blksize=4096,
  blocks=0,
  atime=2015-01-10 22:06:52 +0800 (1420898812),
  mtime=2015-01-10 22:29:49 +0800 (1420900189),
  ctime=2015-01-10 22:29:49 +0800 (1420900189)>
[3] pry(main)> pw = Etc.getpwuid(st.uid)
=> #<struct Struct::Passwd
  name="wangmjc",
  passwd="*****",
  uid=501,
  gid=20,
  gecos="wangmjc",
  dir="/Users/wangmjc",
  shell="/bin/bash",
  change=0,
  uclass="",
  expire=0>
[4] pry(main)> pw.name
=> "wangmjc"
[6] pry(main)> pw = Etc.getgrgid(st.gid)
=> #<struct Struct::Group name="staff", passwd="", gid=20, mem=["root"]>

```

File.ctime(path)

File.mtime(path)

File.atime(path)

这三个方法的执行结果与实例方法 File::Stat#ctime, File::Stat#mtime, File::Stat#atime功能类似

```

[9] pry(main)> File.stat("/Users/wangmjc/Desktop").ctime
=> 2015-01-10 22:44:47 +0800
[10] pry(main)> File.stat("/Users/wangmjc/Desktop").atime
=> 2015-01-10 22:06:52 +0800
[11] pry(main)> File.stat("/Users/wangmjc/Desktop").mtime
=> 2015-01-10 22:44:47 +0800
[12] pry(main)> File.ctime("/Users/wangmjc/Desktop")
=> 2015-01-10 22:44:47 +0800
[13] pry(main)> File.atime("/Users/wangmjc/Desktop")
=> 2015-01-10 22:06:52 +0800
[14] pry(main)> File.mtime("/Users/wangmjc/Desktop")
=> 2015-01-10 22:44:47 +0800

```

File.chmod(mode, path)

```

[15] pry(main)> File.chmod(755, "line1.txt")
=> 1
[16] pry(main)> File.chmod(0755, "line1.txt")
=> 1
wangmjcdeMacBook-Pro:ruby wangmjc$ ls -l |grep line1
-rwxr-xr-x 1 wangmjc staff 67 1 11 18:26 line1.txt

```

当想要为现有的文件追加权限的时候，可以用按位或或者按位与的方法来追加权限，

```
file.rb
1 rfile = "line1.txt"
2 f = File.stat(rfile)
3 File.chmod(f.mode | 0111, rfile)

1. bash
ruby bash
wangmjcdMacBook-Pro:ruby wangmjc$ ls -l |grep line1
-r--r--r-- 1 wangmjc staff 67 1 11 18:26 line1.txt
wangmjcdMacBook-Pro:ruby wangmjc$ ruby file.rb
wangmjcdMacBook-Pro:ruby wangmjc$ ls -l |grep line1
-r-xr-xr-x 1 wangmjc staff 67 1 11 18:26 line1.txt
wangmjcdMacBook-Pro:ruby wangmjc$
```

File.chown(owner, group, path)

改变文件path的所有者，owner表示新的所有者的用户ID，group表示新的所属组ID，同时可以指定多个路径，执行这个命令需要管理员权限

18.4文件名的操作

1.File.basename(path, [, suffix])返回路径path中最后一个/以后的部分，如果指定了扩展名，则会去除返回值中扩展名的部分，不会坚持是否有这个文件

```
wangmjcdMacBook-Pro:ruby wangmjc$ pry
[1] pry(main)> File.basename("/Users/wangmjc/Desktop/ruby")
=> "ruby"
[2] pry(main)> File.basename("/Users/wangmjc/Desktop/ruby/line1.txt")
=> "line1.txt"
[3] pry(main)> File.basename("/Users/wangmjc/Desktop/ruby/line1.txt", ".txt")
=> "line1"
[4] pry(main)> File.basename("/Users/wangmjc/Desktop/ruby/line122.txt", ".txt")
=> "line122"
[5] pry(main)> File.basename("abc.abc", ".txt")
=> "abc.abc"
[6] pry(main)> File.basename("abc.abc", ".abc")
=> "abc"
```

2.File.dirname(path)返回最后一个/之前的内容，路径不包含/的时候返回，表示当前目录

```
[7] pry(main)> File.dirname("/Users/wangmjc/Desktop/ruby/line1.txt")
=> "/Users/wangmjc/Desktop/ruby"
[8] pry(main)> File.dirname("/Users/wangmjc/Desktop/ruby")
=> "/Users/wangmjc/Desktop"
[9] pry(main)> File.dirname("ruby")
=> "."
[10] pry(main)>
```

3.File.split(path)将path目录分割为目录名与文件名两部分，以数组的形式返回

```
[10] pry(main)> File.split("/Users/wangmjc/Desktop/ruby/line1.txt")
=> ["/Users/wangmjc/Desktop/ruby", "line1.txt"]
[11] pry(main)> File.split("/Users/wangmjc/Desktop/ruby")
=> ["/Users/wangmjc/Desktop", "ruby"]
[12] pry(main)> File.split("/ruby")
=> ["/", "ruby"]
[13] pry(main)> File.split("ruby")
=> [".", "ruby"]
[14] pry(main)> a, b = File.split("/Users/wangmjc/Desktop/ruby/line1.txt")
=> ["/Users/wangmjc/Desktop/ruby", "line1.txt"]
[15] pry(main)> a
=> "/Users/wangmjc/Desktop/ruby"
[16] pry(main)> b
=> "line1.txt"
[17] pry(main)>
```

4.File.join(name1, name2)用File::SEPARATOR连接指定参数指定的字符串。File::SEPARATOR的默认值为 "/"

```
[17] pry(main)> File.join("/usr/bin", "ruby", "abc")
=> "/usr/bin/ruby/abc"
[18] pry(main)> File.join(".", "ruby")
=> "./ruby"
```

5.File.expand_path(path[, default_dir])将相对路径path转换为绝对路径，不指定default_dir的时候，则根据当前目录转换

```
[19] pry(main)> File.expand_path("ruby")
=> "/Users/wangmjc/Desktop/ruby/ruby"
[20] pry(main)> File.expand_path("file.rb")
=> "/Users/wangmjc/Desktop/ruby/file.rb"
[21] pry(main)> File.expand_path("file.rb", "usr")
=> "/Users/wangmjc/Desktop/ruby/usr/file.rb"
[22] pry(main)> File.expand_path("../bin")
=> "/Users/wangmjc/Desktop/bin"
[23] pry(main)> File.expand_path("../../bin")
=> "/Users/wangmjc/bin"
[24] pry(main)> File.expand_path("../etc", "usr")
=> "/Users/wangmjc/Desktop/ruby/etc"
[25] pry(main)>
```

18.5与文件相关的库

18.5.1 find库

find库中的find模块用来对指定的目录做递归处理

Find.find(dir){|path| ...} 这个方法将目录dir下所有文件路径逐个穿个路径path

当使用File.find的时候，调用Find.prune方法后，程序会跳过当前查找目录下的所有路径（包含子目录）如果只使用next的时候，则只会跳过当前目录，子目录还会继续查找

```
file.rb
1 require "find"
2
3 IGNORES = [/^\.\/, /^ruby$/ ]
4
5 def list_dir(path)
6   Find.find(path) do |name| #把path内的所有元素分别传给name
7     if FileTest.directory?(name) #判断name是否为文件夹
8       dir, file_name = File.split(name)
9       IGNORES.each do |re|
10        if (re =~ dir) #如果file_name匹配到IGNORES正则，则跳过此文件
11          Find.prune #跳过此文件
12        end
13      end
14      puts name
15    end
16  end
17 end
18 list_dir(ARGV[0])
```

```
1. bash
wangmjcdeMacBook-Pro:ruby wangmjc$ clear
wangmjcdeMacBook-Pro:ruby wangmjc$ ruby file.rb /Users/wangmjc/Desktop
/Users/wangmjc/Desktop
/Users/wangmjc/Desktop/gitkill
/Users/wangmjc/Desktop/gitkill/.git
/Users/wangmjc/Desktop/gitkill/.git/branches
/Users/wangmjc/Desktop/gitkill/.git/hooks
```

18.5.2 tempfile库

tempfile库用于管理临时文件

Tempfile.new(basename[, tempdir])

创建临时文件的时候，实际生成的文件名为 basename + 进程id + 流水号，因此即使用同样地basename，每次用new方法生成的临时文件也都不一样的，如果不知名目录名tempdir，则会按照顺序查找ENV["TMPDIR"], ENV["TMP"], ENV["TEMP"], /tmp，并把最先找到的目录作为临时目录使用

```
[9] pry(main)> tf = Tempfile.new("wang")
=> #<File:/var/folders/z7/qappx2x56ndb9n4sr20nsjg40000gn/T/wang20150111-2657-ptk95i>
[10] pry(main)> tf.path
=> "/var/folders/z7/qappx2x56ndb9n4sr20nsjg40000gn/T/wang20150111-2657-ptk95i"
[11] pry(main)> tf.write("hello new world")
=> 15
[12] pry(main)> tf.read
=> ""
[13] pry(main)> tf.rewind
=> 0
[14] pry(main)> tf.read
=> "hello new world"
[15] pry(main)> tf.close
=> nil
[16] pry(main)> tf.unlink
=> #<File:/var/folders/z7/qappx2x56ndb9n4sr20nsjg40000gn/T/wang20150111-2657-ptk95i (closed)>
```

18.5.3 fileutils库

FileUtils.cp(from, to)

FileUtils.cp_r(from, to) 和上面类似，不同点在于from为目录时，会进行递归拷贝

FileUtils.mv(from,to)

FileUtils.rm(path)

FileUtils.rm_f(paht)

删除的path只能为文件，也可以将path作为数组来一次性删除多个文件，方法在执行删除过程中，若发生异常则中断处理，而

FileUtils.rm_f方法则会忽略错误，继续执行

FileUtils.rm_r(path)

FileUtils.rm_rf(paht)

path为目录时，则进行递归删除，此外也可以将path设置为一个数组来一次性删除多个文件，

FileUtils.compare(from, to)

比较form 和to，相同返回true，否则发挥false

FileUtils.install(from, to[, option])

把文件从from拷贝到to，如果to已经存在，且与from一致，则不会拷贝，option参数用于指定文件的访问权限

FileUtils.install(from, to, :mode => 0755)

FileUtils.mkdir_p(path)

和Dir.mkdir的区别

```
[28] pry(main)> require "fileutils"
=> false
[29] pry(main)> Dir.mkdir("foo/bar/baz")
Errno::ENOENT: No such file or directory - foo/bar/baz
from (pry):23:in `mkdir'
[30] pry(main)> Dir.mkdir("foo")
=> 0
[31] pry(main)> Dir.mkdir("foo/bar")
=> 0
[32] pry(main)> Dir.mkdir("foo/bar/baz")
=> 0
[33] pry(main)> FileUtils.mkdir_p("foo1/bar1/baz1")
=> ["foo1/bar1/baz1"]
```