Class **Dir** < Object

Objects of class Dir are directory streams representing directories in the underlying file system. They provide a variety of ways to list directories and their contents. See also File, page 506.

The directory used in these examples contains the two regular files (config.h and main.rb), the parent directory (..), and the directory itself (.).

Mixes in

Enumerable:

all?, any?, collect, count, cycle, detect, drop, drop_while, each_cons, each_slice, each_with_index, entries, find, find_all, find_index, first, grep, group_by, include?, inject, map, max, max_by, member?, min, min_by, minmax, minmax_by, none?, one?, partition, reduce, reject, select, sort, sort_by, take, take_while, to_a, zip

Class methods

[ ] Dir[*glob_pattern*] → *array*

Equivalent to calling Dir.glob(*glob_pattern*, 0).

chdir Dir.chdir(⟨ *dir* ⟩) → 0
Dir.chdir(⟨ *dir* ⟩) {| path | *block* } → *obj*

Changes the current working directory of the process to the given string. When called without an argument, changes the directory to the value of the environment variable HOME or LOGDIR. Raises a SystemCallError (probably Errno::ENOENT) if the target directory does not exist.

If a block is given, it is passed the name of the new current directory, and the block is executed with that as the current directory. The original working directory is restored when the block exits. The return value of chdir is the value of the block. chdir blocks can be nested, but in a multithreaded program an error will be raised if a thread attempts to open a chdir block while another thread has one open. This is because the underlying operating system only understands the concept of a single current working directory at any one time.

```
Dir.chdir("/var/log")
puts Dir.pwd
Dir.chdir("/tmp") do
  puts Dir.pwd
  Dir.chdir("/usr") do
    puts Dir.pwd
  end
  puts Dir.pwd
end
puts Dir.pwd
```

produces:

```
/var/log
/tmp
/usr
/tmp
/var/log
```

chroot

Dir.chroot(*dirname*) → 0

Changes this process's idea of the file system root. Only a privileged process may make this call. Not available on all platforms. On Unix systems, see chroot(2) for more information.

```
Dir.chdir("/production/secure/root")
Dir.chroot("/production/secure/root")     → 0
Dir.pwd                                   → "/"
```

delete

Dir.delete(*dirname*) → 0

Deletes the named directory. Raises a subclass of SystemCallError if the directory isn't empty.

entries

Dir.entries(*dirname*) → *array*

Returns an array containing all of the filenames in the given directory. Will raise a SystemCallError if the named directory doesn't exist.

```
Dir.entries("testdir")   # =>   [".", "..", "config.h", "main.rb"]
```

exist?

Dir.exist?(*path*) → true or false

1.9

Returns true if *path* exists and is a directory. Alias for File.directory?.

```
Dir.exist?("/tmp")    # =>   true
Dir.exist?("/temp")   # =>   false
```

exists?

Dir.exists?(*path*) → true or false

Alias for Dir.exist?.

foreach

Dir.foreach(*dirname*) {| *filename* | *block* } → nil

Calls the block once for each entry in the named directory, passing the filename of each entry as a parameter to the block.

```
Dir.foreach("testdir") {|x| puts "Got #{x}" }
```

produces:

```
Got .
Got ..
Got config.h
Got main.rb
```

getwd

Dir.getwd → *dirname*

Returns a string containing the canonical path to the current working directory of this process. Note that on some operating systems this name may not be the name you gave to Dir.chdir. On OS X, for example, `/tmp` is a symlink.

```
Dir.chdir("/tmp")   # =>   0
Dir.getwd           # =>   "/private/tmp"
```

glob

Dir.glob(*glob_pattern*, ⟨ *flags* ⟩) → *array*
Dir.glob(*glob_pattern*, ⟨ *flags* ⟩) {| *filename* | *block* } → false

Returns the filenames found by expanding the pattern given in *glob_pattern*, either as elements in *array* or as parameters to the block. Note that this pattern is not a regexp (it's closer to a shell glob). See File.fnmatch on page 509 for the meaning of the *flags* parameter. Case sensitivity depends on your system (so File::FNM_CASEFOLD is ignored). Metacharacters in the pattern are as follows:

*	Any sequence of characters in a filename: * will match all files, c* will match all files beginning with *c*, *c will match all files ending with *c*, and *c* will match all files that have *c* in their name.
**	Matches zero or more directories (so **/fred) matches a file named *fred* in or below the current directory).
?	Matches any one character in a filename.
[*chars*]	Matches any one of *chars*. If the first character in *chars* is ^, matches any character not in the remaining set.
{*patt*,...}	Matches one of the patterns specified between braces. These patterns may contain other metacharacters.
\	Removes any special significance in the next character.

```
Dir.chdir("testdir")                # =>   0
Dir["config.?"]                     # =>   ["config.h"]
Dir.glob("config.?")                # =>   ["config.h"]
Dir.glob("*.[a-z][a-z]")            # =>   ["main.rb"]
Dir.glob("*.[^r]*")                 # =>   ["config.h"]
Dir.glob("*.{rb,h}")                # =>   ["main.rb", "config.h"]
Dir.glob("*")                       # =>   ["config.h", "main.rb"]
Dir.glob("*", File::FNM_DOTMATCH)   # =>   [".", "..", "config.h",
                                           "main.rb"]

Dir.chdir("..")                     # =>   0
Dir.glob("code/**/fib*.rb")         # =>   ["code/fib_up_to.rb",
                                           "code/fiber.rb",
                                           "code/rdoc/fib_example.rb"]
Dir.glob("**/rdoc/fib*.rb")         # =>   ["code/rdoc/fib_example.rb"]
```

mkdir

Dir.mkdir(*dirname* ⟨ , *permissions* ⟩) → 0

Makes a new directory named *dirname*, with permissions specified by the optional parameter *permissions*. The permissions may be modified by the value of File.umask and are

ignored on Windows. Raises a SystemCallError if the directory cannot be created. See also the discussion of permissions on page 506.

new Dir.new(*dirname* ⟨ , :encoding => *enc* ⟩) → *dir*

1.9

Returns a new directory object for the named directory. The optional hash parameter lets you specify the encoding used by filenames. If not given, it defaults to the file system local on the current machine.

open Dir.open(*dirname* ⟨ , :encoding => *enc* ⟩) → *dir*
Dir.open(*dirname* ⟨ , :encoding => *enc* ⟩) {| *dir* | *block* } → *obj*

With no block, open is a synonym for Dir.new. If a block is present, it is passed *dir* as a parameter. The directory is closed at the end of the block, and Dir.open returns the value of the block.

pwd Dir.pwd → *dirname*

Synonym for Dir.getwd.

rmdir Dir.rmdir(*dirname*) → 0

Synonym for Dir.delete.

unlink Dir.unlink(*dirname*) → 0

Synonym for Dir.delete.

Instance methods

close *dir*.close → nil

Closes the directory stream. Any further attempts to access *dir* will raise an IOError.

```
d = Dir.new("testdir")
d.close   # =>   nil
```

each *dir*.each {| *filename* | *block* } → *dir*

Calls the block once for each entry in this directory, passing the filename of each entry as a parameter to the block.

```
d = Dir.new("testdir")
d.each  {|name| puts "Got #{name}" }
```

produces:

```
Got .
Got ..
Got config.h
Got main.rb
```

path *dir*.path → dirname

Returns the path parameter passed to *dir*'s constructor.

```
d = Dir.new("..")
d.path   # =>   ".."
```

pos

dir.pos → *int*

Synonym for Dir#tell.

pos=

dir.pos(*int*) → *int*

Synonym for Dir#seek but returns the position parameter.

```
d = Dir.new("testdir")   # =>   #<Dir:testdir>
d.read                   # =>   "."
i = d.pos                # =>   1
d.read                   # =>   ".."
d.pos = i                # =>   1
d.read                   # =>   ".."
```

read

dir.read → *filename* or nil

Reads the next entry from *dir* and returns it as a string. Returns nil at the end of the stream.

```
d = Dir.new("testdir")
d.read   # =>   "."
d.read   # =>   ".."
d.read   # =>   "config.h"
```

rewind

dir.rewind → *dir*

Repositions *dir* to the first entry.

```
d = Dir.new("testdir")
d.read     # =>   "."
d.rewind   # =>   #<Dir:testdir>
d.read     # =>   "."
```

seek

dir.seek(*int*) → *dir*

Seeks to a particular location in *dir*. *int* must be a value returned by Dir#tell (it is not necessarily a simple index into the entries).

```
d = Dir.new("testdir")   # =>   #<Dir:testdir>
d.read                   # =>   "."
i = d.tell               # =>   1
d.read                   # =>   ".."
d.seek(i)                # =>   #<Dir:testdir>
d.read                   # =>   ".."
```

tell

dir.tell → *int*

Returns the current position in *dir*. See also Dir#seek.

```
d = Dir.new("testdir")
d.tell   # =>   0
d.read   # =>   "."
d.tell   # =>   1
```