Directories

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

There is no installation needed to use these functions; they are part of the PHP core.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

DIRECTORY_SEPARATOR (string)

PATH_SEPARATOR (string)

Note
The **PATH_SEPARATOR** was introduced with PHP 4.3.0-RC2.

Directory Functions

See Also

For related functions such as dirname(), is_dir(), mkdir(), and rmdir(), see the Filesystem section.

chdir

chdir -- Change directory

Description

bool **chdir** (string `$directory`)

Changes PHP's current directory to *`directory`*.

Parameters

`directory`
The new current directory

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #1 - chdir() example

```
<?php

// current directory
echo getcwd() . "\n";

chdir('public_html');

// current directory
echo getcwd() . "\n";

?>
```

The above example will output something similar to:

```
/home/vincent
/home/vincent/public_html
```

Notes

Note
When safe mode is enabled, PHP checks whether the directory in which the script is operating has the same UID (owner) as the script that is being executed.

See Also

- getcwd()

chroot

chroot -- Change the root directory

Description

bool **chroot** (string `$directory`)

Changes the root directory of the current process to *`directory`*.

This function is only available if your system supports it and you're using the CLI, CGI or Embed SAPI. Also, this function requires root privileges.

Parameters

`directory`
The new directory

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
This function is not implemented on Windows platforms.

dir

dir -- Return an instance of the Directory class

Description

Directory

Directory {

string *path*;

resource *handle*;

string **Directory::read** (void)

void **Directory::rewind** (void)

void **Directory::close** (void)
}

A pseudo-object oriented mechanism for reading a directory. The given `directory` is opened. Two properties are available once the directory has been opened. The handle property can be used with other directory functions such as readdir(), rewinddir() and closedir(). The path property is set to path the directory that was opened. Three methods are available: read, rewind and close.

Examples

Example #2 - dir() example

Please note the fashion in which **dir::read()** 's return value is checked in the example below. We are explicitly testing whether the return value is identical to (equal to and of the same type as - see Comparison Operators for more information) **FALSE** since otherwise, any directory entry whose name evaluates to **FALSE** will stop the loop.

```
<?php
$d = dir("/etc/php5");
echo "Handle: " . $d->handle . "\n";
echo "Path: " . $d->path . "\n";
while (false !== ($entry = $d->read())) {
  echo $entry."\n";
}
$d->close();
?>
```

The above example will output something similar to:

```
Handle: Resource id #2
Path: /etc/php5
.
..
apache
cgi
cli
```

Notes

Note
The order in which directory entries are returned by the read method is system-dependent.

closedir

closedir -- Close directory handle

Description

void **closedir** (resource `$dir_handle`)

Closes the directory stream indicated by *`dir_handle`*. The stream must have previously been opened by opendir().

Parameters

`dir_handle`
The directory handle resource previously opened with opendir().

Examples

Example #3 - closedir() example

```
<?php
$dir = "/etc/php5/";

// Open a known directory, read directory into variable and then close
if (is_dir($dir)) {
   if ($dh = opendir($dir)) {
       $directory = readdir($dh);
       closedir($dh);
   }
}
?>
```

getcwd

getcwd -- Gets the current working directory

Description

string **getcwd** (void)

Gets the current working directory.

Return Values

Returns the current working directory on success, or **FALSE** on failure.

On some Unix variants, getcwd() will return **FALSE** if any one of the parent directories does not have the readable or search mode set, even if the current directory does. See chmod() for more information on modes and permissions.

Examples

Example #4 - getcwd() example

```
<?php

// current directory
echo getcwd() . "\n";

chdir('cvs');

// current directory
echo getcwd() . "\n";

?>
```

The above example will output something similar to:

```
/home/didou
/home/didou/cvs
```

See Also

- chdir()
- chmod()

opendir

opendir -- Open directory handle

Description

resource **opendir** (string `$path` [, resource `$context`])

Opens up a directory handle to be used in subsequent closedir(), readdir(), and rewinddir() calls.

Parameters

`path`
The directory path that is to be opened

`context`
For a description of the *`context`* parameter, refer to the streams section of the manual.

Return Values

Returns a directory handle resource on success, or **FALSE** on failure.

If *`path`* is not a valid directory or the directory can not be opened due to permission restrictions or filesystem errors, opendir() returns **FALSE** and generates a PHP error of level E_WARNING. You can suppress the error output of opendir() by prepending ' @ ' to the front of the function name.

ChangeLog

Version	Description
5.0.0	*`path`* supports the *ftp://* URL wrapper.
4.3.0	*`path`* can also be any URL which supports directory listing, however only the *file://* URL wrapper supports this in PHP 4

Examples

Example #5 - opendir() example

```
<?php
$dir = "/etc/php5/";

// Open a known directory, and proceed to read its contents
if (is_dir($dir)) {
   if ($dh = opendir($dir)) {
       while (($file = readdir($dh)) !== false) {
           echo "filename: $file : filetype: " . filetype($dir . $file) .
"\n";
       }
       closedir($dh);
   }
}
?>
```

The above example will output something similar to:

```
filename: . : filetype: dir
filename: .. : filetype: dir
filename: apache : filetype: dir
filename: cgi : filetype: dir
filename: cli : filetype: dir
```

See Also

- is_dir()
- readdir()
- Dir

readdir

readdir -- Read entry from directory handle

Description

string **readdir** (resource `$dir_handle`)

Returns the filename of the next file from the directory. The filenames are returned in the order in which they are stored by the filesystem.

Parameters

`dir_handle`
The directory handle resource previously opened with opendir().

Return Values

Returns the filename on success, or **FALSE** on failure.

Warning
This function may return Boolean **FALSE**, but may also return a non-Boolean value which evaluates to **FALSE**, such as *0* or "". Please read the section on Booleans for more information. Use the === operator for testing the return value of this function.

Examples

Example #6 - List all files in a directory

Please note the fashion in which readdir() 's return value is checked in the examples below. We are explicitly testing whether the return value is identical to (equal to and of the same type as--see Comparison Operators for more information) **FALSE** since otherwise, any directory entry whose name evaluates to **FALSE** will stop the loop (e.g. a directory named "0").
 	
`<?php`
`// Note that !== did not exist until 4.0.0-RC2`
 	
`if ($handle = opendir('/path/to/files')) {`
`  echo "Directory handle: $handle\n";`
`  echo "Files:\n";`
 	
`  /* This is the correct way to loop over the directory. */`

```
    while (false !== ($file = readdir($handle))) {
        echo "$file\n";
    }

    /* This is the WRONG way to loop over the directory. */
    while ($file = readdir($handle)) {
        echo "$file\n";
    }

    closedir($handle);
}
?>
```

Example #7 - List all files in the current directory and strip out. and..

```
<?php
if ($handle = opendir('.')) {
    while (false !== ($file = readdir($handle))) {
        if ($file != "." && $file != "..") {
            echo "$file\n";
        }
    }
    closedir($handle);
}
?>
```

See Also

- is_dir()
- glob()

rewinddir

rewinddir -- Rewind directory handle

Description

void **rewinddir** (resource `$dir_handle`)

Resets the directory stream indicated by *`dir_handle`* to the beginning of the directory.

Parameters

`dir_handle`
The directory handle resource previously opened with opendir().

scandir

scandir -- List files and directories inside the specified path

Description

array **scandir** (string `$directory` [, int `$sorting_order` [, resource `$context`]])

Returns an array of files and directories from the *`directory`*.

Parameters

`directory`
The directory that will be scanned.

`sorting_order`
By default, the sorted order is alphabetical in ascending order. If the optional *`sorting_order`* is used (set to 1), then the sort order is alphabetical in descending order.

`context`
For a description of the *`context`* parameter, refer to the streams section of the manual.

Return Values

Returns an array of filenames on success, or **FALSE** on failure. If *`directory`* is not a directory, then boolean **FALSE** is returned, and an error of level **E_WARNING** is generated.

Examples

Example #8 - A simple scandir() example

```
<?php
$dir    = '/tmp';
$files1 = scandir($dir);
$files2 = scandir($dir, 1);

print_r($files1);
print_r($files2);
?>
```

The above example will output something similar to:

```
Array
(
```

```
   [0] => .
   [1] => ..
   [2] => bar.php
   [3] => foo.txt
   [4] => somedir
)
Array
(
   [0] => somedir
   [1] => foo.txt
   [2] => bar.php
   [3] => ..
   [4] => .
)
```

Example #9 - PHP 4 alternatives to scandir()

```
<?php
$dir = "/tmp";
$dh  = opendir($dir);
while (false !== ($filename = readdir($dh))) {
   $files[] = $filename;
}

sort($files);

print_r($files);

rsort($files);

print_r($files);

?>
```

The above example will output something similar to:

```
Array
(
   [0] => .
   [1] => ..
   [2] => bar.php
   [3] => foo.txt
   [4] => somedir
)
Array
(
   [0] => somedir
   [1] => foo.txt
   [2] => bar.php
   [3] => ..
   [4] => .
)
```

Notes

Tip
A URL can be used as a filename with this function if the fopen wrappers have been enabled. See fopen() for more details on how to specify the filename and List of Supported Protocols/Wrappers for a list of supported URL protocols.

See Also

- opendir()
- readdir()
- glob()
- is_dir()
- sort()