PHP Filesystem Functions

PHP Filesystem Introduction

The filesystem functions allow you to access and manipulate the filesystem.

Installation

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

Runtime Configuration

The behavior of the filesystem functions is affected by settings in php.ini.

Filesystem configuration options:

Name	Default	Description	Changeable
allow_url_fopen	"1"	Allows fopen()-type functions to work with URLs (available since PHP 4.0.4)	PHP_INI_SYSTEM
user_agent	NULL	Defines the user agent for PHP to send (available since PHP 4.3)	PHP_INI_ALL
default_socket_timeout	"60"	Sets the default timeout, in seconds, for socket based streams (available since PHP 4.3)	PHP_INI_ALL
from	""	Defines the anonymous FTP password (your email address)	PHP_INI_ALL
auto_detect_line_endings	"0"	When set to "1", PHP will examine the data read by fgets() and file() to see if it is using Unix, MS-Dos or Mac line-ending characters (available since PHP 4.3)	PHP_INI_ALL

Unix / Windows Compatibility

When specifying a path on Unix platforms, the forward slash (/) is used as directory separator. However, on Windows platforms, both forward slash (/) and backslash (\) can be used.

PHP Filesystem Functions

PHP: indicates the earliest version of PHP that supports the function.

Function	Description	PHP
basename()	Returns the filename component of a path	3
chgrp()	Changes the file group	3
chmod()	Changes the file mode	3
chown()	Changes the file owner	3
clearstatcache()	Clears the file status cache	3
copy()	Copies a file	3
delete()	See unlink() or unset()	
dirname()	Returns the directory name component of a path	3
disk_free_space()	Returns the free space of a directory	4
disk_total_space()	Returns the total size of a directory	4
diskfreespace()	Alias of disk_free_space()	3
fclose()	Closes an open file	3
feof()	Tests for end-of-file on an open file	3
fflush()	Flushes buffered output to an open file	4
fgetc()	Returns a character from an open file	3
fgetcsv()	Parses a line from an open file, checking for CSV fields	3
fgets()	Returns a line from an open file	3
fgetss()	Returns a line, with HTML and PHP tags removed, from an open file	3
file()	Reads a file into an array	3
file_exists()	Checks whether or not a file or directory exists	3
file_get_contents()	Reads a file into a string	4
file_put_contents	Writes a string to a file	5
fileatime()	Returns the last access time of a file	3

filectime()	Returns the last change time of a file	3
filegroup()	Returns the group ID of a file	3
fileinode()	Returns the inode number of a file	3
filemtime()	Returns the last modification time of a file	3
fileowner()	Returns the user ID (owner) of a file	3
fileperms()	Returns the permissions of a file	3
filesize()	Returns the file size	3
filetype()	Returns the file type	3
flock()	Locks or releases a file	3
fnmatch()	Matches a filename or string against a specified pattern	4
fopen()	Opens a file or URL	3
fpassthru()	Reads from an open file, until EOF, and writes the result to the output buffer	3
fputcsv()	Formats a line as CSV and writes it to an open file	5
fputs()	Alias of fwrite()	3
fread()	Reads from an open file	3
fscanf()	Parses input from an open file according to a specified format	4
fseek()	Seeks in an open file	3
fstat()	Returns information about an open file	4
ftell()	Returns the current position in an open file	3
ftruncate()	Truncates an open file to a specified length	4
fwrite()	Writes to an open file	3
glob()	Returns an array of filenames / directories matching a specified pattern	4
is_dir()	Checks whether a file is a directory	3
is_executable()	Checks whether a file is executable	3
is_file()	Checks whether a file is a regular file	3
is_link()	Checks whether a file is a link	3
is_readable()	Checks whether a file is readable	3
is_uploaded_file()	Checks whether a file was uploaded via HTTP POST	3
is_writable()	Checks whether a file is writeable	4
is_writeable()	Alias of is_writable()	3
link()	Creates a hard link	3
linkinfo()	Returns information about a hard link	3
lstat()	Returns information about a file or symbolic link	3
mkdir()	Creates a directory	3
move_uploaded_file()	Moves an uploaded file to a new location	4
parse_ini_file()	Parses a configuration file	4

pathinfo()	Returns information about a file path	4
pclose()	Closes a pipe opened by popen()	3
popen()	Opens a pipe	3
readfile()	Reads a file and writes it to the output buffer	3
readlink()	Returns the target of a symbolic link	3
realpath()	Returns the absolute pathname	4
rename()	Renames a file or directory	3
rewind()	Rewinds a file pointer	3
rmdir()	Removes an empty directory	3
set_file_buffer()	Sets the buffer size of an open file	3
stat()	Returns information about a file	3
symlink()	Creates a symbolic link	3
tempnam()	Creates a unique temporary file	3
tmpfile()	Creates a unique temporary file	3
touch()	Sets access and modification time of a file	3
umask()	Changes file permissions for files	3
unlink()	Deletes a file	3

PHP Filesystem Constants

PHP: indicates the earliest version of PHP that supports the constant.

Constant	Description	PHP
GLOB_BRACE		
GLOB_ONLYDIR		
GLOB_MARK		
GLOB_NOSORT		
GLOB_NOCHECK		
GLOB_NOESCAPE		
PATHINFO_DIRNAME		
PATHINFO_BASENAME		
PATHINFO_EXTENSION		
FILE_USE_INCLUDE_PATH		
FILE_APPEND		
FILE_IGNORE_NEW_LINES		
FILE_SKIP_EMPTY_LINES		

1.Definition and Usage

The basename() function returns the filename from a path.

Syntax

```
basename(path,suffix)
```

Parameter	Description
path	Required. Specifies the path to check
suffix	Optional. Specifies a file extension. If the filename has this file extension, the file extension will not show

---

Example

```
<?php
$path = "/testweb/home.php";
//Show filename with file extension
echo basename($path) ."<br/>";
//Show filename without file extension
echo basename($path,".php");
?>
```

The output of the code above will be:

```
home.php
home
```

2.Definition and Usage

The chgrp() function changes the usergroup of the specified file.

Returns TRUE on success and FALSE on failure.

Syntax

```
chgrp(file,group)
```

Parameter	Description
file	Required. Specifies the file to check
group	Required. Specifies the new group. Can be a group name or a group ID

Example

```
<?php
chgrp("test.txt","admin")
?>
```

3.Definition and Usage

The chmod() function changes permissions of the specified file.

Returns TRUE on success and FALSE on failure.

Syntax

```
chmod(file,mode)
```

Parameter	Description
file	Required. Specifies the file to check
mode	Required. Specifies the new permissions. The mode parameter consists of four numbers: • The first number is always zero • The second number specifies permissions for the owner • The third number specifies permissions for the owner's user group • The fourth number specifies permissions for everybody else Possible values (to set multiple permissions, add up the following numbers): • 1 = execute permissions

	• 2 = write permissions • 4 = read permissions

Example

```
<?php
// Read and write for owner, nothing for everybody else
chmod("test.txt",0600);
// Read and write for owner, read for everybody else
chmod("test.txt",0644);
// Everything for owner, read and execute for everybody else
chmod("test.txt",0755);
// Everything for owner, read for owner's group
chmod("test.txt",0740);
?>
```

4.Definition and Usage

The chown() function changes the owner of the specified file.

Returns TRUE on success and FALSE on failure.

Syntax

```
chown(file,owner)
```

Parameter	Description
file	Required. Specifies the file to check
owner	Required. Specifies the new owner. Can be a user name or a user ID.

Example

```
<?php
chown("test.txt","charles")
```

```
?>
```

5.Definition and Usage

The clearstatcache() function clears the file status cache.

PHP caches data for some functions for better performance. If a file is being checked several times in a script, you might want to avoid caching to get correct results. To do this, use the clearstatcache() function.

Syntax

```
clearstatcache()
```

6.Tips and Notes

Tip: Functions that are caching:

- stat()
- lstat()
- file_exists()
- is_writable()
- is_readable()
- is_executable()
- is_file()
- is_dir()
- is_link()
- filectime()
- fileatime()
- filemtime()
- fileinode()
- filegroup()
- fileowner()
- filesize()
- filetype()
- fileperms()

Example

```
<?php
//check filesize
echo filesize("test.txt");
echo "<br />";
$file = fopen("test.txt", "a+");
// truncate file
ftruncate($file,100);
fclose($file);
//Clear cache and check filesize again
clearstatcache();
echo filesize("test.txt");
?>
```

The output of the code above could be:

```
792
100
```

7.Definition and Usage

The copy() function copies a file.

This function returns TRUE on success and FALSE on failure.

Syntax

```
copy(file,to_file)
```

Parameter	Description
file	Required. Specifies the file to copy
to_file	Required. Specifies the file to copy to

Tips and Notes

Note: If the destination file already exists, it will be overwritten.

Example

```
<?php
echo copy("source.txt","target.txt");
?>
```

The output of the code above will be:

```
1
```

8.Definition and Usage

The dirname() function returns the directory name from a path.

Syntax

```
dirname(path)
```

Parameter	Description
path	Required. Specifies the path to check

Example

```
<?php
echo dirname("c:/testweb/home.php") . "<br />";
echo dirname("/testweb/home.php");
?>
```

The output of the code above will be:

```
c:/testweb
/testweb
```

The disk_free_space() function returns the free space, in bytes, of the specified directory.

Syntax

```
disk_free_space(directory)
```

Parameter	Description
directory	Required. Specifies the directory to check

Example

```
<?php
echo disk_free_space("C:");
?>
```

The output of the code above could be:

```
109693288448
```

9.Definition and Usage

The disk_total_space() function returns the total space, in bytes, of the specified directory.

Syntax

```
disk_total_space(directory)
```

Parameter	Description
directory	Required. Specifies the directory to check

Example

```
<?php
echo disk_total_space("C:");
?>
```

The output of the code above could be:

```
119990349824
```

10. Definition and Usage

The diskfreespace() function returns the free space, in bytes, of the specified directory.

This function is an alias of the disk_free_space() function.

Syntax

```
diskfreespace(directory)
```

Parameter	Description
directory	Required. Specifies the directory to check

Example

```
<?php
echo diskfreespace("C:");
?>
```

The output of the code above could be:

```
109693288448
```

11. Definition and Usage

The fclose() function closes an open file.

This function returns TRUE on success or FALSE on failure.

Syntax

```
fclose(file)
```

Parameter	Description
file	Required. Specifies the file to close

Example

```
<?php
$file = fopen("test.txt","r");
//some code to be executed
fclose($file);
?>
```

12. Definition and Usage

The feof() function checks if the "end-of-file" (EOF) has been reached.

This function returns TRUE if an error occurs, or if EOF has been reached. Otherwise it returns FALSE.

Syntax

```
feof(file)
```

Parameter	Description
file	Required. Specifies the open file to check

Tips and Notes

Tip: The feof() function is useful for looping through data of unknown length.

Example

```
<?php
$file = fopen("test.txt", "r");
//Output a line of the file until the end is reached
while(! feof($file))
  {
  echo fgets($file). "<br />";
  }
fclose($file);
?>
```

The output of the code above will be:

```
Hello, this is a test file.
There are three lines here.
This is the last line.
```

13. Definition and Usage

The fflush() function writes all buffered output to an open file.

Returns TRUE on success and FALSE on failure.

Syntax

```
fflush(file)
```

Parameter	Description
file	Required. Specifies the open file stream to check

Example

```
<?php
file = fopen("test.txt","r+");
// some code
fflush($file);
?>
```

14. Definition and Usage

The fgetc() function returns a single character from an open file.

Syntax

```
fgetc(file)
```

Parameter	Description
file	Required. Specifies the file to check

Tips and Notes

Note: This function is slow and should not be used on large files. If you need to read one character at a time from a large file, use fgets() to read data one line at a time and then process the line one character at a time with fgetc().

Example 1

```
<?php
$file = fopen("test2.txt","r");
echo fgetc($file);
fclose($file);
?>
```

The output of the code above will be:

```
H
```

Example 2

Read file character by character:

```
<?php
$file = fopen("test2.txt","r");
while (! feof ($file))
  {
  echo fgetc($file);
  }
fclose($file);
?>
```

The output of the code above will be:

```
Hello, this is a test file.
```

15. Definition and Usage

The fgetcsv() function parses a line from an open file, checking for CSV fields.

The fgetcsv() function stops returning on a new line, at the specified length, or at EOF, whichever comes first.

This function returns the CSV fields in an array on success, or FALSE on failure and EOF.

Syntax

```
fgetcsv(file,length,separator,enclosure)
```

Parameter	Description
file	Required. Specifies the file to check
length	Optional. Specifies the maximum length of a line. Must be greater than the longest line (in characters) in the CSV file. Omitting this parameter (or setting it to 0) the line length is not limited, which is slightly slower. **Note:** This parameter is required in versions prior to PHP 5
separator	Optional. A character that specifies the field separator. Default is comma (,)
enclosure	Optional. A character that specifies the field enclosure character. Default is "

Tips and Notes

Tip: Also see the fputcsv() function.

Example 1

```
<?php
$file = fopen("contacts.csv","r");
print_r(fgetcsv($file));
fclose($file);
?>
```

The CSV file:

```
Kai Jim, Refsnes, Stavanger, Norway
Hege, Refsnes, Stavanger, Norway
```

The output of the code above will be:

```
Array
(
[0] => Kai Jim
[1] => Refsnes
[2] => Stavanger
[3] => Norway
)
```

Example 2

```
<?php
$file = fopen("contacts.csv","r");
while(! feof($file))
  {
  print_r(fgetcsv($file));
  }
fclose($file);
?>
```

The CSV file:

```
Kai Jim, Refsnes, Stavanger, Norway
Hege, Refsnes, Stavanger, Norway
```

The output of the code above will be:

```
Array
(
[0] => Kai Jim
[1] => Refsnes
[2] => Stavanger
[3] => Norway
)
Array
(
[0] => Hege
[1] => Refsnes
[2] => Stavanger
[3] => Norway
)
```

16. Definition and Usage

The fgets() function returns a line from an open file.

The fgets() function stops returning on a new line, at the specified length, or at EOF, whichever comes first.

This function returns FALSE on failure.

Syntax

```
fgets(file,length)
```

Parameter	Description
file	Required. Specifies the file to read from
length	Optional. Specifies the number of bytes to read. Default is 1024 bytes.

Example 1

```
<?php
$file = fopen("test.txt","r");
echo fgets($file);
fclose($file);
?>
```

The output of the code above will be:

```
Hello, this is a test file.
```

Example 2

Read file line by line:

```
<?php
$file = fopen("test.txt","r");
while(! feof($file))
  {
  echo fgets($file). "<br />";
  }
fclose($file);
?>
```

The output of the code above will be:

```
Hello, this is a test file.
There are three lines here.
This is the last line.
```

17. Definition and Usage

The fgetss() function returns a line, with HTML and PHP tags removed, from an open file.

The fgetss() function stops returning on a new line, at the specified length, or at EOF, whichever comes first.

This function returns FALSE on failure.

Syntax

```
fgetss(file,length,tags)
```

Parameter	Description
file	Required. Specifies the file to check
length	Optional. Specifies the number of bytes to read. Default is 1024 bytes. **Note:** This parameter is required in versions prior to PHP 5
tags	Optional. Specifies tags that will not be removed

Example 1

```
<?php
$file = fopen("test.htm","r");
echo fgetss($file);
fclose($file);
?>
```

The output of the code above will be:

```
This is a paragraph.
```

Example 2

```
<?php
$file = fopen("test.htm","r");
echo fgetss($file,1024,"<p>,<b>");
fclose($file);
?>
```

The output of the code above will be:

```
This is a paragraph.
```

The source of the output above will be:

```
<p><b>This is a paragraph.</b></p>
```

18. Definition and Usage

The file() reads a file into an array.

Each array element contains a line from the file, with newline still attached.

Syntax

```
file(path,include_path,context)
```

Parameter	Description
path	Required. Specifies the file to read
include_path	Optional. Set this parameter to '1' if you want to search for the file in the include_path (in php.ini) as well
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream. Can be skipped by using NULL.

Tips and Notes

Tip: This function became binary-safe in PHP 4.3. (meaning that both binary data, like images, and character data can be written with this function).

Example

```
<?php
print_r(file("test.txt"));
?>
```

The output of the code above will be:

```
Array
(
[0] => Hello World. Testing testing!
[1] => Another day, another line.
[2] => If the array picks up this line,
[3] => then is it a pickup line?
)
```

20. Definition and Usage

The file_exists() function checks whether or not a file or directory exists.

This function returns TRUE if the file or directory exists, otherwise it returns FALSE.

Syntax

```
file_exists(path)
```

Parameter	Description
path	Required. Specifies the path to check

Example

```
<?php
echo file_exists("test.txt");
?>
```

The output of the code above will be:

```
1
```

21. Definition and Usage

The file_get_contents() reads a file into a string.

This function is the preferred way to read the contents of a file into a string. Because it will use memory mapping techniques, if this is supported by the server, to enhance performance.

Syntax

```
file_get_contents(path,include_path,context,start,max_length)
```

Parameter	Description
path	Required. Specifies the file to read
include_path	Optional. Set this parameter to '1' if you want to search for the file in the include_path (in php.ini) as well
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream. Can be skipped by using NULL.
start	Optional. Specifies where in the file to start reading. This parameter was added in PHP 5.1
max_length	Optional. Specifies how many bytes to read. This parameter was added in PHP 5.1

Tips and Notes

Tip: This function is binary-safe (meaning that both binary data, like images, and character data can be written with this function).

Example

```
<?php
echo file_get_contents("test.txt");
?>
```

The output of the code above will be:

```
This is a test file with test text.
```

22. Definition and Usage

The file_put_contents() writes a string to a file.

This function follows these rules when accessing a file:

1. If FILE_USE_INCLUDE_PATH is set, check the include path for a copy of *filename*
2. Create the file if it does not exist
3. Open the file
4. Lock the file if LOCK_EX is set
5. If FILE_APPEND is set, move to the end of the file. Otherwise, clear the file content
6. Write the data into the file
7. Close the file and release any locks

This function returns the number of character written into the file on success, or FALSE on failure.

Syntax

```
file_put_contents(file,data,mode,context)
```

Parameter	Description
file	Required. Specifies the file to write to. If the file does not exist, this function will create one
data	Required. The data to write to the file. Can be a string, an array or a data stream
mode	Optional. Specifies how to open/write to the file. Possible values: • FILE_USE_INCLUDE_PATH • FILE_APPEND • LOCK_EX
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream.

Tips and Notes

Note: Use FILE_APPEND to avoid deleting the existing content of the file.

Example

```
<?php
echo file_put_contents("test.txt","Hello World. Testing!");
?>
```

The output of the code above will be:

```
21
```

23.Definition and Usage

The fileatime() function returns the last access time of the specified file.

This function returns the last access time as a Unix timestamp on success, FALSE on failure.

Syntax

```
fileatime(filename)
```

Parameter	Description
filename	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Note: Some Unix systems have access time updates disabled, because this function reduces performance for applications that regularly accesses a large number of files.

Example

```
<?php
echo fileatime("test.txt");
echo "<br />";
echo "Last access: ".date("F d Y H:i:s.",fileatime("test.txt"));
?>
```

The output of the code above could be:

```
1140684501
Last access: February 23 2006 09:48:21.
```

24. Definition and Usage

The filectime() function returns the last time the specified file was changed.

This function checks for the inode changes as well as regular changes. Inode changes is when permissions, owner, group or other metadata is changed.

This function returns the last change time as a Unix timestamp on success, FALSE on failure.

Syntax

```
filectime(filename)
```

Parameter	Description
filename	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Tip: Use the filemtime() function to return the last time the file content was changed.

Example

```
<?php
echo filectime("test.txt");
echo "<br />";
echo "Last change: ".date("F d Y H:i:s.",filectime("test.txt"));
?>
```

The output of the code above could be:

```
1138609592
Last change: January 30 2006 09:26:32
```

25. Definition and Usage

The filegroup() function returns the group ID of the specified file.

This function returns the group ID on success or FALSE on failure.

Syntax

```
filegroup(filename)
```

Parameter	Description
filename	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Note: This function doesn't produce meaningful results on Windows systems.

Tip: Use posix_getgrgid() to convert the group ID to a group name.

Example

```
<?php
echo filegroup("test.txt");
?>
```

26. Definition and Usage

The fileinode() function returns the inode of the specified file.

This function returns the inode number of the file on success, FALSE on failure.

Syntax

```
fileinode(filename)
```

Parameter	Description
filename	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Note: This function doesn't produce meaningful results on Windows systems.

Example

```
<?php
echo fileinode("test.txt");
?>
```

27. Definition and Usage

The filemtime() function returns the last time the file content was modified.

This function returns the last change time as a Unix timestamp on success, FALSE on failure.

Syntax

```
filemtime(filename)
```

Parameter	Description
filename	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Example

```
<?php
echo filemtime("test.txt");
echo "<br />";
echo "Last modified: ".date("F d Y H:i:s.",filemtime("test.txt"));
?>
```

The output of the code above could be:

```
1139919766
Last modified: February 14 2006 13:22:46.
```

28. Definition and Usage

The fileowner() function returns the user ID (owner) of the specified file.

This function returns the user ID on success or FALSE on failure.

Syntax

```
fileowner(filename)
```

Parameter	Description
filename	Required. Specifies the file to check

Tips and Notes

Note: The results of this function are cached. Use clearstatcache() to clear the cache.

Note: This function doesn't produce meaningful results on Windows systems.

Tip: Use posix_getpwuid() to convert the user ID to a user name.

Example

```
<?php
echo fileowner("test.txt");
?>
```

29. Definition and Usage

The fileperms() function returns the permissions for a file or directory.

This function returns the permission as a number on success or FALSE on failure.

Syntax

```
fileperms(filename)
```

Parameter	Description
filename	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Example 1

```
<?php
echo fileperms("test.txt");
?>
```

The output of the code above could be:

```
33206
```

Example 2

Display permissions as an octal value:

```
<?php
echo substr(sprintf("%o",fileperms("test.txt")),-4);
?>
```

The output of the code above could be:

```
1777
```

30. Definition and Usage

The filesize() function returns the size of the specified file.

This function returns the file size in bytes on success or FALSE on failure.

Syntax

```
filesize(filename)
```

Parameter	Description
filename	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Example

```
<?php
echo filesize("test.txt");
?>
```

The output of the code above will be:

```
20
```

31. Definition and Usage

The filetype() function returns the file type of a specified file or directory.

This function returns the one of seven possible values on success or FALSE on failure.

Possible return values:

- fifo
- char
- dir
- block
- link
- file
- unknown

Syntax

```
filetype(filename)
```

Parameter	Description
filename	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Example 1

```
<?php
echo filetype("test.txt");
?>
```

The output of the code above will be:

```
file
```

Example 2

```
<?php
echo filetype("images");
?>
```

The output of the code above will be:

```
dir
```

32. Definition and Usage

The flock() function locks or releases a file.

This function returns TRUE on success or FALSE on failure.

Syntax

```
flock(file,lock,block)
```

Parameter	Description
file	Required. Specifies an open file to lock or release
lock	Required. Specifies what kind of lock to use. Possible values: • LOCK_SH - Shared lock (reader). Allow other processes to access the file • LOCK_EX - Exclusive lock (writer). Prevent other processes from accessing the file • LOCK_UN - Release a shared or exclusive lock

	• LOCK_NB - Avoids blocking other processes while locking
block	Optional. Set to 1 to block other processes while locking

---

Tips and Notes

Note: These locks only apply to the current PHP process. Other processes can modify or delete a PHP-locked file if permissions allow.

Note: flock() is mandatory under Windows.

Tip: The lock is released also by fclose(), which is called automatically when script is finished.

Example

```
<?php
$file = fopen("test.txt","w+");
// exclusive lock
if (flock($file,LOCK_EX))
  {
  fwrite($file,"Write something");
  // release lock
  flock($file,LOCK_UN);
  }
else
  {
  echo "Error locking file!";
  }
fclose($file);
?>
```

33. Definition and Usage

The fnmatch() function matches a filename or string against the specified pattern.

Syntax

```
fnmatch(pattern,string,flags)
```

Parameter	Description
pattern	Required. Specifies the pattern to search for
string	Required. Specifies the string or file check
flags	Optional.

Tips and Notes

Note: This function is not implemented on Windows platforms.

Example

Checking a color name against a shell wildcard pattern:

```
<?php
$txt = "My car is darkgrey..."
if (fnmatch("*gr[ae]y",$txt))
  {
  echo "some form of gray ...";
  }
?>
```

34. Definition and Usage

The fopen() function opens a file or URL.

If fopen() fails, it returns FALSE and an error on failure. You can hide the error output by adding an '@' in front of the function name.

Syntax

```
fopen(filename,mode,include_path,context)
```

Parameter	Description
filename	Required. Specifies the file or URL to open
mode	Required. Specifies the type of access you require to the file/stream.

	Possible values: • "r" (Read only. Starts at the beginning of the file) • "r+" (Read/Write. Starts at the beginning of the file) • "w" (Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist) • "w+" (Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist) • "a" (Write only. Opens and writes to the end of the file or creates a new file if it doesn't exist) • "a+" (Read/Write. Preserves file content by writing to the end of the file) • "x" (Write only. Creates a new file. Returns FALSE and an error if file already exists) • "x+" (Read/Write. Creates a new file. Returns FALSE and an error if file already exists)
include_path	Optional. Set this parameter to '1' if you want to search for the file in the include_path (in php.ini) as well
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

---

Tips and Notes

Note: When writing to a text file, be sure to use the correct line-ending character! Unix systems use \n, Windows systems use \r\n, and Macintosh systems use \r as the line ending character. Windows offers a translation flag ('t') which will translate \n to \r\n when working with the file. You can also use 'b' to force binary mode. To use these flags, specify either 'b' or 't' as the last character of the mode parameter.

Example

```
<?php
$file = fopen("test.txt","r");
$file = fopen("/home/test/test.txt","r");
$file = fopen("/home/test/test.gif","wb");
$file = fopen("http://www.example.com/","r");
$file = fopen("ftp://user:password@example.com/test.txt","w");
?>
```

35. Definition and Usage

The fpassthru() function reads all data from the current position in an open file, until EOF, and writes the result to the output buffer.

This function returns the number of characters passed or FALSE on failure.

Syntax

```
fpassthru(file)
```

Parameter	Description
file	Required. Specifies the open file or resource to read from

---

Tips and Notes

Note: When using fpassthru() on a binary file on Windows, remember to open the file in binary mode.

Tip: Call rewind() to set the file pointer to the beginning of the file if you have already written to the file.

Tip: If you just want to dump the contents of a file to the output buffer, without first modifying it, use the readfile() function instead.

Example 1

```
<?php
$file = fopen("test.txt","r");
// Read first line
fgets($file);
// Send rest of the file to the output buffer
echo fpassthru($file);
fclose($file);
?>
```

The output of the code above could be:

```
There are three lines in this file.
This is the last line.59
```

59 indicates the number of characters passed.

Example 2

Dump index page of a www server:

```
<?php
$file = fopen("http://www.example.com","r");
fpassthru($file);
?>
```

36. Definition and Usage

The fputcsv() function formats a line as CSV and writes it to an open file.

This function returns the length of the written string, or FALSE on failure.

Syntax

```
fputcsv(file,fields,seperator,enclosure)
```

Parameter	Description
file	Required. Specifies the open file to write to
fields	Required. Specifies which array to get the data from
separator	Optional. A character that specifies the field separator. Default is comma (,)
enclosure	Optional. A character that specifies the field enclosure character. Default is "

Tips and Notes

Tip: Also see the fgetcsv() function.

Example

```
<?php
$list = array
(
"Peter,Griffin,Oslo,Norway",
"Glenn,Quagmire,Oslo,Norway",
);
$file = fopen("contacts.csv","w");
foreach ($list as $line)
  {
  fputcsv($file,split(',',$line));
  }
fclose($file);
?>
```

The CSV file will look like this after the code above has been executed:

```
Peter,Griffin,Oslo,Norway
Glenn,Quagmire,Oslo,Norway
```

37. Definition and Usage

The fputs() writes to an open file.

The function will stop at the end of the file or when it reaches the specified length, whichever comes first.

This function returns the number of bytes written on success, or FALSE on failure.

The fputs() function is an alias of the fwrite() function.

Syntax

```
fputs(file,string,length)
```

Parameter	Description
file	Required. Specifies the open file to write to
string	Required. Specifies the string to write to the open file
length	Optional. Specifies the maximum number of bytes to write

Tips and Notes

Tip: This function is binary-safe (meaning that both binary data, like images, and character data can be written with this function).

Example

```
<?php
$file = fopen("test.txt","w");
echo fputs($file,"Hello World. Testing!");
fclose($file);
?>
```

The output of the code above will be:

```
21
```

38. Definition and Usage

The fread() reads from an open file.

The function will stop at the end of the file or when it reaches the specified length, whichever comes first.

This function returns the read string, or FALSE on failure.

Syntax

```
fread(file,length)
```

Parameter	Description
file	Required. Specifies the open file to read from
length	Required. Specifies the maximum number of bytes to read

Tips and Notes

Tip: This function is binary-safe (meaning that both binary data, like images, and character data can be written with this function).

Example 1

Read 10 bytes from file:

```
<?php
$file = fopen("test.txt","r");
fread($file,"10");
fclose($file);
?>
```

Example 2

Read entire file:

```
<?php
$file = fopen("test.txt","r");
fread($file,filesize("test.txt"));
fclose($file);
?>
```

39. Definition and Usage

The fscanf() function parses the input from an open file according to the specified format.

Syntax

```
fscanf(file,format,mixed)
```

Parameter	Description
file	Required. Specifies the file to check
format	Required. Specifies the format. Possible format values: • %% - Returns a percent sign

	• %b - Binary number • %c - The character according to the ASCII value • %d - Signed decimal number • %e - Scientific notation (e.g. 1.2e+2) • %u - Unsigned decimal number • %f - Floating-point number (local settings aware) • %F - Floating-point number (not local settings aware) • %o - Octal number • %s - String • %x - Hexadecimal number (lowercase letters) • %X - Hexadecimal number (uppercase letters) Additional format values. These are placed between the % and the letter (example %.2f): • + (Forces both + and - in front of numbers. By default, only negative numbers are marked) • ' (Specifies what to use as padding. Default is space. Must be used together with the width specifier. Example: %'x20s (this uses "x" as padding) • - (Left-justifies the variable value) • [0-9] (Specifies the minimum width held of to the variable value) • .[0-9] (Specifies the number of decimal digits or maximum string length) **Note:** If multiple additional format values are used, they must be in the same order as above.
mixed	Optional.

Tips and Notes

Note: Any whitespace in the format string matches any whitespace in the input stream. This means that a tab (\t) in the format string can match a single space character in the input stream.

40. Definition and Usage

The fseek() function seeks in an open file.

This function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes.

This function returns 0 on success, or -1 on failure. Seeking past EOF will not generate an error.

Syntax

```
fseek(file,offset,whence)
```

Parameter	Description
file	Required. Specifies the open file to seek in
offset	Required. Specifies the new position (measured in bytes from the beginning of the file)
whence	Optional. (added in PHP 4). Possible values: • SEEK_SET - Set position equal to offset. Default • SEEK_CUR - Set position to current location plus offset • SEEK_END - Set position to EOF plus offset (to move to a position before EOF, the offset must be a negative value)

Tips and Notes

Tip: Find the current position by using ftell()!

Example

```
<?php
$file = fopen("test.txt","r");
// read first line
fgets($file);
// move back to beginning of file
fseek($file,0);
?>
```

41. Definition and Usage

The fstat() function returns information about an open file.

This function returns an array with the following elements:

- [0] or [dev] - Device number
- [1] or [ino] - Inode number
- [2] or [mode] - Inode protection mode
- [3] or [nlink] - Number of links
- [4] or [uid] - User ID of owner
- [5] or [gid] - Group ID of owner
- [6] or [rdev] - Inode device type
- [7] or [size] - Size in bytes
- [8] or [atime] - Last access (as Unix timestamp)
- [9] or [mtime] - Last modified (as Unix timestamp)
- [10] or [ctime] - Last inode change (as Unix timestamp)
- [11] or [blksize] - Blocksize of filesystem IO (if supported)
- [12] or [blocks] - Number of blocks allocated

Syntax

```
fstat(file)
```

Parameter	Description
file	Required. Specifies the open file to check

Tips and Notes

Note: The results from this function will differ from server to server. The array may contain the number index, the name index, or both.

Tip: This function is similar to stat(), except that with this function the file must be open.

Example

```
<?php
$file = fopen("test.txt","r");
print_r(fstat($file));
fclose($file);
?>
```

The output of the code above could be:

```
Array
(
[0] => 0
[1] => 0
[2] => 33206
[3] => 1
[4] => 0
[5] => 0
[6] => 0
[7] => 92
[8] => 1141633430
[9] => 1141298003
[10] => 1138609592
[11] => -1
[12] => -1
[dev] => 0
[ino] => 0
[mode] => 33206
[nlink] => 1
[uid] => 0
[gid] => 0
[rdev] => 0
[size] => 92
[atime] => 1141633430
[mtime] => 1141298003
[ctime] => 1138609592
[blksize] => -1
[blocks] => -1
)
```

---

42. Definition and Usage

The ftruncate() function truncates an open file to the specified length.

Returns TRUE on success, or FALSE on failure.

Syntax

```
ftruncate(file,size)
```

Parameter	Description
file	Required. Specifies the open file to truncate
size	Required. Specifies the new file size

Example

```
<?php
//check filesize
echo filesize("test.txt");
echo "<br />";
$file = fopen("test.txt", "a+");
ftruncate($file,100);
fclose($file);
//Clear cache and check filesize again
clearstatcache();
echo filesize("test.txt");
?>
```

The output of the code above will be:

```
792
100
```

43. Definition and Usage

The fwrite() writes to an open file.

The function will stop at the end of the file or when it reaches the specified length, whichever comes first.

This function returns the number of bytes written, or FALSE on failure.

Syntax

```
fwrite(file,string,length)
```

Parameter	Description
file	Required. Specifies the open file to write to
string	Required. Specifies the string to write to the open file
length	Optional. Specifies the maximum number of bytes to write

Tips and Notes

Tip: This function is binary-safe (meaning that both binary data, like images, and character data can be written with this function).

Example

```
<?php
$file = fopen("test.txt","w");
echo fwrite($file,"Hello World. Testing!");
fclose($file);
?>
```

The output of the code above will be:

```
21
```

44. Definition and Usage

The glob() function returns an array of filenames or directories matching a specified pattern.

This function returns an array of files/directories, or FALSE on failure.

Syntax

```
glob(pattern,flags)
```

Parameter	Description
pattern	Required. Specifies the pattern to search for
flags	Optional. Specifies special settings. Possible values: • GLOB_MARK - Adds a slash to each item returned • GLOB_NOSORT - Return files as they appear in the directory (unsorted) • GLOB_NOCHECK - Returns the search pattern if no match were found

	• GLOB_NOESCAPE - Backslashes do not quote metacharacters • GLOB_BRACE - Expands {a,b,c} to match 'a', 'b', or 'c' • GLOB_ONLYDIR - Return only directories which match the pattern • GLOB_ERR - (added in PHP 5.1) Stop on errors (errors are ignored by default)

Example 1

```
<?php
print_r(glob("*.txt"));
?>
```

The output of the code above could be:

```
Array
(
[0] => target.txt
[1] => source.txt
[2] => test.txt
[3] => test2.txt
)
```

Example 2

```
<?php
print_r(glob("*.*"));
?>
```

The output of the code above could be:

```
Array
(
[0] => contacts.csv
[1] => default.php
[2] => target.txt
[3] => source.txt
[4] => tem1.tmp
[5] => test.htm
[6] => test.ini
[7] => test.php
[8] => test.txt
[9] => test2.txt
```

```
)
```

45. Definition and Usage

The is_dir() function checks whether the specified file is a directory.

This function returns TRUE if the directory exists.

Syntax

```
is_dir(file)
```

Parameter	Description
file	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Example

```
<?php
$file = "images";
if(is_dir($file))
  {
  echo ("$file is a directory");
  }
else
  {
  echo ("$file is not a directory");
  }
?>
```

The output of the code above could be:

```
images is a directory
```

46. Definition and Usage

The is_executable() function checks whether the specified file is executable.

This function returns TRUE if the file is executable.

Syntax

```
is_executable(file)
```

Parameter	Description
file	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Note: The is_executable() function became available with Windows in PHP 5.0

Example

```
<?php
$file = "setup.exe";
if(is_executable($file))
  {
  echo ("$file is executable");
  }
else
  {
  echo ("$file is not executable");
  }
?>
```

The output of the code above could be:

```
setup.exe is executable
```

47. Definition and Usage

The is_file() function checks whether the specified file is a regular file.

This function returns TRUE if it is a file.

Syntax

```
is_file(file)
```

Parameter	Description
file	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Example

```
<?php
$file = "test.txt";
if(is_file($file))
  {
  echo ("$file is a regular file");
  }
else
  {
  echo ("$file is not a regular file");
  }
?>
```

The output of the code above could be:

```
test.txt is a regular file
```

48. Definition and Usage

The is_readable() function checks whether the specified file is readable.

This function returns TRUE if the file is readable.

Syntax

```
is_readable(file)
```

Parameter	Description
file	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Example

```
<?php
$file = "test.txt";
if(is_readable($file))
  {
  echo ("$file is readable");
  }
else
  {
  echo ("$file is not readable");
  }
?>
```

The output of the code above could be:

```
test.txt is readable
```

49. Definition and Usage

The is_uploaded_file() function checks whether the specified file is uploaded via HTTP POST.

This function returns TRUE if the file is uploaded via HTTP POST.

Syntax

```
is_uploaded_file(file)
```

Parameter	Description
file	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Example

```
<?php
$file = "test.txt";
if(is_uploaded_file($file))
  {
  echo ("$file is uploaded via HTTP POST");
  }
else
  {
  echo ("$file is not uploaded via HTTP POST");
  }
?>
```

The output of the code above could be:

```
test.txt is not uploaded via HTTP POST
```

50. Definition and Usage

The is_writable() function checks whether the specified file is writeable.

This function returns TRUE if the file is writeable.

Syntax

```
is_writable(file)
```

Parameter	Description
file	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Example

```
<?php
$file = "test.txt";
if(is_writable($file))
  {
  echo ("$file is writeable");
  }
else
  {
  echo ("$file is not writeable");
  }
?>
```

The output of the code above could be:

```
test.txt is writeable
```

51. Definition and Usage

The is_writeable() function checks whether the specified file is writeable.

This function returns TRUE if the file is writeable.

This function is an alias of the is_writable() function.

Syntax

```
is_writeable(file)
```

Parameter	Description

file	Required. Specifies the file to check

Tips and Notes

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Example

```
<?php
$file = "test.txt";
if(is_writeable($file))
  {
  echo ("$file is writeable");
  }
else
  {
  echo ("$file is not writeable");
  }
?>
```

The output of the code above could be:

```
test.txt is writeable
```

52. Definition and Usage

The link() function creates a hard link from the existing target with the specified name link.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
link(target,link)
```

Parameter	Description
target	Required.
link	Required.

Tips and Notes

Note: This is not an HTML link, but a link in the filesystem.

Note: This function does not work on Windows platforms.

Note: This function will not work on remote files.

53. Definition and Usage

The linkinfo() function returns information about a hard link.

This function returns a device ID, or FALSE on failure.

Syntax

```
linkinfo(path)
```

Parameter	Description
path	Required. Specifies the path to check

Tips and Notes

Note: This is not an HTML link, but a link in the filesystem.

Note: This function is not implemented on Windows.

54. Definition and Usage

The lstat() function returns information about a file or symbolic link.

This function returns an array with the following elements:

- [0] or [dev] - Device number
- [1] or [ino] - Inode number
- [2] or [mode] - Inode protection mode
- [3] or [nlink] - Number of links
- [4] or [uid] - User ID of owner
- [5] or [gid] - Group ID of owner
- [6] or [rdev] - Inode device type
- [7] or [size] - Size in bytes
- [8] or [atime] - Last access (as Unix timestamp)
- [9] or [mtime] - Last modified (as Unix timestamp)
- [10] or [ctime] - Last inode change (as Unix timestamp)
- [11] or [blksize] - Blocksize of filesystem IO (if supported)
- [12] or [blocks] - Number of blocks allocated

Syntax

```
lstat(file)
```

Parameter	Description
file	Required. Specifies the file to check

Tips and Notes

Note: The results from this function will differ from server to server. The array may contain the number index, the name index, or both.

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Tip: This function is similar to stat(), except that if the file parameter is a symbolic link, the status of the symlink is returned (not the status of the file pointed to by the symlink).

Example

```
<?php
print_r(lstat("test.txt"));
?>
```

The output of the code above could be:

```
Array
```

```
(
[0] => 0
[1] => 0
[2] => 33206
[3] => 1
[4] => 0
[5] => 0
[6] => 0
[7] => 92
[8] => 1141633430
[9] => 1141298003
[10] => 1138609592
[11] => -1
[12] => -1
[dev] => 0
[ino] => 0
[mode] => 33206
[nlink] => 1
[uid] => 0
[gid] => 0
[rdev] => 0
[size] => 92
[atime] => 1141633430
[mtime] => 1141298003
[ctime] => 1138609592
[blksize] => -1
[blocks] => -1
)
```

55. Definition and Usage

The mkdir() function creates a directory.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
mkdir(path,mode,recursive,context)
```

Parameter	Description
path	Required. Specifies the name of the directory to create
mode	Optional. Specifies permissions. By default, the mode is 0777 (widest possible access).

	The mode parameter consists of four numbers: • The first number is always zero • The second number specifies permissions for the owner • The third number specifies permissions for the owner's user group • The fourth number specifies permissions for everybody else Possible values (to set multiple permissions, add up the following numbers): • 1 = execute permissions • 2 = write permissions • 4 = read permissions
recursive	Optional. Specifies if the recursive mode is set (added in PHP 5)
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream (added in PHP 5)

Tips and Notes

Note: The mode parameters is ignored on Windows platforms.

Example

```
<?php
mkdir("testing");
?>
```

56.definition and Usage

The move_uploaded_file() function moves an uploaded file to a new location.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
move_uploaded_file(file,newloc)
```

Parameter	Description
file	Required. Specifies the file to be moved
newloc	Required. Specifies the new location for the file

Tips and Notes

Note: This function only works on files uploaded via HTTP POST.

Note: If the destination file already exists, it will be overwritten.

57. Definition and Usage

The parse_ini_file() function parses a configuration (ini) file and returns the settings in it in an array.

Syntax

```
parse_ini_file(file,process_sections)
```

Parameter	Description
file	Required. Specifies the ini file to check
process_sections	Optional. If set to TRUE, it returns is a multidimensional array with section names and settings included. Default is FALSE

Tips and Notes

Tip: This function can be used to read in your own application's configuration files, and has nothing to do with the php.ini file.

Note: The following reserved words must not be used as keys for ini files: null, yes, no, true, and false. Furthermore, there are also some reserved characters that must not be used in the keys: {}|&~![()".

Example 1

Contents of "test.ini":

```
[names]
me = Robert
you = Peter
[urls]
first = "http://www.example.com"
second = "http://www.w3schools.com"
```

PHP code:

```
<?php
print_r(parse_ini_file("test.ini"));
?>
```

The output of the code above will be:

```
Array
(
[me] => Robert
[you] => Peter
[first] => http://www.example.com
[second] => http://www.w3schools.com
)
```

Example 2

Contents of "test.ini":

```
[names]
me = Robert
you = Peter
[urls]
first = "http://www.example.com"
second = "http://www.w3schools.com"
```

PHP code (with process_sections set to true):

```
<?php
print_r(parse_ini_file("test.ini",true));
?>
```

The output of the code above will be:

```
Array
(
[names] => Array
  (
  [me] => Robert
  [you] => Peter
  )
[urls] => Array
  (
  [first] => http://www.example.com
  [second] => http://www.w3schools.com
  )
)
```

58. Definition and Usage

The unlink() function deletes a file.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
unlink(filename,context)
```

Parameter	Description
filename	Required. Specifies the file to delete
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

---

Example

```
<?php
$file = "test.txt";
if (!unlink($file))
  {
  echo ("Error deleting $file");
```

```
  }
else
  {
  echo ("Deleted $file");
  }
?>
```

59. Definition and Usage

The umask() function changes the file permissions for files.

This function sets PHP's umask to *mask* & 0777 and returns the old umask. However, if you call umask() without any arguments, it returns the current umask.

Syntax

```
umask(mask)
```

Parameter	Description
mask	Optional. Specifies the new permissions. Default is 0777 The mode parameter consists of four numbers: • The first number is always zero • The second number specifies permissions for the owner • The third number specifies permissions for the owner's user group • The fourth number specifies permissions for everybody else Possible values (to set multiple permissions, add up the following numbers): • 1 = execute permissions • 2 = write permissions • 4 = read permission

60Definition and Usage

The touch() function sets the access and modification time of the specified file.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
touch(filename,time,atime)
```

Parameter	Description
filename	Required. Specifies the file to touch
time	Optional. Sets the time. The current system time is set by default
atime	Optional. Sets the access time. Default is the current system time if no parameters are set, or the same as the time parameter if that parameter is set

---

Tips and Notes

Note: If the specified file does not exist, it will be created.

Example

```
<?php
touch("test.txt");
?>
```

61. Definition and Usage

The tmpfile() function creates a temporary file with a unique name in read-write (w+) mode.

Syntax

```
tmpfile()
```

Tips and Notes

Note: The temporary file is automatically removed when it is closed with fclose(), or when the script ends.

Tip: See also tempnam()

Example

```
<?php
$temp = tmpfile();
fwrite($temp, "Testing, testing.");
//Rewind to the start of file
rewind($temp);
//Read 1k from file
echo fread($temp,1024);
//This removes the file
fclose($temp);
?>
```

The output of the code above will be:

```
Testing, testing.
```

62. Definition and Usage

The tempnam() function creates a temporary file with a unique filename in the specified directory.

This function returns the new temporary filename, or FALSE on failure.

Syntax

```
tempnam(dir,prefix)
```

Parameter	Description
dir	Required. Specifies the directory of where to create the temp file
prefix	Required. Specifies the start of the filename

Tips and Notes

Note: If the specified directory does not exist, tempnam() may generate a file in the system's temporary directory.

Tip: See also tmpfile()

Example

```
<?php
echo tempnam("C:\inetpub\testweb","TMP0");
?>
```

The output of the code above could be:

```
C:\inetpub\testweb\TMP1.tmp
```

63. Definition and Usage

The symlink() function creates a symbolic link from the existing target with the specified name link.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
link(target,link)
```

Parameter	Description
target	Required.
link	Required.

Tips and Notes

Note: This is not an HTML link, but a link in the filesystem.

Note: This function does not work on Windows platforms.

64. Definition and Usage

The stat() function returns information about a file.

This function returns an array with the following elements:

- [0] or [dev] - Device number
- [1] or [ino] - Inode number
- [2] or [mode] - Inode protection mode
- [3] or [nlink] - Number of links
- [4] or [uid] - User ID of owner
- [5] or [gid] - Group ID of owner
- [6] or [rdev] - Inode device type
- [7] or [size] - Size in bytes
- [8] or [atime] - Last access (as Unix timestamp)
- [9] or [mtime] - Last modified (as Unix timestamp)
- [10] or [ctime] - Last inode change (as Unix timestamp)
- [11] or [blksize] - Blocksize of filesystem IO (if supported)
- [12] or [blocks] - Number of blocks allocated

Syntax

```
stat(file)
```

Parameter	Description
file	Required. Specifies the file to check

Tips and Notes

Note: The results from this function will differ from server to server. The array may contain the number index, the name index, or both.

Note: The result of this function are cached. Use clearstatcache() to clear the cache.

Tip: This function is similar to fstat(), except that with this function the file does not have to be open.

Example

```
<?php
$file = fopen("test.txt","r");
print_r(stat($file));
fclose($file);
?>
```

The output of the code above could be:

```
Array
(
[0] => 0
[1] => 0
[2] => 33206
[3] => 1
[4] => 0
[5] => 0
[6] => 0
[7] => 92
[8] => 1141633430
[9] => 1141298003
[10] => 1138609592
[11] => -1
[12] => -1
[dev] => 0
[ino] => 0
[mode] => 33206
[nlink] => 1
[uid] => 0
[gid] => 0
[rdev] => 0
[size] => 92
[atime] => 1141633430
[mtime] => 1141298003
[ctime] => 1138609592
[blksize] => -1
[blocks] => -1
)
```

65. Definition and Usage

The set_file_buffer() function sets the buffer size of an open file.

Output using fwrite() is normally buffered at 8K. So, if two processes writes to the same file, each will write up to 8K before pausing, and allow the other to write. If buffer is 0, write operations are unbuffered (meaning that the first write process will be completed before allowing other processes to write).

This function returns 0 on success, otherwise it returns EOF.

Syntax

```
set_file_buffer(file,buffer)
```

Parameter	Description
file	Required. Specifies the open file
buffer	Required. Specifies the buffer size in bytes

Tips and Notes

Tip: This function is an alias of stream_set_write_buffer().

Example

Create an unbuffered stream:

```
<?php
$file = fopen("test.txt","w");
if ($file)
  {
  set_file_buffer($file,0);
  fwrite($file,"Hello World. Testing!");
  fclose($file);
  }
?>
```

66. Definition and Usage

The rmdir() function removes an empty directory.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
rmdir(dir,context)
```

Parameter	Description
dir	Required. Specifies the directory to be removed
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

---

Example

```
<?php
$path = "images";
if(!rmdir($path))
  {
  echo ("Could not remove $path");
  }
?>
```

67. Definition and Usage

The rewind() function "rewinds" the position of the file pointer to the beginning of the file.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
rewind(file)
```

Parameter	Description
file	Required. Specifies the open file

Example

```
<?php
$file = fopen("test.txt","r");
//Change position of file pointer
fseek($file,"15");
```

```
//Set file pointer to 0
rewind($file);
fclose($file);
?>
```

68. Definition and Usage

The rename() function renames a file or directory.

This function returns TRUE on success, or FALSE on failure.

Syntax

```
rename(oldname,newname,context)
```

Parameter	Description
oldname	Required. Specifies the file or directory to be renamed
newname	Required. Specifies the new name of the file or directory
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

---

Example

```
<?php
rename("images","pictures");
?>
```

69. Definition and Usage

The realpath() function returns the absolute pathname.

This function removes all symbolic links (like '/./', '/../' and extra '/') and returns the absolute pathname.

This function returns FALSE on failure.

Syntax

```
realpath(path)
```

Parameter	Description
path	Required. Specifies the path to check

Example

```
<?php
echo realpath("test.txt");
?>
```

The output of the code above will be:

```
C:\Inetpub\testweb\test.txt
```

70. Definition and Usage

The readlink() function returns the target of a symbolic link.

This function returns the target of the link on success, or FALSE on failure.

Syntax

```
readlink(linkpath)
```

Parameter	Description
linkpath	Required. Specifies the link path to check

Tips and Notes

Note: This function is not implemented on Windows platforms.

Example

```
<?php
echo readlink("/user/testlink");
?>
```

71. Definition and Usage

The readfile() function reads a file and writes it to the output buffer.

This function returns the number of bytes read on success, or FALSE and an error on failure. You can hide the error output by adding an '@' in front of the function name.

Syntax

```
readfile(filename,include_path,context)
```

Parameter	Description
filename	Required. Specifies the file to read
include_path	Optional. Set this parameter to '1' if you want to search for the file in the include_path (in php.ini) as well
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

---

Tips and Notes

Tip: You can use a URL as a filename with this function if the fopen wrappers have been enabled in the php.ini file.

Example

```
<?php
echo readfile("test.txt");
?>
```

The output of the code above will be:

```
There are two lines in this file.
This is the last line.
57
```

72. Definition and Usage

The popen() function opens a pipe to the program specified in the command parameter.

This function returns FALSE if an error occurs.

Syntax

```
popen(command,mode)
```

Parameter	Description
command	Required. Specifies the command to execute
mode	Required. Specifies the connection mode. Possible values: • r: Read only • w: Write only (opens and clears existing file or creates a new file)

Example

```
<?php
$file = popen("/bin/ls","r");
//some code to be executed
pclose($file);
?>
```

73. Definition and Usage

The pclose() function closes a pipe opened by popen().

This function returns FALSE on failure.

Syntax

```
pclose(pipe)
```

Parameter	Description
pipe	Required. Specifies the pipe opened by popen()

Example

```
<?php
$file = popen("/bin/ls","r");
//some code to be executed
pclose($file);
?>
```

74. Definition and Usage

The pathinfo() function returns an array that contains information about a path.

The following array elements are returned:

- [dirname]
- [basename]
- [extension]

Syntax

```
pathinfo(path,options)
```

Parameter	Description
path	Required. Specifies the path to check
options	Optional. Specifies which array elements to return. Default is all Possible values: • PATHINFO_DIRNAME - return only dirname

	• PATHINFO_BASENAME - return only basename • PATHINFO_EXTENSION - return only extension

---

Tips and Notes

Note: The pathinfo() function returns a string if not all elements are requested.

Example 1

```
<?php
print_r(pathinfo("/testweb/test.txt"));
?>
```

The output of the code above will be:

```
Array
(
[dirname] => /testweb
[basename] => test.txt
[extension] => txt
)
```

Example 2

```
<?php
print_r(pathinfo("/testweb/test.txt",PATHINFO_BASENAME));
?>
```

The output of the code above will be:

```
test.txt
```

75.

76.

77.

78.

79.

80.

81.