POSIX

Introduction

This module contains an interface to those functions defined in the IEEE 1003.1 (POSIX.1) standards document which are not accessible through other means.

Warning
Sensitive data can be retrieved with the POSIX functions, e.g. posix_getpwnam() and friends. None of the POSIX function perform any kind of access checking when safe mode is enabled. It's therefore *strongly* advised to disable the POSIX extension at all (use *--disable-posix* in your configure line) if you're operating in such an environment.

Note
This extension is not available on Windows platforms.

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

POSIX functions are enabled by default. You can disable POSIX-like functions with *--disable-posix*.

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.

POSIX_F_OK (integer)
Check whether the file exists.

POSIX_R_OK (integer)
Check whether the file exists and has read permissions.

POSIX_W_OK (integer)
Check whether the file exists and has write permissions.

POSIX_X_OK (integer)
Check whether the file exists and has execute permissions.

POSIX_S_IFBLK (integer)
Block special file

POSIX_S_IFCHR (integer)
Character special file

POSIX_S_IFIFO (integer)
FIFO (named pipe) special file

POSIX_S_IFREG (integer)
Normal file

POSIX_S_IFSOCK (integer)
Socket

Note
These constants are available since PHP 5.1.0. Please also note that some of them may not be available in your system.

POSIX Functions

See Also

The section about Process Control Functions maybe of interest for you.

posix_access

posix_access -- Determine accessibility of a file

Description

bool **posix_access** (string `$file` [, int `$mode`])

posix_access() checks the user's permission of a file.

Parameters

`file`
The name of the file to be tested.

`mode`
A mask consisting of one or more of **POSIX_F_OK**, **POSIX_R_OK**, **POSIX_W_OK** and **POSIX_X_OK**. Defaults to **POSIX_F_OK**. **POSIX_R_OK**, **POSIX_W_OK** and **POSIX_X_OK** request checking whether the file exists and has read, write and execute permissions, respectively. **POSIX_F_OK** just requests checking for the existence of the file.

Return Values

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

Examples

Example #1 - posix_access() example

This example will check if the $file is readable and writable, otherwise will print an error message.

```
<?php

$file = 'some_file';

if (posix_access($file, POSIX_R_OK | POSIX_W_OK)) {
   echo 'The file is readable and writable!';

} else {
   $error = posix_get_last_error();

   echo "Error $error: " . posix_strerror($error);
}
```

```
?>
```

Notes

Note
When safe mode is enabled, PHP checks whether the files or directories being operated upon have the same UID (owner) as the script that is being executed.

See Also

- posix_get_last_error()
- posix_strerror()

posix_ctermid

posix_ctermid -- Get path name of controlling terminal

Description

string **posix_ctermid** (void)

Generates a string which is the pathname for the current controlling terminal for the process. On error this will set errno, which can be checked using posix_get_last_error()

Return Values

Upon successful completion, returns string of the pathname to the current controlling terminal. Otherwise **FALSE** is returned and errno is set, which can be checked with posix_get_last_error().

Examples

Example #2 - posix_ctermid() example

This example will display the path to the current TTY.

```
<?php
echo "I am running from ".posix_ctermid();
?>
```

See Also

- posix_ttyname()
- posix_get_last_error()

posix_get_last_error

posix_get_last_error -- Retrieve the error number set by the last posix function that failed

Description

int **posix_get_last_error** (void)

Retrieve the error number set by the last posix function that failed. The system error message associated with the errno may be checked with posix_strerror().

Return Values

Returns the errno (error number) set by the last posix function that failed. If no errors exist, 0 is returned.

Examples

Example #3 - posix_get_last_error() example

This example attempt to kill a bogus process id, which will set the last error. We will then print out the last errno.

```
<?php
posix_kill(999459,SIGKILL);
echo 'Your error returned was '.posix_get_last_error(); //Your error was ___
?>
```

See Also

- posix_strerror()

posix_getcwd

posix_getcwd -- Pathname of current directory

Description

string **posix_getcwd** (void)

Gets the absolute pathname of the script's current working directory. On error, it sets errno which can be checked using posix_get_last_error()

Return Values

Returns a string of the absolute pathname on success. On error, returns **FALSE** and sets errno which can be checked with posix_get_last_error().

Examples

Example #4 - posix_getcwd() example

This example will return the absolute path of the current working directory of the script.

```
<?php
echo 'My current working directory is '.posix_getcwd();
?>
```

Notes

Note

This function can fail on

- Read or Search permission was denied
- Pathname no longer exists

posix_getegid

posix_getegid -- Return the effective group ID of the current process

Description

int **posix_getegid** (void)

Return the numeric effective group ID of the current process.

Return Values

Returns an integer of the effective group ID.

Examples

Example #5 - posix_getegid() example

This example will print out the effective group id, once it is changed with posix_setegid().

```
<?php
echo 'My real group id is '.posix_getgid(); //20
posix_setegid(40);
echo 'My real group id is '.posix_getgid(); //20
echo 'My effective group id is '.posix_getegid(); //40
?>
```

Notes

posix_getegid() is different than posix_getgid() because effective group ID can be changed by a calling process using posix_setegid().

See Also

- posix_getgrgid() for information on how to convert this into a useable group name
- posix_getgid() get real group id.
- posix_setgid() change the effective group id

posix_geteuid

posix_geteuid -- Return the effective user ID of the current process

Description

int **posix_geteuid** (void)

Return the numeric effective user ID of the current process. See also posix_getpwuid() for information on how to convert this into a useable username.

Return Values

Returns the user id, as an integer

Examples

Example #6 - posix_geteuid() example

This example will show the current user id then set the effective user id to a separate id using posix_seteuid(), then show the difference between the real id and the effective id.

```
<?php
echo posix_getuid()."\n"; //10001
echo posix_geteuid()."\n"; //10001
posix_seteuid(10000);
echo posix_getuid()."\n"; //10001
echo posix_geteuid()."\n"; //10000
?>
```

See Also

- posix_getpwuid() for more information about the user.
- posix_getuid() get real user id.
- posix_setuid() change the effective user id
- POSIX man page GETEUID(2)

posix_getgid

posix_getgid -- Return the real group ID of the current process

Description

int **posix_getgid** (void)

Return the numeric real group ID of the current process.

Return Values

Returns the real group id, as an integer.

Examples

Example #7 - posix_getgid() example

This example will print out the real group id, even once the effective group id has been changed.

```
<?php
echo 'My real group id is '.posix_getgid(); //20
posix_setegid(40);
echo 'My real group id is '.posix_getgid(); //20
echo 'My effective group id is '.posix_getegid(); //40
?>
```

See Also

- posix_getgrgid() for information on how to convert this into a useable group name
- posix_getegid() get effective group id.
- posix_setgid() change the effective group id
- POSIX man page GETGID(2)

posix_getgrgid

posix_getgrgid -- Return info about a group by group id

Description

array **posix_getgrgid** (int `$gid`)

Gets information about a group provided its id.

Parameters

`gid`
The group id.

Return Values

The array elements returned are:

The group information array

Element	Description
name	The name element contains the name of the group. This is a short, usually less than 16 character "handle" of the group, not the real, full name.
passwd	The passwd element contains the group's password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
gid	Group ID, should be the same as the *`gid`* parameter used when calling the function, and hence redundant.
members	This consists of an array of string 's for all the members in the group.

ChangeLog

Version	Description
4.2.0	Prior to this version, members was simply an integer representing the number of members in the group, and the member names were returned with numerical indices.

Examples

Example #8 - Example use of posix_getgrgid()

```
<?php

$groupid   = posix_getegid();
$groupinfo = posix_getgrgid($groupid);

print_r($groupinfo);
?>
```

The above example will output something similar to:

```
Array
(
   [name]    => toons
   [passwd]  => x
   [members] => Array
       (
           [0] => tom
           [1] => jerry
       )
   [gid]     => 42
)
```

See Also

- posix_getegid()
- posix_getgrnam()
- filegroup()
- stat()
- safe_mode_gid
- POSIX man page GETGRNAM(3)

posix_getgrnam

posix_getgrnam -- Return info about a group by name

Description

array **posix_getgrnam** (string `$name`)

Gets information about a group provided its name.

Parameters

`name`
The name of the group

Return Values

The array elements returned are:

The group information array

Element	Description
name	The name element contains the name of the group. This is a short, usually less than 16 character "handle" of the group, not the real, full name. This should be the same as the *`name`* parameter used when calling the function, and hence redundant.
passwd	The passwd element contains the group's password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
gid	Group ID of the group in numeric form.
members	This consists of an array of string 's for all the members in the group.

ChangeLog

Version	Description

4.2.0	Prior to this version, members was simply an integer representing the number of members in the group, and the member names were returned with numerical indices.

Examples

Example #9 - Example use of posix_getgrnam()

```
<?php

$groupinfo = posix_getgrnam("toons");

print_r($groupinfo);
?>
```

The above example will output something similar to:

```
Array
(
   [name]    => toons
   [passwd]  => x
   [members] => Array
       (
           [0] => tom
           [1] => jerry
       )
   [gid]     => 42
)
```

See Also

- posix_getegid()
- posix_getgrgid()
- filegroup()
- stat()
- safe_mode_gid
- POSIX man page GETGRNAM(3)

posix_getgroups

posix_getgroups -- Return the group set of the current process

Description

array **posix_getgroups** (void)

Gets the group set of the current process.

Return Values

Returns an array of integers containing the numeric group ids of the group set of the current process.

Examples

Example #10 - Example use of posix_getgroups()

```
<?php

$groups = posix_getgroups();

print_r($groups);
?>
```

The above example will output something similar to:

```
Array
(
   [0] => 4
   [1] => 20
   [2] => 24
   [3] => 25
   [4] => 29
   [5] => 30
   [6] => 33
   [7] => 44
   [8] => 46
   [9] => 104
   [10] => 109
   [11] => 110
   [12] => 1000
)
```

See Also

- posix_getgrgid()

posix_getlogin

posix_getlogin -- Return login name

Description

string **posix_getlogin** (void)

Returns the login name of the user owning the current process.

Return Values

Returns the login name of the user, as a string.

Examples

Example #11 - Example use of posix_getlogin()

```
<?php
echo posix_getlogin(); //apache
?>
```

See Also

- posix_getpwnam()
- POSIX man page GETLOGIN(3)

posix_getpgid

posix_getpgid -- Get process group id for job control

Description

int **posix_getpgid** (int `$pid`)

Returns the process group identifier of the process *`pid`*.

Parameters

`pid`
The process id.

Return Values

Returns the identifier, as an integer.

Examples

Example #12 - Example use of posix_getpgid()

```
<?php
$pid = posix_getppid();
echo posix_getpgid($pid); //35
?>
```

Notes

Note

This is a not POSIX function, but is common on BSD and System V systems. If the system does not support this function, then it will not be included at compile time. This may be checked with function_exists().

See Also

- posix_getppid()

- man page SETPGID(2)

posix_getpgrp

posix_getpgrp -- Return the current process group identifier

Description

int **posix_getpgrp** (void)

Return the process group identifier of the current process.

Return Values

Returns the identifier, as an integer.

See Also

- POSIX.1 and the getpgrp(2) manual page on the POSIX system for more information on process groups.

posix_getpid

posix_getpid -- Return the current process identifier

Description

int **posix_getpid** (void)

Return the process identifier of the current process.

Return Values

Returns the identifier, as an integer.

Examples

Example #13 - Example use of posix_getpid()

```
<?php
echo posix_getpid(); //8805
?>
```

See Also

- posix_kill() to kill a process.
- POSIX man page GETPID(2)

posix_getppid

posix_getppid -- Return the parent process identifier

Description

int **posix_getppid** (void)

Return the process identifier of the parent process of the current process.

Return Values

Returns the identifier, as an integer.

Examples

Example #14 - Example use of posix_getppid()

```
<?php
echo posix_getppid(); //8259
?>
```

posix_getpwnam

posix_getpwnam -- Return info about a user by username

Description

array **posix_getpwnam** (string `$username`)

Returns an array of information about the given user.

Parameters

`username`
An alphanumeric username.

Return Values

The array elements returned are:

The user information array

Element	Description
name	The name element contains the username of the user. This is a short, usually less than 16 character "handle" of the user, not the real, full name. This should be the same as the *`username`* parameter used when calling the function, and hence redundant.
passwd	The passwd element contains the user's password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
uid	User ID of the user in numeric form.
gid	The group ID of the user. Use the function posix_getgrgid() to resolve the group name and a list of its members.
gecos	GECOS is an obsolete term that refers to the finger information field on a Honeywell batch processing system. The field, however, lives on, and its contents have been formalized by POSIX. The field contains a comma separated list containing

	the user's full name, office phone, office number, and home phone number. On most systems, only the user's full name is available.
dir	This element contains the absolute path to the home directory of the user.
shell	The shell element contains the absolute path to the executable of the user's default shell.

Examples

Example #15 - Example use of posix_getpwnam()

```
<?php

$userinfo = posix_getpwnam("tom");

print_r($userinfo);
?>
```

The above example will output something similar to:

```
Array
(
   [name]    => tom
   [passwd]  => x
   [uid]     => 10000
   [gid]     => 42
   [geocs]   => "tom,,,"
   [dir]     => "/home/tom"
   [shell]   => "/bin/bash"
)
```

See Also

- posix_getpwuid()
- POSIX man page GETPWNAM(3)

posix_getpwuid

posix_getpwuid -- Return info about a user by user id

Description

array **posix_getpwuid** (int `$uid`)

Returns an array of information about the user referenced by the given user ID.

Parameters

`uid`
The user identifier.

Return Values

Returns an associative array with the following elements:

The user information array

Element	Description
name	The name element contains the username of the user. This is a short, usually less than 16 character "handle" of the user, not the real, full name.
passwd	The passwd element contains the user's password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
uid	User ID, should be the same as the *`uid`* parameter used when calling the function, and hence redundant.
gid	The group ID of the user. Use the function posix_getgrgid() to resolve the group name and a list of its members.
gecos	GECOS is an obsolete term that refers to the finger information field on a Honeywell batch processing system. The field, however, lives on, and its contents have been formalized by POSIX. The field contains a comma separated list containing

	the user's full name, office phone, office number, and home phone number. On most systems, only the user's full name is available.
dir	This element contains the absolute path to the home directory of the user.
shell	The shell element contains the absolute path to the executable of the user's default shell.

Examples

Example #16 - Example use of posix_getpwuid()

```
<?php

$userinfo = posix_getpwuid(10000);

print_r($userinfo);
?>
```

The above example will output something similar to:

```
Array
(
   [name]    => tom
   [passwd]  => x
   [uid]     => 10000
   [gid]     => 42
   [geocs]   => "tom,,,"
   [dir]     => "/home/tom"
   [shell]   => "/bin/bash"
)
```

See Also

- posix_getpwnam()
- POSIX man page GETPWNAM(3)

posix_getrlimit

posix_getrlimit -- Return info about system resource limits

Description

array **posix_getrlimit** (void)

posix_getrlimit() returns an array of information about the current resource's soft and hard limits.

Each resource has an associated soft and hard limit. The soft limit is the value that the kernel enforces for the corresponding resource. The hard limit acts as a ceiling for the soft limit. An unprivileged process may only set its soft limit to a value from 0 to the hard limit, and irreversibly lower its hard limit.

Return Values

Returns an associative array of elements for each limit that is defined. Each limit has a soft and a hard limit.

List of possible limits returned

Limit name	Limit description
core	The maximum size of the core file. When 0, not core files are created. When core files are larger than this size, they will be truncated at this size.
totalmem	The maximum size of the memory of the process, in bytes.
virtualmem	The maximum size of the virtual memory for the process, in bytes.
data	The maximum size of the data segment for the process, in bytes.
stack	The maximum size of the process stack, in bytes.
rss	The maximum number of virtual pages resident in RAM
maxproc	The maximum number of processes that can be created for the real user ID of the calling process.

memlock	The maximum number of bytes of memory that may be locked into RAM.
cpu	The amount of time the process is allowed to use the CPU.
filesize	The maximum size of the data segment for the process, in bytes.
openfiles	One more than the maximum number of open file descriptors.

Examples

Example #17 - Example use of posix_getrlimit()

```
<?php

$limits = posix_getrlimit();

print_r($limits);
?>
```

The above example will output something similar to:

```
Array
(
   [soft core] => 0
   [hard core] => unlimited
   [soft data] => unlimited
   [hard data] => unlimited
   [soft stack] => 8388608
   [hard stack] => unlimited
   [soft totalmem] => unlimited
   [hard totalmem] => unlimited
   [soft rss] => unlimited
   [hard rss] => unlimited
   [soft maxproc] => unlimited
   [hard maxproc] => unlimited
   [soft memlock] => unlimited
   [hard memlock] => unlimited
   [soft cpu] => unlimited
   [hard cpu] => unlimited
   [soft filesize] => unlimited
   [hard filesize] => unlimited
   [soft openfiles] => 1024
   [hard openfiles] => 1024
)
```

Notes

Note
This is a not POSIX function, but is common on BSD and System V systems. If the system does not support this function, then it will not be included at compile time. This may be checked with function_exists().

See Also

- man page GETRLIMIT(2)

posix_getsid

posix_getsid -- Get the current sid of the process

Description

int **posix_getsid** (int `$pid`)

Return the session id of the process *`pid`*. The session id of a process is the process group id of the session leader.

Parameters

`pid`
: The process identifier. If set to 0, the current process is assumed. If an invalid *`pid`* is specified, then **FALSE** is returned and an error is set which can be checked with posix_get_last_error().

Return Values

Returns the identifier, as an integer.

Examples

Example #18 - Example use of posix_getsid()

```
<?php
$pid = posix_getpid();
echo posix_getsid($pid); //8805
?>
```

See Also

- posix_getpgid()
- posix_setsid()
- POSIX man page GETSID(2)

posix_getuid

posix_getuid -- Return the real user ID of the current process

Description

int **posix_getuid** (void)

Return the numeric real user ID of the current process.

Return Values

Returns the user id, as an integer

Examples

Example #19 - Example use of posix_getuid()

```
<?php
echo posix_getuid(); //10000
?>
```

See Also

- posix_getpwuid() for information on how to convert this into a useable username
- POSIX man page GETUID(2)

posix_initgroups

posix_initgroups -- Calculate the group access list

Description

bool **posix_initgroups** (string `$name`, int `$base_group_id`)

Calculates the group access list for the user specified in name.

Parameters

`name`
The user to calculate the list for.

`base_group_id`
Typically the group number from the password file.

Return Values

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

See Also

- The Unix manual page for initgroups(3).

posix_isatty

posix_isatty -- Determine if a file descriptor is an interactive terminal

Description

bool **posix_isatty** (int `$fd`)

Determines if the file descriptor *`fd`* refers to a valid terminal type device.

Parameters

`fd`
: The file descriptor.

Return Values

Returns **TRUE** if *`fd`* is an open descriptor connected to a terminal and **FALSE** otherwise.

See Also

- posix_ttyname()

posix_kill

posix_kill -- Send a signal to a process

Description

bool **posix_kill** (int `$pid`, int `$sig`)

Send the signal *`sig`* to the process with the process identifier *`pid`*.

Parameters

`pid`
: The process identifier.

`sig`
: One of the PCNTL signals constants.

Return Values

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

See Also

- The kill(2) manual page of the POSIX system, which contains additional information about negative process identifiers, the special pid 0, the special pid -1, and the signal number 0.

posix_mkfifo

posix_mkfifo -- Create a fifo special file (a named pipe)

Description

bool **posix_mkfifo** (string `$pathname`, int `$mode`)

posix_mkfifo() creates a special *FIFO* file which exists in the file system and acts as a bidirectional communication endpoint for processes.

Parameters

`pathname`
: Path to the *FIFO* file.

`mode`
: The second parameter *`mode`* has to be given in octal notation (e.g. 0644). The permission of the newly created *FIFO* also depends on the setting of the current umask(). The permissions of the created file are (mode & ~umask).

Return Values

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

Notes

Note
When safe mode is enabled, PHP checks whether the files or directories being operated upon have the same UID (owner) as the script that is being executed.

posix_mknod

posix_mknod -- Create a special or ordinary file (POSIX.1)

Description

bool **posix_mknod** (string `$pathname`, int `$mode` [, int `$major` [, int `$minor`]])

Creates a special or ordinary file.

Parameters

`pathname`
: The file to create

`mode`
: This parameter is constructed by a bitwise OR between file type (one of the following constants: **POSIX_S_IFREG**, **POSIX_S_IFCHR**, **POSIX_S_IFBLK**, **POSIX_S_IFIFO** or **POSIX_S_IFSOCK**) and permissions.

`major`
: The major device kernel identifier (required to pass when using **S_IFCHR** or **S_IFBLK**).

`minor`
: The minor device kernel identifier (defaults to 0).

Return Values

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

Examples

Example #20 - A posix_mknod() example

```
<?php

$file = '/tmp/tmpfile';  // file name
$type = POSIX_S_IFBLK;   // file type
$permissions = 0777;     // octal
$major = 1;
$minor = 8;              // /dev/random

if (!posix_mknod($file, $type | $permissions, $major, $minor)) {
   die('Error ' . posix_get_last_error() . ': ' .
posix_strerror(posix_get_last_error()));
```

```
}

?>
```

See Also

- posix_mkfifo()

posix_setegid

posix_setegid -- Set the effective GID of the current process

Description

bool **posix_setegid** (int `$gid`)

Set the effective group ID of the current process. This is a privileged function and needs appropriate privileges (usually root) on the system to be able to perform this function.

Parameters

`gid`
The group id.

Return Values

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

Examples

Example #21 - posix_setegid() **example**

This example will print out the effective group id, once changed.

```
<?php
echo 'My real group id is '.posix_getgid(); //20
posix_setegid(40);
echo 'My real group id is '.posix_getgid(); //20
echo 'My effective group id is '.posix_getegid(); //40
?>
```

See Also

- posix_getgrgid() for information on how to convert a group id into a useable group name
- posix_getgid() get real group id.
- posix_setgid() change the effective group id

posix_seteuid

posix_seteuid -- Set the effective UID of the current process

Description

bool **posix_seteuid** (int `$uid`)

Set the real user ID of the current process. This is a privileged function and needs appropriate privileges (usually root) on the system to be able to perform this function.

Parameters

`uid`
: The user id.

Return Values

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

See Also

- posix_setgid()

posix_setgid

posix_setgid -- Set the GID of the current process

Description

bool **posix_setgid** (int `$gid`)

Set the real group ID of the current process. This is a privileged function and needs appropriate privileges (usually root) on the system to be able to perform this function. The appropriate order of function calls is posix_setgid() first, posix_setuid() last.

Note
If the caller is a super user, this will also set the effective group id.

Parameters

`gid`
The group id.

Return Values

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

Examples

Example #22 - posix_setgid() **example**

This example will print out the effective group id, once it is changed.

```
<?php
echo 'My real group id is '.posix_getgid(); //20
posix_setgid(40);
echo 'My real group id is '.posix_getgid(); //40
echo 'My effective group id is '.posix_getegid(); //40
?>
```

See Also

- posix_getgrgid() for information on how to convert this into a useable group name
- posix_getgid() get real group id.

posix_setpgid

posix_setpgid -- Set process group id for job control

Description

bool **posix_setpgid** (int `$pid`, int `$pgid`)

Let the process *`pid`* join the process group *`pgid`*.

Parameters

`pid`
: The process id.

`pgid`
: The process group id.

Return Values

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

See Also

- See POSIX.1 and the setsid(2) manual page on the POSIX system for more informations on process groups and job control.

posix_setsid

posix_setsid -- Make the current process a session leader

Description

int **posix_setsid** (void)

Make the current process a session leader.

Return Values

Returns the session id, or -1 on errors.

See Also

- The POSIX.1 and the setsid(2) manual page on the POSIX system for more information on process groups and job control.

posix_setuid

posix_setuid -- Set the UID of the current process

Description

bool **posix_setuid** (int `$uid`)

Set the real user ID of the current process. This is a privileged function that needs appropriate privileges (usually root) on the system to be able to perform this function.

Parameters

`uid`
The user id.

Return Values

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

Examples

Example #23 - posix_setuid() **example**

This example will show the currrent user id and then set it to a different value.

```
<?php
echo posix_getuid()."\n"; //10001
echo posix_geteuid()."\n"; //10001
posix_setuid(10000);
echo posix_getuid()."\n"; //10000
echo posix_geteuid()."\n"; //10000
?>
```

See Also

- posix_setgid()
- posix_seteuid()
- posix_getuid()
- posix_geteuid()

posix_strerror

posix_strerror -- Retrieve the system error message associated with the given errno

Description

string **posix_strerror** (int `$errno`)

Returns the POSIX system error message associated with the given *`errno`*. You may get the *`errno`* parameter by calling posix_get_last_error().

Parameters

`errno`
: A POSIX error number, returned by posix_get_last_error(). If set to 0, then the string "Success" is returned.

Return Values

Returns the error message, as a string.

Examples

Example #24 - posix_strerror() example

This example will attempt to kill a process which does not exist, then will print out the corresponding error message.

```
<?php
posix_kill(50,SIGKILL);
echo posix_strerror(posix_get_last_error())."\n";
?>
```

The above example will output something similar to:

```
No such process
```

See Also

- posix_get_last_error()

posix_times

posix_times -- Get process times

Description

array **posix_times** (void)

Gets information about the current CPU usage.

Return Values

Returns a hash of strings with information about the current process CPU usage. The indices of the hash are:

- ticks - the number of clock ticks that have elapsed since reboot.
- utime - user time used by the current process.
- stime - system time used by the current process.
- cutime - user time used by current process and children.
- cstime - system time used by current process and children.

Notes

Warning
This function isn't reliable to use, it may return negative values for high times.

Examples

Example #25 - Example use of posix_times()

```
<?php

$times = posix_times();

print_r($times);
?>
```

The above example will output something similar to:

```
Array
(
```

```
    [ticks] => 25814410
    [utime] => 1
    [stime] => 1
    [cutime] => 0
    [cstime] => 0
)
```

posix_ttyname

posix_ttyname -- Determine terminal device name

Description

string **posix_ttyname** (int `$fd`)

Returns a string for the absolute path to the current terminal device that is open on the file descriptor *`fd`*.

Parameters

`fd`
: The file descriptor.

Return Values

On success, returns a string of the absolute path of the *`fd`*. On failure, returns **FALSE**

posix_uname

posix_uname -- Get system name

Description

array **posix_uname** (void)

Gets information about the system.

Posix requires that assumptions must not be made about the format of the values, e.g. the assumption that a release may contain three digits or anything else returned by this function.

Return Values

Returns a hash of strings with information about the system. The indices of the hash are

- sysname - operating system name (e.g. Linux)
- nodename - system name (e.g. valiant)
- release - operating system release (e.g. 2.2.10)
- version - operating system version (e.g. #4 Tue Jul 20 17:01:36 MEST 1999)
- machine - system architecture (e.g. i586)
- domainname - DNS domainname (e.g. example.com)

domainname is a GNU extension and not part of POSIX.1, so this field is only available on GNU systems or when using the GNU libc.

Examples

Example #26 - Example use of posix_uname()

```
<?php
$uname=posix_uname();
print_r($uname);
?>
```

The above example will output something similar to:

```
Array
(
   [sysname] => Linux
   [nodename] => funbox
   [release] => 2.6.20-15-server
```

```
    [version] => #2 SMP Sun Apr 15 07:41:34 UTC 2007
    [machine] => i686
)
```