Direct IO

Introduction

PHP supports the direct io functions as described in the Posix Standard (Section 6) for performing I/O functions at a lower level than the C-Language stream I/O functions (fopen(), fread(),..). The use of the DIO functions should be considered only when direct control of a device is needed. In all other cases, the standard filesystem functions are more than adequate.

Note
This extension has been moved to the » PECL repository and is no longer bundled with PHP as of PHP 5.1.0.

This extension is only available on Windows Platforms as of PHP 5.0.0

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

To get these functions to work, you have to configure PHP with *--enable-dio*.

Runtime Configuration

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

Resource Types

One resource type is defined by this extension: a file descriptor returned by dio_open().

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.

F_DUPFD (integer)

F_GETFD (integer)

F_GETFL (integer)

F_GETLK (integer)

F_GETOWN (integer)

F_RDLCK (integer)

F_SETFL (integer)

F_SETLK (integer)

F_SETLKW (integer)

F_SETOWN (integer)

F_UNLCK (integer)

F_WRLCK (integer)

O_APPEND (integer)

O_ASYNC (integer)

O_CREAT (integer)

O_EXCL (integer)

O_NDELAY (integer)

O_NOCTTY (integer)

O_NONBLOCK (integer)

O_RDONLY (integer)

O_RDWR (integer)

O_SYNC (integer)

O_TRUNC (integer)

O_WRONLY (integer)

S_IRGRP (integer)

S_IROTH (integer)

S_IRUSR (integer)

S_IRWXG (integer)

S_IRWXO (integer)

S_IRWXU (integer)

S_IWGRP (integer)

S_IWOTH (integer)

S_IWUSR (integer)

S_IXGRP (integer)

S_IXOTH (integer)

S_IXUSR (integer)

Direct IO Functions

dio_close

dio_close -- Closes the file descriptor given by fd

Description

void **dio_close** (resource `$fd`)

The function dio_close() closes the file descriptor *`fd`*.

Parameters

`fd`
The file descriptor returned by dio_open().

Return Values

No value is returned.

Examples

Example #1 - Closing an open file descriptor

```
<?php
$fd = dio_open('/dev/ttyS0', O_RDWR);

dio_close($fd);
?>
```

See Also

- dio_open()

dio_fcntl

dio_fcntl -- Performs a c library fcntl on fd

Description

mixed **dio_fcntl** (resource `$fd`, int `$cmd` [, mixed `$args`])

The dio_fcntl() function performs the operation specified by *`cmd`* on the file descriptor *`fd`*. Some commands require additional arguments *`args`* to be supplied.

Parameters

`fd`
: The file descriptor returned by dio_open().

`cmd`
: Can be one of the following operations:

- **F_SETLK** - Lock is set or cleared. If the lock is held by someone else dio_fcntl() returns -1.
- **F_SETLKW** - like **F_SETLK**, but in case the lock is held by someone else, dio_fcntl() waits until the lock is released.
- **F_GETLK** - dio_fcntl() returns an associative array (as described above) if someone else prevents lock. If there is no obstruction key "type" will set to **F_UNLCK**.
- **F_DUPFD** - finds the lowest numbered available file descriptor greater than or equal to *`args`* and returns them.
- **F_SETFL** - Sets the file descriptors flags to the value specified by *`args`*, which can be **O_APPEND**, **O_NONBLOCK** or **O_ASYNC**. To use **O_ASYNC** you will need to use the PCNTL extension.

`args`
: *`args`* is an associative array, when *`cmd`* is **F_SETLK** or **F_SETLLW**, with the following keys:

- "start" - offset where lock begins
- "length" - size of locked area. zero means to end of file
- "wenth" - Where l_start is relative to: can be **SEEK_SET**, **SEEK_END** and **SEEK_CUR**
- "type" - type of lock: can be **F_RDLCK** (read lock), **F_WRLCK** (write lock) or **F_UNLCK** (unlock)

Return Values

Returns the result of the C call.

Examples

Example #2 - Setting and clearing a lock

```
<?php

$fd = dio_open('/dev/ttyS0', O_RDWR);

if (dio_fcntl($fd, F_SETLK, Array("type"=>F_WRLCK)) == -1) {
  // the file descriptor appears locked
  echo "The lock can not be cleared. It is held by someone else.";
} else {
  echo "Lock succesfully set/cleared";
}

dio_close($fd);
?>
```

Notes

Note

This function is not implemented on Windows platforms.

dio_open

dio_open -- Opens a new filename with specified permissions of flags and creation permissions of mode

Description

resource **dio_open** (string `$filename`, int `$flags` [, int `$mode`])

dio_open() opens a file and returns a new file descriptor for it.

Parameters

`filename`
The opened file.

`flags`
The *`flags`* parameter can also include any combination of the following flags:

- **O_CREAT** - creates the file, if it doesn't already exist.
- **O_EXCL** - if both, **O_CREAT** and **O_EXCL** are set, dio_open() fails, if the file already exists.
- **O_TRUNC** - if the file exists, and its opened for write access, the file will be truncated to zero length.
- **O_APPEND** - write operations write data at the end of the file.
- **O_NONBLOCK** - sets non blocking mode.

`mode`
If *`flags`* is **O_CREAT**, *`mode`* will set the mode of the file (creation permissions).

- **O_RDONLY** - opens the file for read access.
- **O_WRONLY** - opens the file for write access.
- **O_RDWR** - opens the file for both reading and writing.

Return Values

A file descriptor or **FALSE** on error.

Examples

Example #3 - Opening a file descriptor

```
<?php

$fd = dio_open('/dev/ttyS0', O_RDWR | O_NOCTTY | O_NONBLOCK);

dio_close($fd);
?>
```

See Also

- dio_close()

dio_read

dio_read -- Reads bytes from a file descriptor

Description

string **dio_read** (resource `$fd` [, int `$len`])

The function dio_read() reads and returns *`len`* bytes from file with descriptor *`fd`*.

Parameters

`fd`
: The file descriptor returned by dio_open().

`len`
: The number of bytes to read. If not specified, dio_read() reads 1K sized block.

Return Values

The bytes read from *`fd`*.

See Also

- dio_write()

dio_seek

dio_seek -- Seeks to pos on fd from whence

Description

int **dio_seek** (resource `$fd`, int `$pos` [, int `$whence`])

The function dio_seek() is used to change the file position of the given file descriptor.

Parameters

`fd`
The file descriptor returned by dio_open().

`pos`
The new position.

`whence`
Specifies how the position *`pos`* should be interpreted:

- **SEEK_SET** (default) - specifies that *`pos`* is specified from the beginning of the file.
- **SEEK_CUR** - Specifies that *`pos`* is a count of characters from the current file position. This count may be positive or negative.
- **SEEK_END** - Specifies that *`pos`* is a count of characters from the end of the file. A negative count specifies a position within the current extent of the file; a positive count specifies a position past the current end. If you set the position past the current end, and actually write data, you will extend the file with zeros up to that position.

Return Values

Examples

Example #4 - Positioning in a file

```
<?php

$fd = dio_open('/dev/ttyS0', O_RDWR);

dio_seek($fd, 10, SEEK_SET);
```

```
// position is now at 10 characters from the start of the file

dio_seek($fd, -2, SEEK_CUR);
// position is now at 8 characters from the start of the file

dio_seek($fd, -5, SEEK_END);
// position is now at 5 characters from the end of the file

dio_seek($fd, 10, SEEK_END);
// position is now at 10 characters past the end of the file.
// The 10 characters between the end of the file and the current
// position are filled with zeros.

dio_close($fd);
?>
```

dio_stat

dio_stat -- Gets stat information about the file descriptor fd

Description

array **dio_stat** (resource `$fd`)

dio_stat() returns information about the given file descriptor.

Parameters

`fd`
: The file descriptor returned by dio_open().

Return Values

Returns an associative array with the following keys:

- "device" - device
- "inode" - inode
- "mode" - mode
- "nlink" - number of hard links
- "uid" - user id
- "gid" - group id
- "device_type" - device type (if inode device)
- "size" - total size in bytes
- "blocksize" - blocksize
- "blocks" - number of blocks allocated
- "atime" - time of last access
- "mtime" - time of last modification
- "ctime" - time of last change

On error dio_stat() returns **NULL**.

dio_tcsetattr

dio_tcsetattr -- Sets terminal attributes and baud rate for a serial port

Description

bool **dio_tcsetattr** (resource `$fd`, array `$options`)

dio_tcsetattr() sets the terminal attributes and baud rate of the open *`fd`*.

Parameters

`fd`
The file descriptor returned by dio_open().

`options`
The currently available options are:

- 'baud' - baud rate of the port - can be 38400,19200,9600,4800,2400,1800, 1200,600,300,200,150,134,110,75 or 50, default value is 9600.
- 'bits' - data bits - can be 8,7,6 or 5. Default value is 8.
- 'stop' - stop bits - can be 1 or 2. Default value is 1.
- 'parity' - can be 0,1 or 2. Default value is 0.

Return Values

No value is returned.

Examples

Example #5 - Setting the baud rate on a serial port

```
<?php

$fd = dio_open('/dev/ttyS0', O_RDWR | O_NOCTTY | O_NONBLOCK);

dio_fcntl($fd, F_SETFL, O_SYNC);

dio_tcsetattr($fd, array(
 'baud' => 9600,
 'bits' => 8,
 'stop'  => 1,
 'parity' => 0
```

```
));

while (1) {

 $data = dio_read($fd, 256);

 if ($data) {
     echo $data;
 }
}

?>
```

Notes

Note
This function is not implemented on Windows platforms.

dio_truncate

dio_truncate -- Truncates file descriptor fd to offset bytes

Description

bool **dio_truncate** (resource `$fd`, int `$offset`)

dio_truncate() truncates a file to at most *`offset`* bytes in size.

If the file previously was larger than this size, the extra data is lost. If the file previously was shorter, it is unspecified whether the file is left unchanged or is extended. In the latter case the extended part reads as zero bytes.

Parameters

`fd`
: The file descriptor returned by dio_open().

`offset`
: The offset in bytes.

Return Values

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

Notes

Note
This function is not implemented on Windows platforms.

dio_write

dio_write -- Writes data to fd with optional truncation at length

Description

int **dio_write** (resource `$fd`, string `$data` [, int `$len`])

dio_write() writes up to *`len`* bytes from *`data`* to file *`fd`*.

Parameters

`fd`
: The file descriptor returned by dio_open().

`data`
: The written data.

`len`
: The length of data to write in bytes. If not specified, the function writes all the data to the specified file.

Return Values

Returns the number of bytes written to *`fd`*.

See Also

- dio_read()