Shared Memory

Introduction

Shmop is an easy to use set of functions that allows PHP to read, write, create and delete Unix shared memory segments.

Note
Versions of Windows previous to Windows 2000 do not support shared memory. Under Windows, Shmop will only work when PHP is running as a web server module, such as Apache or IIS (CLI and CGI will not work).

Note
In PHP 4.0.3, these functions were prefixed by *shm* rather than *shmop*.

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

To use shmop you will need to compile PHP with the *--enable-shmop* parameter in your configure line.

Runtime Configuration

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

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Examples

Example #1 - Shared Memory Operations Overview

```
<?php

// Create 100 byte shared memory block with system id of 0xff3
$shm_id = shmop_open(0xff3, "c", 0644, 100);
if (!$shm_id) {
   echo "Couldn't create shared memory segment\n";
}

// Get shared memory block's size
$shm_size = shmop_size($shm_id);
echo "SHM Block Size: " . $shm_size . " has been created.\n";

// Lets write a test string into shared memory
$shm_bytes_written = shmop_write($shm_id, "my shared memory block", 0);
if ($shm_bytes_written != strlen("my shared memory block")) {
   echo "Couldn't write the entire length of data\n";
}

// Now lets read the string back
$my_string = shmop_read($shm_id, 0, $shm_size);
if (!$my_string) {
   echo "Couldn't read from shared memory block\n";
}
echo "The data inside shared memory was: " . $my_string . "\n";

//Now lets delete the block and close the shared memory segment
if (!shmop_delete($shm_id)) {
   echo "Couldn't mark shared memory block for deletion.";
}
shmop_close($shm_id);

?>
```

Shared Memory Functions

shmop_close

shmop_close -- Close shared memory block

Description

void **shmop_close** (int `$shmid`)

shmop_close() is used to close a shared memory block.

Parameters

`shmid`
The shared memory block identifier created by shmop_open()

Return Values

No value is returned.

Examples

Example #2 - Closing shared memory block

```
<?php
shmop_close($shm_id);
?>
```

This example will close shared memory block identified by *$shm_id*.

See Also

- shmop_open()

shmop_delete

shmop_delete -- Delete shared memory block

Description

bool **shmop_delete** (int `$shmid`)

shmop_delete() is used to delete a shared memory block.

Parameters

`shmid`
The shared memory block identifier created by shmop_open()

Return Values

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

Examples

Example #3 - Deleting shared memory block

```
<?php
shmop_delete($shm_id);
?>
```

This example will delete shared memory block identified by *$shm_id*.

shmop_open

shmop_open -- Create or open shared memory block

Description

int **shmop_open** (int `$key`, string `$flags`, int `$mode`, int `$size`)

shmop_open() can create or open a shared memory block.

Parameters

`key`
System's id for the shared memory block. Can be passed as a decimal or hex.

`flags`
The flags that you can use:

- "a" for access (sets SHM_RDONLY for shmat) use this flag when you need to open an existing shared memory segment for read only
- "c" for create (sets IPC_CREATE) use this flag when you need to create a new shared memory segment or if a segment with the same key exists, try to open it for read and write
- "w" for read & write access use this flag when you need to read and write to a shared memory segment, use this flag in most cases.
- "n" create a new memory segment (sets IPC_CREATE|IPC_EXCL) use this flag when you want to create a new shared memory segment but if one already exists with the same flag, fail. This is useful for security purposes, using this you can prevent race condition exploits.

`mode`
The permissions that you wish to assign to your memory segment, those are the same as permission for a file. Permissions need to be passed in octal form, like for example *0644*

`size`
The size of the shared memory block you wish to create in bytes

Note
Note: the 3rd and 4th should be entered as 0 if you are opening an existing memory segment.

Return Values

On success shmop_open() will return an id that you can use to access the shared memory segment you've created. **FALSE** is returned on failure.

Examples

Example #4 - Create a new shared memory block

```
<?php
$shm_key = ftok(__FILE__, 't');
$shm_id = shmop_open($shm_key, "c", 0644, 100);
?>
```

This example opened a shared memory block with a system id returned by ftok().

See Also

- shmop_close()
- shmop_delete()

shmop_read

shmop_read -- Read data from shared memory block

Description

string **shmop_read** (int `$shmid`, int `$start`, int `$count`)

shmop_read() will read a string from shared memory block.

Parameters

`shmid`
The shared memory block identifier created by shmop_open()

`start`
Offset from which to start reading

`count`
The number of bytes to read

Return Values

Returns the data or **FALSE** on failure.

Examples

Example #5 - Reading shared memory block

```
<?php
$shm_data = shmop_read($shm_id, 0, 50);
?>
```

This example will read 50 bytes from shared memory block and place the data inside *$shm_data*.

See Also

- shmop_write()

shmop_size

shmop_size -- Get size of shared memory block

Description

int **shmop_size** (int `$shmid`)

shmop_size() is used to get the size, in bytes of the shared memory block.

Parameters

`shmid`
The shared memory block identifier created by shmop_open()

Return Values

Returns an int, which represents the number of bytes the shared memory block occupies.

Examples

Example #6 - Getting the size of the shared memory block

```
<?php
$shm_size = shmop_size($shm_id);
?>
```

This example will put the size of shared memory block identified by *$shm_id* into *$shm_size*.

shmop_write

shmop_write -- Write data into shared memory block

Description

int **shmop_write** (int `$shmid`, string `$data`, int `$offset`)

shmop_write() will write a string into shared memory block.

Parameters

`shmid`
The shared memory block identifier created by shmop_open()

`data`
A string to write into shared memory block

`offset`
Specifies where to start writing data inside the shared memory segment.

Return Values

The size of the written *`data`*, or **FALSE** on failure.

Examples

Example #7 - Writing to shared memory block

```
<?php
$shm_bytes_written = shmop_write($shm_id, $my_string, 0);
?>
```

This example will write data inside *$my_string* into shared memory block, *$shm_bytes_written* will contain the number of bytes written.

See Also

- shmop_read()