NAME

shm_open – open a shared memory object

SYNOPSIS

cc [*flag* ...] *file* ... **–lposix4** [*library* ...]

#include <sys/mman.h>

int shm_open(const char **name*, **int** *oflag*, **mode_t** *mode* **);**

MT-LEVEL

MT-Safe

DESCRIPTION

shm_open( ) either opens a file descriptor for the shared memory object with the name referenced by *name.* If successful, **shm_open( )** returns a file descriptor for the shared memory object that is the lowest numbered file descriptor not currently open for that process. Since the open file description is new, the new file descriptor is not as yet shared with any other processes.

The file status flags and file access modes of the open file descriptor are set according to the value of *oflag:* the bitwise inclusive OR of the following flags, defined in the header <fcntl.h>. (Applications must specify exactly one of the first two values below in the value of *oflag):*

O_RDONLY Open for read access only.

O_RDWR Open for read or write access.

Any combination of the remaining flags may be bitwise inclusive OR- ed with the value of *oflag:*

O_CREAT If *name* does not exist, the shared memory object is created, it's user ID is set to the effective user ID of the process, and it's group ID is set to a system default group ID or to the effective group ID of the process. The shared memory object's permission bits will be set to the value of *mode*, modified by clearing all bits set in the file mode creation mask of the process (see **umask**(2)).

mode does not affect whether the shared memory object is opened for reading, for writing, or for both. The new shared memory object has a size of zero.

If the shared memory object does exist, this flag will have no effect, except as specified under **O_EXCL** below.

O_EXCL If both **OEXCL** and **O_CREAT** are set, **shm_open( )** fails if the shared memory object, *name,* exists. The check for the existence of the shared memory object and the creation of the object if it does not exist is atomic with respect to other processes executing **shm_open( )** naming the same shared memory object with
OEXCL and **O_CREAT** set.

O_TRUNC If the shared memory object exists, and it is successfully opened **O_RDWR,** the object is truncated to zero length and the mode and ownership are unchanged by this function call.

RETURN VALUES

If successful, **shm_open( )** returns a nonnegative integer representing the lowest numbered unused file descriptor, otherwise it returns **-1** and sets **errno** to indicate the error condition.

ERRORS

EACCES The shared memory object exists and the permissions specified by *oflag* are denied, or the shared memory object does not exist and permission to create the shared memory object is denied, or **O_TRUNC** is specified and write permission is denied.

EEXIST **O_CREAT** and **O_EXCL** are set and the named shared memory object already exists.

EINTR The **shm_open( )** operation was interrupted by a signal.

EINVAL *name* is an invalid file description.

EMFILE The number of open file descriptors in this process exceeds **{OPEN_MAX}**.

ENAMETOOLONG The length of the *name* string exceeds **{PATH_MAX}**, or a pathname component is longer than **{NAME_MAX}** while **_POSIX_NO_TRUNC** is in effect.

ENFILE The system file table is full

ENOENT **O_CREAT** is not set and the named shared memory object does not exist.

ENOSPC There is insufficient space for the creation of the new shared memory object.

ENOSYS **shm_open( )** is not supported by this implementation.

FILES

/usr/include/fcntl.h

SEE ALSO

close(2), **dup**(2), **exec**(2), **fcntl**(2), **mmap**(2), **umask**(2), **sysconf**(3C), **shm_unlink**(3R), **fcntl**(5)

NOTES

When a shared memory object is created, the state of the shared memory object, including all data associated with the shared memory object, persists until the shared memory object is unlinked and all other references are gone.

BUGS

In Solaris 2.5, these functions always return –**1** and set **errno** to **ENOSYS,** because this release does not support the Shared Memory Objects option. It is our intention to provide support for these interfaces in future releases.