Rocky Enterprise Linux 9.2 Manual Pages on command 'shm_unlink.3'

C:\>man shm_unlink.3

SHM_OPEN(3) Linux Programmer's Manual SHM_OPEN(3)

NAME

shm_open, shm_unlink - create/open or unlink POSIX shared memory objects

SYNOPSIS

```
#include <sys/mman.h>
#include <sys/stat.h>        /* For mode constants */
#include <fcntl.h>           /* For O_* constants */
int shm_open(const char *name, int oflag, mode_t mode);
int shm_unlink(const char *name);
```

Link with -lrt.

DESCRIPTION

shm_open() creates and opens a new, or opens an existing, POSIX shared memory ob? ject. A POSIX shared memory object is in effect a handle which can be used by un? related processes to mmap(2) the same region of shared memory. The shm_unlink() function performs the converse operation, removing an object previously created by shm_open().

The operation of shm_open() is analogous to that of open(2). name specifies the shared memory object to be created or opened. For portable use, a shared memory object should be identified by a name of the form /somename; that is, a null-termi? nated string of up to NAME_MAX (i.e., 255) characters consisting of an initial slash, followed by one or more characters, none of which are slashes.

oflag is a bit mask created by ORing together exactly one of O_RDONLY or O_RDWR and

any of the other flags listed here:

O_RDONLY Open the object for read access. A shared memory object opened in this way can be mmap(2)ed only for read (PROT_READ) access.

O_RDWR Open the object for read-write access.

O_CREAT Create the shared memory object if it does not exist. The user and group ownership of the object are taken from the corresponding effective IDs of the calling process, and the object's permission bits are set ac? cording to the low-order 9 bits of mode, except that those bits set in the process file mode creation mask (see umask(2)) are cleared for the new object. A set of macro constants which can be used to define mode is listed in open(2). (Symbolic definitions of these constants can be obtained by including <sys/stat.h>.)

A new shared memory object initially has zero length?the size of the ob? ject can be set using ftruncate(2). The newly allocated bytes of a shared memory object are automatically initialized to 0.

O_EXCL If O_CREAT was also specified, and a shared memory object with the given name already exists, return an error. The check for the existence of the object, and its creation if it does not exist, are performed atomi? cally.

O_TRUNC If the shared memory object already exists, truncate it to zero bytes.

Definitions of these flag values can be obtained by including <fcntl.h>.

On successful completion shm_open() returns a new file descriptor referring to the shared memory object. This file descriptor is guaranteed to be the lowest-numbered file descriptor not previously opened within the process. The FD_CLOEXEC flag (see fcntl(2)) is set for the file descriptor.

The file descriptor is normally used in subsequent calls to ftruncate(2) (for a newly created object) and mmap(2). After a call to mmap(2) the file descriptor may be closed without affecting the memory mapping.

The operation of shm_unlink() is analogous to unlink(2): it removes a shared memory object name, and, once all processes have unmapped the object, de-allocates and de? stroys the contents of the associated memory region. After a successful shm_un? link(), attempts to shm_open() an object with the same name fail (unless O_CREAT was specified, in which case a new, distinct object is created).

RETURN VALUE

On success, shm_open() returns a nonnegative file descriptor. On failure, shm_open() returns -1. shm_unlink() returns 0 on success, or -1 on error.

ERRORS

On failure, errno is set to indicate the cause of the error. Values which may ap? pear in errno include the following:

EACCES Permission to shm_unlink() the shared memory object was denied.

EACCES Permission was denied to shm_open() name in the specified mode, or O_TRUNC was specified and the caller does not have write permission on the object.

EEXIST Both O_CREAT and O_EXCL were specified to shm_open() and the shared memory object specified by name already exists.

EINVAL The name argument to shm_open() was invalid.

EMFILE The per-process limit on the number of open file descriptors has been reached.

ENAMETOOLONG

The length of name exceeds PATH_MAX.

ENFILE The system-wide limit on the total number of open files has been reached.

ENOENT An attempt was made to shm_open() a name that did not exist, and O_CREAT was not specified.

ENOENT An attempt was to made to shm_unlink() a name that does not exist.

VERSIONS

These functions are provided in glibc 2.2 and later.

ATTRIBUTES

For an explanation of the terms used in this section, see attributes(7).

??

?Interface ? Attribute ? Value ?

??

?shm_open(), shm_unlink() ? Thread safety ? MT-Safe locale ?

??

CONFORMING TO

POSIX.1-2001, POSIX.1-2008.

POSIX.1-2001 says that the group ownership of a newly created shared memory object is set to either the calling process's effective group ID or "a system default

group ID". POSIX.1-2008 says that the group ownership may be set to either the calling process's effective group ID or, if the object is visible in the filesys? tem, the group ID of the parent directory.

NOTES

POSIX leaves the behavior of the combination of O_RDONLY and O_TRUNC unspecified. On Linux, this will successfully truncate an existing shared memory object?this may not be so on other UNIX systems.

The POSIX shared memory object implementation on Linux makes use of a dedicated tmpfs(5) filesystem that is normally mounted under /dev/shm.

SEE ALSO

close(2), fchmod(2), fchown(2), fcntl(2), fstat(2), ftruncate(2), memfd_create(2), mmap(2), open(2), umask(2), shm_overview(7)

COLOPHON

This page is part of release 5.05 of the Linux man-pages project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man-pages/.