NAME

shmat, shmdt - System V shared memory operations

SYNOPSIS

#include <sys/types.h>
#include <sys/shm.h>

void *shmat(int *shmid***, const void *** *shmaddr***, int** *shmflg***);**

int shmdt(const void * *shmaddr***);**

DESCRIPTION

shmat()

shmat() attaches the System V shared memory segment identified by *shmid* to the address space of the calling process. The attaching address is specified by *shmaddr* with one of the following criteria:

* If *shmaddr* is NULL, the system chooses a suitable (unused) address at which to attach the segment.
* If *shmaddr* isn't NULL and **SHM_RND** is specified in *shmflg*, the attach occurs at the address equal to *shmaddr* rounded down to the nearest multiple of **SHMLBA**.
* Otherwise, *shmaddr* must be a page-aligned address at which the attach occurs.

In addition to **SHM_RND**, the following flags may be specified in the *shmflg* bit-mask argument:

SHM_EXEC (Linux-specific; since Linux 2.6.9)
Allow the contents of the segment to be executed. The caller must have execute permission on the segment.

SHM_RDONLY
Attach the segment for read-only access. The process must have read permission for the segment. If this flag is not specified, the segment is attached for read and write access, and the process must have read and write permission for the segment. There is no notion of a write-only shared memory segment.

SHM_REMAP (Linux-specific)
This flag specifies that the mapping of the segment should replace any existing mapping in the range starting at *shmaddr* and continuing for the size of the segment. (Normally, an **EINVAL** error would result if a mapping already exists in this address range.) In this case, *shmaddr* must not be NULL.

The brk(2) value of the calling process is not altered by the attach. The segment will automatically be detached at process exit. The same segment may be attached as a read and as a read-write one, and more than once, in the process's address space.

A successful **shmat**() call updates the members of the *shmid_ds* structure (see shmctl(2)) associated with the shared memory segment as follows:

shm_atime is set to the current time.

shm_lpid is set to the process-ID of the calling process.

shm_nattch is incremented by one.

shmdt()

shmdt() detaches the shared memory segment located at the address specified by *shmaddr* from the address space of the calling process. The to-be-detached segment must be currently attached with *shmaddr* equal to the value returned by the attaching **shmat**() call.

On a successful **shmdt**() call, the system updates the members of the *shmid_ds* structure associated with the shared memory segment as follows:

shm_dtime is set to the current time.

shm_lpid is set to the process-ID of the calling process.

shm_nattch is decremented by one. If it becomes 0 and the segment is marked for deletion, the segment is deleted.

RETURN VALUE

On success, **shmat**() returns the address of the attached shared memory segment; on error, *(void *) -1* is returned, and *errno* is set to indicate the cause of the error.

On success, **shmdt**() returns 0; on error -1 is returned, and *errno* is set to indicate the cause of the error.

ERRORS

When **shmat**() fails, *errno* is set to one of the following:

EACCES
The calling process does not have the required permissions for the requested attach type, and does not have the **CAP_IPC_OWNER** capability in the user namespace that governs its IPC namespace.

EIDRM
shmid points to a removed identifier.

EINVAL
Invalid *shmid* value, unaligned (i.e., not page-aligned and **SHM_RND** was not specified) or invalid *shmaddr* value, or can't attach segment at *shmaddr*, or **SHM_REMAP** was specified and *shmaddr* was NULL.

ENOMEM
Could not allocate memory for the descriptor or for the page tables.

When **shmdt**() fails, *errno* is set as follows:

EINVAL
There is no shared memory segment attached at *shmaddr*; or, *shmaddr* is not aligned on a page boundary.

CONFORMING TO

POSIX.1-2001, POSIX.1-2008, SVr4.

In SVID 3 (or perhaps earlier), the type of the *shmaddr* argument was changed from *char* * into *const void* *, and the returned type of **shmat**() from *char* * into *void* *.

NOTES

After a fork(2), the child inherits the attached shared memory segments.

After an execve(2), all attached shared memory segments are detached from the process.

Upon _exit(2), all attached shared memory segments are detached from the process.

Using **shmat**() with *shmaddr* equal to NULL is the preferred, portable way of attaching a shared memory segment. Be aware that the shared memory segment attached in this way may be attached at different addresses in different processes. Therefore, any pointers maintained within the shared memory must be made relative (typically to the starting address of the segment), rather than absolute.

On Linux, it is possible to attach a shared memory segment even if it is already marked to be deleted. However, POSIX.1 does not specify this behavior and many other implementations do not support it.

The following system parameter affects **shmat**():

SHMLBA
Segment low boundary address multiple. When explicitly specifying an attach address in a call to **shmat**(), the caller should ensure that the address is a multiple of this value. This is necessary on some architectures, in order either to ensure good CPU cache performance or to ensure that different attaches of the same segment have consistent views within the CPU cache. **SHMLB A** is normally some multiple of the system page size (on many Linux architectures, it is the same as the system page size).

The implementation places no intrinsic per-process limit on the number of shared memory segments (**SHMSEG**).

SEE ALSO

brk(2), mmap(2), shmctl(2), shmget(2), capabilities(7), shm_overview(7), svipc(7)

COLOPHON

This page is part of release 4.10 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/.