NAME

shmget – allocates a System V shared memory segment

SYNOPSIS

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

int shmget(key_t *key***, size_t** *size***, int** *shmflg***);**

DESCRIPTION

shmget() returns the identifier of the System V shared memory segment associated with the value of the argument *key*. It may be used either to obtain the identifier of a previously created shared memory segment (when *shmflg* is zero and *key* does not have the value **IPC_PRIVATE**), or to create a new set.

A new shared memory segment, with size equal to the value of *size* rounded up to a multiple of **PAGE_SIZE**, is created if *key* has the value **IPC_PRIVATE** or *key* isn't **IPC_PRIVATE**, no shared memory segment corresponding to *key* exists, and **IPC_CREAT** is specified in *shmflg*.

If *shmflg* specifies both **IPC_CREAT** and **IPC_EXCL** and a shared memory segment already exists for *key*, then **shmget**() fails with *errno* set to **EEXIST**. (This is analogous to the effect of the combination **O_CREAT | O_EXCL** for **open**(2).)

The value *shmflg* is composed of:

IPC_CREAT
Create a new segment. If this flag is not used, then **shmget**() will find the segment associated with *key* and check to see if the user has permission to access the segment.

IPC_EXCL This flag is used with **IPC_CREAT** to ensure that this call creates the segment. If the segment already exists, the call fails.

SHM_HUGETLB (since Linux 2.6)
Allocate the segment using "huge pages." See the Linux kernel source file *Documentation/admin-guide/mm/hugetlbpage.rst* for further information.

SHM_HUGE_2MB, **SHM_HUGE_1GB** (since Linux 3.8)
Used in conjunction with **SHM_HUGETLB** to select alternative hugetlb page sizes (respectively, 2 MB and 1 GB) on systems that support multiple hugetlb page sizes.

More generally, the desired huge page size can be configured by encoding the base-2 logarithm of the desired page size in the six bits at the offset **SHM_HUGE_SHIFT**. Thus, the above two constants are defined as:

```
#define SHM_HUGE_2MB    (21 << SHM_HUGE_SHIFT)
#define SHM_HUGE_1GB    (30 << SHM_HUGE_SHIFT)
```

For some additional details, see the discussion of the similarly named constants in **mmap**(2).

SHM_NORESERVE (since Linux 2.6.15)
This flag serves the same purpose as the **mmap**(2) **MAP_NORESERVE** flag. Do not reserve swap space for this segment. When swap space is reserved, one has the guarantee that it is possible to modify the segment. When swap space is not reserved one might get **SIGSEGV** upon a write if no physical memory is available. See also the discussion of the file */proc/sys/vm/overcommit_memory* in **proc**(5).

In addition to the above flags, the least significant 9 bits of *shmflg* specify the permissions granted to the owner, group, and others. These bits have the same format, and the same meaning, as the *mode* argument of **open**(2). Presently, execute permissions are not used by the system.

When a new shared memory segment is created, its contents are initialized to zero values, and its associated data structure, *shmid_ds* (see **shmctl**(2)), is initialized as follows:

shm_perm.cuid and *shm_perm.uid* are set to the effective user ID of the calling process.

shm_perm.cgid and *shm_perm.gid* are set to the effective group ID of the calling process.

The least significant 9 bits of *shm_perm.mode* are set to the least significant 9 bit of *shmflg*.

shm_segsz is set to the value of *size*.

shm_lpid, *shm_nattch*, *shm_atime*, and *shm_dtime* are set to 0.

shm_ctime is set to the current time.

If the shared memory segment already exists, the permissions are verified, and a check is made to see if it is marked for destruction.

RETURN VALUE

On success, a valid shared memory identifier is returned. On error, −1 is returned, and *errno* is set to indicate the error.

ERRORS

On failure, *errno* is set to one of the following:

EACCES
The user does not have permission to access the shared memory segment, and does not have the **CAP_IPC_OWNER** capability in the user namespace that governs its IPC namespace.

EEXIST
IPC_CREAT and **IPC_EXCL** were specified in *shmflg*, but a shared memory segment already exists for *key*.

EINVAL
A new segment was to be created and *size* is less than **SHMMIN** or greater than **SHMMAX**.

EINVAL
A segment for the given *key* exists, but *size* is greater than the size of that segment.

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

ENOENT
No segment exists for the given *key*, and **IPC_CREAT** was not specified.

ENOMEM
No memory could be allocated for segment overhead.

ENOSPC
All possible shared memory IDs have been taken (**SHMMNI**), or allocating a segment of the requested *size* would cause the system to exceed the system-wide limit on shared memory (**SHMALL**).

EPERM
The **SHM_HUGETLB** flag was specified, but the caller was not privileged (did not have the **CAP_IPC_LOCK** capability).

CONFORMING TO

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

SHM_HUGETLB and **SHM_NORESERVE** are Linux extensions.

NOTES

The inclusion of *<sys/types.h>* and *<sys/ipc.h>* isn't required on Linux or by any version of POSIX. However, some old implementations required the inclusion of these header files, and the SVID also documented their inclusion. Applications intended to be portable to such old systems may need to include these header files.

IPC_PRIVATE isn't a flag field but a *key_t* type. If this special value is used for *key*, the system call ignores all but the least significant 9 bits of *shmflg* and creates a new shared memory segment.

Shared memory limits

The following limits on shared memory segment resources affect the **shmget**() call:

SHMALL
System-wide limit on the total amount of shared memory, measured in units of the system page size.

On Linux, this limit can be read and modified via */proc/sys/kernel/shmall*. Since Linux 3.16, the default value for this limit is:

ULONG_MAX - 2^24

The effect of this value (which is suitable for both 32-bit and 64-bit systems) is to impose no limitation on allocations. This value, rather than **ULONG_MAX**, was chosen as the default to prevent some cases where historical applications simply raised the existing limit without first checking its current value. Such applications would cause the value to overflow if the limit was set at **ULONG_MAX**.

From Linux 2.4 up to Linux 3.15, the default value for this limit was:

SHMMAX / PAGE_SIZE * (SHMMNI / 16)

If **SHMMAX** and **SHMMNI** were not modified, then multiplying the result of this formula by the page size (to get a value in bytes) yielded a value of 8 GB as the limit on the total memory used by all shared memory segments.

SHMMAX
Maximum size in bytes for a shared memory segment.

On Linux, this limit can be read and modified via */proc/sys/kernel/shmmax*. Since Linux 3.16, the default value for this limit is:

ULONG_MAX - 2^24

The effect of this value (which is suitable for both 32-bit and 64-bit systems) is to impose no limitation on allocations. See the description of **SHMALL** for a discussion of why this default value (rather than **ULONG_MAX**) is used.

From Linux 2.2 up to Linux 3.15, the default value of this limit was 0x2000000 (32 MB).

Because it is not possible to map just part of a shared memory segment, the amount of virtual memory places another limit on the maximum size of a usable segment: for example, on i386 the largest segments that can be mapped have a size of around 2.8 GB, and on x86-64 the limit is around 127 TB.

SHMMIN
Minimum size in bytes for a shared memory segment: implementation dependent (currently 1 byte, though **PAGE_SIZE** is the effective minimum size).

SHMMNI
System-wide limit on the number of shared memory segments. In Linux 2.2, the default value for this limit was 128; since Linux 2.4, the default value is 4096.

On Linux, this limit can be read and modified via */proc/sys/kernel/shmmni*.

The implementation has no specific limits for the per-process maximum number of shared memory segments (**SHMSEG**).

Linux notes

Until version 2.3.30, Linux would return **EIDRM** for a **shmget**() on a shared memory segment scheduled for deletion.

BUGS

The name choice **IPC_PRIVATE** was perhaps unfortunate, **IPC_NEW** would more clearly show its function.

SEE ALSO

memfd_create(2), **shmat**(2), **shmctl**(2), **shmdt**(2), **ftok**(3), **capabilities**(7), **shm_overview**(7), **sysvipc**(7)

COLOPHON

This page is part of release 5.02 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/.