**NAME**

semget − get a System V semaphore set identifier

**SYNOPSIS**

**#include <sys/types.h>**
**#include <sys/ipc.h>**
**#include <sys/sem.h>**

**int semget(key_t** *key***, int** *nsems***, int** *semflg***);**

**DESCRIPTION**

The **semget**() system call returns the System V semaphore set identifier associated with the argument *key*. It may be used either to obtain the identifier of a previously created semaphore set (when *semflg* is zero and *key* does not have the value **IPC_PRIVATE**), or to create a new set.

A new set of *nsems* semaphores is created if *key* has the value **IPC_PRIVATE** or if no existing semaphore set is associated with *key* and **IPC_CREAT** is specified in *semflg*.

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

Upon creation, the least significant 9 bits of the argument *semflg* define the permissions (for owner, group and others) for the semaphore set. These bits have the same format, and the same meaning, as the *mode* argument of **open**(2) (though the execute permissions are not meaningful for semaphores, and write permissions mean permission to alter semaphore values).

When creating a new semaphore set, **semget**() initializes the set's associated data structure, *semid_ds* (see **semctl**(2)), as follows:

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

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

> The least significant 9 bits of *sem_perm.mode* are set to the least significant 9 bits of *semflg*.

> *sem_nsems* is set to the value of *nsems*.

> *sem_otime* is set to 0.

> *sem_ctime* is set to the current time.

The argument *nsems* can be 0 (a don't care) when a semaphore set is not being created. Otherwise, *nsems* must be greater than 0 and less than or equal to the maximum number of semaphores per semaphore set (**SEMMSL**).

If the semaphore set already exists, the permissions are verified.

**RETURN VALUE**

If successful, the return value will be the semaphore set identifier (a nonnegative integer), otherwise, −1 is returned, with *errno* indicating the error.

**ERRORS**

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

**EACCES**

> A semaphore set exists for *key*, but the calling process does not have permission to access the set, 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 *semflg*, but a semaphore set already exists for *key*.

**EINVAL**

*nsems* is less than 0 or greater than the limit on the number of semaphores per semaphore set (**SEMMSL**).

**EINVAL**

A semaphore set corresponding to *key* already exists, but *nsems* is larger than the number of semaphores in that set.

**ENOENT**

No semaphore set exists for *key* and *semflg* did not specify **IPC_CREAT**.

**ENOMEM**

A semaphore set has to be created but the system does not have enough memory for the new data structure.

**ENOSPC**

A semaphore set has to be created but the system limit for the maximum number of semaphore sets (**SEMMNI**), or the system wide maximum number of semaphores (**SEMMNS**), would be exceeded.

## CONFORMING TO

SVr4, POSIX.1-2001.

## 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 *semflg* and creates a new semaphore set (on success).

### Semaphore initialization

The values of the semaphores in a newly created set are indeterminate. (POSIX.1-2001 and POSIX.1-2008 are explicit on this point, although POSIX.1-2008 notes that a future version of the standard may require an implementation to initialize the semaphores to 0.) Although Linux, like many other implementations, initializes the semaphore values to 0, a portable application cannot rely on this: it should explicitly initialize the semaphores to the desired values.

Initialization can be done using **semctl**(2) **SETVAL** or **SETALL** operation. Where multiple peers do not know who will be the first to initialize the set, checking for a nonzero *sem_otime* in the associated data structure retrieved by a **semctl**(2) **IPC_STAT** operation can be used to avoid races.

### Semaphore limits

The following limits on semaphore set resources affect the **semget**() call:

**SEMMNI**

System-wide limit on the number of semaphore sets. On Linux systems before version 3.19, the default value for this limit was 128. Since Linux 3.19, the default value is 32,000. On Linux, this limit can be read and modified via the fourth field of */proc/sys/kernel/sem*.

**SEMMSL**

Maximum number of semaphores per semaphore ID. On Linux systems before version 3.19, the default value for this limit was 250. Since Linux 3.19, the default value is 32,000. On Linux, this limit can be read and modified via the first field of */proc/sys/kernel/sem*.

**SEMMNS**

System-wide limit on the number of semaphores: policy dependent (on Linux, this limit can be read and modified via the second field of */proc/sys/kernel/sem*). Note that the number of semaphores system-wide is also limited by the product of **SEMMSL** and **SEMMNI**.

**BUGS**

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

**SEE ALSO**

      **semctl**(2), **semop**(2), **ftok**(3), **capabilities**(7), **sem_overview**(7), **sysvipc**(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/.