NAME

semget − get a 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 semaphore set identifier associated with the argument *key*. 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).

The values of the semaphores in a newly created set are indeterminate. (POSIX.1-2001 is explicit on this point.) 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.

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 non-negative 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.

EEXIST
A semaphore set exists for *key* and *semflg* specified both **IPC_CREAT** and **IPC_EXCL**.

EINVAL
nsems is less than 0 or greater than the limit on the number of semaphores per semaphore set (**SEMMSL**), or a semaphore set corresponding to *key* already exists, and *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
IPC_PRIVATE isn't a flag field but a *key_t* type. If this special value is used for *key*, the system call ignores everything but the least significant 9 bits of *semflg* and creates a new semaphore set (on success).

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

SEMMNI
System wide maximum number of semaphore sets: policy dependent (on Linux, this limit can be read and modified via the fourth field of */proc/sys/kernel/sem*).

SEMMSL
Maximum number of semaphores per semid: implementation dependent (on Linux, this limit can be read and modified via the first field of */proc/sys/kernel/sem*).

SEMMNS
System wide maximum number of semaphores: policy dependent (on Linux, this limit can be read and modified via the second field of */proc/sys/kernel/sem*). Values greater than **SEMMSL * SEMMNI** makes it irrelevant.

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

The semaphores in a set are not initialized by **semget**(). In order to initialize the semaphores, **semctl**(2) must be used to perform a **SETVAL** or a **SETALL** operation on the semaphore set. (Where multiple peers do not know who will be the first to initialize the set, checking for a non-zero *sem_otime* in the associated data structure retrieved by a **semctl**(2) **IPC_STAT** operation can be used to avoid races.)

SEE ALSO
semctl(2), **semop**(2), **ftok**(3), **capabilities**(7), **sem_overview**(7), **svipc**(7)

COLOPHON
This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.