NAME

semctl − semaphore control operations

SYNOPSIS

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

int semctl(int *semid***, int** *semnum***, int** *cmd***, ...);**

DESCRIPTION

semctl() performs the control operation specified by *cmd* on the semaphore set identified by *semid*, or on the *semnum*-th semaphore of that set. (The semaphores in a set are numbered starting at 0.)

This function has three or four arguments, depending on *cmd*. When there are four, the fourth has the type *union semun*. The *calling program* must define this union as follows:

```
union semun {
    int              val;    /* Value for SETVAL */
    struct semid_ds *buf;    /* Buffer for IPC_STAT, IPC_SET */
    unsigned short  *array;  /* Array for GETALL, SETALL */
    struct seminfo  *__buf;  /* Buffer for IPC_INFO
                                (Linux-specific) */
};
```

The *semid_ds* data structure is defined in <*sys/sem.h*> as follows:

```
struct semid_ds {
    struct ipc_perm sem_perm;  /* Ownership and permissions */
    time_t          sem_otime; /* Last semop time */
    time_t          sem_ctime; /* Last change time */
    unsigned short  sem_nsems; /* No. of semaphores in set */
};
```

The *ipc_perm* structure is defined in <*sys/ipc.h*> as follows (the highlighted fields are settable using **IPC_SET**):

```
struct ipc_perm {
    key_t          __key; /* Key supplied to semget(2) */
    uid_t          uid;   /* Effective UID of owner */
    gid_t          gid;   /* Effective GID of owner */
    uid_t          cuid;  /* Effective UID of creator */
    gid_t          cgid;  /* Effective GID of creator */
    unsigned short mode;  /* Permissions */
    unsigned short __seq; /* Sequence number */
};
```

Valid values for *cmd* are:

IPC_STAT
Copy information from the kernel data structure associated with *semid* into the *semid_ds* structure pointed to by *arg.buf*. The argument *semnum* is ignored. The calling process must have read permission on the semaphore set.

IPC_SET Write the values of some members of the *semid_ds* structure pointed to by *arg.buf* to the kernel data structure associated with this semaphore set, updating also its *sem_ctime* member. The following members of the structure are updated: *sem_perm.uid*, *sem_perm.gid*, and (the least significant 9 bits of) *sem_perm.mode*. The effective UID of the calling process must match the owner (*sem_perm.uid*) or creator (*sem_perm.cuid*) of the semaphore set, or the caller must be privileged. The argument *semnum* is ignored.

IPC_RMID
Immediately remove the semaphore set, awakening all processes blocked in **semop**(2) calls on the set (with an error return and *errno* set to **EIDRM**). The effective user ID of the calling process must match the creator or owner of the semaphore set, or the caller must be privileged. The argument *semnum* is ignored.

IPC_INFO (Linux-specific)
Returns information about system-wide semaphore limits and parameters in the structure pointed to by *arg.__buf*. This structure is of type *seminfo*, defined in <*sys/sem.h*> if the **_GNU_SOURCE** feature test macro is defined:

```
struct  seminfo {
    int semmap;  /* Number of entries in semaphore
                    map; unused within kernel */
    int semmni;  /* Maximum number of semaphore sets */
    int semmns;  /* Maximum number of semaphores in all
                    semaphore sets */
    int semmnu;  /* System-wide maximum number of undo
                    structures; unused within kernel */
    int semmsl;  /* Maximum number of semaphores in a
                    set */
    int semopm;  /* Maximum number of operations for
                    semop(2) */
    int semume;  /* Maximum number of undo entries per
                    process; unused within kernel */
    int semusz;  /* Size of struct sem_undo */
    int semvmx;  /* Maximum semaphore value */
    int semaem;  /* Max. value that can be recorded for
                    semaphore adjustment (SEM_UNDO) */
};
```

The *semmsl*, *semmns*, *semopm*, and *semmni* settings can be changed via */proc/sys/kernel/sem*; see **proc**(5) for details.

SEM_INFO (Linux-specific)
Returns a *seminfo* structure containing the same information as for **IPC_INFO**, except that the following fields are returned with information about system resources consumed by semaphores: the *semusz* field returns the number of semaphore sets that currently exist on the system; and the *semaem* field returns the total number of semaphores in all semaphore sets on the system.

SEM_STAT (Linux-specific)
Returns a *semid_ds* structure as for **IPC_STAT**. However, the *semid* argument is not a semaphore identifier, but instead an index into the kernel's internal array that maintains information about all semaphore sets on the system.

GETALL Return **semval** (i.e., the current value) for all semaphores of the set into *arg.array*. The argument *semnum* is ignored. The calling process must have read permission on the semaphore set.

GETNCNT
The system call returns the value of **semncnt** (i.e., the number of processes waiting for the value of this semaphore to increase) for the *semnum*−th semaphore of the set (i.e., the number of processes waiting for an increase of **semval** for the *semnum*−th semaphore of the set). The calling process must have read permission on the semaphore set.

GETPID The system call returns the value of **sempid** for the *semnum*−th semaphore of the set (i.e., the PID of the process that executed the last **semop**(2) call for the *semnum*−th semaphore of the set). The calling process must have read permission on the semaphore set.

GETVAL The system call returns the value of **semval** for the *semnum*−th semaphore of the set. The calling process must have read permission on the semaphore set.

GETZCNT
The system call returns the value of **semzcnt** (i.e., the number of processes waiting for the value of this semaphore to become zero) for the *semnum*−th semaphore of the set (i.e., the number of processes waiting for **semval** of the *semnum*−th semaphore of the set to become 0). The calling process must have read permission on the semaphore set.

SETALL Set **semval** for all semaphores of the set using *arg.array*, updating also the *sem_ctime* member of the *semid_ds* structure associated with the set. Undo entries (see **semop**(2)) are cleared for altered semaphores in all processes. If the changes to semaphore values would permit blocked **semop**(2) calls in other processes to proceed, then those processes are woken up. The argument *semnum* is ignored. The calling process must have alter (write) permission on the semaphore set.

SETVAL Set the value of **semval** to *arg.val* for the *semnum*−th semaphore of the set, updating also the *sem_ctime* member of the *semid_ds* structure associated with the set. Undo entries are cleared for altered semaphores in all processes. If the changes to semaphore values would permit blocked **semop**(2) calls in other processes to proceed, then those processes are woken up. The calling process must have alter permission on the semaphore set.

RETURN VALUE

On failure **semctl**() returns −1 with *errno* indicating the error.

Otherwise the system call returns a non-negative value depending on *cmd* as follows:

GETNCNT	the value of **semncnt**.
GETPID	the value of **sempid**.
GETVAL	the value of **semval**.
GETZCNT	the value of **semzcnt**.
IPC_INFO	the index of the highest used entry in the kernel's internal array recording information about all semaphore sets. (This information can be used with repeated **SEM_STAT** operations to obtain information about all semaphore sets on the system.)
SEM_INFO	As for **IPC_INFO**.
SEM_STAT	the identifier of the semaphore set whose index was given in *semid*.

All other *cmd* values return 0 on success.

ERRORS

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

EACCES
The argument *cmd* has one of the values **GETALL**, **GETPID**, **GETVAL**, **GETNCNT**, **GETZCNT**, **IPC_STAT**, **SEM_STAT**, **SETALL**, or **SETVAL** and the calling process does not have the required permissions on the semaphore set and does not have the **CAP_IPC_OWNER** capability.

EFAULT
The address pointed to by *arg.buf* or *arg.array* isn't accessible.

EIDRM
The semaphore set was removed.

EINVAL
Invalid value for *cmd* or *semid*. Or: for a **SEM_STAT** operation, the index value specified in *semid* referred to an array slot that is currently unused.

EPERM
The argument *cmd* has the value **IPC_SET** or **IPC_RMID** but the effective user ID of the calling process is not the creator (as found in *sem_perm.cuid*) or the owner (as found in *sem_perm.uid*) of the semaphore set, and the process does not have the **CAP_SYS_ADMIN** capability.

ERANGE
The argument *cmd* has the value **SETALL** or **SETVAL** and the value to which **semval** is to be set (for some semaphore of the set) is less than 0 or greater than the implementation limit **SEMVMX**.

CONFORMING TO

SVr4, POSIX.1-2001.

NOTES

The **IPC_INFO**, **SEM_STAT** and **SEM_INFO** operations are used by the **ipcs**(8) program to provide information on allocated resources. In the future these may modified or moved to a /proc file system interface.

Various fields in a *struct semid_ds* were typed as *short* under Linux 2.2 and have become *long* under Linux 2.4. To take advantage of this, a recompilation under glibc-2.1.91 or later should suffice. (The kernel distinguishes old and new calls by an **IPC_64** flag in *cmd*.)

In some earlier versions of glibc, the *semun* union was defined in <*sys/sem.h*>, but POSIX.1-2001 requires that the caller define this union. On versions of glibc where this union is *not* defined, the macro **_SEM_SEMUN_UNDEFINED** is defined in <*sys/sem.h*>.

The following system limit on semaphore sets affects a **semctl**() call:

SEMVMX
Maximum value for **semval**: implementation dependent (32767).

For greater portability it is best to always call **semctl**() with four arguments.

SEE ALSO

ipc(2), **semget**(2), **semop**(2), **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/.