Rocky Enterprise Linux 9.2 Manual Pages on command 'semctl.2'

$ man semctl.2

SEMCTL(2)   Linux Programmer's Manual   SEMCTL(2)

NAME

semctl - System V 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 Sys?

tem V 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 pro?

gram 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; /* Creation time/time of last
                                      modification via semctl() */
        unsigned long   sem_nsems; /* No. of semaphores in set */
    };
```

The fields of the semid_ds structure are as follows:

sem_perm   This is an ipc_perm structure (see below) that specifies the access permissions on the semaphore set.

sem_otime Time of last semop(2) system call.

sem_ctime Time of creation of semaphore set or time of last semctl() IPCSET, SETVAL, or SETALL operation.

sem_nsems Number of semaphores in the set. Each semaphore of the set is referenced by a nonnegative integer ranging from 0 to sem_nsems-1.

The ipc_perm structure is defined 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 */
    };
```

The least significant 9 bits of the mode field of the ipc_perm struc?

ture define the access permissions for the shared memory segment. The permission bits are as follows:

0400 Read by user

0200 Write by user

0040 Read by group

0020 Write by group

0004 Read by others

0002 Write by others

In effect, "write" means "alter" for a semaphore set. Bits 0100, 0010, and 0001 (the execute bits) are unused by the system.

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 ig? nored.

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)

Return information about system-wide semaphore limits and param?
eters 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)

Return 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 num?
ber of semaphores in all semaphore sets on the system.

SEM_STAT (Linux-specific)

Return 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.

SEM_STAT_ANY (Linux-specific, since Linux 4.17)

Return a seminfo structure containing the same information as for SEM_STAT. However, sem_perm.mode is not checked for read access for semid meaning that any user can employ this operation (just as any user may read /proc/sysvipc/sem to obtain the same information).

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

Return the semncnt value for the semnum-th semaphore of the set (i.e., the number of processes waiting for the semaphore's value to increase). The calling process must have read permission on the semaphore set.

GETPID Return the sempid value for the semnum-th semaphore of the set. This is the PID of the process that last performed an operation on that semaphore (but see NOTES). The calling process must have read permission on the semaphore set.

GETVAL Return semval (i.e., the semaphore value) for the semnum-th sem? aphore of the set. The calling process must have read permis? sion on the semaphore set.

GETZCNT

Return the semzcnt value for the semnum-th semaphore of the set (i.e., the number of processes waiting for the semaphore value to become 0). The calling process must have read permission on the semaphore set.

SETALL Set the semval values 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 semaphore value (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 nonnegative 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 ar?
ray recording information about all semaphore sets. (This in?
formation can be used with repeated SEM_STAT or SEM_STAT_ANY op?
erations 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.

SEM_STAT_ANY

as for SEM_STAT.

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, SEM_STAT_ANY, SETALL, or

SETVAL and the calling process does not have the required per?

missions on the semaphore set and does not have the

CAP_IPC_OWNER capability in the user namespace that governs its

IPC namespace.

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 ef?

fective 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

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

POSIX.1 specifies the sem_nsems field of the semid_ds structure as hav?

ing the type unsigned short, and the field is so defined on most other

systems. It was also so defined on Linux 2.2 and earlier, but, since

Linux 2.4, the field has the type unsigned long.

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.

The IPC_INFO, SEM_STAT, and SEM_INFO operations are used by the ipcs(1) program to provide information on allocated resources. In the future these may modified or moved to a /proc filesystem 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 re?compilation 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 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.

The sempid value

POSIX.1 defines sempid as the "process ID of [the] last operation" on a semaphore, and explicitly notes that this value is set by a successful semop(2) call, with the implication that no other interface affects the sempid value.

While some implementations conform to the behavior specified in POSIX.1, others do not. (The fault here probably lies with POSIX.1 inasmuch as it likely failed to capture the full range of existing im?plementation behaviors.) Various other implementations also update sempid for the other operations that update the value of a semaphore: the SETVAL and SETALL operations, as well as the semaphore adjustments performed on process termination as a consequence of the use of the SEM_UNDO flag (see semop(2)).

Linux also updates sempid for SETVAL operations and semaphore adjust?

ments. However, somewhat inconsistently, up to and including Linux 4.5, the kernel did not update sempid for SETALL operations. This was rectified in Linux 4.6.

EXAMPLES

See shmop(2).

SEE ALSO

ipc(2), semget(2), semop(2), capabilities(7), sem_overview(7), sysvipc(7)

COLOPHON

This page is part of release 5.10 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/.

Linux 2020-12-21 SEMCTL(2)