

## NAME

semget - get set of semaphores

## 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()` function returns the semaphore identifier associated with `_k_e_y`.

A semaphore identifier and associated data structure and set containing `_n_s_e_m_s` semaphores (see `intro(3)`) are created for `_k_e_y` if one of the following is true:

- + o `_k_e_y` is equal to `IPC_PRIVATE`.
- + o `_k_e_y` does not already have a semaphore identifier associated with it, and `(_s_e_m_f_l_g & IPC_CREAT)` is true.

On creation, the data structure associated with the new semaphore identifier is initialized as follows:

- + o `sem_perm.cuid`, `sem_perm.uid`, `sem_perm.cgid`, and `sem_perm.gid` are set equal to the effective user ID and effective group ID, respectively, of the calling process.
- + o The access permission bits of `sem_perm.mode` are set equal to the access permission bits of `_s_e_m_f_l_g`.
- + o `sem_nsems` is set equal to the value of `_n_s_e_m_s`.
- + o `sem_otime` is set equal to 0 and `sem_ctime` is set equal to the current time.

## RETURN VALUES

Upon successful completion, a non-negative integer representing a semaphore identifier is returned. Otherwise, -1 is returned and `errno` is set to indicate the error.

## ERRORS

The `semget()` function will fail if:

EACCES

A semaphore identifier exists for `_k_e_y`, but operation permission (see `intro(3)`) as specified by the low-order 9 bits of `_s_e_m_f_l_g` would not be granted.

SunOS 5.8      Last change: 30 Nov 1993      1

System Calls      `semget(2)`

#### EEXIST

A semaphore identifier exists for `_k_e_y` but both `(_s_e_m_f_l_g&IPC_CREAT)` and `(_s_e_m_f_l_g&IPC_EXCL)` are both true.

#### EINVAL

The `_n_s_e_m_s` argument is either less than or equal to 0 or greater than the system-imposed limit; or a semaphore identifier exists for `_k_e_y`, but the number of semaphores in the set associated with it is less than `_n_s_e_m_s` and `_n_s_e_m_s` is not equal to 0.

#### ENOENT

A semaphore identifier does not exist for `_k_e_y` and `(_s_e_m_f_l_g&IPC_CREAT)` is false.

#### ENOSPC

A semaphore identifier is to be created but the system-imposed limit on the maximum number of allowed semaphores or semaphore identifiers system-wide would be exceeded.

System Calls      `semctl(2)`

#### 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

The `semctl()` function provides a variety of semaphore control operations as specified by `_c_m_d`. The fourth argument is optional, depending upon the operation requested. If required, it is of type `union semun`, which must be explicitly declared by the application program.

```
union semun {
    int          val;
    struct semid_ds *buf;
    ushort_t     *array;
} arg ;
```

The permission required for a semaphore operation is given as `{_t_o_k_e_n}`, where `_t_o_k_e_n` is the type of permission needed. The types of permission are interpreted as follows:

```
00400    READ by user
00200    ALTER by user
00040    READ by group
00020    ALTER by group
00004    READ by others
00002    ALTER by others
```

See the Semaphore Operation Permissions subsection of the DEFINITIONS section of `intro(2)` for more information. The following semaphore operations as specified by `_c_m_d` are executed with respect to the semaphore specified by `_s_e_m_i_d` and `_s_e_m_n_u_m`.

### GETVAL

Return the value of `semval` (see `intro(2)`). {READ}

### SETVAL

Set the value of `semval` to `_a_r_g_v_a_l`. {ALTER} When this command is successfully executed, the `semadj` value corresponding to the specified semaphore in all processes is cleared.

### GETPID

Return the value of (int) `sempid`. {READ}

### GETNCNT

Return the value of `semncnt`. {READ}

### GETZCNT

Return the value of `semzcnt`. {READ}

The following operations return and set, respectively, every `semval` in the set of semaphores.

### GETALL

Place `semvals` into array pointed to by `_a_r_g.array`. {READ}

## SETALL

Set semvals according to the array pointed to by `_a_r_g.array`. {ALTER}. When this cmd is successfully executed, the `semadj` values corresponding to each specified semaphore in all processes are cleared.

The following operations are also available.

## IPC\_STAT

Place the current value of each member of the data structure associated with `_s_e_m_i_d` into the structure pointed to by `_a_r_g.buf`. The contents of this structure are defined in `intro(2)`. {READ}

## IPC\_SET

Set the value of the following members of the data structure associated with `_s_e_m_i_d` to the corresponding value found in the structure pointed to by `_a_r_g.buf`:

```
sem_perm.uid
sem_perm.gid
sem_perm.mode /* access permission bits only */
```

This command can be executed only by a process that has an effective user ID equal to either that of super-user, or to the value of `sem_perm.cuid` or `sem_perm.uid` in the data structure associated with `_s_e_m_i_d`.

## IPC\_RMID

Remove the semaphore identifier specified by `_s_e_m_i_d` from the system and destroy the set of semaphores and data structure associated with it. This command can only be executed by a process that has an effective user ID equal to either that of super-user, or to the value of `sem_perm.cuid` or `sem_perm.uid` in the data structure associated with `_s_e_m_i_d`.

## RETURN VALUES

Upon successful completion, the value returned depends on `_c_m_d` as follows:

### GETVAL

the value of `semval`

### GETPID

the value of (int) `sempid`

### GETNCNT

the value of `semncnt`

### GETZCNT

the value of `semzcnt`

All other successful completions return 0; otherwise, -1 is

returned and errno is set to indicate the error.

## ERRORS

The semctl() function will fail if:

### EACCES

Operation permission is denied to the calling process (see intro(2)).

### EINVAL

The `_s_e_m_ i_ d` argument is not a valid semaphore identifier; the `_s_e_m_n_u_m` argument is less than 0 or greater than `sem_nsems - 1`; or the `_c_m_d` argument is not a valid command or is `IPC_SET` and `sem_perm.uid` or `sem_perm.gid` is not valid.

**EPERM** The `_c_m_d` argument is equal to `IPC_RMID` or `IPC_SET` and the effective user of the calling process is not super-user, or `_c_m_d` is equal to the value of `sem_perm.cuid` or `sem_perm.uid` in the data structure associated with `_s_e_m_ i_ d`.

### EOVERFLOW

The `_c_m_d` argument is `IPC_STAT` and `_u_ i_ d` or `_g_ i_ d` is too large to be stored in the structure pointed to by `_a_r_g_b_u_f`.

### ERANGE

The `_c_m_d` argument is `SETVAL` or `SETALL` and the value to which `semval` is to be set is greater than the system imposed maximum.

## System Calls

semop(2)

## NAME

semop - semaphore operations

## SYNOPSIS

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

```
int semop(int semid, struct sembuf *sops, size_t nsops);
```

## DESCRIPTION

The `semop()` function is used to perform atomically an array of semaphore operations on the set of semaphores associated with the semaphore identifier specified by `_s_e_m_ i_ d`. The `_s_o_p_s` argument is a pointer to the array of semaphore-operation structures. The `_n_s_o_p_s` argument is the number of such structures in the array.

Each `sembuf` structure contains the following members:

```
short sem_num; /* semaphore number */
short sem_op; /* semaphore operation */
short sem_flg; /* operation flags */
```

Each semaphore operation specified by `sem_op` is performed on the corresponding semaphore specified by `sem_id` and `sem_num`. The permission required for a semaphore operation is given as `{_t_o_k_e_n}`, where `_t_o_k_e_n` is the type of permission needed. The types of permission are interpreted as follows:

```
00400 READ by user
00200 ALTER by user
00040 READ by group
00020 ALTER by group
00004 READ by others
00002 ALTER by others
```

See the `_S_e_m_a_p_h_o_r_e_O_p_e_r_a_t_i_o_n` and `_P_e_r_m_i_s_s_i_o_n_s` section of `intro(3)` for more information.

The `sem_op` member specifies one of three semaphore operations:

1. The `sem_op` member is a negative integer; `{ALTER}`

- + o If `semval` (see `intro(3)`) is greater than or equal to the absolute value of `sem_op`, the absolute value of `sem_op` is subtracted from `semval`. Also, if `(sem_flg & SEM_UNDO)` is true, the absolute value of `sem_op` is added to the calling process's `semadj`

value (see `exit(2)`) for the specified semaphore.

- + o If `semval` is less than the absolute value of `sem_op` and `(sem_flg & IPC_NOWAIT)` is true, `semop()` returns immediately.

- + o If `semval` is less than the absolute value of `sem_op` and `(sem_flg & IPC_NOWAIT)` is false, `semop()` increments the `semncnt` associated with the specified semaphore and suspends execution of the calling process until one of the following conditions occur:

- + o The value of `semval` becomes greater than or equal to the absolute value of `sem_op`. When this occurs, the value of `semncnt` associated with the specified semaphore is decremented, the absolute value of `sem_op` is subtracted from `semval` and, if `(sem_flg & SEM_UNDO)` is

true, the absolute value of `sem_op` is added to the calling process's `semadj` value for the specified semaphore.

- + o The `_s_e_m_i_d` for which the calling process is awaiting action is removed from the system (see `semctl(2)`). When this occurs, `errno` is set to `EIDRM` and `-1` is returned.
- + o The calling process receives a signal that is to be caught. When this occurs, the value of `semncnt` associated with the specified semaphore is decremented, and the calling process resumes execution in the manner prescribed in `signal(3C)`.

2. The `sem_op` member is a positive integer; `{ALTER}`

The value of `sem_op` is added to `semval` and, if `(_s_e_m_f_l_g&SEM_UNDO)` is true, the value of `sem_op` is subtracted from the calling process's `semadj` value for the specified semaphore.

3. The `sem_op` member is 0; `{READ}`

- + o If `semval` is 0, `semop()` returns immediately.
- + o If `semval` is not equal to 0 and `(_s_e_m_f_l_g&IPC_NOWAIT)` is true, `semop()` returns immediately.
- + o If `semval` is not equal to 0 and `(_s_e_m_f_l_g&IPC_NOWAIT)` is false, `semop()` increments

the `semzcnt` associated with the specified semaphore and suspends execution of the calling process until one of the following occurs:

- + o The value of `semval` becomes 0, at which time the value of `semzcnt` associated with the specified semaphore is decremented.
- + o The `_s_e_m_i_d` for which the calling process is awaiting action is removed from the system. When this occurs, `errno` is set to `EIDRM` and `-1` is returned.
- + o The calling process receives a signal that is to be caught. When this occurs, the value of `semzcnt` associated with the specified semaphore is decremented, and the calling process resumes execution in the manner prescribed in `signal(3C)`.

Upon successful completion, the value of `sempid` for each semaphore specified in the array pointed to by `_s_o_p_s` is set

to the process ID of the calling process.

#### RETURN VALUES

Upon successful completion, 0 is returned. Otherwise, -1 is returned and errno is set to indicate the error.

#### ERRORS

The semop() function will fail if:

**E2BIG** The `_n_s_o_p_s` argument is greater than the system-imposed maximum.

#### **EACCES**

Operation permission is denied to the calling process (see intro(3)).

#### **EAGAIN**

The operation would result in suspension of the calling process but (`_s_e_m_f_l_g&IPC_NOWAIT`) is true.

#### **EFAULT**

The `_s_o_p_s` argument points to an illegal address.

**EFBIG** The value of `sem_num` is less than 0 or greater than or equal to the number of semaphores in the set associated with `_s_e_m_i_d`.

**EIDRM** A `_s_e_m_i_d` was removed from the system.

**EINTR** A signal was received.

#### **EINVAL**

The `_s_e_m_i_d` argument is not a valid semaphore identifier, or the number of individual semaphores for which the calling process requests a `SEM_UNDO` would exceed the limit.

#### **ENOSPC**

The limit on the number of individual processes requesting an `SEM_UNDO` would be exceeded.

#### **ERANGE**

An operation would cause a `semval` or a `semadj` value to overflow the system-imposed limit.