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

semop( ) is used to perform atomically an array of semaphore operations on the set of semaphores associated with the semaphore identifier specified by *semid*. *sops* is a pointer to the array of semaphore-operation structures. *nsops* is the number of such structures in the array. The contents of each structure includes 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 *semid* and *sem_num*. The permission required for a semaphore operation is given as {*token*}, where *token* 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* section of **intro**(2) for more information.

sem_op specifies the {ALTER} token if its value is negative or positive, and the {READ} token if its value is zero. Depending on the value of *sem_op,* the following may occur:

sem_op **is a negative integer;** {ALTER}

- If **semval** (see **intro**(2)) 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.
- If **semval** is less than the absolute value of *sem_op* and (*sem_flg***&IPC_NOWAIT**) is true, **semop( )** returns immediately.
- 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:

- **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.
- The *semid* for which the calling process is awaiting action is removed from the system (see **semctl**(2)). When this occurs, **errno** is set equal to **EIDRM**, and a value of −1 is returned.
- 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).

sem_op **is a positive integer;** {ALTER}

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

sem_op **is zero;** {READ}

- If **semval** is zero, **semop( )** returns immediately.
- If **semval** is not equal to zero and (*sem_flg*&**IPC_NOWAIT**) is true, **semop( )** returns immediately.
- If **semval** is not equal to zero and (*sem_flg*&**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:
  - **semval** becomes zero, at which time the value of **semzcnt** associated with the specified semaphore is decremented.
  - The *semid* for which the calling process is awaiting action is removed from the system. When this occurs, **errno** is set equal to **EIDRM**, and a value of −1 is returned.
  - 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).

RETURN VALUES

Upon successful completion, a value of zero is returned. Otherwise, a value of −1 is returned and **errno** is set to indicate the error.

ERRORS

semop( ) fails if one or more of the following are true for any of the semaphore operations specified by *sops*:

E2BIG *nsops* is greater than the system-imposed maximum.

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

EAGAIN The operation would result in suspension of the calling process but (*sem_flg*&**IPC_NOWAIT**) is true.

EFAULT *sops* points to an illegal address.

EFBIG *sem_num* is less than zero or greater than or equal to the number of semaphores in the set associated with *semid*.

EIDRM **semop( )** A *semid* was removed from the system.

EINTR A signal was received.

EINVAL *semid* 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.

Upon successful completion, the value of **sempid** for each semaphore specified in the array pointed to by *sops* is set equal to the process ID of the calling process.

SEE ALSO

ipcs(1), **intro**(2), **exec**(2), **exit**(2), **fork**(2), **semctl**(2), **semget**(2)