NAME

semop - atomic array of operations on a semaphore set

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

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

int
semop(*int semid, struct sembuf *array, size_t nops*);

DESCRIPTION

The **semop**() system call atomically performs the array of operations indicated by *array* on the semaphore set indicated by *semid*. The length of *array* is indicated by *nops*. Each operation is encoded in a *struct sembuf*, which is defined as follows:

```
struct sembuf {
        u_short sem_num;        /* semaphore # */
        short   sem_op;         /* semaphore operation */
        short   sem_flg;        /* operation flags */
};
```

For each element in *array*, *sem_op* and *sem_flg* determine an operation to be performed on semaphore number *sem_num* in the set. The values SEM_UNDO and IPC_NOWAIT may be *OR*'ed into the *sem_flg* member in order to modify the behavior of the given operation.

The operation performed depends as follows on the value of *sem_op*:

- When *sem_op* is positive and the process has alter permission, the semaphore's value is incremented by *sem_op*'s value. If SEM_UNDO is specified, the semaphore's adjust on exit value is decremented by *sem_op*'s value. A positive value for *sem_op* generally corresponds to a process releasing a resource associated with the semaphore.

- The behavior when *sem_op* is negative and the process has alter permission, depends on the current value of the semaphore:

  - If the current value of the semaphore is greater than or equal to the absolute value of *sem_op*,

then the value is decremented by the absolute value of *sem_op*. If SEM_UNDO is specified, the semaphore's adjust on exit value is incremented by the absolute value of *sem_op*.

- If the current value of the semaphore is less than the absolute value of *sem_op*, one of the following happens:
  - If IPC_NOWAIT was specified, then **semop**() returns immediately with a return value of EAGAIN.
  - Otherwise, the calling process is put to sleep until one of the following conditions is satisfied:
    - Some other process removes the semaphore with the IPC_RMID option of semctl(2). In this case, **semop**() returns immediately with a return value of EIDRM.
    - The process receives a signal that is to be caught. In this case, the process will resume execution as defined by sigaction(2).
    - The semaphore's value is greater than or equal to the absolute value of *sem_op*. When this condition becomes true, the semaphore's value is decremented by the absolute value of *sem_op*, the semaphore's adjust on exit value is incremented by the absolute value of *sem_op*.

  A negative value for *sem_op* generally means that a process is waiting for a resource to become available.

- When *sem_op* is zero and the process has read permission, one of the following will occur:
  - If the current value of the semaphore is equal to zero then **semop**() can return immediately.
  - If IPC_NOWAIT was specified, then **semop**() returns immediately with a return value of EAGAIN.
  - Otherwise, the calling process is put to sleep until one of the following conditions is satisfied:
    - Some other process removes the semaphore with the IPC_RMID option of semctl(2). In this case, **semop**() returns immediately with a return value of EIDRM.
    - The process receives a signal that is to be caught. In this case, the process will resume execution as defined by sigaction(2).

- The semaphore’s value becomes zero.

For each semaphore a process has in use, the kernel maintains an "adjust on exit" value, as alluded to earlier. When a process exits, either voluntarily or involuntarily, the adjust on exit value for each semaphore is added to the semaphore’s value. This can be used to ensure that a resource is released if a process terminates unexpectedly.

RETURN VALUES

The **semop**() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **semop**() system call will fail if:

[EINVAL]	No semaphore set corresponds to *semid*, or the process would exceed the system-defined limit for the number of per-process SEM_UNDO structures.
[EACCES]	Permission denied due to mismatch between operation and mode of semaphore set.
[EAGAIN]	The semaphore’s value would have resulted in the process being put to sleep and IPC_NOWAIT was specified.
[E2BIG]	Too many operations were specified. [SEMOPM]
[EFBIG]	*sem_num* was not in the range of valid semaphores for the set.
[EIDRM]	The semaphore set was removed from the system.
[EINTR]	The **semop**() system call was interrupted by a signal.
[ENOSPC]	The system SEM_UNDO pool [SEMMNU] is full.
[ERANGE]	The requested operation would cause either the semaphore’s current value [SEMVMX] or its adjust on exit value [SEMAEM] to exceed the system-imposed limits.

SEE ALSO

semctl(2), semget(2), sigaction(2)

BUGS

The **semop**() system call may block waiting for memory even if IPC_NOWAIT was specified.