NAME

sem_open – initialize/open a named semaphore

SYNOPSIS

cc [*flag* ...] *file* ... **–lposix4** [*library* ...]

#include <semaphore.h>

sem_t ∗**sem_open(const char** ∗*name*, **int** *oflag*,
    /∗ **unsigned long** *mode*, **unsigned int** *value* ∗**/ ...);**

typedef struct {
 ...
} sem_t;   /∗opaque POSIX.4 semaphore∗/

MT-LEVEL

MT-Safe

DESCRIPTION

sem_open( ) establishes a connection to a semaphore, *name,* returning the address of the semaphore to the calling process (or LWP or thread) for subsequent calls to **sem_wait**(3R), **sem_trywait**(3R), **sem_post**(3R), and **sem_close**(3R). The semaphore remains usable by this process until the semaphore is closed.

name points to a string naming a semaphore object. The *name* argument should conform to the construction rules for a pathname. If a process makes multiple successful calls to **sem_open( )** with the same value for *name,* the same semaphore address will be returned for each such successful call, provided that there have been no calls to **sem_unlink**(3R) for this semaphore.

oflag determines whether the semaphore is created or merely accessed by the call to **sem_open( ).** The three valid values for *oflag* are **0, O_CREAT**, or the bitwise inclusive OR of **O_CREAT** and **O_EXCL**. Setting the *oflag* bits to **O_CREAT** will create the semaphore if it does not already exist. Setting both **O_CREAT** and **O_EXCL** will fail if the semaphore already exists. The check for the existence of the semaphore and the creation of the semaphore if it does not exist is atomic with respect to other processes executing **sem_open( ).** After the semaphore named *name* has been created by **sem_open( )** with the **O_CREAT** flag, other processes can connect to this semaphore by calling **sem_open( )** with the same value of *name,* and nobits set in *oflag.*

Using the **O_CREAT** flag requires a third and a fourth argument: *mode* and *value.* The semaphore is created with an initial count of *value. value* must be less than or equal to **{SEM_VALUE_MAX}.** The semaphore's user ID acquires the effective user ID of the process; the semaphore's group ID is set to a system default group ID or to the effective group ID of the process. The semaphore's permission bits is set to the value of *mode*, modified by clearing all bits set in the file creation mask of the process (see **umask**(2)).

RETURN VALUES

If successful, **sem_open( )** returns the address of the semaphore, otherwise it returns **-1** and sets **errno** to indicate the error condition.

ERRORS

EACCES The named semaphore exists and the **O_RDWR** permissions are denied, or the named semaphore does not exist and permission to create the named semaphore is denied.

EEXIST **O_CREAT** and **O_EXCL** are set and the named semaphore already exists.

EINTR **sem_open( )** was interrupted by a signal.

EINVAL *name* is not a valid name.

O_CREAT was set in *oflag* and *value* is greater than **{SEM_VALUE_MAX}**.

EMFILE The number of open semaphore descriptors in this process exceeds **{SEM_NSEMS_MAX}**.

The number of open file descriptors in this process exceeds **{OPEN_MAX}**.

ENAMETOOLONG
The string-length of *name* exceeds **{PATH_MAX}**, or a pathname component is longer than **{NAME_MAX}** while **_POSIX_NO_TRUNC** is in effect.

ENFILE The system file table is full.

ENOENT **O_CREAT** is not set and the named semaphore does not exist.

ENOSPC There is insufficient space for the creation of the new named semaphore.

ENOSYS **sem_open( )** is not supported by this implementation.

SEE ALSO

exec(2), **exit**(2), **umask**(2), **sysconf**(3C), **sem_close**(3R), **sem_post**(3R), **sem_unlink**(3R), **sem_wait**(3R)

BUGS

In Solaris 2.5, these functions always return −**1** and set **errno** to **ENOSYS,** because this release does not support the Semaphores option. It is our intention to provide support for these interfaces in future releases.