NAME

sema, **sema_init**, **sema_destroy**, **sema_post**, **sema_wait**, **sema_timedwait**, **sema_trywait**, **sema_value** - kernel counting semaphore

SYNOPSIS

#include <sys/types.h>
#include <sys/lock.h>
#include <sys/sema.h>

void
sema_init(*struct sema *sema, int value, const char *description*);

void
sema_destroy(*struct sema *sema*);

void
sema_post(*struct sema *sema*);

void
sema_wait(*struct sema *sema*);

int
sema_timedwait(*struct sema *sema, int timo*);

int
sema_trywait(*struct sema *sema*);

int
sema_value(*struct sema *sema*);

DESCRIPTION

Counting semaphores provide a mechanism for synchronizing access to a pool of resources. Unlike mutexes, semaphores do not have the concept of an owner, so they can also be useful in situations where one thread needs to acquire a resource, and another thread needs to release it. Each semaphore has an integer value associated with it. Posting (incrementing) always succeeds, but waiting (decrementing) can only successfully complete if the resulting value of the semaphore is greater than or equal to zero.

Semaphores should not be used where mutexes and condition variables will suffice. Semaphores are a more complex synchronization mechanism than mutexes and condition variables, and are not as efficient.

Semaphores are created with **sema_init**(), where *sema* is a pointer to space for a *struct sema*, *value* is the initial value of the semaphore, and *description* is a pointer to a null-terminated character string that describes the semaphore. Semaphores are destroyed with **sema_destroy**(). A semaphore is posted (incremented) with **sema_post**(). A semaphore is waited on (decremented) with **sema_wait**(), **sema_timedwait**(), or **sema_trywait**(). The *timo* argument to **sema_timedwait**() specifies the minimum time in ticks to wait before returning with failure. **sema_value**() is used to read the current value of the semaphore.

RETURN VALUES

The **sema_value**() function returns the current value of the semaphore.

If decrementing the semaphore would result in its value being negative, **sema_trywait**() returns 0 to indicate failure. Otherwise, a non-zero value is returned to indicate success.

The **sema_timedwait**() function returns 0 if waiting on the semaphore succeeded; otherwise a non-zero error code is returned.

ERRORS

The **sema_timedwait**() function will fail if:

[EWOULDBLOCK] Timeout expired.

SEE ALSO

condvar(9), locking(9), mtx_pool(9), mutex(9), rwlock(9), sx(9)