NAME

sema, p, v, test, post, block, setsem, rdsem, lock, unlock, tlock, noulk − semaphore operations

SYNOPSIS

p (sema)
v (sema)
test (sema)
post (sema)
block (sema)
setsem (sema,value)
rdsem (sema)
lock (sema)
unlock (sema)
tlock (sema)
noulk ( )

DESCRIPTION

The indicated function is performed on the specified semaphore. Semaphores are assigned on a system-wide basis. By convention the file **/usr/include/sema.h** contains define symbols for the usage of semaphores. Also by convention, semaphore numbers less than zero are reserved for system programs such as the line printer spooling system.

The various semaphore operations are defined as follows:

post causes all users doing a *block* on the specified semaphore to be awakened. As a side effect, the semaphore is incremented.

block causes the current user to roadblock until a subsequent *post* on the specified semaphore.

p causes the current user to roadblock if the specified semaphore's value is zero until it becomes nonzero. If the semaphore's value is nonzero, the semaphore is decremented and the user is not roadblocked.

v causes the specified semaphore to be incremented.

test causes the specified semaphore to be decremented if the current value is nonzero.

rdsem returns the current value of the specified semaphore.

setsem sets the specified semaphore to the given value.

lock roadblocks if the specified semaphore is nonzero until it becomes zero. Once the semaphore is zero, the negative of the current process' pid is stored in the semaphore and *lock* returns a zero.

unlock sets the specified semaphore to zero if its current value is the negative of the current process' pid; an error is returned otherwise.

tlock is like *lock* except the current process is not roadblocked if the semaphore's value is nonzero. Instead, the value is returned.

noulk prevents the system from automatically unlocking any semaphores which may be locked by the current process at termination time.

In all cases, if the function is successfully performed, the system returns the old semaphore value (in R0).

Users of semaphores should be wary of the interaction between caught signals and the use of *p*, *block*, and *lock*. If a signal is caught while waiting for a semaphore, the call to the semaphore primative will return with a −1 error and the error number of EINTR. The base level routine

should then call the primative again if this is desired.

The use of the various semaphore primatives in an intermixed manner may produce undefined results. In particular a single semaphore should be used with only one of the following tuples: *block-post*, *p-v-test*, or *lock-unlock-tlock*. *Rdsem* may be used on any semaphore at any time. *Setsem* should only be used to set a semaphore to an inital value.

The system will only automatically *unlock* those semaphores which have been *locked* or *tlocked*.

A counting semaphore (*p-v-test*) may only assume values between zero and 32767. If an overflow occurs the new value is set to one.

SEE ALSO

sema(1)

DIAGNOSTICS

From C, a −1 value indicates an error.

ASSEMBLER

(semas = 63.; not in assembler)
(new value in R0)
sys semas; func; sema
(old value in R0)