NAME

lock – semaphores (USG Version)

SYNOPSIS

(lock = 62.; not in assembler)
sys lock; function; number

allocsem(number, type)
freesem(number)

lock(number)
unlock(number)
tlock(number)

p(number)
v(number)
test(number)

DESCRIPTION

Semaphores are useful for the synchronization of processes and controlling access to resources. There are two versions of semaphores available for these purposes: the lock-unlock type and the P-V (counting) type. A semaphore must be allocated by a process before it may be used; the semaphore's access permission and type are established by the first process to allocate it and remain in effect until freed by every process that has it allocated. Allocated semaphores are inherited across *fork* but not across *exec*.

From assembly language, the low byte of the *function* argument specifies the request type.

0 Lock the semaphore. If it is already locked, suspend execution until it is unlocked (Lock Operation).

1 Unlock the semaphore. Awaken any processes waiting for it to be unlocked (Unlock Operation).

2 Lock the semaphore if it is not already locked. Return immediately if it is locked (Conditional Lock Operation).

3 Decrement the semaphore's value by one if it is positive. If the value is zero, suspend execution until it becomes positive, then decrement it (P Operation).

4 Increment the semaphore's value by one and, if necessary, awaken any processes waiting for the value to become positive (V Operation).

5 Decrement the value of the semaphore by one, but never below zero. Do not suspend execution if the value was already zero (Conditional P operation).

6 Allocate a semaphore to the process. The high byte of *function* indicates how the semaphore is to be used and what class of processes are also eligible to also allocate and use the semaphore. A value of 0, 1, or 2 allocates a lock-unlock semaphore with an access class of anyone, same userid only, and same groupid only, respectively. A value of 3, 4, or 5 allocates a P-V semaphore for use by anyone, same userid only, and same groupid only, respectively.

7 Deallocate (free) the indicated semaphore(s).

The *number* argument designates the semaphore number of the semaphore in question. In allocation requests, a negative value for *number* implies that an available semaphore from the system pool is to be chosen and allocated. A negative value for *number* in deallocation requests indicates that all of the semaphores allocated to the process are to be freed.

After successful calls, r0 contains the semaphore's previous value for lock, unlock, conditional lock, P, V, and conditional P operations; the number of the allocated semaphore for allocation requests; and zero for deallocation requests. Note that a *locked* semaphore's value is the pro-

LOCK(f)

cessid of the process that *locked* it; it is zero when not locked.

From C, *allocsem* is used to allocate semaphore *number* for the calling process. If *number* is −1, the first available semaphore from the system pool is allocated. The value of *type* has the same meaning as described above for *function 6*. Semaphore number *number* is deallocated by the *freesem* function. If *number* is −1, all allocated semaphores are freed. The number of the allocated semaphore is returned by *allocsem; freesem* returns zero.

The *lock, unlock,* and *tlock* functions perform lock, unlock, and conditional lock operations, respectively, on the lock-unlock semaphore *number*. Similarly, *p, v,* and *test* perform P, V, and conditional P operations on the P-V semaphore *number*.

Note: From C, if a *lock, unlock, tlock, p, v,* or *test* function is used without the semaphore having been previously allocated by the process, the semaphore will first be allocated before the requested operation is performed.

SEE ALSO

—

DIAGNOSTICS

The error bit (c-bit) is set for any one of a number of error conditions. Allocation errors occur if there are no semaphores available, a process tries to allocate too many semaphores, access permission is denied, or the intended usage is incorrect (e.g. allocating a P-V semaphore for lock-unlock use). When attempting to use a semaphore, an error occurs when the number is out of range, the semaphore was not previously allocated, or an invalid operation is attempted (e.g. doing a P operation on a lock-unlock semaphore). From C, a −1 return indicates an error.