NC STATE UNIVERSITY Message Board

Course: CSC501 - 001
Forum: PA2 discussion

Go to: Different Course | Forum List | Topic List

Author **Topic: Lock Deletion**

mjoshi Tue Feb 11 9:10 who

My question is regarding below line in the specification:
"As mentioned before, there is a slight problem with XINU semaphores. The way XINU handles sdelete may have undesirable effects if a semaphore is deleted while a process or processes are waiting on it. Examining the code for wait and sdelete, you will notice that sdelete readies processes waiting on a semaphore being deleted. So they will return from wait with OK."

Then we are asked to implement locks such that they return DELETED constant on deleting the lock.

If you look at the sdelete() code, you can see that it indeed readies all the processes which are in wait state but the implementation differs from the specification at one point. It sets the pwaitret value of processes to DELETED while making them ready.

Hence, when the process is rescheduled, it returns DELETED from wait() and not OK.

Is my understanding correct? If it is, then what do we exactly need to change regarding the DELETED constant? I guess it will work in similar way in case of locks.

Kindly confirm.

hcnguye3 Tue Feb 11 10:03 who

You must implement your lock system such that waiting on a lock will return a new constant DELETED instead of OK when returning due to a deleted lock

It is true that "It sets the pwaitret value of processes to DELETED while making them ready", but wait() still returns OK, please read wait.c again

gpingal Tue Feb 11 11:36 who

Hi ,

My understanding on the above:

When in sdelete for some semaphore , for all waiting processes on this semaphore , the pwaitret is assigned DELETED . and all of them are put into the ready list.

Now when one of them is resched , the excecution of this process starts right after the resched() call in wait . After which it returns with pwaitret .(which now is DELETED)

I am not sure what I am missing here . Kindly help.

nbadlan Tue Feb 11 14:14

Same Doubt . It already returns DELETED instead of OK. Please clarify , if possible with some example

fmkavath Tue Feb 11 14:57

Same_Doubt++;

nbadlan Tue Feb 11 15:02

Also : I tested with some code . This statement in the spec:
you will notice that sdelete readies processes waiting on a semaphore being deleted. So they will return from wait with OK .

In My test case , My Main process was waiting for the semaphore , and after deleting that sempahore , it returned the value -6 which is the constant for DELETED. Also I checked the status of main after deleting the semaphore using proctab[currpid].pwaitret , it showed -6 . Can you please clarify this too ?

hcnguye3 Tue Feb 11 22:28

This is the code of wait():

```
SYSCALL wait(int sem)
{
STATWORD ps;
struct sentry *sptr;
struct pentry *pptr;

disable(ps);
if (isbadsem(sem) || (sptr= &semaph[sem])->sstate==SFREE) {
restore(ps);
return(SYSERR);
}
```

```
if (--(sptr->semcnt) < 0) {
(pptr = &proctab[currpid])->pstate = PRWAIT;
pptr->psem = sem;
enqueue(currpid,sptr->sqtail);
pptr->pwaitret = OK;
resched();
restore(ps);
return pptr->pwaitret;
}
restore(ps);
return(OK);
}
```

Did you guys see that it returns OK?

adhanas Tue Feb 11 23:18 Who

But, the code in sdelete() sets the waiting procs' pwaitret to DELETED whenever a semaphore is deleted.

So, when the waiting procs are scheduled again, in wait(), they return pptr->pwaitret which is essentially DELETED now.

nbadlan Tue Feb 11 23:23 Who

I still dont understand. Will you be holding office hours tomorrow morning as usual ?

hcnguye3 Wed Feb 12 0:21 Who

```
pptr->pwaitret = OK ---> This is why it returns OK
resched();
restore(ps);
return pptr->pwaitret;
```

@ nbadlan: yes

adhanas Wed Feb 12 0:36 Who

```
SYSCALL sdelete(int sem)
{
/* blah */
sptr = &semaph[sem];
sptr->sstate = SFREE;
if (nonempty(sptr->sqhead)) {
while( (pid=getfirst(sptr->sqhead)) != EMPTY)
{
proctab[pid].pwaitret = DELETED; ---> the value is changed from OK to DELETED here
```

```
ready(pid,RESCHNO);
}
resched();
}
restore(ps);
return(OK);
}
```

So, when the control returns to wait(), the value in pwaitret is DELETED and that is returned to the caller.

hcnguye3 Wed Feb 12 2:50 Who

Sorry to make the confusion. Apparently, our version already has the fix for that semaphore problem (the semaphore problem described in this thread is only applied for some other versions such as http://www.xinu.cs.purdue.edu/files/Xinu-code-x86.tar.gz). And yes, you can borrow the same idea to design the lock.

skypa Fri Feb 14 16:18 Who

In continuation to this post,

Can you please help me understand the following,

When a semaphore is deleted, the state is set to SFREE. So in any other operation on this semaphore would return SYSERR because wait,signal has following check in the very beginning.

```
if (isbadsem(sem) || (sptr= &semaph[sem])->sstate==SFREE) {
restore(ps);
return(SYSERR);
}
```

I feel there is no case where DELETED is returned. its SYSERR that is returned.

I think I am missing some flow here where a DELETED is returned. Can you please help me understand this?

hcnguye3 Fri Feb 14 23:36 Who

That is only the sanity check. The actual WAIT happens when the process changes the state to PRWAIT and rescheduling happens (check wait() again)

Post New Topic Post a Reply

WolfWare - Message Board - last update 30-Aug-2001