CS 1550

Week 5 – Synchronization with xv6

Teaching Assistant

Maher Khan

(Slides credited to Henrique Potter)

Keep in mind the different qemu

- qemu with xv6 (Labs) - Refer to Lab 1 if needed!
- qemu-x86 i386 (Project 1 and 2)

SpinLocks are Busy waiting

```
void lock(struct spinlock * lk) {
    while(xchg(&lk->locked, 1) != 0)
      ;
}
```

Locks – Processes sharing CPU

```
void
acquiresleep(struct sleeplock *lk)
{
}
```

Locks – Processes sharing CPU

```
void
acquiresleep(struct sleeplock *lk)
{
    while (lk->locked) {
        sleep(lk, &lk->lk);
    }
}
```

Locks – Processes sharing CPU

```
void
acquiresleep(struct sleeplock *lk)
{
    acquire(&lk->lk);
    while (lk->locked) {
        sleep(lk, &lk->lk);
    }
    release(&lk->lk);
}
```

Locks – Processes sharing CPU

```
void
acquiresleep(struct sleeplock *lk)
{
    acquire(&lk->lk);
    while (lk->locked) {
        sleep(lk, &lk->lk);
    }
    release(&lk->lk);
}
```

```
void
releasesleep(struct sleeplock *lk)
{
    acquire(&lk->lk);

    wakeup(lk);

    release(&lk->lk);
}
```

Locks – Processes sharing CPU

```
void
acquiresleep(struct sleeplock *lk)
{
    acquire(&lk->lk);
    while (lk->locked) {
        sleep(lk, &lk->lk);
    }
    release(&lk->lk);
}
```

Locks – Processes sharing CPU

```
void
acquiresleep(struct sleeplock *lk)
{
    acquire(&lk->lk);
    while (lk->locked) {
        sleep(lk, &lk->lk);
    }
    release(&lk->lk);
}
```

```
void
sleep(void *chan, struct spinlock *lk)
{
  struct proc *p = myproc();
    …
  p->chan = chan;
  p->state = SLEEPING;

  sched();
    …
}
```

```
void
sleep(void *chan, struct spinlock *lk)
{
  struct proc *p = myproc();

  …
}
```

Process control Block

```
void
sleep(void *chan, struct spinlock *lk)
{
  struct proc *p = myproc();

  if(p == 0)
    panic("sleep");
  if(lk == 0)
    panic("sleep without lk");

 …
}
```

Control checks. This should be impossible

```
void
sleep(void *chan, struct spinlock *lk)
{
  struct proc *p = myproc();

  if(p == 0)
    panic("sleep");
  if(lk == 0)
    panic("sleep without lk");

  …
  p->chan = chan;
  p->state = SLEEPING;

  sched();
  …
}
```

Change process state to sleep. Call scheduler

```
void
sleep(void *chan, struct spinlock *lk)
{
  struct proc *p = myproc();

  if(p == 0)
    panic("sleep");
  if(lk == 0)
    panic("sleep without lk");


  acquire(&ptable.lock);

  p->chan = chan;
  p->state = SLEEPING;

  sched();
  p->chan = 0

  release(&ptable.lock);

}
```

Global Lock

```
void
sleep(void *chan, struct spinlock *lk)
{
  struct proc *p = myproc();

  if(p == 0)
    panic("sleep");
  if(lk == 0)
    panic("sleep without lk");


  acquire(&ptable.lock);

  p->chan = chan;
  p->state = SLEEPING;

  sched();
  p->chan = 0

  release(&ptable.lock);

}
```

Once sched() is called this process execution is held “at this line”

```
void
sleep(void *chan, struct spinlock *lk)
{
  struct proc *p = myproc();

  if(p == 0)
    panic("sleep");
  if(lk == 0)
    panic("sleep without lk");


  acquire(&ptable.lock);

  p->chan = chan;
  p->state = SLEEPING;

  sched();
  p->chan = 0

  release(&ptable.lock);

}
```

When process awakes his is removed from the sleep channel

Locks – Processes sharing CPU

```
void
releasesleep(struct sleeplock *lk)
{
    acquire(&lk->lk);

    wakeup(lk);

    release(&lk->lk);
}
```

Locks – Processes sharing CPU

```
void
releasesleep(struct sleeplock *lk)
{
    acquire(&lk->lk);

    wakeup(lk);

    release(&lk->lk);
}
```

```
void
wakeup(void *chan)
{
    acquire(&ptable.lock);

    wakeup_channel(chan);

    release(&ptable.lock);
}
```

Locks – Processes sharing CPU

```
static void
wakeup_channel(void *chan)
{
    struct proc *p;
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)

}
```

A processes is awaken from the sleeping channel

Locks – Processes sharing CPU

```
static void
wakeup_channel(void *chan)
{
    struct proc *p;
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
        if(p->state == SLEEPING && p->chan == chan)

}
```

Locks – Processes sharing CPU

```
static void
wakeup_channel(void *chan)
{
    struct proc *p;
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
        if(p->state == SLEEPING && p->chan == chan)
            p->state = RUNNABLE;
}
```

Locks – Processes sharing CPU

```
static void
wakeup_channel(void *chan)
{
    struct proc *p;
    for(p = ptable.proc; p < &ptable.proc[NPROC]; p++)
        if(p->state == SLEEPING && p->chan == chan)
            p->state = RUNNABLE;
}
```

Locks – Processes sharing CPU

- Who needs to be a syscall?
  - SpinLocks
  - Sleep/Wakeup

CS 1550 – Lab exercise 2

- **PROCESS SYNCHRONIZATION IN XV6**
  - **Due**: Friday, February 22, 2019 @11:59pm

  - Part 2 - step 5: user.h
    - Add declaration for init_lock()
      - void init_lock(struct spinlock *);
    - struct condvar;
    - struct spinlock;

  - Part 3 - step 8: defs.h
    - Add declaration for sleep1()

CS 1550

Week 5 – Synchronization with xv6

Teaching Assistant

Maher Khan

(Slides credited to Henrique Potter)