CS219 Lecture 27

Synchronization primitives.

Spinlock, sleep mutex lock

~ C-like pseudo code for mutex.

```
struct mutex {
    int id;        // identifier of mutex
    int lock;      // 0 — available
                   // 1 — not available
}

mutexlock (mutex * m)
{
    if (m->lock == 1)        } not-
        sleep (m->id);       } atomic
    m->lock = 1;
}
```

needs to sleep till lock available

```
struct mutex {
    int id;
    int lock;
    Spinlock L;
}
```

LQ3
18. Mar
2pm

Q2
25 - 830 am

Today
Post 2:30 pm
xv6 help session
SyncG Lab
KR 3x1
(near the lifts)

```
mutexlock (mutex *m)
{
    spinlock (m->l);
    while (m->lock == 1) {
        sleep(m->id, m->l);
    }
    m->lock = 1;
    spinunlock(m->l);
}
```

Spinunlock(L) ⇒ lock is released & then proc updates assume mutex-lock = 1

interrupt & releases mutex lock.

```
sleep (id, spinlock L);
{
    proc->state = WAITING;
    proc->chan = id;
        (condition)
    spinunlock (L);
    scheduler();
    spinlock(L);
}
```

```
mutexlock (mutex * m)
{
    spinlock (m->L);
    while (m->lock == 1) {
        sleep(m->id, m->L);
    }
    m->lock = 1;
    spinunlock(m->L);
}
```

```
mutexunlock( m)
{
    spinlock(m->L);
    m->lock = 0;
    wakeup(m->id);

    spinunlock(m->L);
}
```

```
sleep ( id, spinlock L);
{
    proc->state = WAITING;
    proc->chan = id;
        condition
    spinunlock (L);
    scheduler ();
    spinlock(L);
}
```

```
wakeup ( id )
{
    ∀ all processes p
    s.t p->chan == id
        p->state = READY;
        p->chan = 0;
}
```