CS 3214 lecture # 15 locking & condition variables

Multithreads

```
vtt;
```

```
mov  mem  %r
inc  %r
mov  %r  mem
```

critical section

not atomic!

"locking"

Mutual exclusion

Mutex

as if it was atomic ||

critical section

acquire lock

thread holds lock

...critical section...

release lock

```
int shared_global; // protected by mutex1
```

lock & unlock should be paired

don't have any return while holding lock (error)

don't use different locks for same data

don't hold locks while blocking

don't acquire if you already hold lock

mutual exclusion is just a piece of synchronization

→ signaling & waiting

wait(cv*, mutex*)

- assume lock held
- put caller to sleep & release lock (atomically)
- reacquire lock on wakeup

signal(cv*)

wake single waiting thread (if no waiters, just returns)

state variable is needed!

sv & cv go together
+ mutex

while vs. if

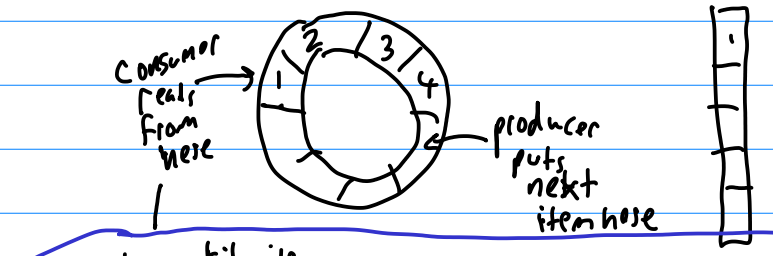
```
func {
    .lock(m)
     if(err)
      ...return;
     sv,cv.
     cv_wait(c,m);
    unlock(m)
}
```

"monitor pattern"

Java
Synchronized

Java concurrent

producer/consumer problem "Bounded buffer"



wait until items are available

Signal consumer if item available

wait if no more slots

Signal more slots are available

Unix pipe
device driver
web servers