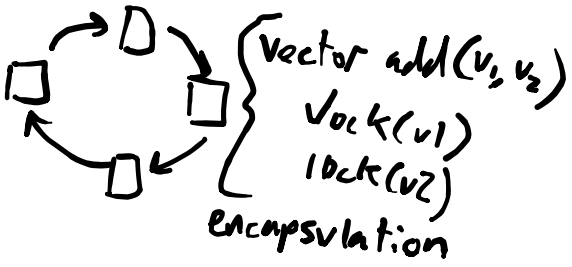
- Project due tonight (4/8) at 11:59 PM EDT
- Concurrency Exam
  - Github Classroom
  - txt file: edit directly with your answers
  - commit/push to remote
  - timed take home — Set a timer!
  - 75 min: time starts when you "accept assignment"
  - have until 4/15 9am EDT

Resources:
- Colbly office hours
- Reach out to me for Zoom

Today:
1. finish up deadlock
2. review concurrency unit
   → your questions

Deadlock



1. mutual exclusion
2. hold and wait
3. no preemption
4. circular wait

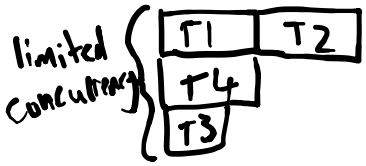
Solutions

1. prevent
   1. mutex: lock-free data structures → atomic H/W prim.
   2. hold-and-wait: grab all at same time "mega"
   3. mutex_trylock: "livelock"
   4. total order
2. Avoid
3. Detect + Recover
4. Ignore

Deadlock avoidance

Smart scheduler

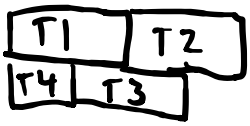
limited env.



locks \ threads	T1	T2	T3	T4
L1	y	y	n	n
L2	y	y	y	n

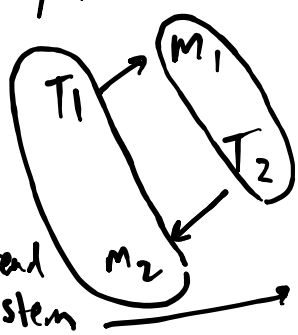
Dijkstra: Bankers Alg.

Not used much in practice



3. Detect + Recover

↳ Keeping track of "waits for" graph



detect cycle

→ tell human

→ restart - thread
- system

4. Ignore

DBMS

make sure state is consistent

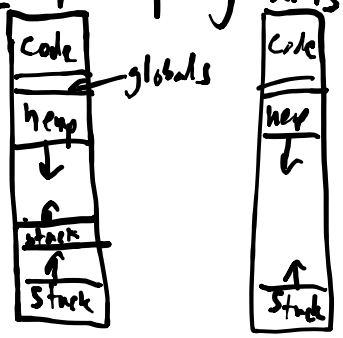
Review:

Concurrency: mutiple programs running at same time

↳ processes

↳ threads

↳ share address space



still have their own
- reg. (PC)
- stack

- interleaving I/O
- parallel programming
- OS

threads { cooperating on some task + want lots of implicit sharing

Why is concurrency hard?

programmer is not in control of scheduling

counter++;

```
mov addr, reg
add 1, reg
mov reg, addr
```

```
mov
add
mov
```

. critical section
. atomically

Think like the EVIL scheduler

- non-det:
- indeterminate
- race condition

one thread at a time

mutual exclusion (mutex) locks

```
pthread_mutex_lock(&m)
<crit. section>
pthread_mutex_unlock(&m)
```

How to build a lock

spinlock → flag: keep trying (spin) to check it's 0, then update

```
lock(l)
while(l==0)
    ;
l=1
```

H/w primitives:
- Test and set
- CAS
- xchg

Spinlocks: wasteful, unfair: starvation

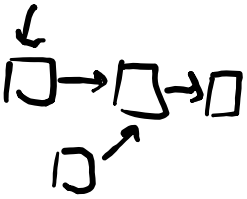
high vs. low contention

queues: park, unpark

(setpark)

How to apply locks to data structure

1. find critical section
2. add locks



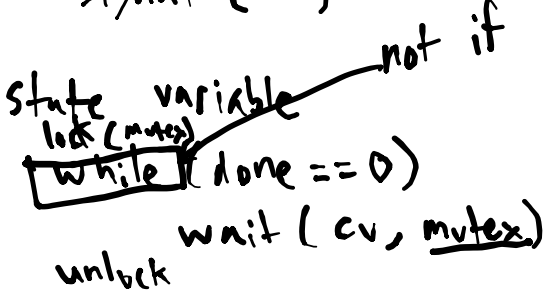
EVIL → set next
update head

Other synch primitives: CV's

join problem pthread_join();

wait(cv,m)

signal(cv)

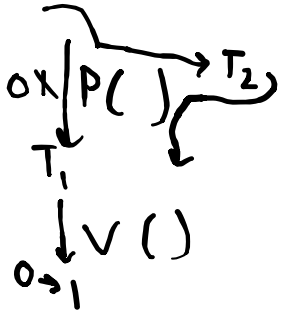


Mesa Semantics
- spurious wakeups

Producer/Consumer (Bounded buffer)
↳ cv for each

Semaphores: kind of do both mutex + cv

Binary semaphore (1)



P down wait if counter <=0
sleep(wait)
V up post ↳ dec counter
inc counter
wakeup
single waiter

Elegant solution to join problem init(0)

parent: wait()

child: post()

How to build semaphores from mutex/cv

Finally, problems.
↳ deadlock