Concurrency: Semaphores

⇒ Locks : Atomicity

⇒ Condition Variables : Ordering

e.g., fork/join, producer/consumer

P ... P (bounded)
{ { ⇒ [ ]
C ... C
{ {

new:
Semaphore
⇒ lock + CV in one

lock version:

```
lock(); //acquire
critical section  e.g. counter++;
unlock(); //release
```

semaphore version:

```
sem_wait (&lock);
critical section
sem_post (&lock);
```

{ acquire

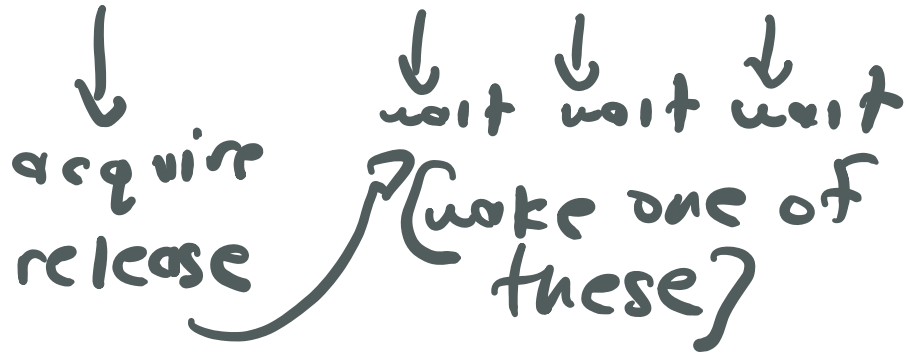
{ tries to acquire ⇒ wait

initialization: to use semaphore as lock
initialize it to 1

(binary semaphore)

(binary semaphore)

case:
many threads to acquie at "same time"




```
child () {          // child
  ...
  -> sem_post(&s);
}

main () {           // parent
  sem_init(&s, 0);
  pthread_create(child);
  -> sem_wait(&s);
}
```

parent runs first ✓

child runs first ✓

Midterm #2

"it always gets worse"

[Focus: Concurrency]

1 page "helper" page

Project 3a: parallel compression

partner via pizza (?)