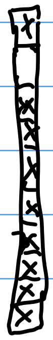
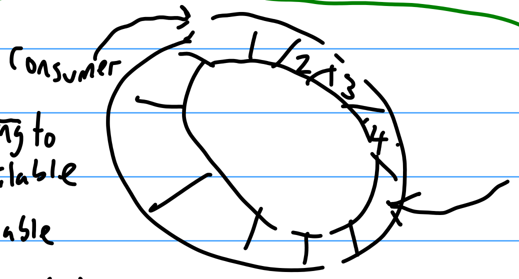
CS 3214 lecture # 16 more synchronization + semaphores

2 uses of synch.
- mutual exclusion mutex/lock pthread_mutex_lock
- signaling pthread_cond_wait



idx % LEN

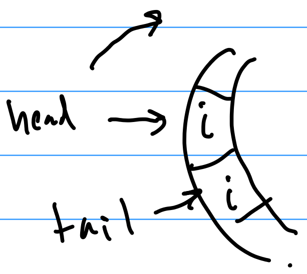
producer / consumer (bounded buffer)

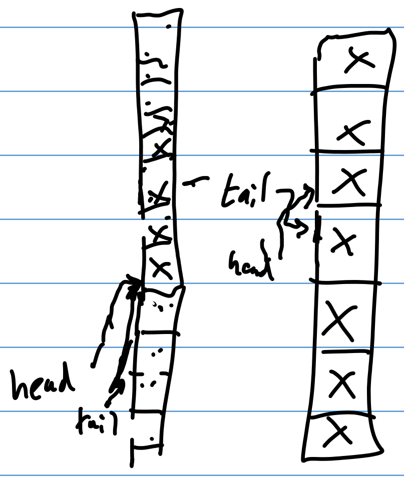


1. wait for ~~something~~ item to be ~~ready~~ available
2. read next available
3. Signal a new slot is freed

producer
1. put something in next available + signal ready
2. wait for next available slot

1. what if there was no synch?
2. figure out who need to wait (for what)
3. figure out when waiters should be signaled





"mutex"
"signal/wait"
} both can be represented by

Dijkstra : semaphore ← classic

$<= 0$

wait

Semaphores   counter
set of waiting threads

P   "prolaag"   try decrease / down / wait

V   "verhoog"   increase / up / post   "signal"

$$\begin{array}{ll} & \} \{ \} \\ \text{sem\_wait} & \overline{} \\ & \} \\ \text{sem post} & \overline{} \end{array}$$

"binary semaphore"   (lock)
initial counter value = 1

Semophores vs condition vars
wait/post match ← "signals" don't get lost if no waiters
simple "state" (counter)
↑ one off signals

```
while(head-tail %.....)
    cond_wait(...)
```

Concurrency problems   locks!
1. Atomicity violation

```
T1:  if (thd → procinfo)
         fputs( thd → procinfo)
T2:  thd→procinfo = NULL
```