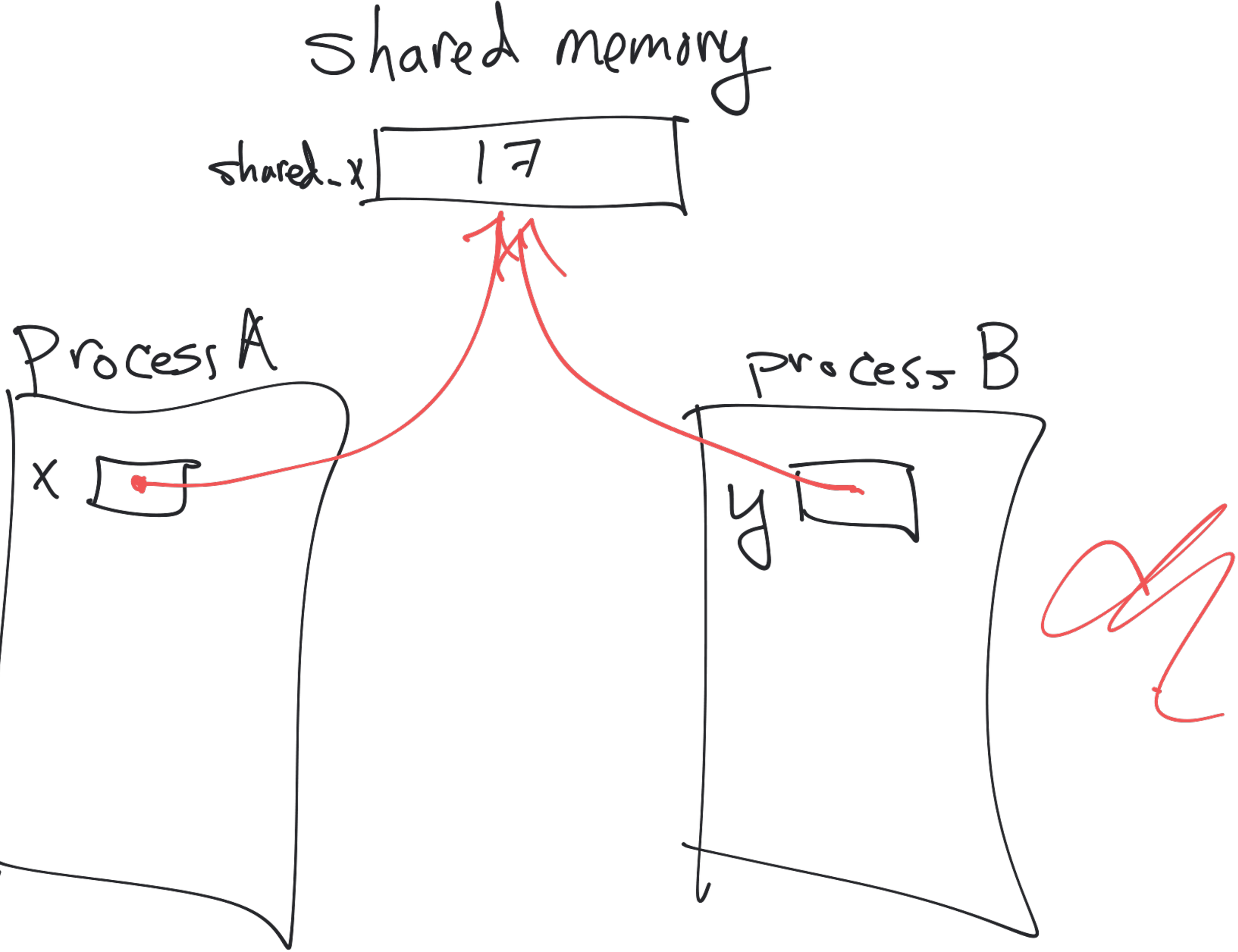
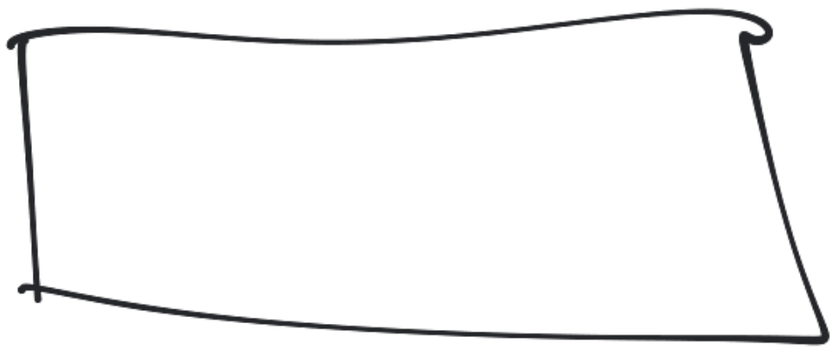
CS 208

Wed / Fri

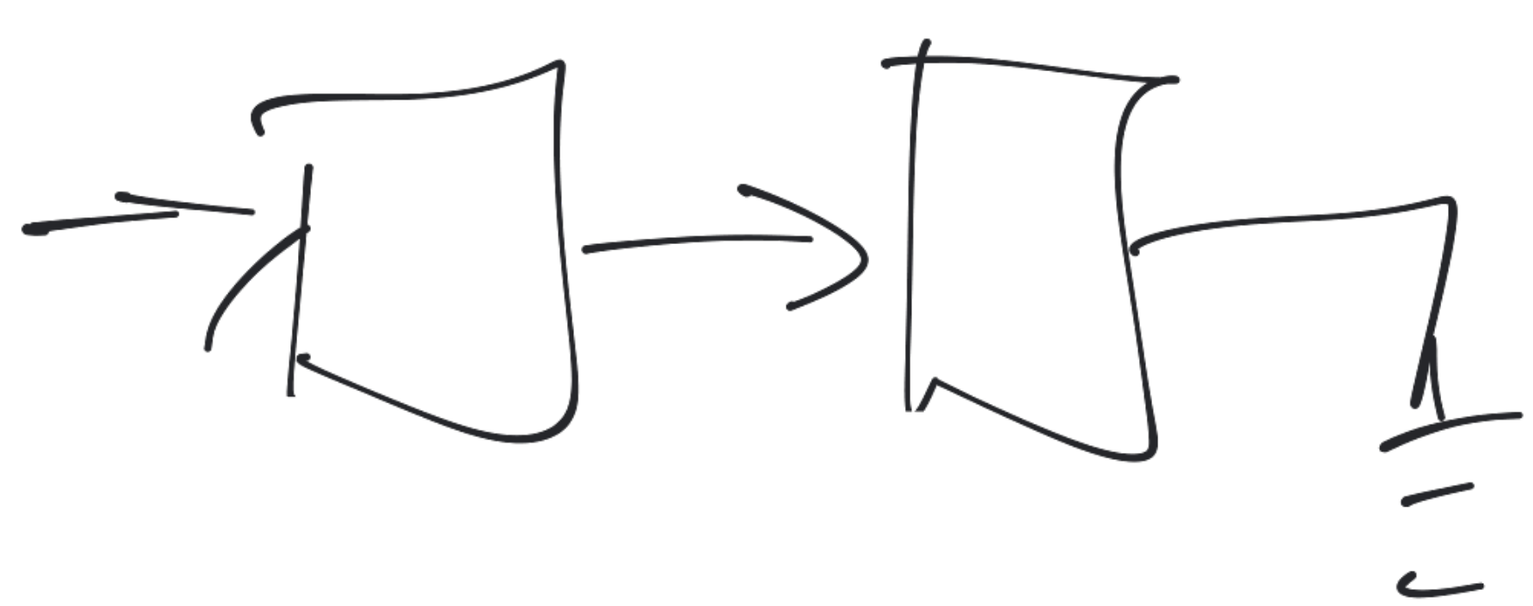
Feb 23 - 25, 2022


Shared memory
shared_x
17
Process A
x
process B
y

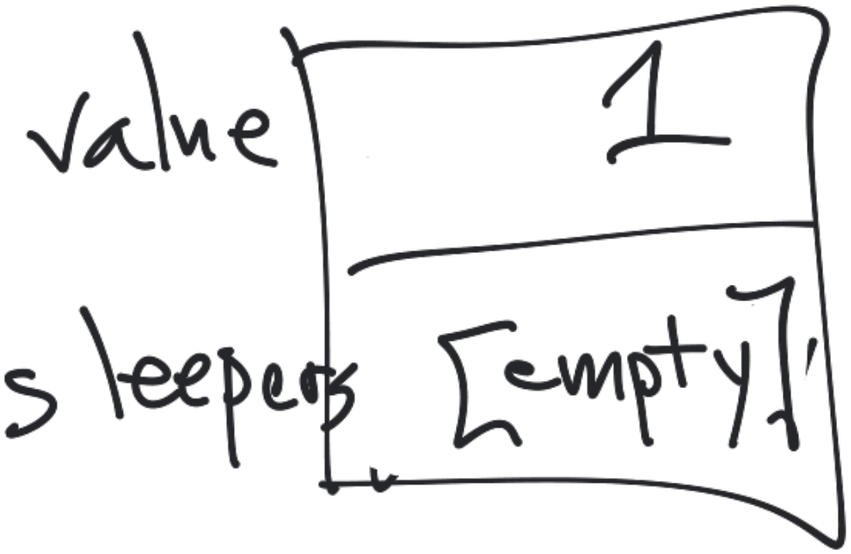
Semaphore

int 

queue of sleeping processes



Initialize Sem.

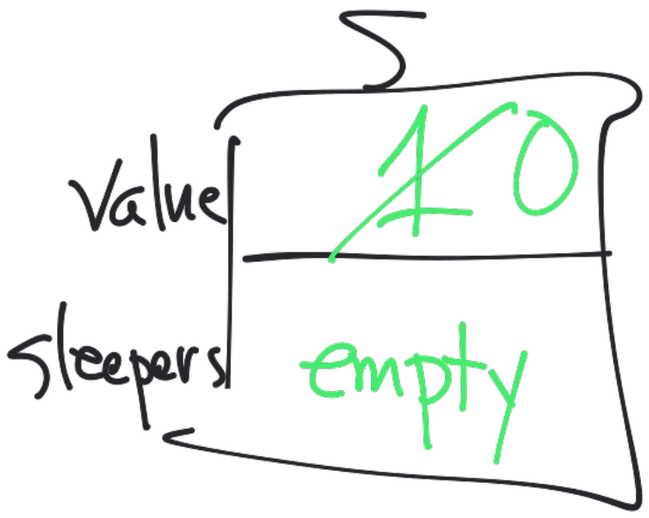


(share among some processes)

Two ops.

Down

Up



① Process A

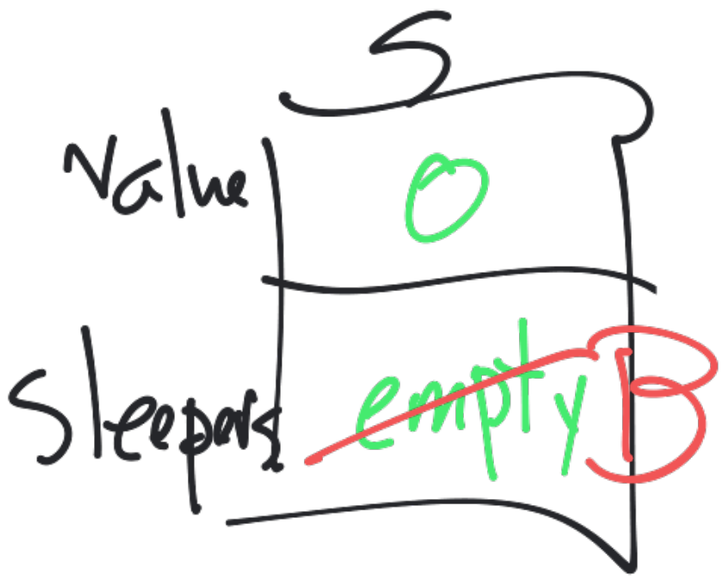
② Down(s);

critical region ← preemptive interrupt

Up(s);

Down

```
if s.value > 0
    s.value --
else
    put calling process to sleep
    add calling process to s.sleepers
```



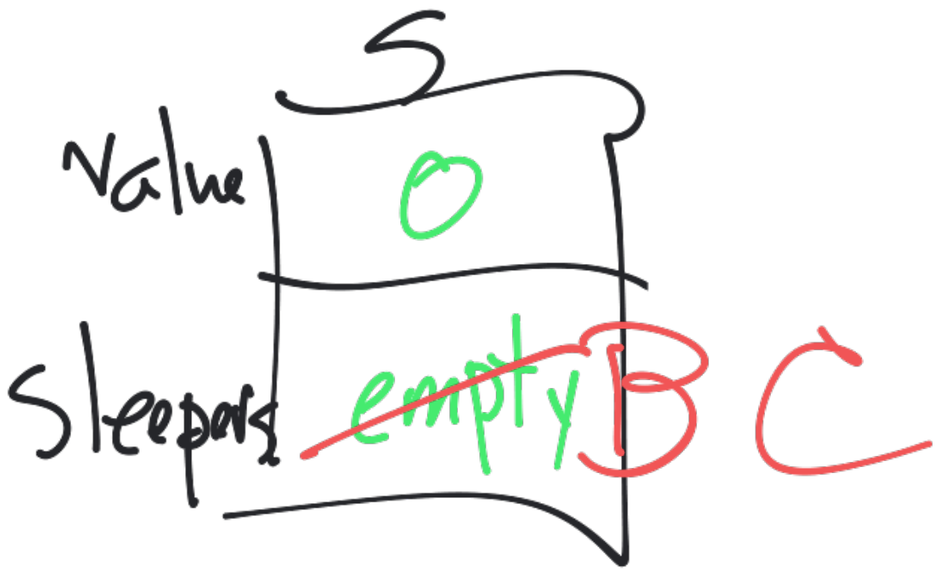
A is in its critical region

process B

① Down(S);

crit. region B

Up(S);



A is in its critical region

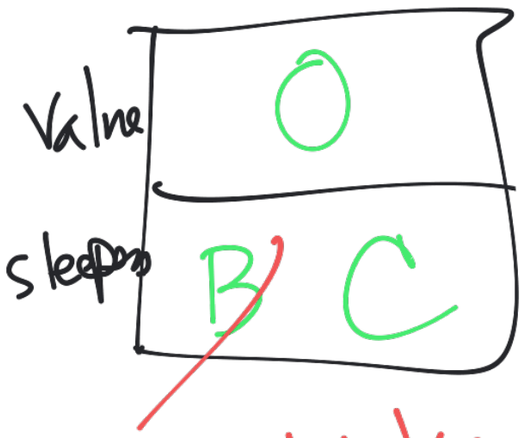
Process C

① Down(S);

crit. region C

Up(S);

S



A

```
Down(s);
critical region
Up(s);
```

Wake up little B!

```
Up:
if s.sleepers
   is non-empty
{
   wake up the front sleeper
      (& remove from s.sleepers)
}else
   s.value++;
```

A Down — 0, empty — (A in CR)

B Down — 0, B — (B asleep, A^{B} in CR)

C Down — 0, BC — (B, C sleep; A ^{BC} CR)

A Up — 0, C — (C sleep; B, C – CR)

B Up — 0, x — (– ; C in CR)

A Down — 0, A — (A sleep; A, C in CR)

C Up — 0, x — (– ; A in CR)

A Up — 1, ~~x~~ — (– ; –)

A "mutex" (enforces "mutual exclusion")

"binary semaphore" (Lock, Unlock operations)

Weird problems

"Dining Philosophers"

" "

" . . . "

(check Wikipedia)