OS - Semaphores

1. Producer Consumer Problem
   - → Mutex | Lock
   - → Synchronised
   - → Semaphore
2. Practice Problems
3. Atomic Data types

Producer Consumer

(1) We have a conveyor

belt which can hold
n items.

② If no items on the belt,
# of producers ⇒ n

③ If conveyon belt is full
# of consumers ⇒ n

19
Producer
Consumer

```
if (q.size() <= 20)
    FI food = new FI();
    q.add(food)
```

```
if (q.size > 0)
    q.remove()
```


P 1

P 2

// Create

// add to the q

// Create

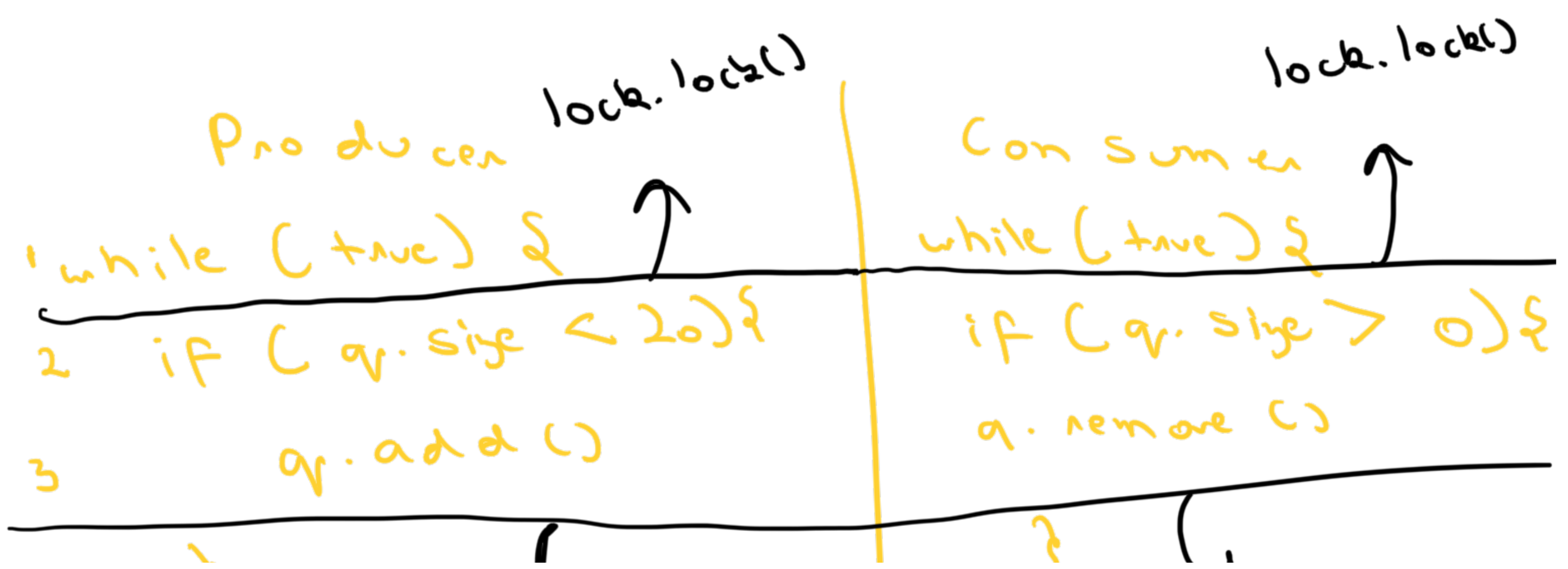
// add to the q

Underflow

Only happens

→ Multiple producer consumer

Solutions — Mutex lock



lock.unlock();

lock.unlock();

6:13 - 6:18

10:48

run() {

read(x);

write(x);

Sys.out();

{

read(y);

write(y);

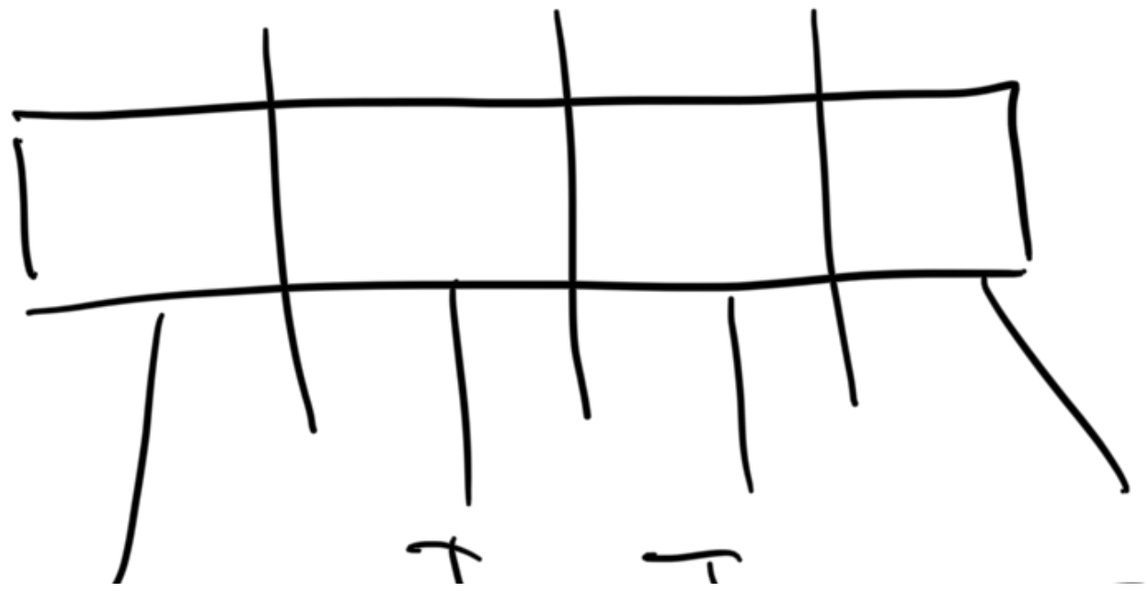
}

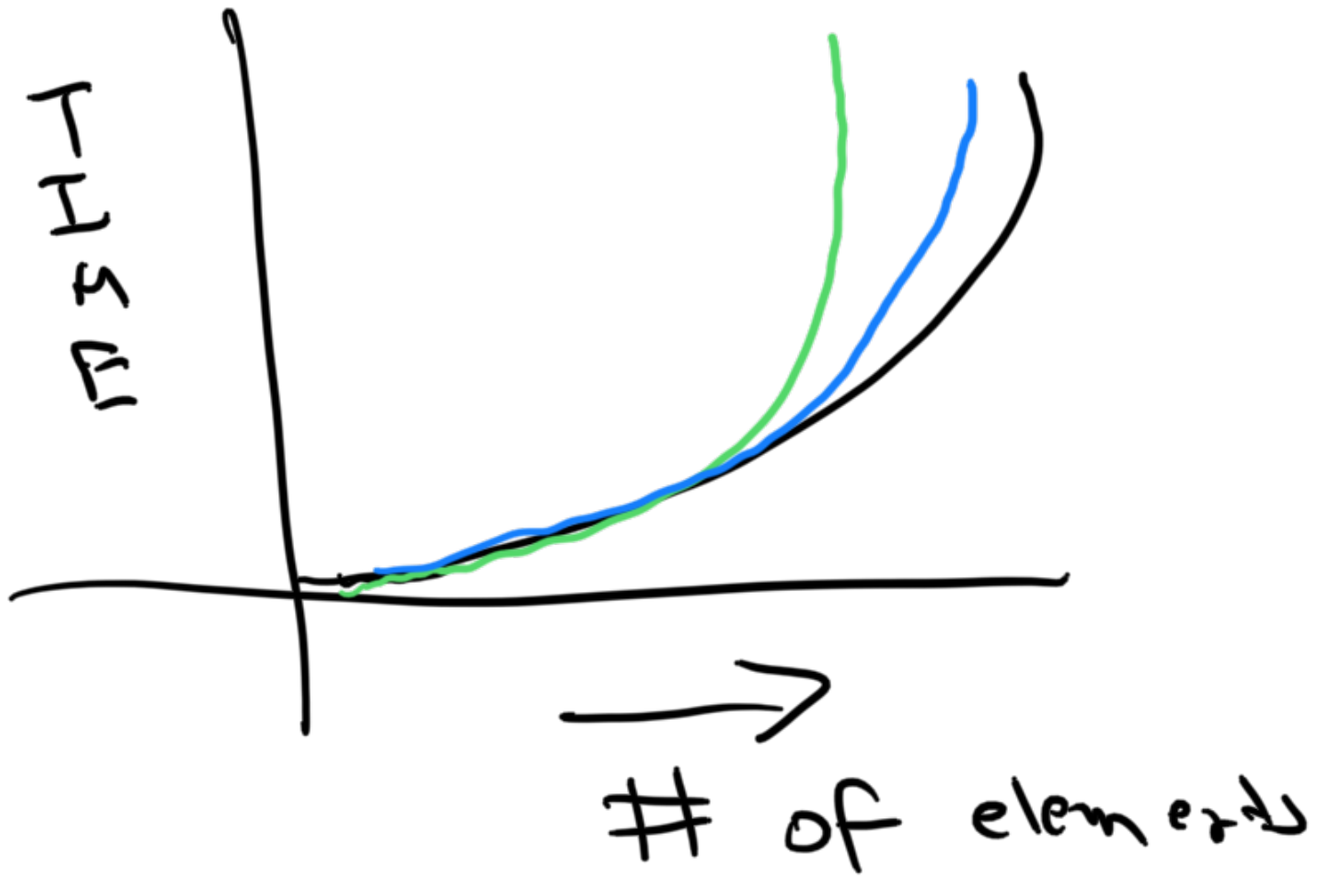
1. Write your own getters
   - More work
   - Customisable

2. IDE - generators

- adds the code

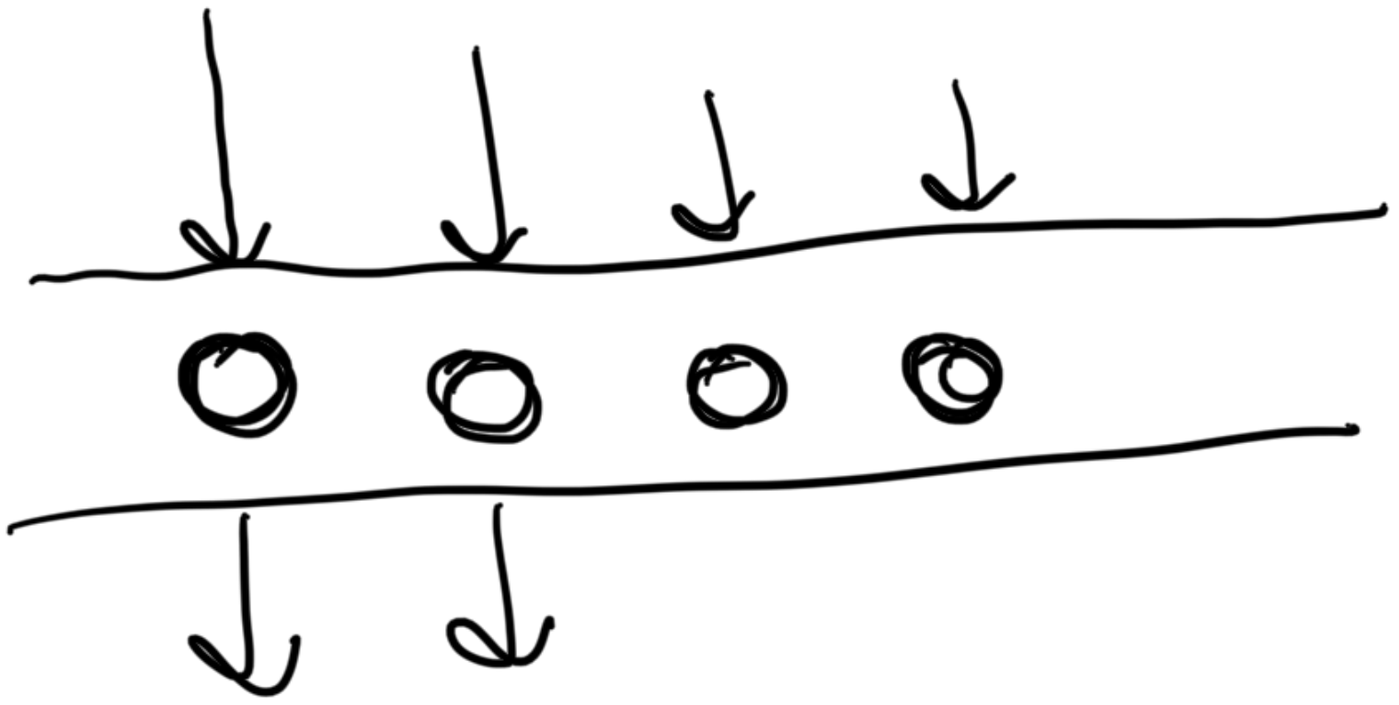
③ Lombok = Create
- not pollute
↓ customisable
↓
getCount(type)





Mutex

→ One thread into the CS

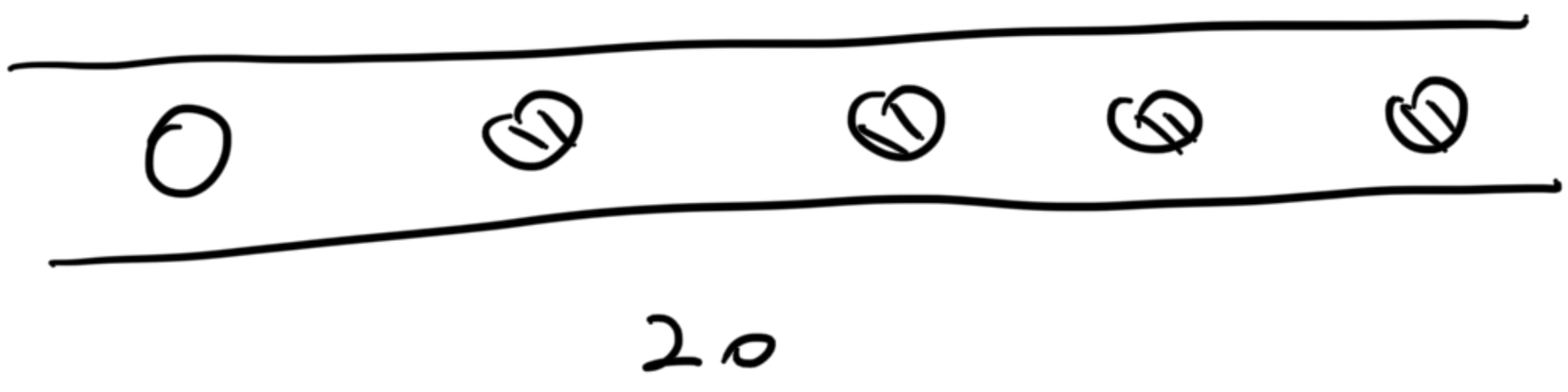


Produces

# of produces that can work parallely = 20, 19, 18

20

# of producers ⇒ # of free/avl. slots



20

# of consumers = 20
= 19

= # of filled slots

Semaphores

Semaphore s = new Semaphore(MAX);

new Semaphore(1)

Lock.lock() → Semaphore.acquire()

Lock.unlock() → Semaphore.release()

Semaphore(20)

Producer

S.acquire()

Consumer

Q = READ();

if (Q < 20)

WRITE();

Q = READ();

→ if (Q > 0)

WRITE();

↳ S.release()

S = 20

P1

Q = 1

S = 19 ....

P2

Q = 2

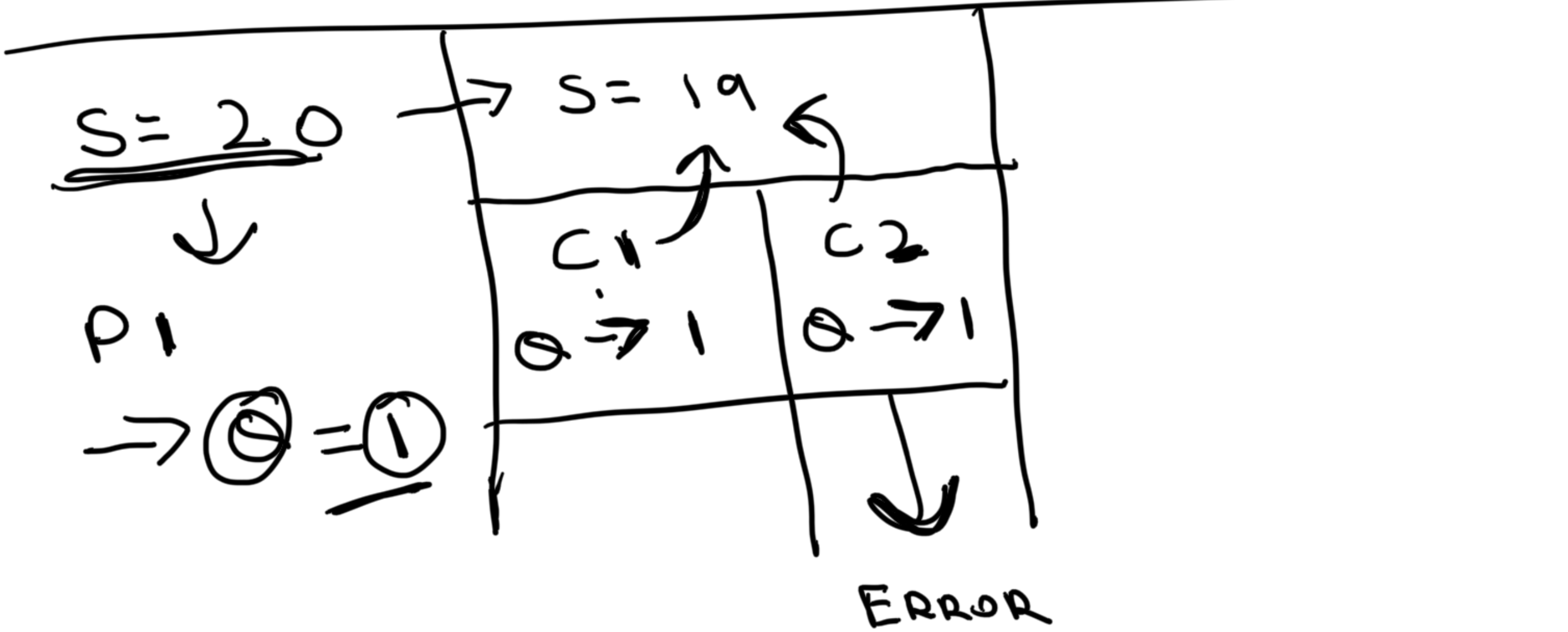
....

S = 0

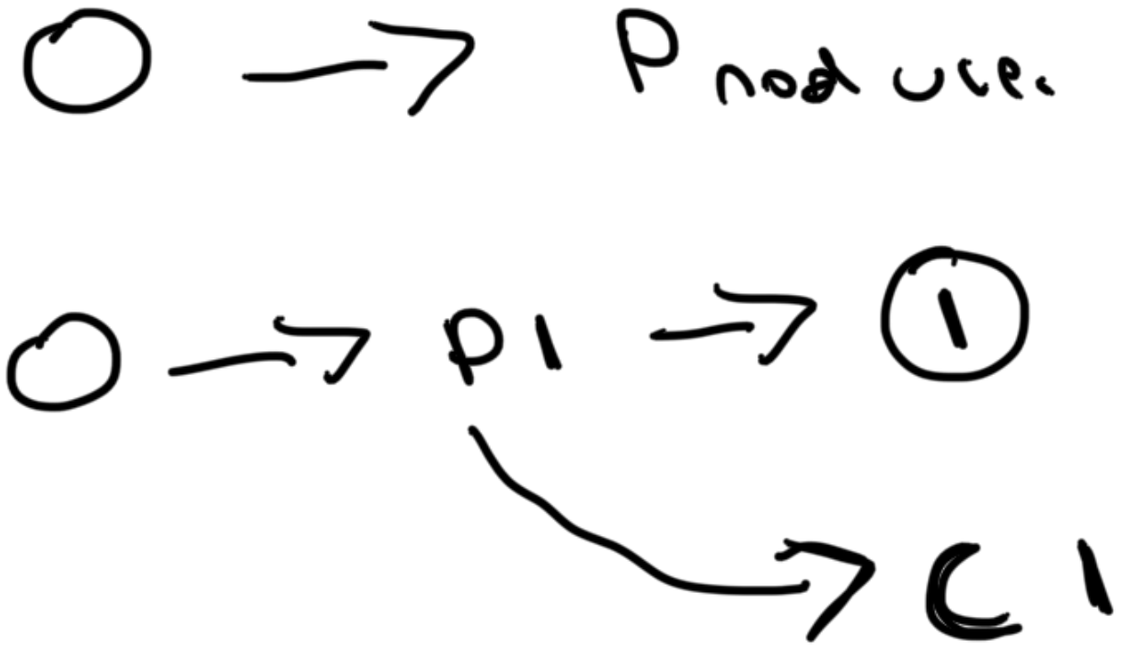
WAIT

↓

S > 0



Semaphore → signal



Semaphore (20) → All empty slots

Semaphore (0) → All filled slots

P1 → Create one item

1

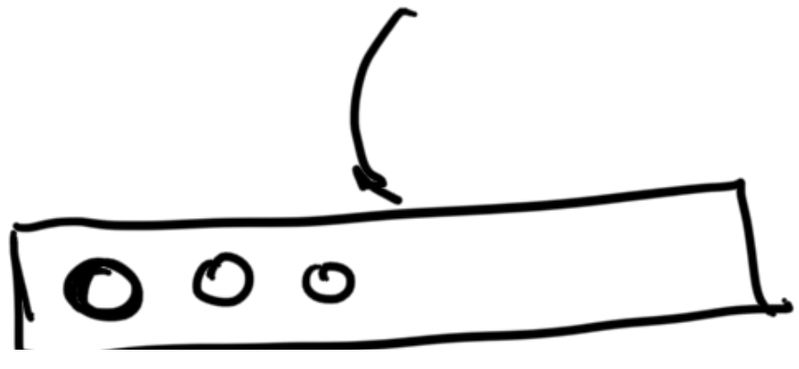
S1 → 19

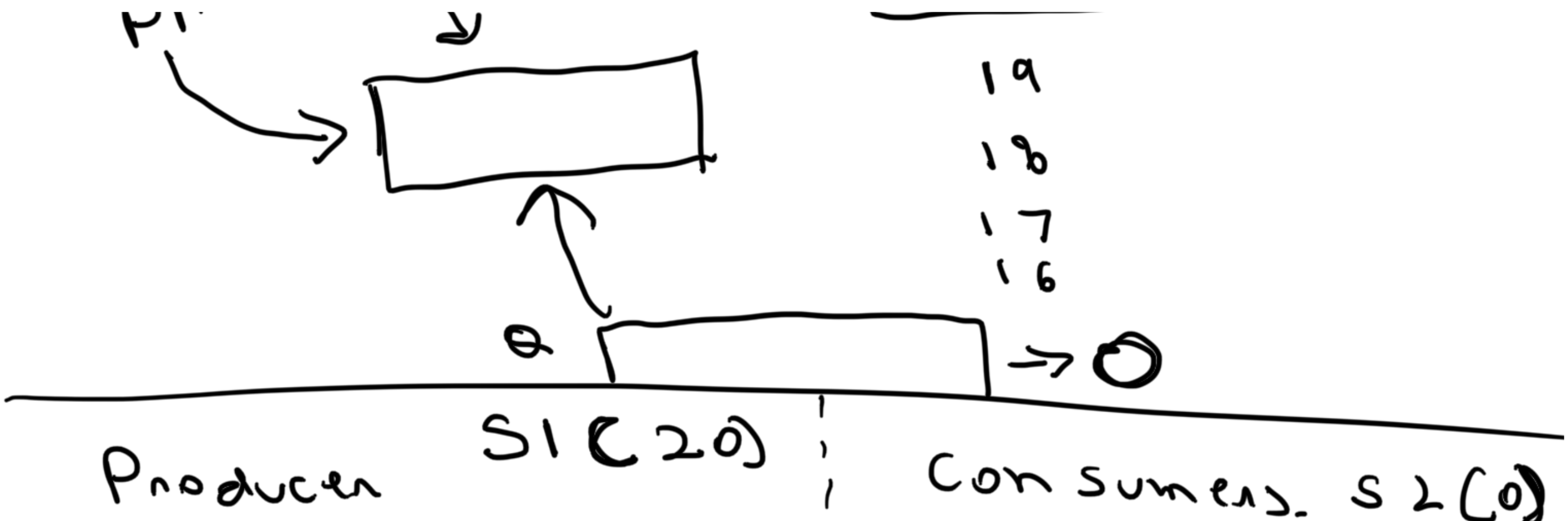
S2 → 1

S1.acquire() → 20 − 19

⋮

S2.release()





S1.acquire() → 19

Q = Read();

if (Q < 20) → Q

Food = new Food();

WRITE (Q, Food)

Q → 1

S2.acquire(); → Q

Q = Read();

if (Q > 0)

Food = Q.remove();

WRITE (Q);

S1. release()

S2. release()

↳ S2(1)

Q=0	Q=1		Q=0
S1 =20 S2 =0	S1 =19 S2 =1		S1 =20 S2= 0
P1	C1	C2	P3
P1 → Q=1 → S1 =19 S2=1	S2 Q S1 0 0 20	S2 Q S1 0 1 1	S1 Q S2 19 1 1

S1 → S

S2 → 15

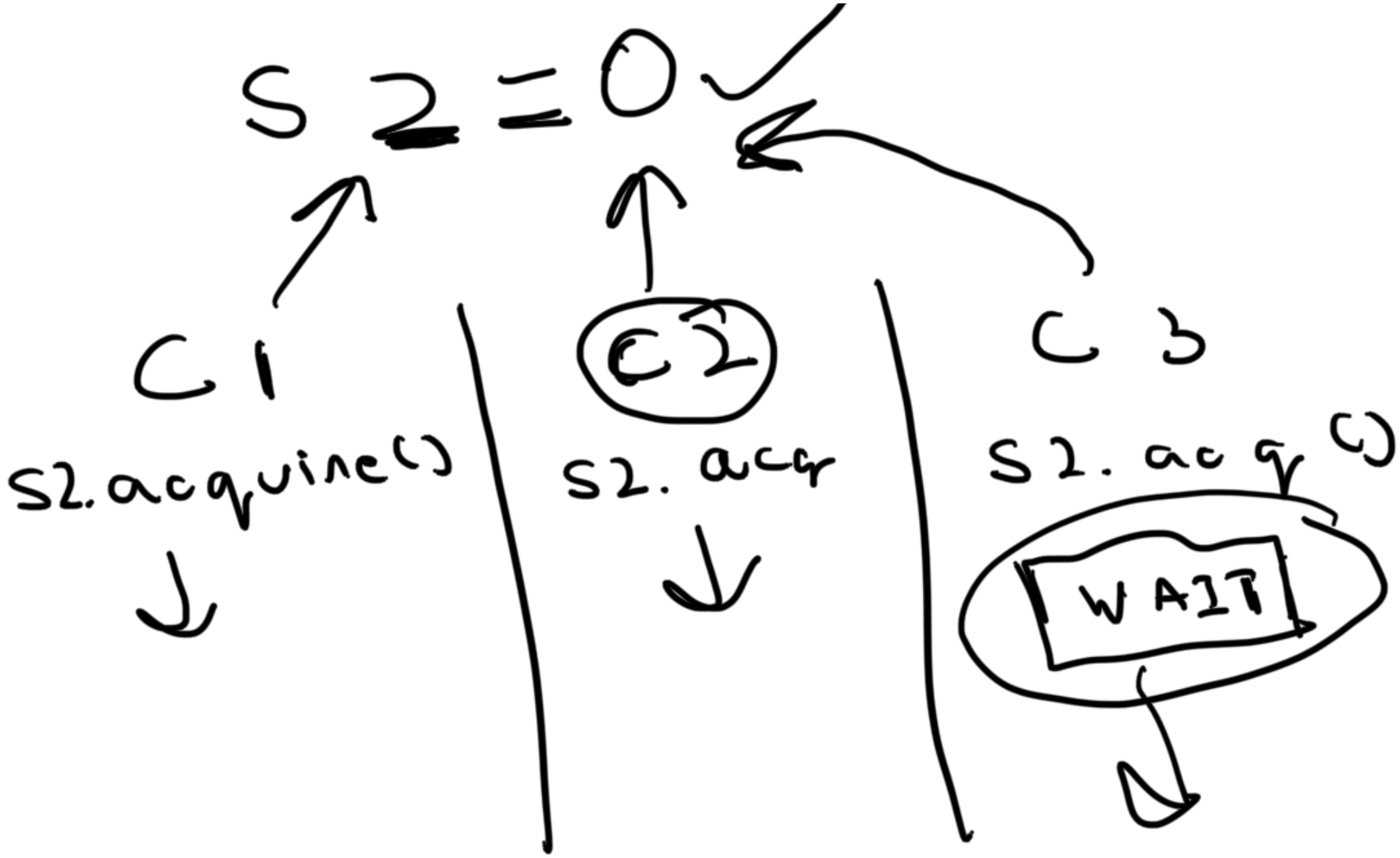
P1 P2 P5

S1 = 20

S1. release() → 1

S2 20

19


S 2 = 0
C 1
S2.acquire()
C 2
S2. acq
C 3
S 2. acq ()
WAIT