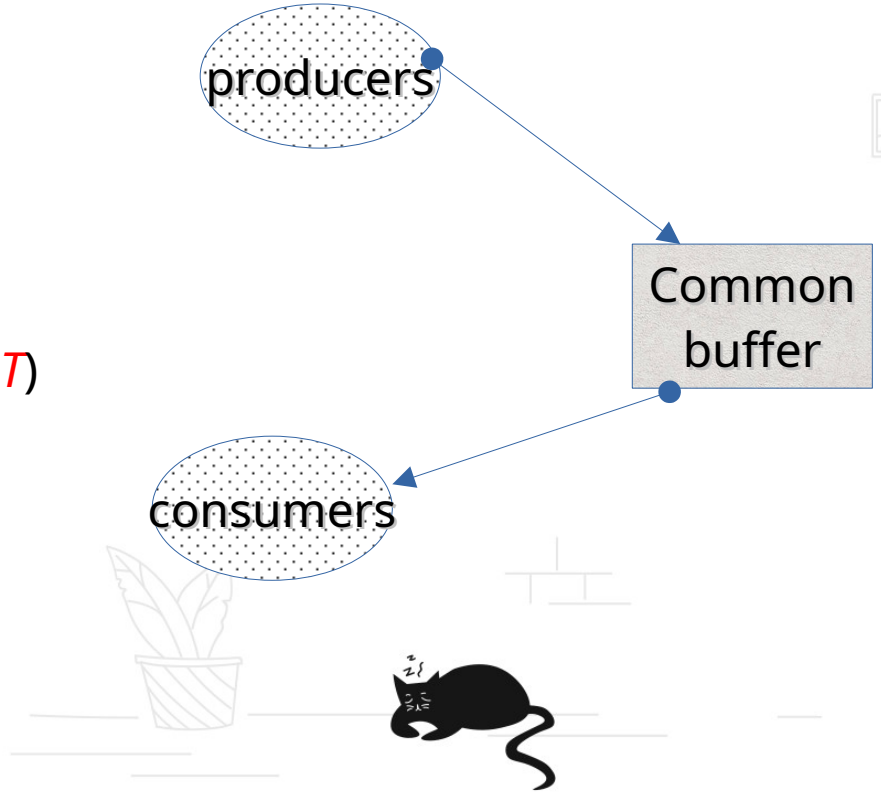
Producer Consumer Problem

Aagaman Luitel
Feburary 28, 2024

LibreOffice

Producer Consumer Problem (Bounded buffer problem)

M Producers → Produces
N Consumers → Consumes
Common shared array/list/buffer (data *T*)



6 producer threads
6 consumer threads
buffer size 3 bytes

What happens if 6 threads try to put value in the buffer at the same time?

Assumption: producer generates 1 byte data in the buffer

Rules
- on buffer full lock the producer threads
- on buffer empty lock the consumer threads
- mutually exclusive data using mutexes

semaphores are signals that can be used by multiple thread to request an access

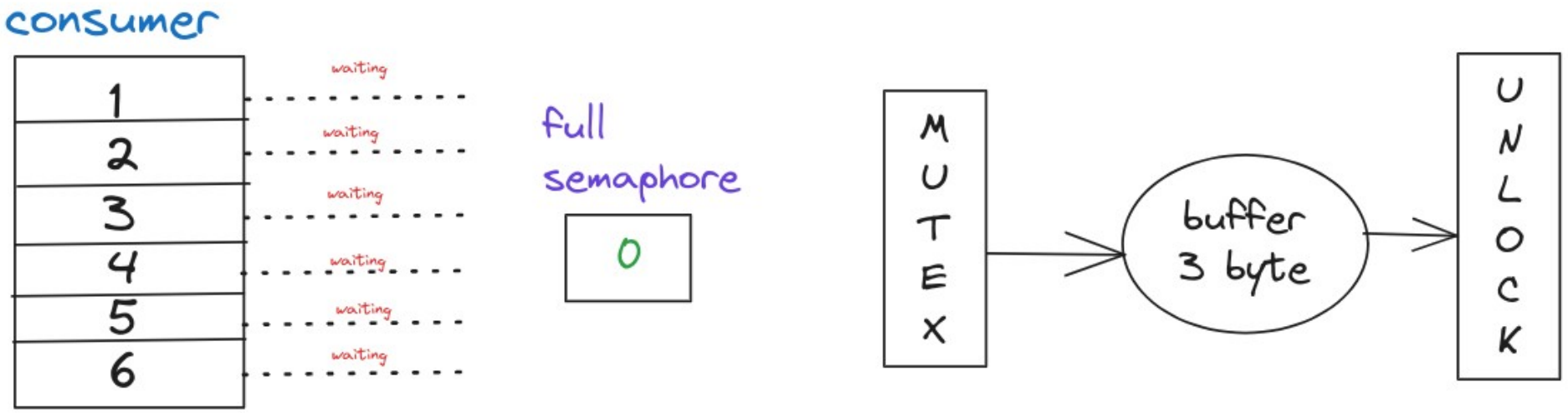
mutex is like a unique key that can be used by only one thread

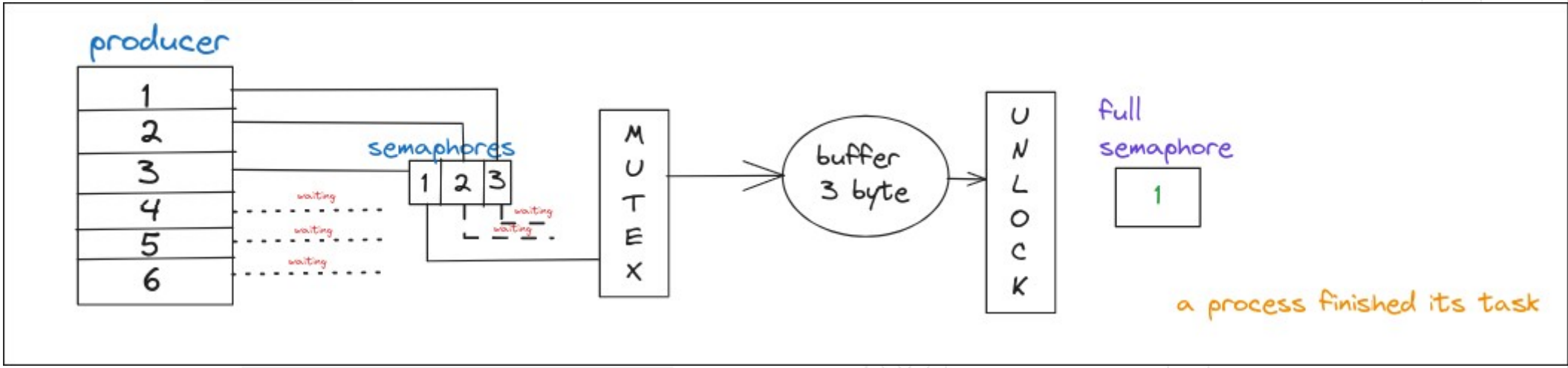
2 binary semaphores and 1 mutex is needed

- empty semaphore[P]
  → tracks no of empty slots available in the buffer
  → producer [*M*] have to wait if there are no semaphores available

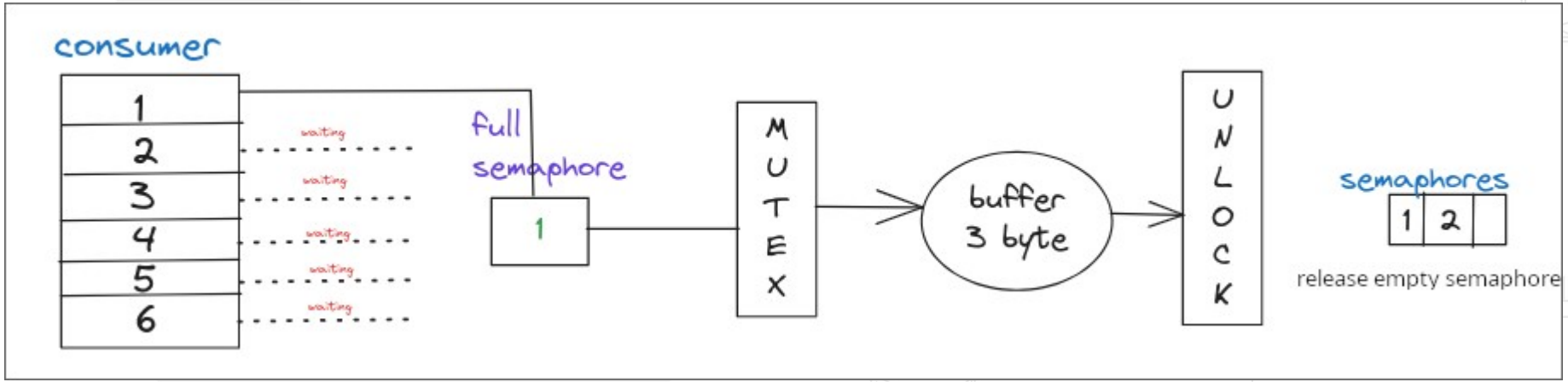
- full semaphore[0]
  → acts as a flag if buffer is empty or not empty
  → consumer [*N*] consumes if buffer is not empty i.e 1

- mutex
  → does not allows other thread to access the buffer


consumer
1
2
3
4
5
6
waiting
waiting
waiting
waiting
waiting
waiting
full
semaphore
0
MUTEX
buffer
3 byte
UNLOCK


producer
1
2
3
4
5
6
semaphores
1
2
3
waiting
waiting
waiting
waiting
waiting
MUTEX
buffer
3 byte
UNLOCK
full
semaphore
1
a process finished its task


zzz...


consumer
1
2
3
4
5
6
waiting
waiting
waiting
waiting
waiting
full
semaphore
1
MUTEX
buffer
3 byte
UNLOCK
semaphores
1
2
release empty semaphore


zzz...

```
fn producer(){
    // prodcue some value
    V item = compute_some_value
    // [3 – 1] = 2 threads can still access the buffer
    empty_semaphore-- // acquire

    // lock the thread like a good boy
    mutex.lock()
    buffer.push(item)
    // unlock the thread for other to put items in the buffer
    mutex.unlock()

    // consumer knows its safe to consume, i.e buffer is not empty
    full_semaphore++ // releasae
  }
}
```

```
fn consumer(){
    // consumer waits until producer says full semaphore is 1
    full_semaphore-- // acquire

    mutex.lock()
    V item = buffer.pop()
    mutex.unlock()

    // [2 + 1] = 3 threads can access the buffer
    empty_semaphore++ // release
}
```