The Producer/Consumer Problem

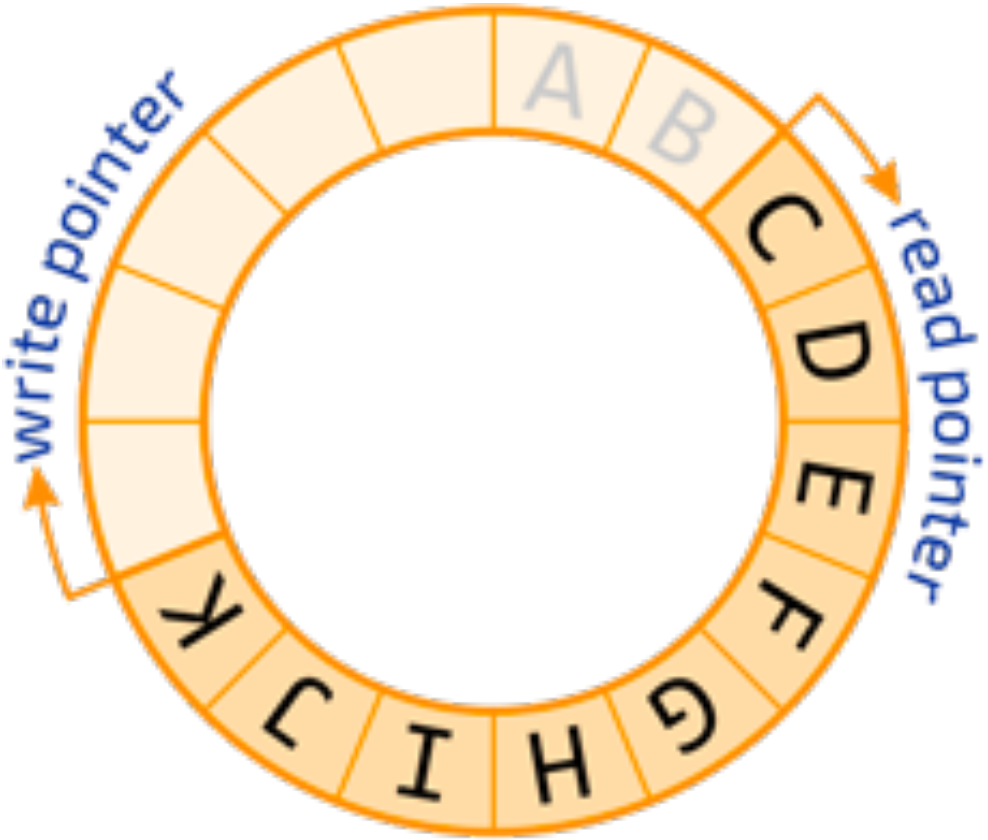
- Also called the Bounded Buffer problem.

Producer

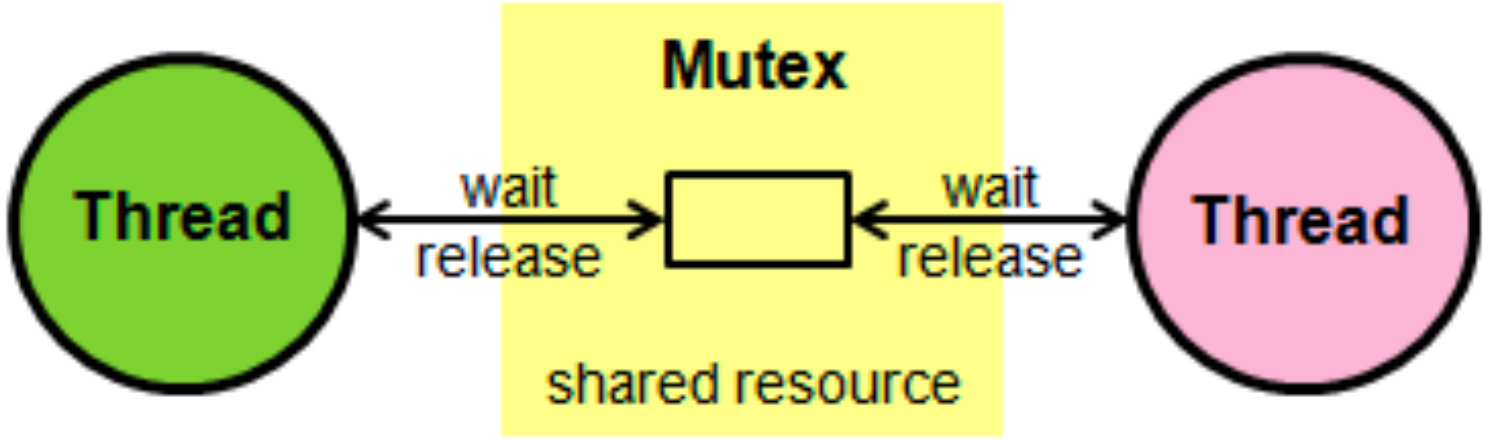
Mmmm... donuts

Consumer

- Producer pushes items into the buffer.
- Consumer pulls items from the buffer.
- Producer needs to wait when buffer is full.
- Consumer needs to wait when the buffer is empty.

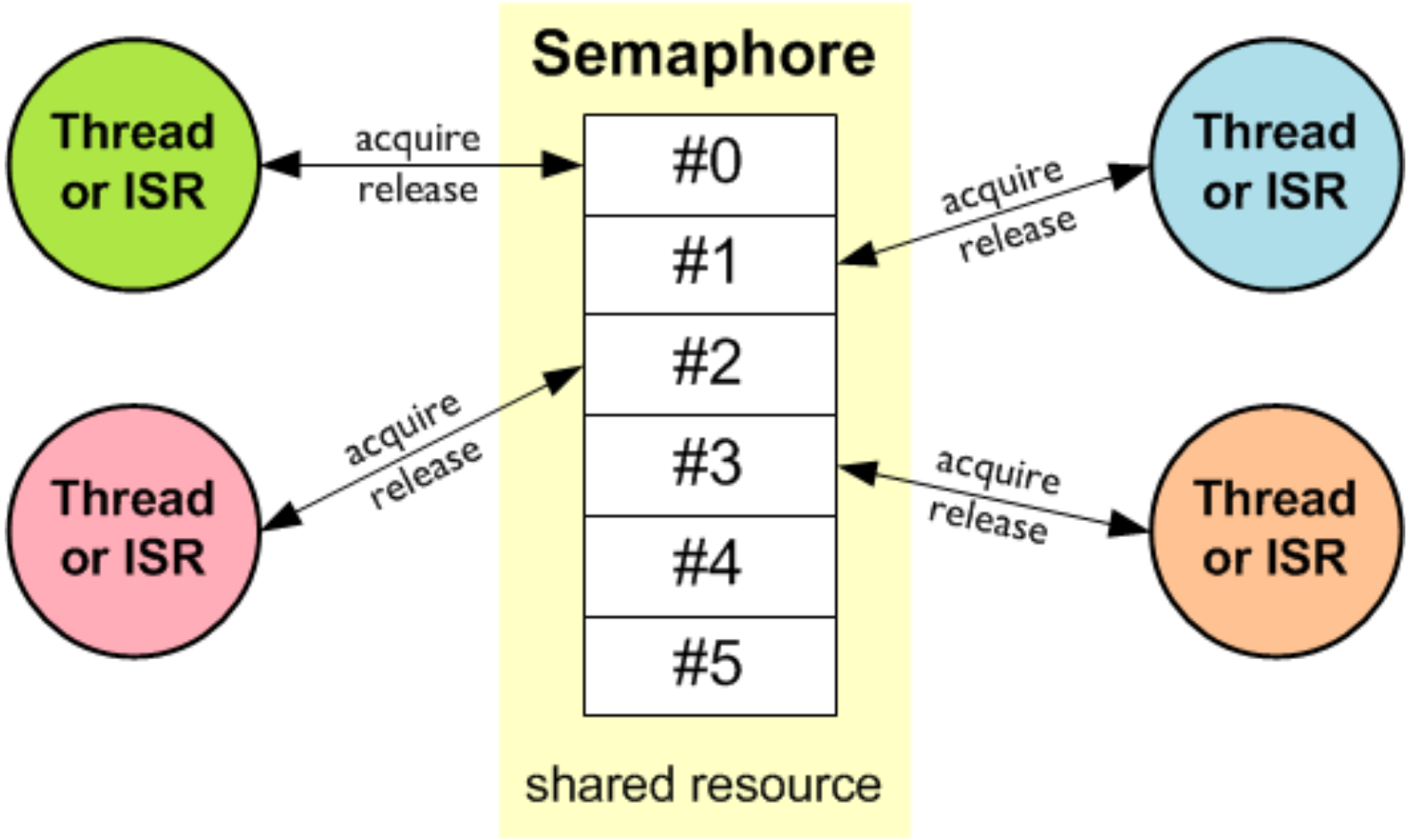

write pointer
read pointer
A
B
C
D
E
F
G
H
I
J
K

Mutexes



- Mutexes are used to prevent data inconsistencies due to race conditions. A race condition often occurs when two or more threads need to perform operations on the same memory area, but the results of computations depends on the order in which these operations are performed. Mutexes are used for serializing shared resources.

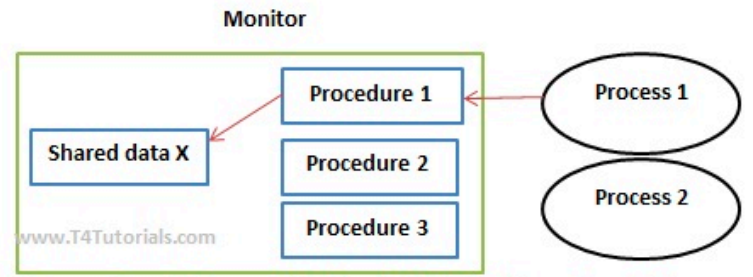
Semaphore


Semaphore
Thread or ISR
acquire
release
#0
#1
#2
#3
#4
#5
Thread or ISR
acquire
release
Thread or ISR
acquire
release
Thread or ISR
acquire
release
shared resource

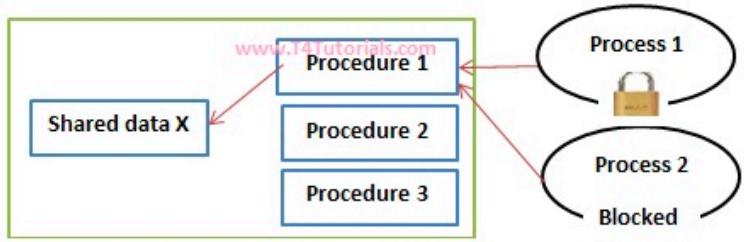
Monitors

```
Monitor Demo //Name of Monitor
{
variables;
condition variables;

procedure p1 {....}
prodecure p2 {....}


}
```

Syntax of Monitor

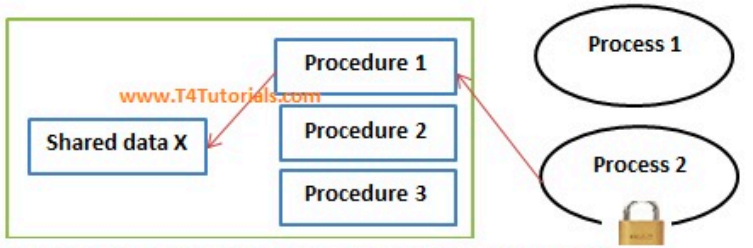


Process 1 wants to access the shared data X. So Process 1 gets the permissions from monitor, and here in this example, Procedure 1 grants permission to the process 1 to access X. Now, monitor locks the shared data by setting a lock for Process 1



Process 2 wants to access the shared data X. Procedure 1 in the monitor not give the shared data - X access to the process 2. Process 2 remains blocked to access the X until Process 1 leaves the X.

In this case how Process 2 will access the shared data X?



Process 2 wants to access the shared data X. Procedure 1 in the monitor give the shared data - X access to the process 2.

- #include <semaphore.h>
- sem_t semaphore;
- sem_init(&*sem*, 0, count);
- sem_wait(&semaphore);
- sem_post(&semaphore);

- pthread_mutex_t mutex;
- pthread_mutex_lock(&mutex);
- pthread_mutex_unlock(&mutex);

- pthread_cond_t wait_for_cons;
- pthread_cond_wait(&wait_for_cons, &mutex);
- pthread_cond_signal(&wait_for_prod);