# Producer Consumer Problem

## Introduction

Producer consumer problem is an example of synchronization between multi processes in which producer and consumer operate on a shared data buffer(queue). The producer tries to insert data in the queue and the consumer tries to remove data from the queue. The producer will not insert any data item in the queue if it is full. Similarly, the consumer will stop removing data from the queue if it is empty.

The above problem can be solved by using two approaches described in brief below:

1. **Mutex**

   Mutex can be used to acquire/release a lock by producer/consumer on the critical section to access the shared data buffer. The mutex ensures mutual exclusion of the data access and avoid faulty situations. A separate check inside the critical section is needed to handle the boundary conditions.

2. **Counting Semaphores**

   Counting semaphores along with a mutex can be used to solve the problem. Two counting semaphores (full & empty) can be maintained to maintain the boundary conditions along with a mutex to ensure exclusive access to the critical section by the producer/consumer.

## Potential problems and faulty situations

1. **Data Race**

   As producer and consumer are trying to read/modify the same shared buffer, there could be read-write/write-write data races without the use of any synchronization techniques. e.g. Two producers may try to write the same data location while generating a new data item.

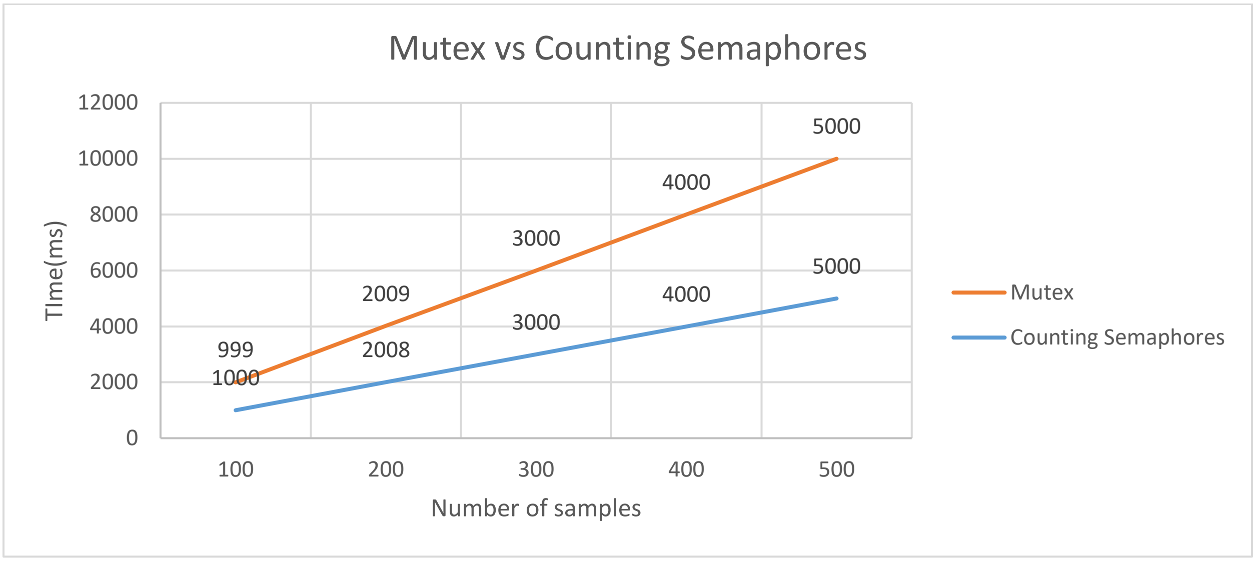
2. **Data loss**

   Without any synchronization techniques, the producer may override the data e.g. in a circular data buffer which has not been consumed by the consumer yet and hence there might be data loss. Another example is a producer overriding the data produced by another producer.

3. **Deadlocks**

If producer/consumer use sleep while waiting to be awaken by the other, there could be deadlock as both might wait for each other if no synchronization techniques are used.

### Performance Graph: Mutex vs Counting Semaphores



**(Note: Please do not see y-axis to scale as counting semaphore has been displayed separately to avoid overlap between the two lines)**

## Conclusion

The producer-consumer problem can be successfully solved by using both mutex and counting semaphores.

As per the graph shown above, it is clear that with initial negligible differences, the time taken by **mutex** and **counting semaphores** are almost identical to solve the producer consumer problem. This happens because the producer and consumer take almost equal processing times to operate in our example. When we consider situations where there is a difference in the amount of time taken by the producer and consumer, the performance of mutex and counting semaphores will vary. Counting semaphores are more suited to be used in the producer-consumer problem as they take care of the **buffer full** & **buffer empty** case which needs to be handled separately in solution using only mutex.  So, multiple producers and consumers will perform better using counting semaphores.