Assignment 1: Producer & Consumer

Introduction:

In this assignment Producer consumer are implemented using mutex and counting semaphore. File main_mutex.c and main_countsem.c are the actual implementation of above two mentioned ways.

Mutex Implementation: File : main_mutex.c
I have maintained a structure which has an integer value index. I have an array of structure which is global. Producer process assigns a value to index for different indices of this structure array. Consumer process prints the index value on console. To protect the critical section I have initialized one mutex. After producer completes work in critical section it release the lock and consumer acquires the lock and makes producer wait. Thus producer and consumer access critical section one at a time. Thus avoiding and data race and inconsistency.

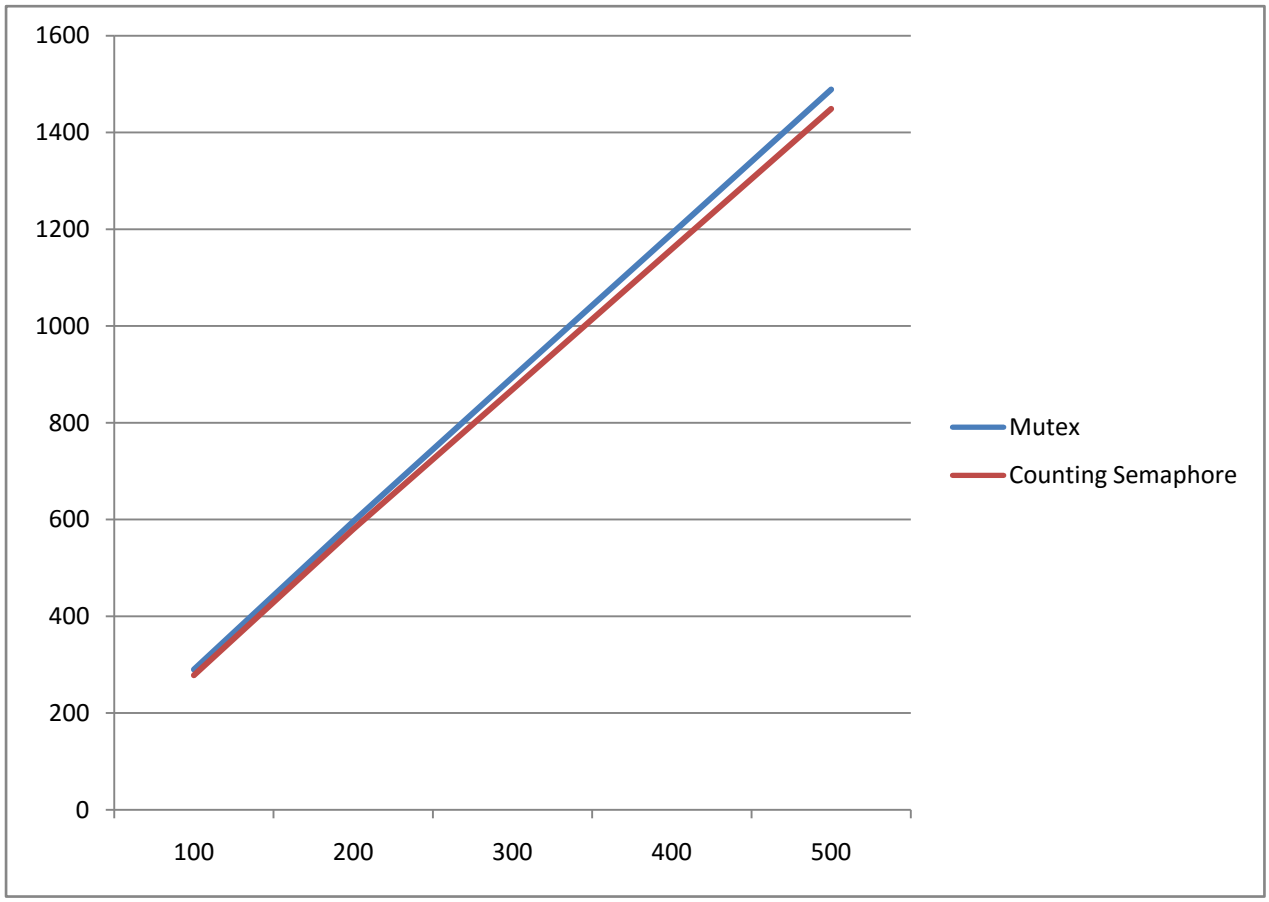
Counting semaphore : File : main_countsem.c
For this implementation I have initialized two semaphores sem_empty and sem_full. One is initialized with buffer size and one with 0. Producer is acquires sem_empty and consumer acquires sem_full. When producer signals sem_full consumer enters critical region. During this time since consumer has lock on sem_empty producer can enter critical region and modify buffer so I have introduced a mutex for buffer so at a time only one process can access buffer. Once consumer is done it signals sem_empty.

Problems due to no synchronization:

- Data Race: if consumer is trying to read data in critical section and perform some calculations using it and concurrently produces modifies that data, consumer will calculate wrong data.
- Buffer overflow and buffer underflow: Without synchronization producer will not know when to stop producing and will overwrite existing data. Similarly consumer would not know when to stop reading as there will be no data once it consumes all the data.
- Multiple write and multiple read: In a multi producer and consumer scenario, co ordination between producers is important. Producers should be in synchronization with what and where they are producing otherwise it may lead to data loss.

Rishabh Pandita
UFID : 49129151

Timing Graph:



Conclusion

Using semaphore and mutex co ordination/synchronization problem were handled successfully. On analyzing the timing graph it is clear that counting semaphores take less time compared to mutex. The timing difference is not considerable because in this simulated environment producer and consumer are light and they execute quickly, they have same priority and take same time which would not be case when multiple producers and consumers are running in OS.

Mutex is a locking mechanism which can be used for locking a resource, when we implement producer consumer using mutex we need to take care of synchronization between processes. Counting semaphore is a signaling mechanism. It lets other processes know when they are done and can carry on. So its prefrable to use counting semaphore for producer and consumer problems.

Rishabh Pandita
UFID : 49129151