33231

CLASSMATE
Date :
Page :

ASSIGNMENT 4

Title : Thread Synchronization using Counting Semaphore

Problem Statement : Thread synchronisation using counting semaphores. Application to demonstrate : producer - consumer problem with counting semaphores and mutex.

Theory :-

Semaphore - a variable which is non - negative and shared between threads.

Binary Semaphore - This is also known as mutex lock. It can have only two values 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problem with multiple processes.

Counting Semaphore - Its value can range over an unrestricted domain. It is used to control access to a resource that has multiple instances.

Producer - Consumer Problem - Given a buffer of fixed size, a producer can produce an item and place it in the buffer. A consumer can pick items and consume them. We need to ensure that when a producer is placing an item in the buffer, then at the same time consumer should not consume any item. In this problem, buffer is the critical section.

Algorithm :

Consumer Function :-

```
void consumer () {
// consumes items and finally pops from buffer and processes
    int itemc ;
    while (true) {
        while (count == 0) ;   // buffer empty
        itemc = Buffer (out) ;
        (out) = (out +1) mod n ;
        count - = 1 ;
        process_item (itemc) ;
        //I1 load Rc , count
        // I2 dec Rc
        // I3 store count , Rc
    }
}
int count = 0; // global variable shared by both
// buffer common to both , shared
```

Producer Function

```
void Producer () {
    // produces items and puts to buffer
     int itemp ;
     while (true) {
         while (count == n) ; // buffer full
         Buffer [in] = itemp ;
         in = (in + 1) mod n ;
         count + = 1 ;
         // I1 load Rp, count
         // I2 inc Rp
         // I3 store count , Rc
     }
}
```

Execution :

- Producer arrives executes I_1, I_2
- Before executing I_3, consumer arrives, so producer preempts
- Consumer execution begins executes I_1, I_2
- Before executing I_3, producer resumes and executes I_3
- After producer terminates consumer I_3 executes

Conclusion : I have successfully implemented producer - consumer problem with counting semaphores and mutex.