Report on Assignment 01

CS 5323 – Design and Implementation of Operating System II

Submitted by – S M Rafiuddin

CWID: A20387294

The code creates four threads, two producer threads, and two consumer threads. The producers produce numbers from 1 to 30 and put them in a shared buffer. The consumers consume the numbers from the buffer and add them to a sum variable. The buffer has a fixed size of 20. The producers wait when the buffer is full, and the consumers wait when the buffer is empty. This is ensured using the condition variables 'buffer_not_full' and 'buffer_not_empty'. The mutex lock mutex is used to ensure that only one thread accesses the buffer at a time. *(Here 4 is the number of readers passed from command prompt)*

To ensure synchronization between the producer and consumer threads, mutex and condition variables are used. Mutex is used to protect the shared data and avoid race conditions, while condition variables are used to signal and wait for threads that are either waiting to produce or consume. The buffer_not_full and buffer_not_empty condition variables are used to signal the producer threads to start producing when there is space in the buffer and signal the consumer threads to start consuming when there is data in the buffer.

In the producer function, the mutex is locked before adding a new item to the buffer. If the buffer is full, the thread waits for the buffer_not_full condition variable to be signaled. When signaled, it adds an item to the buffer, increments the index, signals the consumer threads, and unlocks the mutex.

In the consumer function, the mutex is locked before retrieving an item from the buffer. If the buffer is empty, the thread waits for the buffer_not_empty condition variable to be signaled. When signaled, it retrieves an item from the buffer, increments the index, calculates the temporary sum, signals the producer threads, and unlocks the mutex.

Mutexes only require a single atomic operation to lock and unlock, while semaphores require multiple atomic operations, making them slower. Since only one thread at a time can access the buffer in this implementation, a mutex is a better choice than a semaphore. However, in situations where multiple threads can access a shared resource at the same time, semaphores would be a better choice.

Screenshot of the code running in CSX server-