Operating Systems - II: CS3523

Programming Assignment - III

Solving Producer Consumer Problem using Semaphores & Locks

Assignment Report

Sagar Jain - CS17BTECH11034

March 6, 2019

Contents

Salient Features of Program Design

The problem has been solved using semaphores and locks.

Semaphores

Semaphores have been implemented using the header ***semaphore.h***

1. I have used three semaphores, ***full***, ***empty*** and ***locker***.
2. ***locker*** is initialised as 1, ***full*** as 0 and ***empty*** as the capacity of the buffer.
3. ***full*** is used in the consumer thread to ensure that no consumer can consume if there are no full buffers and we increment it in the producer thread everytime something is produced.
4. ***empty*** is used in the producer thread to ensure that no producer can produce if there are no empty buffers and we decrement it in the consumer thread everytime something is consumed.
5. ***locker*** is used to ensure mutual exclusion between multiple producers and consumers.

Locks

Mutex has been implemented using the header ***mutex***.

1. I have used two locks ***check_lock*** and ***update_lock***. ***counter*** is used to keep the count of the number of filled buffer cells.
2. ***check_lock*** is used to ensure that no two processes read the same value of counter and update them leading to a race condition.
3. ***update_lock*** is used to ensure that no two processes are reading and weiting at the same time.
4. ***check_lock*** is unlocked after ***update_lock*** to make sure that another thread can only check the condition once current thread has updated the value of ***counter***.

Program Output

The programs output log files (*output-semaphore.txt*, *output-lock.txt*). These files have the data about the point in time at which any consumer or producr consumes or produces any item from or into the buffer. For Example:

14th item produced by thread 4 at 19:58:09 into buffer location 61
0th item read from the buffer by thread 0 at 19:58:09 from buffer location 60
1th item read from the buffer by thread 0 at 19:58:11 from buffer location 59

Results & Graphs

Average Waiting Time vs Number of Threads

Maximum Waiting Time vs Number of Threads

Explaination of Results