Kshitij Deshpande K11 Roll No: 33312

Assignment 4 - A

Problem Statement:

Thread synchronization using counting semaphores. Application to demonstrate: producer-consumer problem with counting semaphores and mutual exclusion.

Theory

A) Producer Consumer Problem

• The producer consumer problem is a problem involving the use of synchronization to ensure efficient utilization of common resources.
• In this problem, the producer produces some items and places them on a platform termed as the buffer.
• The consumer consumes the items produced by the producer which are placed on the buffer.
• Here, the buffer is the shared resource and is required to be accessed by multiple people with different roles.
• The constraint involved in this problem is that at a particular moment of time,either the consumer or the producer can access the buffer.
• This involves the need of introducing the principal of exclusion.

B) Mapping of the problem to the solution

• Producer and Consumer : Threads
• Buffer : Shared Resource
• Task of Accessing Buffer : Critical Section
• Number of Empty and Full Items : Counting Semaphores

C) Threads

• Thread can be termed as an atomic entity inside a process initialized to perform a designated sub task.
• The process of thread creation, synchronization, communication and deletion are much faster than that of a process.
• Threads can be of 2 types :
    – User Level Thread
    – Kernel Level Thread
• Both types of threads are of the same importance and are intended for similar purposes.
• The difference lies in the creation process and levels of abstractions.
• In case of the producer consumer problem, each producer and consumer can be assigned an individual thread.
• The producer thread is responsible for creating the item and placing it on the buffer.
• The consumer thread is responsible for accessing the buffer and consuming the item.

D) Critical Section

• The critical section is termed as that section where the thread accesses the shared resources.
• The thread that accesses the shared resources can either modify, update or wipe out the underlying data structures which may cause conflicts with other threads if not handled properly.
• Hence, a section of code (Instruction trace) is defined as the critical section of the thread wherein the thread accesses the shared resources.
• In order to avoid the conflicts, the principle of mutual exclusion comes into picture.
• Here, the section where the producer or consumer threads request access to the buffer, that is termed as the critical section.

E) Mutual Exclusion

• The critical section, if not handled correctly, can cause conflicts directly associated with the value or existence of the shared resources.
• To avoid such a conflict, the mutual exclusion principle is introduced.
• It states that only 1 thread shall be allowed to be in its critical section.
• Only those threads can requests access to the resources i.e. execute its critical section for which the remaining threads are in their remainder section (non critical).

F) Counting Semaphores

• Semaphores are typically used to coordinate access to resources, with the semaphore count initialized to the number of free resources.
• Threads then atomically increment the count when resources are added and atomically decrement the count when resources are removed.
• When the semaphore count becomes zero, indicating that no more resources are present.
• Threads trying to decrement the semaphore block wait until the count becomes greater than zero.
• The life cycle of a thread involves the following operations :
    – Initialize : Initialize a semaphore (allocate memory and assign value).
    – Increment : Increment value of semaphore.
    – Decrement : Decrement value of semaphore.
    – Destroy : Destroy Semaphore
• Here, the semaphores are in charge of keeping a track of the empty and full slots available in the buffer.

Conclusion

• Concepts of threads and thread synchronization were understood.
• Principle of Mutual Exclusion was understood.
• Relevant terminologies and their applicability (e.g. Critical Section, Lock and Unlock) were understood.
• Producer Consumer problem was implemented using pthread library.