ELEC 377 Lab 3: Producer Consumer – Description of Solution

Section 003, Group 16 – Thursday Lab

Erhowvosere Otubu, #20293052

Ivan Samardzic, #20296563

Date of Submission: October 31, 2023

Problem Solution

The purpose of Lab 3 is to use pthreads mutexes and condition variables to synchronize threads and implement the classic synchronization problem known as the “Producer-Consumer” problem in C programming.

Solution Approach

The following sections will explain in-depth the process of the group’s solution.

Pthread Synchronization

The lab uses two pthread types,

- *pthread_mutex_t*: a binary semaphore that is used to provide mutual exclusion to critical sections. It is used with the pthread functions *pthread_mutext_lock()*, and *pthread_mutex_unlock()*. When a thread enters a critical section, it calls *pthread_mutex_lock()* to effectively block other threads from entering the same critical section until the current thread releases the lock. After a thread completes its task within the critical section, it calls *pthread_mutex_unlock()* to release the mutex lock and allow other threads to access the same critical section, preventing contention and race conditions.
- *pthread_cond_t*: variable used to coordinate and synchronize the execution of threads, allowing them to efficiently wait for specific conditions to be met before proceeding. Functions like *pthread_cond_wait(&full, &mutex)*, *pthread_cond_signal(&full)*, and *pthread_cond_broadcast(&full) are used*.

Lab Structure – Producer Function

The producer function opens a data file, reads each line in turn, and converts its content into numerical values, and places these values into a shared buffer. The condition variable *full* is employed to manage buffer capacity, ensuring that it waits when the buffer is full. To signal the consumer threads when to exit, the function decrements the global variable *numProdRunning* while safeguarding this operation with a mutex, thus preventing synchronization issues and ensuring a coordinated termination process.

Lab Structure – Consumer Function

The consumer function runs in an infinite loop, with a critical section for retrieving values from the shared buffer. It employs a mutex for synchronization, ensuring safe access to shared variables. The condition variable *empty* is used to make the consumer wait when the buffer is empty. If the buffer is empty, and there are no products running, then the consumer releases the lock and breaks the loops, where the output file is closed and the function returns. When data is available in the buffer, the consumer function retrieves it and writes to the output file, with this write operation occurring outside the critical section for efficient and parallel execution.

Lab Structure – Main

The main function sets the program parameters based on command-line arguments, initializes thread-related resources, creates the producer and consumer threads, and ensures that all threads complete their tasks before the program terminates. It orchestrates the multi-threaded execution of the producer-consumer problem.

Lab Structure – Interrupt Simulation

This part of the lab introduces a mechanism to simulate interruptions within the code to mimic real-world scenarios where context switches or interruptions can occur. It involves the following actions: function execution, random interruptions, and interrupt simulation. Simulating the interrupts adds a level of unpredictability to the program's execution, mimicking real-world scenarios where threads may be context-switched by the operating system. This helps to test the program's ability to handle interruptions and ensures that synchronization mechanisms are functioning effectively.

Special Features Used

Some special features of the C language demonstrated in the code include the following:

- C language's ability to work with threads and implement multi-threading
- Utilizes pthreads, mutexes, and condition variables to meticulously manage thread synchronization and ensure the safe and efficient operation of multiple threads