Ben Foltz-Miranda

Dr. Zhang

CS 433-02

19 April 2024

# Assignment 4 Program Report

## Problem Description

The problem that I am solving is being able to solve the bounded buffer problem in the textbook by using producer and consumer processes. These producer and consumer processes are able to add and remove items from the buffer without doing it at the same time with the use of mutex locks and semaphores to keep track of if the buffer is full or empty.

## Program Design

The Buffer class is designed with a queue to serve as the main data structure for the buffer object, an int to hold the size of the queue, an int to hold the amount items in the buffer, a mutex lock that protects the actual insertion or removal of items in the buffer, and semaphores that keep track of if the buffer is full or empty. When inserting an item in the buffer, the mutex will be locked to prevent the consumer thread or anything else from manipulating the buffer, then it will run a while loop to check if the buffer is full, and if it is it will wait until the not full semaphore to be signaled, then it will push the item onto the buffer and increment the count, it will then signal that the buffer is not empty and will unlock the mutex. When removing an item from the buffer it does a similar thing as inserting an item, but the opposite basically. The mutex will be locked, and then it will run a while loop to check if the buffer is empty and if it is, then it will wait until the not empty semaphore is signaled, it will then set the item parameter to the first element in the queue and then pop the first element off the queue and then decrement the count. It will then signal that the buffer is not full and then unlocks the mutex. The main function takes the program arguments for the sleep time, number of producers and the number of consumers. It will then create the pthreads for the producers and consumers and then will create the correct number of producer and consumer threads based on program argument input from before. Then the main function will sleep for however long the user told it to sleep for and then will exit.

## System Implementation

I did run into some problems during my implementation, one issue was figuring out how to actually use the mutex locks and the semaphores. I was able to figure it out though and decided to instead add the synchronization code into the Buffer implementation instead of in the main program. I also was making a big mistake by trying to create the buffer data structure with a circular array instead of just using a queue. A lot of the issues that I was having was easily fixed by just using the std queue.

## Results

I believe that all the features that were required are included in my submission. I don’t think that there is anything that I really want to improve on with this coding project. I am very satisfied with the results of this project.

## Conclusion

I was able to solve the intend problem successfully. The program is able to successfully run multiple producer and consumer threads at a time. A lesson that I’ve learned from this assignment is to stop and think about the implementation of the program design before I actually start coding. An issue that I have is just starting with whatever idea I have first and then running with it and then going back later and fixing it when I realize that there is a better way to do it.