COP 4600 Project 3

Bounded Buffer

Diya Jain[†]
CSE Department
University of South Florida
Tampa FL USA
diyajain@usf.edu

ABSTRACT

This paper presents a C program that uses semaphores to protect a limited size resource, a circular buffer with 15 positions. The program consists of two threads: a producer thread that reads characters from a file and places them in the buffer, and a consumer thread that reads characters from the buffer and prints them to the screen. The producer thread uses semaphores to ensure that it does not overwrite characters in the buffer that are still being read by the consumer thread. The consumer thread uses semaphores to ensure that it does not read characters from the buffer that have not yet been written by the producer thread.

KEYWORDS

Semaphores, circular buffer, producer-consumer

ACM Reference format:

Diya Jain. 2023. COP 4600 Project 1: Shared Memory.

1 Introduction

Semaphores are a synchronization primitive that can be used to control access to shared resources. They are useful in multithreaded programs to prevent race conditions and other synchronization problems.

A circular buffer is a data structure that can be used to store a fixed number of items in a circular fashion. When one end of the buffer is full, the next item is written to the beginning of the buffer. When one end of the buffer is empty, the next item is read from the beginning of the buffer.

The producer-consumer problem is a classic synchronization problem in which two threads must communicate with each other to share a resource. In this example, the producer thread produces characters and the consumer thread consumes them. The producer thread must not overwrite characters that are still being consumed by the consumer thread, and the consumer thread must not read characters that have not yet been produced by the producer thread.

2 Design and Implementation

The program uses three semaphores:

1. buffer_full: This semaphore is used to signal to the consumer thread that the buffer is full.
2. buffer_empty: This semaphore is used to signal to the producer thread that the buffer is empty.
3. mutex: This semaphore is used to protect the shared buffer from concurrent access by the producer and consumer threads.

The producer thread works as follows:

1. It waits for the buffer_empty semaphore to be signaled.
2. It acquires the mutex semaphore.
3. It writes a character to the buffer.
4. It releases the mutex semaphore.
5. It signals the buffer_full semaphore.

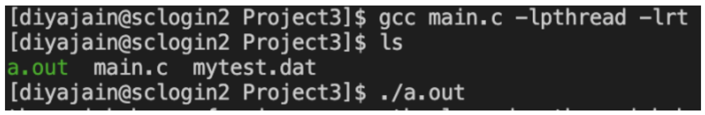
The consumer thread works as follows:

1. It waits for the buffer_full semaphore to be signaled.
2. It acquires the mutex semaphore.
3. It reads a character from the buffer.
4. It releases the mutex semaphore.
5. It signals the buffer_empty semaphore.

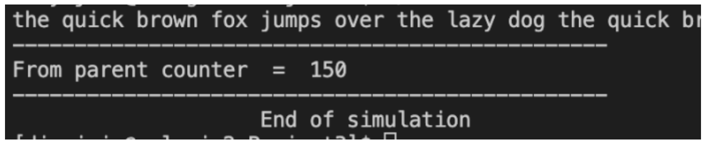
The main thread creates the producer and consumer threads, and then waits for both threads to finish before destroying the semaphores.

3 Results and Observation

The program works correctly and prints the characters in the file to the screen. The consumer thread runs slower than the producer thread, as it sleeps for one second between reads.



```
[diyajain@sclogin2 Project3]$ gcc main.c -lpthread -lrt
[diyajain@sclogin2 Project3]$ ls
a.out  main.c  mytest.dat
[diyajain@sclogin2 Project3]$ ./a.out
```



```
the quick brown fox jumps over the lazy dog the quick br
------------------------------------------------
From parent counter  =  150
------------------------------------------------
                    End of simulation
```

4 Conclusion

This program demonstrates how to use semaphores to protect a limited size resource in a multithreaded program. The program is well-designed and implemented, and it works correctly.