Report: Producer-Consumer Problem Using Semaphores

Ruben Osornio
CS 3113

## Abstract

This project implements the Producer-Consumer Problem using threads and semaphores in C. A shared circular buffer is used to manage communication between the producer and consumer threads. The producer reads characters from a file (mytest.dat) and writes them to the buffer. The consumer reads and prints the characters with a one-second delay to simulate slower processing. Proper synchronization is ensured using semaphores. This report evaluates the performance of the solution and provides an analysis of its behavior.

## Introduction

The Producer-Consumer Problem is a classic example of synchronization in operating systems. It demonstrates the importance of safely sharing resources in a multithreaded environment. In this implementation:

A producer thread reads characters from a file and places them in a circular buffer.
A consumer thread retrieves and prints the characters with a deliberate delay.
Semaphores are used to ensure mutual exclusion and proper synchronization.
Key challenges addressed include:

Managing a fixed-size circular buffer.
Synchronizing access to shared resources.
Ensuring that neither thread produces or consumes incorrect data.

## Design and Implementation

### Circular Buffer

The circular buffer has a fixed size of 15 positions:

Producer writes data at the in index.
Consumer reads data from the out index.
Both in and out wrap around when reaching the end of the buffer using modulo arithmetic (% BUFFER_SIZE).

### Semaphores

sem_items: Counts the number of items in the buffer, ensuring the consumer waits for data.
sem_space: Tracks available space in the buffer, ensuring the producer waits for free space.
sem_mutex: Provides mutual exclusion, preventing race conditions.

### Threads

Producer Thread:
Reads characters from mytest.dat until EOF.
Adds characters to the buffer, signaling the consumer.
Marks the end of data with a special character (*).

Consumer Thread:
Reads characters from the buffer and prints them.
Introduces a one-second delay between reads.
Stops when the EOF marker (*) is encountered.

Results and Performance Evaluation
Execution Steps
Compile the program using:
gcc project3.c -lpthread -lrt -o project3
Prepare the input file mytest.dat with test data:
Hello, this is a test for the producer-consumer problem!

Run the program:
./project3
Output
The program reads and prints the content of mytest.dat to the screen. Example output:
Hello, this is a test for the producer-consumer problem!
Program completed successfully.

Each character appears with a one-second delay, demonstrating proper synchronization.

Performance Metrics

| Input Size | Expected Execution Time | Observed Execution Time |
|---|---|---|
| 50 characters | ~52 seconds | ~52 seconds (estimate) |
| 55 characters | ~57 seconds | 57.009 seconds (measured) |

Analysis
Synchronization: The producer and consumer threads work seamlessly without data loss or duplication.
Timing: The one-second delay ensures the consumer operates slower than the producer.
Efficiency: The program efficiently handles the circular buffer, ensuring continuous operation without buffer overflows or underflows.

Conclusion
This implementation successfully demonstrates the Producer-Consumer Problem using semaphores for synchronization. Key takeaways include:

Proper use of semaphores (sem_items, sem_space, and sem_mutex) ensures safe sharing of resources.
The program handles EOF gracefully with a special marker (*).
The one-second delay effectively simulates a slower consumer.
Possible improvements:

```
cs096@cs096:~/project3$ gcc project3.c -lpthread -lrt -o project3
cs096@cs096:~/project3$ ./project3
Hello, this is a test for the producer-consumer problem!

Program completed successfully.
```