Lucas Ho

CS 3113: Intro to Operating Systems

29 April 2024

Project 3

Report Analysis on Semaphore-Protected Circular Buffer System

Objective: The primary goal of this project is to understand and implement semaphores to safeguard access to a limited-size resource, in this case, a circular buffer with 15 positions. This buffer facilitates communication between two threads: a producer that reads characters from a file, and a consumer that outputs these characters to the terminal.

System Design:

1. **Circular Buffer:**
    - **Size:** 15 characters.
    - **Purpose:** Act as a shared resource between producer and consumer threads.
2. **Producer Thread (thread_1):**
    - **Functionality:** Reads characters from the file "mytest.dat" until it reaches the end-of-file marker. Each character is written into the circular buffer, utilizing semaphores to manage the buffers state and to ensure mutual exclusion during write operations.
3. **Consumer Thread (thread_2):**
    - **Functionality:** Continuously reads from the circular buffer. It prints each character to the screen and processes characters with a one second delay to simulate a slower consumer. It terminates when it encounters the special end-of-file marker "*".
4. **Semaphores:**
    - **empty:** Indicates the number of empty positions in the buffer.
    - **full:** Indicates the number of filled positions.
    - **mutex:** Ensures that only one thread accesses the buffer at a time to avoid data corruption.

Implementation Details:

- The producer uses the semaphore empty to ensure it only writes when there is space available. After writing, it signals full to indicate that there is data to read.
- The consumer waits on full to ensure there is data to be processed, and after reading, it signals empty to indicate that more space is available.
- Both threads utilize mutex to access the buffer safely, ensuring that the producer does not overwrite data that has not been read by the consumer, and vice versa.

- The program uses a special character "*****" to signal the end of file. This mechanism is important for the consumer to know when to terminate properly.

Analysis of Synchronization:

- The use of semaphores is effective in preventing race conditions.
- Semaphore management ensures that the buffer operates within its bounds without overflow, which is critical given its limited size.
- The deliberate one-second delay in the consumer thread successfully simulates a scenario where the consumer is slower than the producer.

```
ho0055@gpel9:~/Project3$ ./main
Characters read by consumer:
project 3 out
End of Program.
ho0055@gpel9:~/Project3$ nano mytest.dat
ho0055@gpel9:~/Project3$ ./main
Characters read by consumer:
project 3 out 2nd try
End of Program.
ho0055@gpel9:~/Project3$
```

Conclusion: This project successfully demonstrates the application of semaphores in managing access to a shared circular buffer in a multi-threaded environment. The correct implementation of semaphores ensures that the buffer is neither accessed by both threads simultaneously or incorrectly managed for its capacity. The program meets the designed specifications and effectively simulates the dynamics between a faster producer and a slower consumer.