ECS 408/608 : Operaitng System
Critical Section Problem

Instructor: Dr. Sukarn Agarwal,
Assistant Professor,
EECS Department,
IISER Bhopal

March 22, 2025

Abstract

This assignment aims to familiarize you with critical section problem concept using semaphores. Note that you can use any one kernel synchronization: Windows, Linux and POSIX to solve the assignment.

Critical Section Problem

- Each process has critical section segment of code
  - Process may be changing common variables, updating table, writing file, etc.
  - When one process in critical section, no other process may be in its critical section.
- Critical section problem is to design protocol to solve this.
- Each process must ask permission to enter critical section in entry section, may follow critical section with exit section, then remainder section.

The genreal structure of process P_i.

```
while (true) {
entry section
critical section
exit section
remainder section
}
```

Problems

1. Consider a system with one parent process and two child processes, C1 and C2. There is a shared integer X initialized to 0. Process C1 increments X by two 10 times in a for loop. Meanwhile, process C2 decrements X by two 10 times in a for loop. After both C1 and C2 are finished, the parent process prints out the final value of X.

To solve this, declare a shared memory variable to hold X (see the calls `shmget()`, `shmat()`, `shmdt()`, and `shmctl` in Linux). Write the programs for processes C1 and C2. Ensure you do not place any synchronization code in the code for C1 and C2. You should write the code in such a way that you can simulate race conditions in your program by slowing down C1 or C2 appropriately by using `sleep()` calls at appropriate points. Note that if there is no race condition, the value of X finally should be 0. Simulating race condition means that if you run the program a few times, sometimes the final value of X printed by your program should be non-zero.

2. Add synchronization code based on semaphores to process C1 and C2 above so that race conditions are not possible. Use the call `semget()`, `semop()`, `semctl()` in linux to create and manage semaphore.

3. In this program, you will write a program to solve the m-producer n-consumer problem, m, n$\geq$ 1. You have a shared circular buffer that can hold 20 integers. Each producer process stores the numbers 1 to 100 in the buffer one by one (in a for loop with 100 iterations) and then exits. Each consumer process reads the number from the buffer and adds them to a shared variable SUM(initialized to 0). Any consumer process can read any of the numbers in the buffer. The only constraint is that the consumers should read every number written by some producer exactly once. However, a producer should not write when the buffer is full; a consumer should not read when the buffer is empty.

   Write a program that creates the shared circular buffer and the shared variable SUM using the shm*() calls in Linux. You can create any other shared variable that you may need. The program then reads in the value of m and n from the user and forks m producers and n consumers. The producer and consumer codes can be written as functions that are called by the child processes. After all the producers and consumers have finished (the consumers exit after all the data produced by all the producers have been read. How does a consumer know this?), the parent process prints the value of SUM.
   Test your program with at least (a) m=1, n=1, (b) m=1, n=2, (c) m=2, n=1, and (d) m=2, n=2.

Assignment Submission Instruction:

- Based on your observed result, create a report with your name and roll number with assignment ID (For example, Sukarn_Agarwal_21056_Assign5.pdf).
- Submitted pdf file contains answers to all the questions above in order. **Any out-of-order answer results in a zero mark.**
- With each question, attach all the screenshots you observed on your screen. Make sure that your name should appear there. **If TA finds any discrepancy in this, zero marks will be awarded.**
- Use of CC computers are prohibited for solving the assignment.

- **The last date to submit the assignment is by the end of day of 29 March 2025 (IST). Any late submission results in the 0 Marks and no requests to be entertained.**