Dealing with Classical Problems

CSL301 - Operating System Lab
Class Assignment 9

October 29, 2025

Q1: Identify Race Condition and Fix with Mutex

Problem Statement

You are given the following producer-consumer code. This code has an issue that can cause data corruption when multiple threads access shared variables simultaneously. Run the code multiple times and observe the final counts. Identify the issue and fix it by adding a mutex for mutual exclusion.

Given Code: `pc_bug.c`

Q2: Add Finite Buffer Constraint

Problem Statement

The previous solution still has a limitation: the buffer is treated as unbounded, meaning producers can keep adding items and potentially overwrite unconsumed data when the buffer wraps around. Modify the code to handle a finite buffer where producers must wait when the buffer is full.

Requirements

- Buffer has a fixed capacity
- Producers must block when all slots are full
- Consumers must block when buffer is empty (Note: already ensured in previous solution)
- Use two semaphores to track empty and full slots

Q3: Identify and Fix Deadlock

Problem Statement

The following code contains a deadlock scenario where threads can permanently block each other. Identify why the deadlock occurs and fix it.

Given Code: `pc_deadlock.c`

Q4: Identify Error in Basic Readers-Writers and Fix It

Problem Statement

You are given a readers-writers implementation where multiple readers can access shared data simultaneously, but writers need exclusive access. The code below has a critical bug that can cause data corruption. Identify the error and provide the correct solution.

Given Code: `rw_bug.c`

Q5: Implement Readers–Writers using Lightswitch Pattern

Problem Statement

The Lightswitch pattern allows multiple readers to access a shared resource simultaneously, while ensuring that writers have exclusive access when needed. Implement a solution using semaphores where:

- Multiple readers can enter the critical section concurrently.
- Only the first reader locks the writers out.
- Only the last reader releases the writers.
- Writers require exclusive access to the shared resource.
- update `void *reader(void *arg)` and `void *writer(void *arg)` code

Given Code: `lightswitch.c`

Q6: Identify and Fix Deadlock in Dining Philosophers

Problem Statement

Five philosophers sit at a round table with five forks (one between each pair of adjacent philosophers). Each philosopher alternates between thinking and eating. To eat, a philosopher needs both the left and right forks. The code below implements this but has a deadlock problem. Run the code, observe the deadlock, identify why it happens, and fix it using the Lower ID First strategy.

Given Code: `dp_deadlock.c`

Q7: Dining Philosophers with User-Defined N (Optional)

Problem Statement

Implement a deadlock-free dining philosophers solution where the user can specify the number of philosophers (n). The solution must work correctly for any $n \geq 2$.

Submission Checklist

- Ensure your code compiles without warnings or errors.
- The program should run and produce correct output.
- Submit all files in a single compressed folder.
- Attach screenshots of your final output in the report.

Good Luck!