Operating System Lab
End Semester Examination

Answer any one of the questions. Submission deadline 11:50 AM. Once you have downloaded this question paper, you can not use browser till the time of submission. Plagiarism will be check and penalised.

Question 1

Implement a synchronization solution for a unisex restroom that enforces specific rules to ensure fairness and avoid awkward situations. This problem focuses on mutual exclusion based on categories and condition synchronization.

Problem Description:

Imagine a single unisex restroom. To maintain decorum, it has the following rules:

1. It cannot be used by men and women simultaneously.
2. There is no limit on the number of people of the same gender who can be inside simultaneously (assume infinite capacity for simplicity, though the logic should work even with a capacity limit).
3. Once someone of a particular gender enters, others of the same gender can enter freely as long as the restroom isn't empty.
4. When the restroom becomes empty, the next person to enter (regardless of gender) determines the gender allowed entry until it becomes empty again.
5. Prevent starvation – ensure that if people of one gender are waiting, they eventually get to use the restroom, even if people of the currently allowed gender keep arriving.

Requirements:

Implement a simulation using threads (in C with Pthreads and).

Create multiple threads representing men and women attempting to use the restroom.

Implement functions like woman_wants_to_enter(), man_wants_to_enter(), woman_leaves(), man_leaves().

Inside these functions, use synchronization primitives (mutexes, or semaphores) to enforce the rules.

Each thread should print messages when it wants to enter, when it successfully enters, and when it leaves. The output should clearly demonstrate that the rules are being followed (e.g., "Man [ID] wants to enter.", "Woman [ID] enters. Women inside: 2.", "Man [ID] leaves. Men inside: 0. Restroom empty.").

Simulate for a reasonable duration or number of entry/exit events.

Implement the fairness enhancement (rule 5). This might involve tracking waiting counts or using separate condition variables.

Hints:

You'll need a mutex to protect the shared state variables (counts, status).

You'll likely need semaphores used for signaling for threads to wait when they cannot enter (e.g., a man waiting because women are inside, or vice-versa, or waiting due to the fairness rule).
Think carefully about when to signal waiting threads (e.g., when the last person of a gender leaves).

Evaluation Criteria:

- Correctness (rules enforced), 10
- proper use of synchronization primitives, 5
- clear output demonstrating behaviour, 5
- starvation free, 10

Question 2:

This assignment on virtual memory is designed to provide a comprehensive understanding of how modern operating systems manage memory efficiently.

Tasks:

You have to implement a paging system using a page table to map virtual addresses to physical addresses and employ the Least Recently Used (LRU) algorithm to handle page faults. This involves simulating a virtual memory system in C, accepting memory access requests, and displaying the page table along with the number of page faults. The page table will contain the valid invalid bit.

The assignment also covers demand paging, where pages are loaded from disk only when needed, reducing memory usage and improving system performance. A critical component of the assignment is the performance analysis, where students evaluate the LRU algorithm's effectiveness in terms of page faults. They compare LRU with other algorithms like FIFO and Optimal page replacement.

The grading criteria, code correctness and efficiency (20 marks), performance analysis (10 Marks).