National University of Computer and Emerging Sciences

Chiniot-Faisalabad Campus

Lab 10

CL2006 – Operating System - Lab

Course Instructor	Juhinah Batool
Lab Instructor	Juhinah Batool
Semester	Fall 2024

FAST School of Computing

Department of AI & DS

Instructions

1. Make a PDF document with the convention "ROLLNO_ LAB#_ SECTION" and put all your source code and snapshots of its output in it.
2. Plagiarism is strictly prohibited, if you take a code snippet off the internet, mention its reference.
3. Do not discuss solutions with one another. Copying the solution from any source can lead to ZERO marks.

Lab Task: Implementing Binary and Counting Semaphores

Task 1: Basics of Semaphores (30 minutes)

1. **Objective**: Understand binary semaphores and basic semaphore operations (wait and signal).
2. **Description**: Implement a basic binary semaphore to control access to a shared resource.
    - Define a binary semaphore and initialize it to control access to a shared counter.
    - Implement two functions that:
        - **Increment** the counter (requires locking the resource).
        - **Decrement** the counter (requires unlocking the resource).
    - Use pthread library functions to create two threads, one incrementing and the other decrementing the counter.
    - Print the counter's value at each step to demonstrate controlled access.
3. **Expected Output**: The counter should only change when a thread has access, showing how binary semaphores prevent race conditions.

Task 2: Implementing Counting Semaphores (1 hour)

1. **Objective**: Use counting semaphaskores to manage multiple instances of a resource.
2. **Description**: Implement a counting semaphore to manage access to a limited number of resources (e.g., three printers).
    - Initialize a counting semaphore with a count of 3, representing the three available printers.
    - Create multiple threads (e.g., 6) to simulate print jobs that require access to a printer.
    - Each thread should:
        - Wait until a printer becomes available.
        - "Use" the printer for a short duration (simulated with a sleep function).
        - Release the printer when done.
    - Track and print the usage of each printer to show that only three threads (print jobs) can use the printers at any given time.
3. **Expected Output**: At any time, no more than three threads should be accessing printers simultaneously.

Task 3: Advanced Synchronization Problem (1 hour 30 minutes)

1. **Objective**: Apply binary and counting semaphores to solve a complex synchronization problem.
2. **Problem Description**: Implement a producer-consumer model with a bounded buffer of fixed size (e.g., 5).
    - Use a counting semaphore to track available slots in the buffer.
    - Use another counting semaphore to track the number of items in the buffer.
    - A binary semaphore should manage mutual exclusion while accessing the buffer.
3. **Instructions**:
    - Create two threads: one for the producer and one for the consumer.
    - The **Producer**:
        - Waits for an available slot.
        - Produces an item and adds it to the buffer.

        - Signals that an item is now available in the buffer.
    - The **Consumer**:
        - Waits for an item in the buffer.
        - Consumes the item and removes it from the buffer.
        - Signals that a slot is now available.
    - Print statements should display the buffer's state at each step, showing items being added and removed.
4. **Expected Output**: The buffer should maintain synchronization without overflows or underflows, showing how binary and counting semaphores work together in a bounded buffer scenario.