Bridge Crossing Simulation with Semaphores and Mutex

Introduction

This project aims to simulate the movement of cars approaching a bridge from both the left and right sides, ensuring they cross the bridge safely and without collisions. It utilizes semaphores and a mutex to synchronize car crossings and prevent data races.

Problem Description

The bridge crossing problem involves managing the movement of multiple cars approaching a single-lane bridge from both the left and right sides. The goal is to ensure that cars from opposite directions do not occupy the bridge simultaneously, preventing collisions and ensuring smooth traffic flow.

Solution Overview

The code implements a bridge crossing simulation using semaphores and a mutex to achieve synchronized and collision-free car movement.

Semaphores

Semaphores are synchronization primitives that control access to shared resources. In this simulation, semaphores are used to manage the number of cars from each side that can be on the bridge at a given time.

- bridgeSem: This semaphore controls overall access to the bridge, allowing a maximum of one car to be on the bridge at a time. It is initialized to a value of 1, representing the bridge's capacity.
- leftSem: This semaphore manages the number of cars from the left side that can be on the bridge at a given time. Only one car from the left side can be on the bridge at a time. It is also initialized to a value of 1.
- rightSem: Similarly, this semaphore manages the number of cars from the right side that can be on the bridge at a given time. It is also initialized to a value of 1.

Mutex

A mutex is a synchronization primitive that ensures that only one thread can access a shared resource at a time. In this simulation, a mutex is used to protect the critical section of code

where car counts are updated. This prevents data races, which occur when multiple threads try to access and update the same shared resource simultaneously.

- mutex: This mutex protects the critical section of code where the leftCount and rightCount variables are updated. These variables track the number of cars from each side that are currently on the bridge. The mutex is acquired before entering the critical section and released after exiting the critical section. This ensures that only one thread can update the car counts at a time, preventing data races.

Function Summary

- passing(int direction, int carNum): Simulates a car crossing the bridge, taking the car's direction and number as parameters.
- left(): Represents a car approaching the bridge from the left. It performs the following steps:
    1. Waits for a slot using `leftSem`.
    2. Acquires `mutex` to protect the critical section.
    3. Updates `leftCount` to reflect its presence on the bridge.
    4. Waits for the bridge to be clear using `bridgeSem`.
    5. Simulates crossing using `passing(0, carNum)`.
    6. Releases `bridgeSem` and `mutex`.
    7. Signals an available slot using `leftSem`.
    8. Exits the thread.
- right(): Similar to `left()`, but manages cars from the right side using `rightSem` and updates `rightCount`.
- main(): Controls the main program flow:
    1. Prompts for the number of cars from the left and right sides.
    2. Checks if the total number of cars exceeds the limit.
    3. Initializes semaphores and mutex.
    4. Creates threads for each car using `pthread_create()`.
    5. Waits for all threads to finish using `pthread_join()`.
    6. Destroys semaphores and mutex.
    7. Exits the program.

Testing Strategy

To ensure the code's effectiveness, run it with varying numbers of cars from both sides. Verify that cars from opposite directions do not simultaneously occupy the bridge, ensuring collision-free traffic flow.

Conclusion

This project effectively demonstrates the use of semaphores and a mutex to synchronize concurrent processes in the bridge crossing problem. Semaphores control access to the shared resource (bridge) and prevent data races, while the mutex protects the critical section of code where car counts are updated. This combination ensures safe, collision-free car movement across the bridge.