CSEN 383 – Assignment 3
Spring 2024

Group 4

Piyush Joshi
Sudarshan Mehta
Patrick Lee
Manasa Madiraju

Objective

This project offers practical learning in building a multithreaded program using the pthread library in C. Our program uses the pthread library to generate threads and mutexes to simulate multiple ticket sellers concurrently selling concert tickets for an hour.

Software Design

Upon program initialization, we establish 10 customer queues, each capable of accommodating N customers. These queues store customer details such as randomly generated customer numbers, arrival times, and service times. Arrival and service times are measured in minutes, with all customers arriving at the start of each minute. To represent one minute, we are incrementing a counter called current_time_slice by 1 which represents a minute passing by. Then, each of the ten sellers is assigned a queue and starts processing customers as per their scheduled arrival. Additionally, each seller is categorized as L, M, or H, determining their processing speed and seat selection strategy. Our shared data is the available concert seats matrix. This means our critical region for our code is the threads trying to access the concert seats matrix whenever a seller is trying to reserve a seat to a customer. As a seller begins serving a customer, it searches for an available seat based on its type, with open seats indicated by an unlocked mutex to ensure only one seller can claim a specific seat at a time. We added mutex locks to make sure no two sellers try accessing the same seat to sell to a customer. This will enable process synchronization throughout the simulation. Once a seller secures a seat mutex, it records details at that seat location, including response time, turnaround time, and seller type. This process continues until all seats are occupied or an hour has passed, with any remaining unserved customers turned away. After the simulation ends, sold seats are analyzed to calculate the average response time, turnaround time, and throughput for each seller type.

The program shares common variables across all functions, including the aggregate count of customers per seller, the global clock counter, pthread arguments, and seat-specific structure variables. We utilized the provided code snippet as a basis, implementing our task by simulating clock ticks using the main thread. This approach allows us to manage critical regions and execute the selling process in child threads dedicated to simulating ticket sales. In our project, the main thread generates clock ticks at regular intervals, simulating a one-minute time duration for each tick.

Presumptions

1. Clock Tick: The smallest unit of time is one minute, and each child thread simulates tasks lasting exactly one minute, including serving customers, waiting for their completion, or finalizing a sale.

2. Seller's Thread State: Each seller's thread can be in one of the following states at any given time:
   - Waiting: Awaiting a new client.
   - Serving: Actively serving a client from the seller's queue.
   - Processing: Engaged in processing and dedicating time to complete the sale.
   - Completing: Finalizing the sale for the client.

3. New Clock Generation: A new clock is generated when all seller threads have finished their tasks for that time unit to maintain time synchronization.

4. Concert Seat Simulation: Concert seats are represented by a 2D matrix. To prevent contention in seat assignment, only one thread manipulates the matrix. The overall workflow of the project is as follows:

- Initialization: Initialize parameters such as mutex locks, display seat matrix, customer queues for each seller, and threads.
- After creating seller threads, they all enter a waiting state for the next clock tick after initialization.
- Simulating Clock Tick: When the main thread triggers a clock tick, all seller threads compete to acquire a lock on the concert seat matrix based on their current state. Each seller thread checks for new customers based on arrival time and serves them one by one.
- Termination Condition: Each seller thread stops processing if either the concert is sold out or the simulation time has expired, requiring the customer to leave.