Advanced Operating Systems
COEN 383
Project - 3
Group 5
Snehal Nikam
Sarthak Patel
Bharadwaj Parthasarathy
Raksha Narasegowda
Smit Patel

Objective

The objective of the project is to apply multithreading concepts and synchronization techniques in a real-world simulation of concert ticket sales, where various aspects of the ticket-selling process, customer arrivals, and seat assignments need to be implemented and analyzed.

Utilizing the template code from the project preview, we constructed our project by using the main thread to simulate the ticking of the clock and the child threads used for simulating ticket sales to regulate crucial regions and sell processes.

The primary thread was responsible for generating clock ticks at one-minute intervals to represent the passage of time during the simulation. When running the simulation, the following assumptions were considered:

1. Clock Ticks: The smallest unit of time used in the simulation is equivalent to one minute. Each child thread is designed to perform its tasks in increments of these one-minute time intervals. These tasks include serving customers, waiting for sales to complete, and finalizing the sales process.

2. Seller Thread States: Each seller thread is assumed to be in one of four states during a given time interval:
   a. Waiting: Waiting for a new customer to arrive from the seller's queue.
   b. Serving: Actively assisting a customer.
   c. Processing: Representing the time required for processing the sale.
   d. Completing: Finalizing a sale with a customer.

3. A new clock tick is generated each time a seller completes a transaction to ensure that the time intervals remain synchronized across various sales.

4. The seating arrangement for the concert is emulated using a two-dimensional matrix, with the assumption that only one thread will be modifying this array at any given moment. Mutex locks are employed to implement this to ensure mutual exclusion and prevent conflicts when accessing the matrix.

Project Workflow

1. Initialization: We started by setting up the initial parameters, including mutex locks, the concert seating matrix, customer queues for each seller, and the threads.

2. After creating the seller threads, we placed all of them in a waiting state, ready to respond to clock ticks, following their initialization.

3. Simulating Clock Ticks: Simulating clock ticks was a challenging part of the process since we needed to ensure that a clock tick signal would be sent only when all threads had completed their tasks for that time interval. To achieve this, we made the main thread wait by keeping track of the number of threads that were still actively processing.

4. When the main thread triggered a clock tick, all the seller threads raced to obtain a lock on the concert seat matrix based on their respective states.

5. Initially, each seller thread checked for new customers by considering their arrival times and began serving customers one at a time.

6. To mimic the passage of time in completing sales, we introduced a time delay, which was reduced with each clock tick. The time delay was randomly generated when new customers arrived, and when it reached zero, the sale for that particular thread was completed.

Note

The project's design guarantees that seat allocation is carried out sequentially, but the actual processing of customers occurs concurrently. As a result, it's feasible for all 10 sellers to simultaneously process customers. This parallel processing is evident in the output, where the number of customers being served concurrently can reach as high as 8 at any given time.