COEN 383 - Advance Operating Systems

Report - Project 3

Group – 1

1) Tishyaketu Atul Deshpande (07700000172)
2) Atharva Abhay Bal (07700005835)
3) Sowmya Baluvu (07700011659)
4) Ziang Chen (07700006146)
5) Simran Dhawan (07700011429)

Introduction:

We began our project using the given template code, enhancing it by simulating clock ticks via the main thread while delegating the management of critical regions and ticket sales to child threads. In essence, this C-based program is designed to create a multi-threaded environment that simulates time through clock ticks in the primary thread. Simultaneously, it processes ticket sales using separate child threads. Thread creation is handled via the pthread library, and synchronization is ensured with the use of mutexes.

Operational Details:

- **Clock Ticks:** The fundamental time unit in the simulation is a minute. Each child thread performs tasks like assisting customers, waiting for the end of sales, or processing transactions within these time intervals.
- **Thread States:** During each time unit, seller threads can occupy various states:
    - **Waiting:** Awaiting the next customer in the queue.
    - **Serving:** Actively helping a customer.
    - **Processing:** The time taken to complete a sale.
    - **Completing:** Finalizing a transaction with a customer.
- **Clock Synchronization:** A new tick is generated whenever a seller finalizes a transaction, ensuring that the time quantum remains consistent across all sales processes.
- **Concert Seating Simulation:** The concert seating is modelled as a two-dimensional array, with mutex locks guaranteeing that only one thread can modify the array at any given moment.

Project Workflow:

1. **Initialization:** We began by configuring the necessary parameters, which included setting up mutex locks, creating a seating matrix for the concert, organizing customer queues for each seller, and initializing threads.
2. **Thread Setup:** Once the threads were created, they entered an idle state and remained there until the initialization process was complete.
3. **Clock Tick Simulation:** This vital part of the system ensures that a clock tick signal is generated only after all threads have finished their tasks for the current time unit. The main thread pauses until all threads are synchronized.
4. **Thread Operation:** When a clock tick occurs, all seller threads attempt to secure a lock on the seating matrix, depending on their current status.
5. **Customer Handling:** Initially, seller threads review incoming customers based on their arrival time and then proceed to serve them one by one.
6. **Sales Completion:** To mimic the delay involved in completing sales, a random time delay is assigned to each thread when a customer arrives. The sale is completed once this delay reaches zero.

Conclusion: The project design enables serialized seat assignments while concurrently processing customer requests. This setup allows for up to eight customers to be served at the same time, as confirmed in the output.

Output generation:

>> gcc -std=c99 *.c -lpthread -o main

>> ./main <N> , N can be 5, 10 or 15