Software Design Report – Ticket Selling Simulation

Overview

This project implements a multi-threaded ticket selling simulation using C and POSIX threads (pthreads). The goal is to model a realistic concert ticket-selling environment where multiple sellers operate concurrently, serve customers over time, and compete for a limited shared resource (seats in a venue).

The simulation runs for 60 simulated minutes, during which sellers process customer queues, assign seats, and track performance metrics such as response time, turnaround time, and throughput.

System Design

Sellers and Threads

- The system consists of 10 sellers, each implemented as a separate thread:
    - 1 High-price seller (H)
    - 3 Medium-price sellers (M)
    - 6 Low-price sellers (L)
- Each seller maintains its own customer queue.
- Sellers differ in service speed and seat selection strategy, which helps simulate real-world priority differences.

Venue

- The venue is modeled as a 10×10 seating chart (100 seats total).
- Seats are stored in a shared 2D array in memory.
- Seat assignment depends on seller type:
    - H sellers fill seats from the front rows first
    - M sellers fill from the middle outward
    - L sellers fill from the back rows forward

Parameters Adjusted for Realism

Several parameters were chosen to make the simulation realistic:

1. Customer Arrival Time
   - Each customer is assigned a random arrival time between 0 and 59 minutes.
   - This models customers arriving throughout the selling window instead of all at once.
2. Service Time by Seller Type
   - High seller: 1–2 minutes
   - Medium seller: 2–4 minutes
   - Low seller: 4–7 minutes
   - Faster service for high-priority sellers reflects real-world preferential treatment.
3. Time-Step Simulation
   - The simulation advances in discrete one-minute intervals.
   - All sellers operate synchronously at each minute to maintain a consistent timeline.

Shared Data and Critical Regions

Shared Data

The following data structures are shared among multiple threads:

- Venue seating chart
- Total seats sold counter
- Console output (logs)

Critical Regions

Critical regions occur when multiple threads may:

- Assign seats
- Update the number of seats sold
- Print logs or seating charts

To prevent race conditions, these regions are protected using a mutex associated with the venue.

Process Synchronization

Mutex Synchronization

- A `pthread_mutex_t` is used to protect the venue structure.
- This ensures that only one seller can modify the seating chart or seat count at a time.

Barrier Synchronization

- Two barriers are used:
  - `barrier_start` ensures all sellers begin each simulated minute together.
  - `barrier_end` ensures all sellers finish processing before moving to the next minute.
- Barriers maintain a consistent notion of simulated time across all threads.

This combination of mutexes and barriers ensures correctness, fairness, and determinism in the simulation.

Statistics and Reporting

Each seller tracks:

- Customers served
- Customers turned away
- Total response time (start time − arrival time)
- Total turnaround time (finish time − arrival time)

At the end of the simulation, a final report is printed to the console summarizing overall performance, including throughput.

Conclusion

This design demonstrates effective use of multithreading, synchronization primitives, and shared-memory coordination to model a real-world ticket-selling system. The use of barriers enforces time-step simulation, while mutexes ensure safe access to shared resources, resulting in a realistic and correct concurrent system.
