COEN383 Advanced Operating System Group No. 3 Report

Name:
Aditya Kanodia(W1650366)

Pujitha Kallu(W1653660)

Ching Yueh Huang(W1649844)

Abdullah Khan(W1652477)

Almas Khan(W1620934)

Objective:

The objective of this C program aims to create a multi-threaded system that simulates the progression of time through clock ticks in the main thread. Concurrently, it manages ticket sales using child threads. The implementation utilizes the pthread library for thread creation and mutex for synchronization.

Assumptions:

1. The simulation operates with a minimum time quanta equivalent to one minute, and each child thread progresses its work in increments of these time quanta.i.e.. The simulation runs for 60 time units.
2. Seller threads are always in one of the following states:
   Sellers serve customers in a round-robin fashion. Each seller type has a different wait time for serving a customer. Seats are assigned to customers, and the venue's seating matrix is updated.
   a. Waiting: Awaiting a new customer arrival from the seller queue.
   b. Serving: Actively serving a customer.
   c. Processing: Engaged in the processing time for the sale.
   d. Completing: Finalizing a sale with the customer.
3. Seating arrangements are simulated using a 2D matrix, assuming that only one thread can sell tickets for a particular row at any given time. Mutex locks are employed to enforce this constraint and ensure thread safety.

Design:

1. The system is initialized with parameters including mutex locks, seat matrix, customer queues for each seller, and thread instances.
2. After creating seller threads, they enter a waiting state for a clock tick signal, indicating the end of initialization.

3. Clock ticks are used to synchronize thread actions, ensuring that the signal is sent only when all threads have
   completed their tasks for the current time quanta.
4. Upon receiving a clock tick, seller threads compete to acquire a lock on the concert seat matrix based on their
   current state.
5. Seller threads check for new customers based on arrival time and serve them sequentially.
6. A random time delay is implemented for each customer, decrementing with each clock tick until the sale for that
   seat is completed.

In summary, the design ensures sequential seat allocation while allowing concurrent processing of customers by all seller threads.

Critical Regions:
Adding and removing nodes from the linked list (e.g., add_node, remove_data, remove_head, add_after).
Sorting the linked list (sort).

Process Synchronization:
Synchronization mechanisms (mutex locks) added in the actual ticket-selling simulation logic to ensure thread safety.

Outputs

For E.g with N=5

```
Final Seat Allocation
=========================
 H101    H102    H103    H104    H105       -       -       -       -       -
    -       -       -       -       -       -       -       -       -       -
    -       -       -       -       -       -       -       -       -       -
    -       -       -       -       -       -       -       -       -       -
    -       -       -       -       -       -       -       -       -       -
 M301    M101    M102    M201    M103    M202    M302    M104    M105    M203
 M303    M304    M204    M205       -       -       -       -       -       -
    -       -       -       -    L605    L205    L105    L404    L505    L604
 L104    L204    L403    L603    L302    L504    L203    L602    L503    L202
 L402    L502    L301    L401    L103    L501    L601    L201    L102    L101


Stat for N = 05
===============
 ============================================
|    | No of Customers | Got Seat | Returned |
 ============================================
|  H |               5 |        5 |        0 |
|  M |              15 |       14 |        1 |
|  L |              30 |       26 |        4 |
 ============================================

Average TAT is 30.00
Average RT is 34.60
Throughput of seller H is 0.43
Throughput of seller M is 0.23
Throughput of seller L is 0.08
(base) adityakapadia@Adityas-MacBook-Pro-2 source_code %
```

Fig 1: Seats Allocation for N=5