Santa Clara University

Department of Computer Engineering

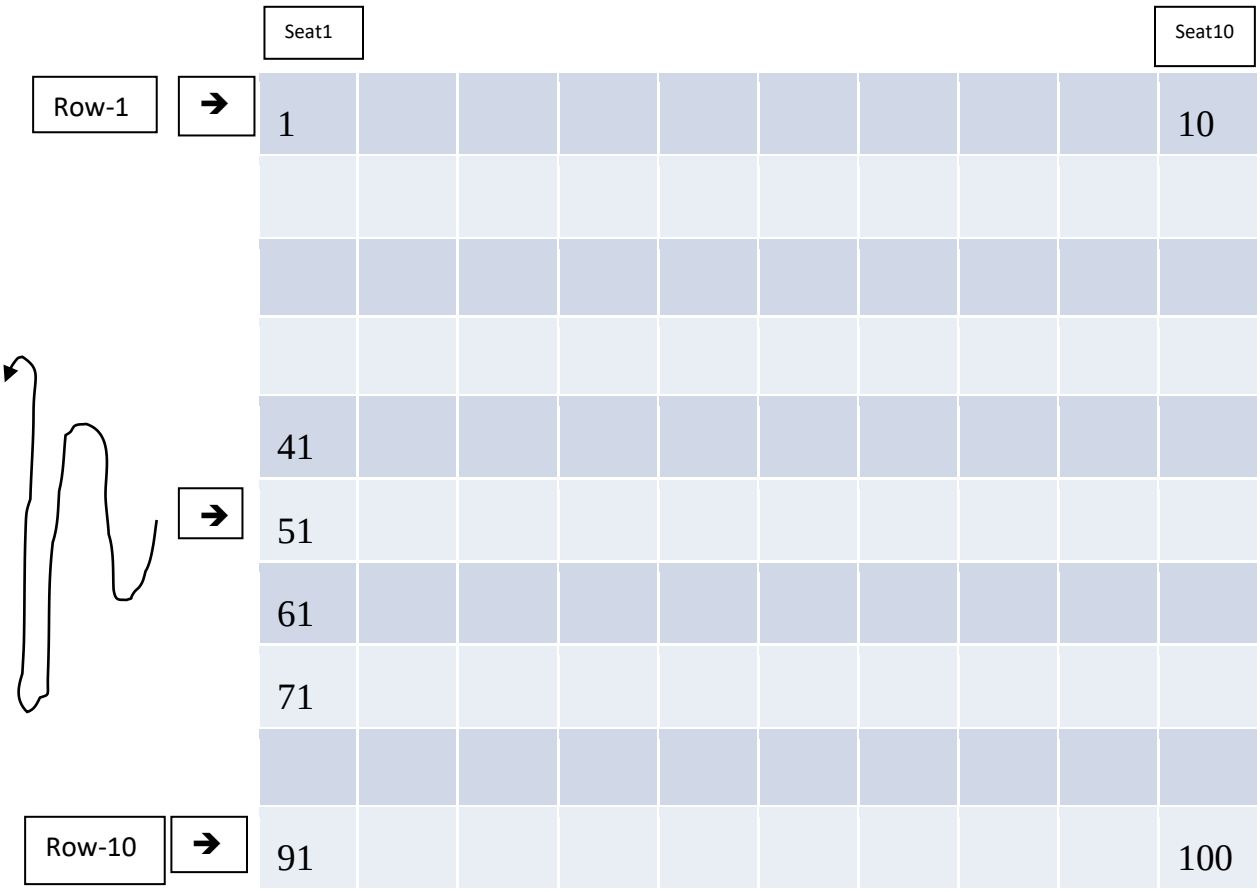
Advanced Operating Systems (COEN 383)

Project-3 Preview (6 pts)

Instructor: Ahmed Ezzat

Multi-threaded Ticket Sellers

We will build simulation written in C or C++ or Java programming language that experiment with 10 ticket sellers to 100 seats concert during one hour. Each ticket seller has their own queue for buyers.



H1 (High) Seller: one seller and sell tickets starting from row-1

(M1, M2, M3) Medium Sellers: 3 sellers start selling from row 5 then 6, then 4, then 7, etc.

(L1, L2, L3, L4, L5, L6) Low Sellers: 6 sellers start selling from row 10 then row-9, etc.

Each seller has in their queue N customers arriving at random time during the one hour; where N is a command-line option.
No 2 sellers can sell the same seat to different customers

Time for H-Seller to complete the ticket sale is random either 1 or 2 minutes.

Time for M-Seller to complete the ticket sale is random either 2 or 3 or 4 minutes.

Time for L-Seller to complete the ticket sale is random either 4 or 5 or 6 or 7 minutes.

Simulator:

1. Assume 10 threads, each represents a ticket seller: H1, M1, M2, M3, L1, L2, L3, L4, L5, L6.
2. Each seller has their own queue and customer stands in the queue using FIFO.
3. Initialize the simulation clock – initially to zero (0:00)
4. Create 10 threads and suspend them.
5. Think of the simulation as main() that include:
    a. Create the necessary data structures including the 10 queues for the different sellers and initialize each queue with its ticket buyers up front based on the N value.
    b. Create the 10 threads and each will be set with a sell() function and seller type as arguments.
    c. Wakeup all 10 threads to execute in parallel; wakeup_all_seller_threads();
    d. Wait for all threads to complete
    e. Exit
6. Output:
    a. Your C or C++ or Java **source files**.

b. A text file containing **output** from your simulation runs.
c. **A 1- or 2-page report** that describes your software design. What parameters did you adjust to make the simulation run realistically? What data was shared and what were the critical regions? What process synchronization was required?
d. **Average response time per customer for a given seller type, average turn-around time per customer for a given seller type, average throughput per seller type**

```
#include <stdio.h>
#include <pthread.h>

pthread_cond_t     cond     = PTHREAD_COND_INITIALIZER;
pthread_mutex_t   mutex = PTHREAD_MUTEX_INITIALIZER;

// A thread can find its own-id: thread-id = pthread_self();
// How a thread can find out about its own customers' queue?
// option-1: thread can find out its own thread-id and check the tid[] array to find the index
// in the workload array where each entry is the head of the customers linked list.
// option-2: when creating the thread, pass the last argument as the head of that thread's
// customers linked list…
// option-3: when creating the thread, pass a struct that includes the index "i" and  seller_type

// seller thread to serve one time slice (1 minute)
void *  sell(void  *s_t)
{
   While (having more work todo)
   {
        seller_t = (char *) s_t;
        pthread_mutex_lock(&mutex);
        pthread_cond_wait(&cond, &mutex);                    // wait until next buyer comes for this seller
        pthread_mutex_unlock(&mutex);
        // Serve any buyer available in this seller queue that is ready
        // now to buy ticket till done with all relevant buyers in their queue
        ……
    }
    return NULL;                  // thread exits
}


void wakeup_all_seller_threads()
{
   pthread_mutex_lock(&mutex);
   pthread_cond_broadcast(&cond);
```

```
    pthread_mutex_unlock(&mutex);
}
int main()
{
    int i;
    pthread_t   tids[10];
    char  Seller_type;

    // Create necessary data structures for the simulator.
    // Create buyers list for each seller ticket queue based on the
    // N value within an hour and have them in the seller queue.

    // Create 10 threads representing the 10 sellers.
    seller-type = “H”;
    pthread_create(&tids[0], NULL, sell, &seller-type);

    seller-type = “M”;
    for (i = 1; i < 4; i++)
        pthread_create(&tids[i], NULL, sell, &seller-type);

    seller-type = “L”;
    for (i = 4; i < 10; i++)
        pthread_create(&tids[i], NULL, sell, &seller-type);


    // wakeup all seller threads
    wakeup_all_seller_threads();

    // wait for all seller threads to exit
    for (i = 0 ; i < 10 ; i++)
          Pthread_join(&tids[i], NULL);

    // Printout simulation results
    ……

    exit(0);
}
```