**CS 5348**

**Operating Systems Concepts**

**Project 2**

**Threads**

**Summary**

**Name:** Madhusudan Pranav Venugopal

**UTD ID:** 2021163441

**netid:** mxv130430

**Project purpose:**

The purpose of the project was to study the coordination on multiple threads using Unix operating systems semaphores. The aim of the project was to simulate the working of a theatre with 300 customer threads, 2 box office agent threads, 1 ticket taker thread and 1 concession stand thread

The functions of each of the threads are as follows:

Customer:

1. 300 customers are in line at the box office when the simulation starts (1 thread per customer).
2. Movie titles and seats available are read from an input file.
3. Each customer randomly chooses a movie.
4. If the movie the customer wants to see is sold out, the customer leaves.
5. The customer must see the ticket taker before entering the theater’s main lobby.
6. Each customer randomly chooses whether to visit the concession stand (50% chance to visit it).
7. Each customer randomly visiting the concession stand randomly chooses popcorn, soda, or both.

Box Office Agents:

1. Two created initially, one thread each.
2. Serves next customer in line.
3. Maintains count of tickets sold for each movie—does not sell ticket if movie is sold out.
4. If movie is sold out, this is communicated to the customer.

Ticket Taker:

1. Created initially, one thread.
2. Serves next customer in line.
3. Takes and tears ticket.

Concession Stand Worker:

1. Created initially, one thread.
2. Serves next customer in line.
3. Gets order from customer.
4. Fills order.

Main

1. Creates all threads and joins all customer threads.

2)When last customer has exited, ends the simulation

The other rules which the project was supposed to satisfy are as follows:

1. Some actions take time (see table). The thread should sleep 1 second for each 60 seconds listed in the table.
2. Each activity of each thread should be printed with identification (e.g., customer 1).
3. All mutual exclusion and coordination must be achieved with semaphores.
4. A thread may not use sleeping as a means of coordination.
5. Busy waiting (polling) is not allowed.
6. Mutual exclusion should be kept to a minimum to allow the most concurrency.
7. The semaphore value may not obtained and used as a basis for program logic.
8. Each customer thread should print when it is created and when it is joined.
9. All activities of a thread should only be output by that thread.

Task Table

|  |  |  |
| --- | --- | --- |
| Actor | Step | Time (seconds) |
| Ticket Taker | Tear Ticket | 15 |
| Concession Stand Worker | Fill Order | 180 |
| Box Office Agent | Sell Ticket | 90 |

**Project Implementation:**

The project was implemented in C++ language with Unix Operating System platform . The project was tested on the CS2 Unix server.

The project was implemented in the following steps:

Step 1: The design and pseudo code for the project was designed

Step 2: The filename was accepted and the contents of the file was read

Step 3: The movie name and number of seats available were separated and stored separately in the respective movie structure

Step 4: The functions containing the tasks to be performed by the various types of threads were designed

Step 5 300 customer threads, 2 box office agents, 1 ticket taker and 1 concession stand thread were created

**Personal Experiences:**

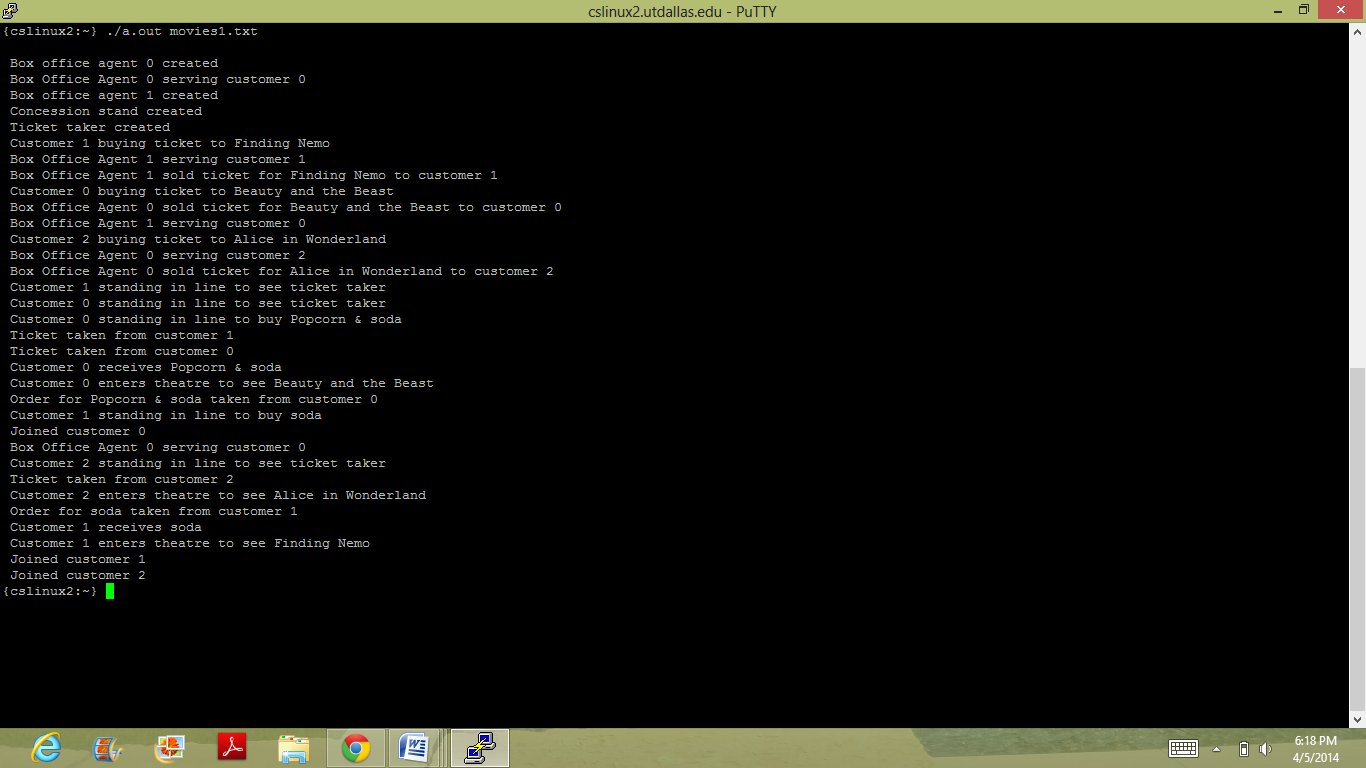
I had to spend some amount of time reading about threads and semaphores. I then had to code the project in stages and check the output at each stage to ensure that the project was working properly up-to that stage.

I had a bit difficulty in preventing one thread from printing it's output while another thread was printing leading to a mix up in the output of two threads.

I overcame this difficulty by replacing the cout command by printf() command

I also had to use GDB debugger to execute the code line by line to correct other small runtime errors that were present in the code. This took some amount of time since line by line execution was taking place and the entire process had to be repeated every time I corrected an error.

**Sample Output:**  ( Number of customer threads reduced to 3 to capture the output)



**Reference:**

The barber shop example given in the slides and text book was used as a reference while designing the design and pseudo code of the project.

The examples put in elearning were used while doing the coding for the threads and semaphores.

The following website was used as reference while doing the line by line execution of the code using GDB debugger:

https://sourceware.org/gdb/onlinedocs/gdb/Threads.html