Discrete Event Simulator

Content

Folder Structure

1. makefile in root of the extracted file
2. qSim executable will be generated on running make alos in root
3. 3 test case as textfiles - test1.txt, test2.txt, test3.txt
    a. First line the command
    b. Following lines the output
4. README.pdf
5. include contains all .h files
6. src contains all .c files
7. ProblemStatement.pdf put in for reference

How to run?

Compile the program: make
Remove compiled files: make clean

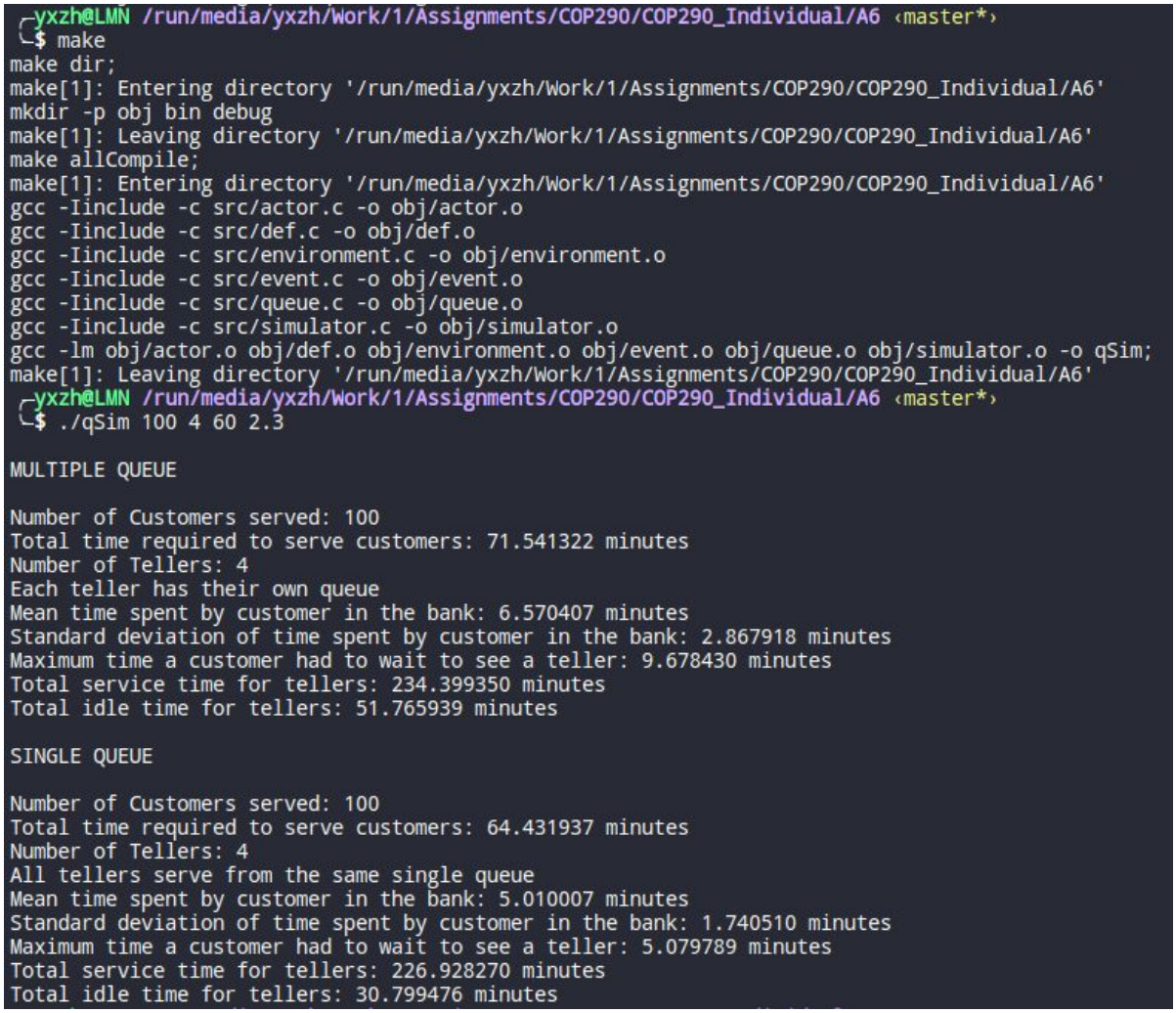
Running Executable: ./qSim 100 4 60 2.3
Arguments -

1. Number of customers
2. Number of tellers
3. Duration of arrivals
4. Average service time

Output -

1. Statistics about both cases - multiple queue and single queue

2. 5 information summary mentioned in the problem statement (Output heading) is printed out on stdout.



Debug mode can be toggled by changing DEBUG (1 or 0) macro set in include/def.h (make clean & make to see the change)
On turning debug mode on the time of start of events can be seen. Events such as -

1. Arrival of a customer
2. Teller start to serve a particular customer
3. Teller leaving for a break
4. Teller taking customer from another queue

```
yxzh@LMN /run/media/yxzh/Work/1/Assignments/COP290/COP290_Individual/A6 ‹master*›
$ ./qSim 100 4 60 2.3
Initialized of multiqueue system
0.000000: teller 0 taking a break for 8.404541 minutes
0.000000: teller 1 taking a break for 3.953923 minutes
0.000000: teller 2 taking a break for 7.834607 minutes
0.000000: teller 3 taking a break for 7.987760 minutes
0.978034: customer 0 entered queue 3
1.201383: customer 1 entered queue 0
2.356821: customer 2 entered queue 2
3.785750: customer 3 entered queue 1
3.850279: customer 4 entered queue 1
3.953923: teller 1 started serving customer 3 for 1.292873 minutes
4.185317: customer 5 entered queue 0
5.163351: customer 6 entered queue 3
5.246796: teller 1 started serving customer 4 for 2.056354 minutes
6.528528: customer 7 entered queue 1
7.303151: teller 1 started serving customer 7 for 0.862652 minutes
7.787427: customer 8 entered queue 1
7.834607: teller 2 started serving customer 2 for 2.559641 minutes
7.987760: teller 3 started serving customer 0 for 1.915906 minutes
8.165803: teller 1 started serving customer 8 for 0.780193 minutes
8.233895: customer 9 entered queue 2
8.404541: teller 0 started serving customer 1 for 0.474587 minutes
8.496153: customer 10 entered queue 1
8.879128: teller 0 started serving customer 5 for 2.279043 minutes
8.945996: teller 1 started serving customer 10 for 3.498186 minutes
9.400745: customer 11 entered queue 1
9.903666: teller 3 started serving customer 6 for 4.301018 minutes
9.958450: customer 12 entered queue 3
10.394249: teller 2 started serving customer 9 for 1.762666 minutes
11.158171: teller 0 took customer from queue 1
11.158171: teller 0 started serving customer 12 for 1.695852 minutes
11.532831: customer 13 entered queue 0
11.853082: customer 14 entered queue 2
12.156915: teller 2 started serving customer 14 for 2.688647 minutes
12.444182: teller 1 started serving customer 11 for 1.124299 minutes
12.854023: teller 0 started serving customer 13 for 0.700993 minutes
12.949498: customer 15 entered queue 1
13.095414: customer 16 entered queue 2
13.555016: teller 0 took customer from queue 1
13.555016: teller 0 started serving customer 16 for 0.754869 minutes
13.568480: teller 1 started serving customer 15 for 3.427328 minutes
```

Design Decisions (elaboration of the specification in the problem statement)

Event

1. Function Pointer
2. char ** Argument Vector

This allows to have multiple types of functions to execute to replicate an action occurring in the real world (bank)

Actors

Structs were used to model instances of actors (Tellers, Customers). These Structs also contained information about the statistics to be done in the summary of the simulation (figure 1).

FIFO Queue

It was noted that in some cases only the order of events was known and not the exact time of event occuring. Case like customers queueing in a queue to be served by tellers. FIFO queue without functionality to insert in between or reordering was sufficient to handle the mentioned scenario.

Environment

1. Arrays of actors
2. Starting FIFO Event Queues
    a. Contained events corresponding to the start of an action
    b. Used here to mark the start of serving of a customer by a teller
3. Ending FIFO Event Queues
    a. Contained events corresponding to end of an action
    b. Used here to mark the end of serving of a customer by a teller. The function pointer is used to determine whether to continue serving the next customer in tellers queue,picking and serving a customer from another queue, or going on a break
4. All other information relevant to the world (bank) was also stored here

Simulator

Simulator here is defined as an entity that proceeds the simulated time and calls necessary events at appropriate time. It leaves the changes to the state of the environment which are consequences of an action to the function pointer getting called. The simulator only needs to simulate discrete moments of time. Moments like the event of a customer starting to be served, teller checking which action to do next.

The only differences in multiple customer queue and single customer queue from the perspective of implementation were -

1. Single starting FIFO queue in case of single customer queue simulator
2. Different event for checking the next customer to serve

Apart from these I was able to generalize the implementation into common parts

Problems Encountered

1. Generalised Functions for modelling events
    a. char ** arguments vectors need to be malloc-ed, parsed, freed at start, during, end of the function call respectively.
2. Heap memory related issues
    a. Used valgrind to narrow down the location of memory corruptions and leakages
3. Debug printing
    a. Wrapper for printf which only runs when DEBUG set to 1
    b. Passed on variable number of argument to printf from stdio.h
4. Checking equality of doubles
    a. Instead of checking equality of variable type double. Check if they were a less than epsilon apart.
    b. Type double was used to model simulated time, used to prevent precision errors in case of equality

Analysis

I wasn't able to complete the GNUPlot deliverable. Adding information required to make the plot below

Variation of average time customer spent in bank with increase in number of tellers

Fixed parameter:

1. Number of customers: 100
2. Arrival duration: 60 minutes
3. Average service time: 2.3 minutes
4. Break duration is between 1 to 600 second (mentioned by problem statement)

Multiple Queue

Number of tellers	Average time spent by customer in bank
4	6.57

6	3.96
8	3.40
10	3.11
20	1.07

Single Queue

Number of tellers	Average time spent by customer in bank
4	5.01
6	3.56
8	3.28
10	3.54
20	1.29

Average time spent by customers decreases with the increase in number of tellers as expected for both cases. But for Multiple Queue scenario the decrease is quicker than the counterpart