EE663- Project 4 Report QNX Bank Simulation

Table of Contents

Overview:

The aim of the project is to simulate a typical banking environment.

- Banks system basically works on single and multiple server queuing concept.
- Customers access the bank and three tellers are available at their service.
- Monitor the performance of the transactions at the end of the day.

Cover Page:

Project title	:	QNX Bank Simulation.
Names	:	Deepak Siddharth Parthipan
E-mail addresses of the authors	:	dp9040@rit.edu.
Date of submission	:	11-21-2016.

Areas of Focus:

Vedant Karia	: Problem Statement analysis, Functional Design, Developing Code, Test plan and Report.
Deepak Siddharth Parthipan	: Problem Statement analysis, Functional Design, Developing Code, Test plan and Report.

Analysis / Design:

The basic idea of the project is to work on a multi-threaded application to create a bank model in a queuing concept.

Hardware configuration:

The purple box is connected to the system and the power is given to the QNX purple box.

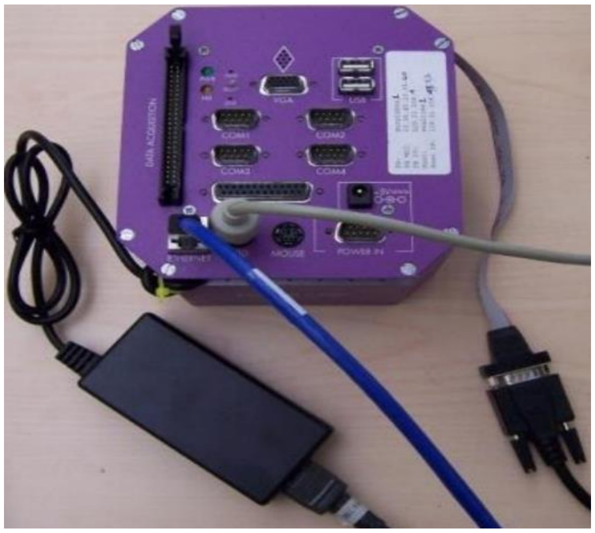


Figure1: Hardware

Software Approach:

The major task for this project was to work with threads. The concept of the threads is that, unlike function call happening in infinite while 1 loops. Here we have an RTOS running at the background to take care of system tasks. Multiple threads run in a time-slicing manner. It would rather seem like threads are running in parallel. Thread creation was executed and checked before going to main application.

The next challenge was to recreate bank model as a software application program. Total bank run time was followed as given in the spec, i.e. 100ms equivalent to 1minute in real world. To get the current time, system clock from QNX was used using the build in function gettimeofday(). The next process was to create a delay in program so that it represents real world time for transaction to complete. For this purpose, usleep() function was used. Another major factor for the delay was transaction time or queue time and for this purpose, a uniform random distribution function rand_r() was used extensively. This rand was strictly limited to min and max time as specified in the problem statement.

The approach for the program was to split the tasks for the customer, teller1, teller2 and teller3 into separate threads. As mentioned in the problem statement for each of the customer, individual task control blocks were created. This task control blocks contains the details of each customer such as token number, teller number, time in, time out, queue in time. And since TCB was declared as a global variable accessing shared data across threads were made possible.

Customer thread was responsible to create: customer queue, updating the token and the queue in time for the customers. It also had the information regarding the max queue length.

Teller thread was designed identical across all tellers. Teller thread updated the information regarding: teller wait time, customer in time, customer out time, random teller transaction time, accumulation of teller waittime for average and the graduate extension for the teller break time.

Once the entire day's transaction in the bank was completed the control returns to the main function. Here the final calculation for total customer visited, max teller wait for customer, teller spent in transaction, avg time teller wait for customer, avg time customer with teller, max time customer in queue was done w.r.t to each of the teller and the finally the output was displayed.

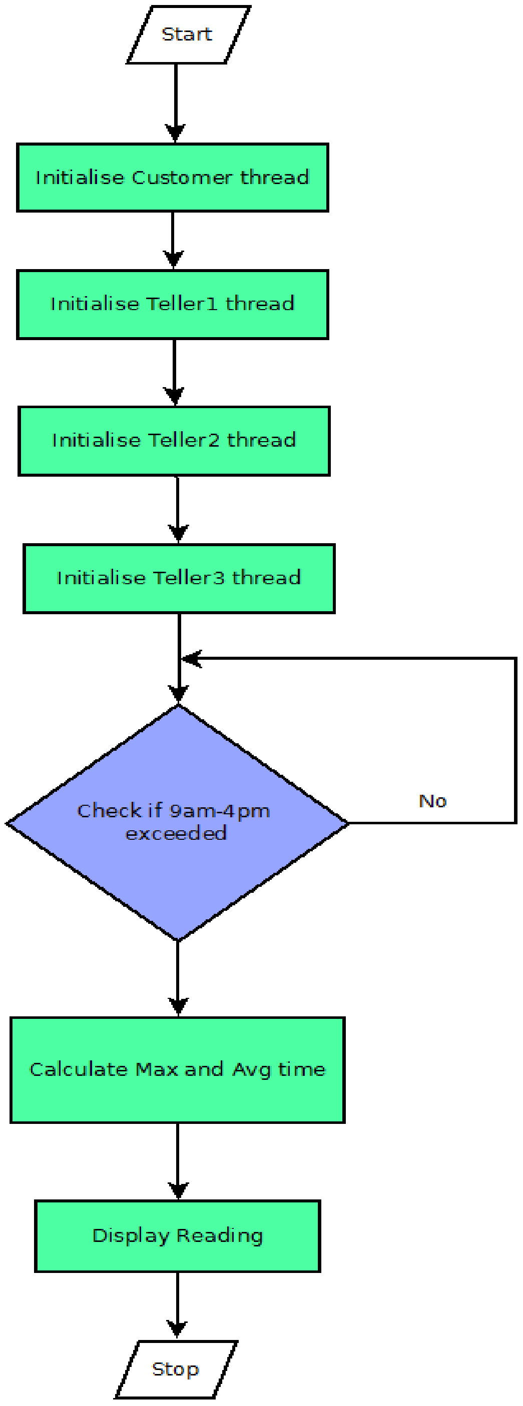

Start
Initialise Customer thread
Initialise Teller1 thread
Initialise Teller2 thread
Initialise Teller3 thread
Check if 9am-4pm exceeded
No
Calculate Max and Avg time
Display Reading
Stop

Figure2: Main function flowchart

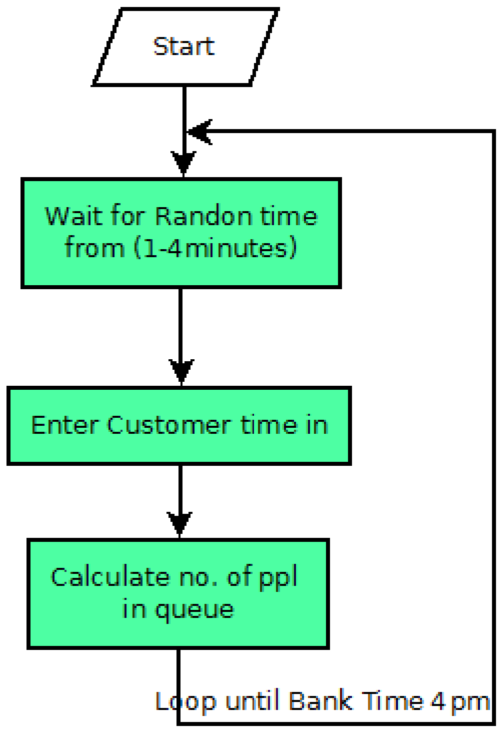

Start
Wait for Randon time from (1-4minutes)
Enter Customer time in
Calculate no. of ppl in queue
Loop until Bank Time 4 pm

Figure3: Customer Thread

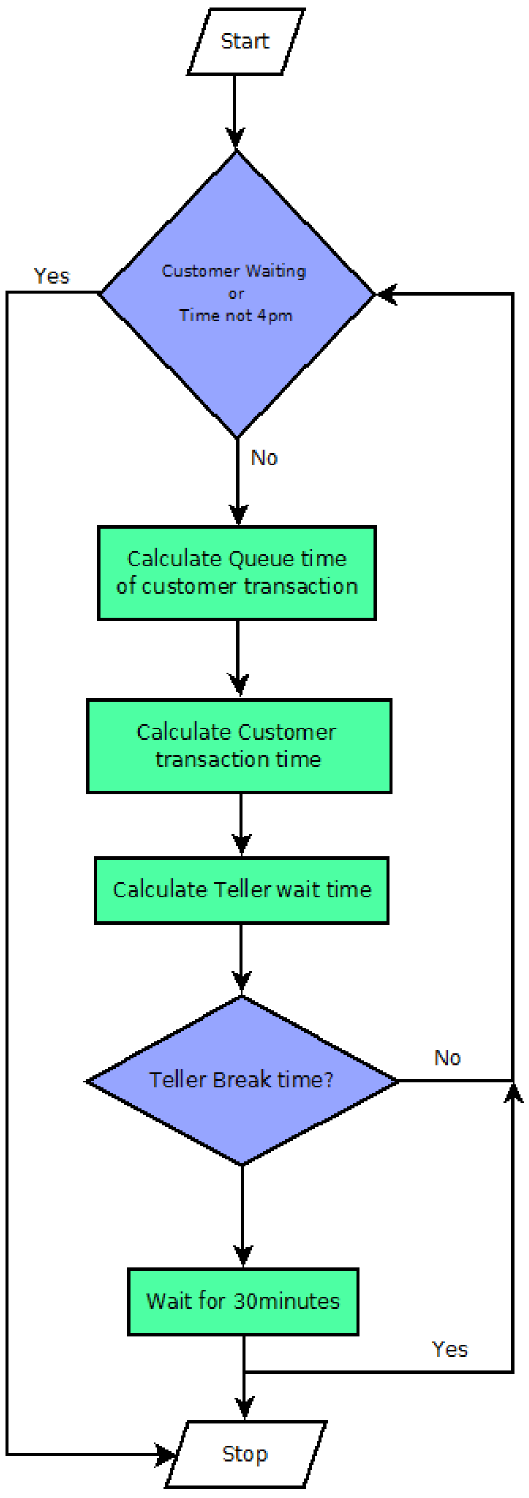
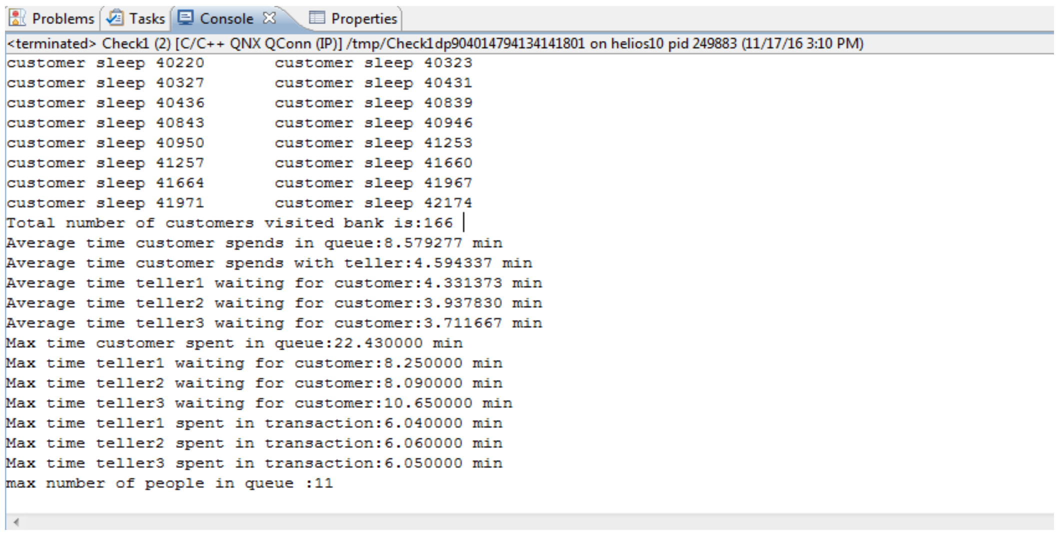

Start
Customer Waiting or Time not 4pm
Yes
No
Calculate Queue time of customer transaction
Calculate Customer transaction time
Calculate Teller wait time
Teller Break time?
No
Wait for 30minutes
Yes
Stop

Figure4: Teller Thread

Test Plan:

- Implement different threads and test how synchronization works.
- From the results check if there is really a need for mutex locks for bank model?
- Run & check, if an average of 160-170 customer arrive and complete transation for a single day.
- Testing goes hand in development. Working by parts. If basic teller tread is working fine add the break time for teller into the thread.

Project Results:



```
<terminated> Check1 (2) [C/C++ QNX QConn (IP)] /tmp/Check1dp904014794134141801 on helios10 pid 249883 (11/17/16 3:10 PM)
customer sleep 40220       customer sleep 40323
customer sleep 40327       customer sleep 40431
customer sleep 40436       customer sleep 40839
customer sleep 40843       customer sleep 40946
customer sleep 40950       customer sleep 41253
customer sleep 41257       customer sleep 41660
customer sleep 41664       customer sleep 41967
customer sleep 41971       customer sleep 42174
Total number of customers visited bank is:166
Average time customer spends in queue:8.579277 min
Average time customer spends with teller:4.594337 min
Average time teller1 waiting for customer:4.331373 min
Average time teller2 waiting for customer:3.937830 min
Average time teller3 waiting for customer:3.711667 min
Max time customer spent in queue:22.430000 min
Max time teller1 waiting for customer:8.250000 min
Max time teller2 waiting for customer:8.090000 min
Max time teller3 waiting for customer:10.650000 min
Max time teller1 spent in transaction:6.040000 min
Max time teller2 spent in transaction:6.060000 min
Max time teller3 spent in transaction:6.050000 min
max number of people in queue :11
```

Figure5: Output screen capture

Lessons Learned:

- Working with thread both creating and synchronization.
- Using mutex's as and when required and not using it, if no critical tasks were missed.
- Understanding and working with multiple server queuing concept.
- Using the in-built function such as rand_r(), usleep() etc.,
- Debugging the code and to work with RTOS in background.

Submission:

The QNX bank simulation application program was developed, tested and demonstrated successfully.