BEYZANUR BEKTAN 2019400174

CMPE 322 - PROJECT 3 - REPORT

In this project, we were intended to develop a multithreaded prepayment system simulation and handle some synchronization issues using pthread mutex locks. My implementation consists of mainly 3 parts;

- sendInfo function,
- makePrepayment function
- main function.

In the beginning of the project, I declared some global variables that I will use later. I created a struct named “prepayment” which holds the customer ID, machine ID, company ID and prepayment amount for each prepayment instance. Then I created an array “prepayments” size of 10 to hold currently operating prepayments. Array index of a prepayment represents the operating machine ID.

In the main function, I opened the log file using input file name. Then I read the first line of the input file and assigned the value to the customerNumber variable. I created a string vector named “fileLines” to hold all lines except the first one. I initialized the mutexes that I will later use in the threads.

I created vending machine threads and customer threads as many as needed. While creating the machine threads, I called the makePrepayment function by machine ID parameter. For customers, I created an array of vector named “datavector” which holds the vectors containing prepayment information of each customer. I read required information from input file and add to the vector in order. Then I add each vector to the array. While creating customer threads, I called the sendInfo function by that array elements. In the end of the main function, I printed the overall balance values of companies to the log file.

“sendInfo” function is the customer thread function and does the information transfer from customers to vending machines using the shared data “prepayments”. It takes a vector instance as a parameter containing features of current prepayment. Firstly, thread sleeps for the given milliseconds, then creates a prepayment object which will be added to prepayments array. Required information comes from input file in an integer vector and passes to machine threads with these prepayment instances. While sending information to the vending machine, customer thread locks that machine to prevent concurrent writing on the “prepayments” shared data item which can lead to inconsistencies. After customer is done, it unlocks the customer mutex to ensure that machine start the operations. At the same time, it locks the vending machine mutex to wait for it finishes its job. Finally, it releases the vending machine.

“makePrepayment” is the machine thread function and does the prepayment operations. It takes machine ID as a parameter and uses it to obtain the information about which prepayment instance that will be realized. I used a while loop because vending

machines are always serving and waiting for a customer to arrive. It checks the company ID and amount from relevant prepayment object and makes the payment to that bank account accordingly. It adds this amount to overall balance of that company using global integer variables “K, B, S, O, D” representing company balances. While doing this, it locks the company[i] mutex to prevent concurrent write on balances. Then it locks the log file mutex and prints the information about current prepayment operation.

In the end, there are some comments that I want to add. According to my implementation, customers are enumerated by their order from the input file. Also, since the mutex unlock operation behaves like FIFO queue, customers with smaller sleeping durations are being served first. In general, prepayment order in the log file shows parallelism with the sleep times but there can be small changes in different compilations of the program.