CmpE 322 Project3

Zeynep Buse Aydın - 2019400066

Input - Output

The project has a one source .cpp file named main.cpp which can be compiled with the Makefile. After the **“make”** command, the user must enter:

./simulation “name_of_the_input_file”.

The logs and results are written on the files with names “**<name_of_the_input_file>_log.txt**”.

Implementation

Global Variables:

- **remainingCus** integer keeps track of how many customers are left that did not done with their payment.
- **companies** array holds the amount of money that is paid to each company “Kevin”, “Bob”, “Stuart”, “Otto” and “Dave”, respectively.
- **inputLines** vector holds the data line of each customer that is read from the input file.
- **prepaymentDone** is a vector of boolean which indicates whether the customer is done with the prepayment or not. Indexes correspond to (customer number - 1) since the customer numbers start from 1. The vector is initialized with all zeroes.
- **vendingQueues** is the array of the queues of the vending ticket machines.
- **companyMutexes** array contains the mutexes of each company “Kevin”, “Bob”, “Stuart”, “Otto” and “Dave”, respectively. These mutexes prevent updating the amount of money a company has simultaneously by more than one vending ticket machine thread.
- **vtmMutexes** array contains the mutexes of each vending ticket machine. They prevent the race condition for the queue of the vending ticket machine.
- **finishedMutex** is used to make the **remainingCus** variable mutually exclusive.
- **outputLock** is used to prevent writing simultaneously to the output file.

Customer Struct:

The struct that contains the necessary information of a customer (the **amount** of money the customer wants to pay, the **company** s/he wants to use and his/her **customer number.** The customer sends its data to vending ticket machine threads using this struct.

Threads:

There are **10** threads for vending ticket machines, and **numCustomers** number of threads for customers. **numCustomers** is read from the input file.

Functions:

- **customerRoutine:**

  This is the function of the customer threads. In this function, threads gets the customer number as the parameter and takes the line that contains its data (the line that is in the customer number index) from the **inputLines** vector.

Then it splits the line from commas, firstly gets the **sleep time** and sleeps for that amount in milliseconds. After that, it gets the vending ticket machine number -**vtmNo**-, the company name -**company**- and the amount of money it is going to pay -**amount**-. With this information, it creates a customer named **cus** and puts it in the queue of the vending ticket machine with the number **vtmNo** while locking the mutex of that machine. The communication between the customer thread and the vending machine thread happens that way. Before it exits, the customer thread waits until the prepayment is done. This check is made via the **prepaymentDone** vector.

- **vtmRoutine:**

  This is the function of vending ticket machine threads. This thread executes until there are no customers left waiting to pay.

  If there is a customer waiting in the vending ticket machine's queue, the thread locks the mutex of that vending ticket machine, gets the information of the customer being served. performs the payment operation with the **performPayment** function and declares that the current customer is done via making the boolean in the **customerNo** index of **prepaymentDone true**.

  Then, it prints the logs in the output file, using the **outputLock** mutex. Finally, it also updates the value of **remainingCus** by decreasing it by one.

- **performPayment:**

  Given the **company** name and the **amount** of money that will be paid, this function increases the money of that company by the amount that it gets. It uses that spesific company's mutex in order to prevent any inaccuracy.

Execution of main:

First, all the mutexes are initialized. Then the input is read. Each line of the input except the first line that contains the number of customers are put into the **inputLines** vector. After that, the vending ticket machine threads are created with the **vtmRoutine** function in which the threads perform the payments of the customers. While they are executing in parallel, the customer threads are created with the **customerRoutine** function in which they send their information to the corresponding vending machine threads. After all the threads exit, they are joined and all the mutexes are destroyed. Lastly, the result is printed on the output file.