Ömer Faruk Ünal
2019400048

BOFUN Fair?

Usage

- To run the project simply write `make run input={input_path}(make run input=case.txt).` This will create three output files with names "case_log.txt" .
- case_log.txt is the logs of the given input file.
- To create an executable file just write `make`. This will create three executable file with name "simulation." To run it type "./simulation {input_path}"
- To clean the executable run `make clean.`

Description

- This program takes an input configuration file. Process each prepayment according to given rules. And writes the logs of given transactions.

Project Outline and Explanations

This project was written in the C++ language, and it consists of one `.cpp` files. This file is:

- `main.cpp`: The whole program is in this file.

Main Thread:

- First the program parses the number of customers and creates threads for each customer enumerated from 0 to n - 1.
- Then creates 10 threads for machines enumerated from 1 to 10.
- After creating threads, the main thread waits for all other threads to complete.

Customer Threads:

- Customer thread first parses the line regarding that customer.
- Sleeps for the given amount.
- Locks the input mutex of the given machine so that until the machine reads the relevant line no other customer will be able to write to buffer of that machine which is just one string in the transaction array.
- Unlock the transaction mutex of the given machine so that the given machine will be able to proceed.

Machine Threads:

- First parses the machine no and creates a local array for storing the prepayments amount for each bank.
- Starts to wait idle until transaction mutex is unlocked which is any of the customer send a request to proceed the payment.
- If the transaction mutex is unlocked machine starts processing and locks the transaction mutex so that no other customer will be able to start a new process until the previous one is finished.
- Reads the given line and unlocks the input mutex so that other customer would be able to write to buffer of that machine.
- Process the payment and starts waiting again.
- If transaction count is equal to number of customers which means all the transactions are completed all mutexes are unlocked and writes their local bank data to global bank data.
- Then each thread breaks the loop and terminates.