Cmpe 322, Project 3

Deniz Ünal, 2019400234

Introduction

In this project we were required to implement a prepayment system with limited number of customers and vending machines. Our goal was to prevent synchronization issues and race conditions when there were multiple customers that wants to access to the same vending machine, while also keeping track of the payments that are completed so far. In order to achieve this goal I used the pthread library and mutex locks.

Program Interface

The program is written in C and in order to compile it make and gcc should be installed on the system. To run the program following call syntax should be followed:

./simulation /path/to/input/file/<input-name>.<file-extension>

or if the file has no extension,

./simulation /path/to/input/file/<input-name>

The program output will be in the current working directory and its name will be:

<input-name>_log.txt

Program Execution

In order to implement customers and vending machines C structures are used. Arrays of these structures are declared as global variables. The code parses through the input and set the local variables of the each customer instance. After parsing through the input, first threads of the vending machines and then threads of the customers will be created.

Thread will be given an index as an argument and it will find the corresponding struct instance using that index. First it will sleep for the amount of time that is given in the input. After that it will lock the vending machine that will be used by the customer and pass the necessary information to the vending machine. Then it will set a request flag and exit.

```c
void *Prepay(void *customerID){
  struct Customer *customer = &g_customerArr[*(int*)customerID];
  usleep(customer->sleepTime * 1000);
  /*
   * Sets a lock to prevent other customers requesting
   * prepayment without the current one is complete.
   */
  pthread_mutex_lock(&g_vendingLocks[customer->vendingID]);

  struct VendingMachine *machine = &g_vendingArr[customer->vendingID];
  /*
   * Sets the necessary data so that vending machine
   * can process the prepayment.
   */
  machine->customerID = customer->id;
  strncpy(machine->company, customer->company, sizeof(machine->company));
  machine->amount = customer->amount;
  // Sets a flag to let vending machine know there is a prepayment to process.
  machine->requested = 1;
}
```

Vending machines will enter in a loop and will check if there is any prepayment request is sent from any customer in each loop itearation. When the request flag is set by the customer it will start the prepayment operation using the data set by customer thread. It will set one of the locks that corresponds to a company based on the company variable set by the customer and update the amount of total money paid to the company so far. Unlocks the company after the update. Then set the lock that corresponds to the output file, then prints the necessary information and unlocks it again. Finally it will unlock the lock of itself that is set by customer that requested it and continue looping until another request arrives.

After all prepayment methods are done and all customers are joined a customerDone flag will be set by the main and the while loops in the vending machine threads will be exited. Then the final output lines will be printed by main and the program will exit.

Program Structure

Functions

Prepay: The function that will be run inside the thread. It will send the data to vending machine, and will set a flag to start the prepaymen operation.

Vending: The function that will be run inside the thread. It will realize the prepayment operation using the data that it is given by customer.

Main: All other things are handled by main function including reading input, initializing struct instances, creating threads and printing last part of the input.

Structs

Customer:

id → id of the customer - 1

sleepTime → the amount of time the thread will sleep in ms

vendingID → the id of the vending machine ithe customer will use – 1

company → name of the company prepayment will be made to

amount → amount of money that will be prepaid

VendingMachine:

id → id of the vending machine – 1

customerID → id of the customer that will prepay using this vending machine

company → name of the company prepayment will be made to

amount → amount of money that will be prepaid

requested → this flag will be set to 1 before each prepayment by the customer, after the operation the vending machine will revert it to its original state (initial value = 0)

Note that customerID, company, amount and requested members will be set by customer before each prepayment.

Mutex Locks

Vending machine locks: Vending machine locks are holded in a array of length 10. They are used to ensure that only one customer will be using the vending machine at a time.

Company locks: Company locks are 5 seperate locks that is declared globally. They are used to ensure that only one vending machine will be updating the total amount of prepaid money to the company.

File lock: There is one file lock declared globally. It is preventing different vending machine threads from printing their outputs at the same time to ensure the integrity of the output file.

In this project the communication between the threads are implemented by using shared memory. Since the assignment that are done on variables are not atomic instructions, usage of mutex locks prevents race conditions and ensure deterministic outcome.

Assumptions

- It is assumed that the correct input will be given. The program checks if the number of arguments are correct but does not check if the given input file exists or not.
- It is assumed that input name and each line in the input will be shorter than 256 characters.
- It is assumed that in the desired output filename the input file extension will be removed by _log.txt . For example given input.txt the output filename will be input_log.txt and not input.txt_log.txt