CMPE322-PROJECT3 DOCUMENTATION

In this project, I implement a multi-threaded application for simulating a fun fair payment system, where the customers can make the prepayments of the various rides through the available ticket vending machines.

Since the prepayments can be made simultaneously through different ticket vending machine instances, the synchronization and the data consistency should be prevented. To do this, I use mutexes. First, I create global variables for saving the records of payments, number of machines, consumers, arrays for mutexes and struct for customer instance. I use this struct for using information of customers. Also, there is a vector behaves like a queue.

In main, I do file I/O operation. I take input file as an argument like “input.txt” then I erase the part after dot and concatenate this string with "_log.txt" to make output file like description. Then I initialize mutexes for customers and machines. Machine mutexes are firstly locked because there is no ready customer. Then, I read file line by line and save each customer’s data to its struct and there is an array for all structs. I create customer and machine threads and join them. Lastly, I print into the output file total values of payments made with each company.

In customer function, I take the consumer struct as a parameter and send to sleep every customer with usleep function. Since usleep suspend execution for microsecond intervals, these intervals are very short and can lead to data inconsistency. So, I multiply every sleep number with 1000. I use consumer mutex array which has 10 mutex for each machine and machine mutex array same properties with same properties. These arrays are used for communication and prevent inconsistency. If a customer awakes then ready to be added to queue so, I locked consumer mutex array’s element to say consumer is ready for this vending machine. Then I added to vector with push_back. Then I unlocked machine mutex array’s element to say this vending machine is ready for take element from array because there is a customer now.

In machine function, I take waiting consumer vector as a parameter and while number of payments are not equal to number of consumer and while this vector is not empty, I take consumer at the 0th index. Then, assign customer’s data to local variables. I locked the machine mutex element to say vending machine is making transaction now, in critical section. I write to output file the results and use writing mutex for ensuring only one writing operation. **I write like [vtm0...9] as in first output files you provided.**

Begüm Yivli-2019400147