CSE 344 HOMEWORK 5 REPORT

Furkan Kalabalık - 161044001

Gebze Technical University
Computer Science and Engineering Department

8 June 2020 06:00 AM

In this homework we're asked to implement a flower delivery system using POSIX threads and their synchronization methods. In a file we have all required florist data and customer request written in exact format. Our purpose in here is to create threads that indicated number in file and delegate customer requests which again written in file to these threads from main thread. By delegating requests of customer we consider two rule. One ,of course, florist must have stock for requested flower type. Other is that, when two or more florist has same type of flower, we will look distance called ***Chebyshev distance***. This distance is calculated by absolute substraction of opposing coordinates of two point. Since we deal with 2D plane our equation is:

$$D_{Chebyshev} = max(|x_2 - x_1|, |y_2 - y_1|)$$

Location of each florist and customer written in file. By that we can find distances and compare by selecting most closest florist to customer.

First, I created two structures, one for florist and other for customers. Florist structure contains information about florist's id, name, what kind of flowers he or she has, where the florist is located by a point structure that consist of two double for x and y, speed of delivery which will be used to find delivery time when one florist delivering to customer and sale stats which is required to print total statics at the end of the day. Customer structure is consist of customer name, customer location and order name of the customer.

After that I started to read file to create florists and learn their total number to create required synchronization tools for them. I just parsed file by considering example file. I do it generally so any number of florist and customer with different number and kind of flower can be apply. I stored location as double since we deal with real plane. I used dynamic arrays to store names.

After learning number of florist, I started creating structures for synchronization of threads and queues for their customers. I created an array which holds queues for each florist and each queue holds requests that delegated to particular florist. This is the array called *customerPerFlorist*. After that I created an integer array that holds number of request for each florist. This is required since we need to stop florist thread to process requests when there is no request. This array is called *requestPerFlorist*. Then I created queue size array for each florist queue. Reason of that array may be queue of florist can be filled and there is no room for next request. Since we need to delegate request as soon as possible, when there is need I enlarge queue size. This array also called *queueSize*. After that I created array of total requests. This array hold all request until program ends even request is processed and delivered by florist. Reason of that array, since we dynamically allocated names for request, we must free them at the end of program. Instead of doing that after delivery, I stored them and at the and I freed the memory.

After that for syncronization to shared variables I create a *pthread_mutex* called *mutex* to lock access to the shared variables between florist threads and main thread. Also I created a mutex called *quitMutex*, that used to control exit state of thread. When a signal is occured by using a integer value *quit* is setted and after that threads look up that value and break their loops and flow goes to exit. Then I created condition values for each florist thread, which required to block thread when queue is empty. After all when all data structures is read I created threads with *pthread_create*.

In main thread I started to read file from where cursor is remained. I read file until encounter a newline. After that char I sended line to function called *processCustomer* and then I constructed request for that customer. After that first I searched I florist that has stock for that particular order. After finded that florist I started to search again a florist that has particular order. If florist has that flower and distance to florist less than precviously found florist, I changed id of current florist. After finded required florist I locked it with *mutex* to access shared data. First I tried to put request to the queue of florist but first I checked if the queue of florist is full or not. If it is full, I reallocate queue again and enlarge the size of queue and increase number of request for that florist. I do that same thing for totalRequest array. After increasing number of request for particular florist, I checked number is equal to 1. If it is one that means florist is waiting for condition variable since there is no request in queue. Then if it is the case I signaled that particular florist's condition variable by using *pthread_cond_signal*. Then I release the lock to give access to the florists.

Each florist thread takes a florist structure. This structure includes informations about florist. Each

florist first runs a while loop. Before start loop it checks quit flag is setted or not. If it is setted it simply returns. Thread starts with try to lock *mutex* to access shared data. After owned *mutex* it looks *requestPerFlorist* defined by its id to check if there is a request. If there is not a request thread calls *pthread_cond_wait* by its condition defined with its id with *mutex*. After main thread gives request to the queue of thread, signal thread and thread continues. When all requests is processed, main thread puts a request with order "poison" to the queue of each florist. After each florist processed its requests, then it also request "poison" and dies and returns to main thread. If this is not the case florist process next request by first calculating time. Time is change with respect to distance and randomly for each order. Preparing each order requires a random ms between 1 and 250. Also we have distance and speed information. We can find required time easily by dividing them. Each florist process first element of the queue and then moves other elements to the left by one. Then florist release lock, increase its stats then sleep *time* ms by *usleep* function. Then it prints delivery time and informations to standart output. Main thread puts "poison" to the queues when all requests are processed. After that main threads joins with florist threads by calling *pthread_join*. Threads returns their argumets with filled in stats. After that stats for today's sales written in nice format.

1 In case of Ctrl-C

When Ctrl-C occurs, this mean **SIGINT** will be signaled to process. First, I blocked **SIGINT** before creating any of dynamic memories and threads. After that I registered a handler for that signal called *termHandler*. This handler simply set the quit flag and put process in exit flow. After handler is registered, I unblocked **SIGINT** after creating all threads. Now, when a signal comes or if a signal sended but pending, is delivered to the process. Handler sets quit flag. When the quit flag is setted, while loops of threads is check quit flag and breaks its loops an returns. Main thread also break its while loop that reads customers. If a thread is waiting for signal or it cannot break its loop until it is signaled. For that after main thread break its loop, it do again, puts "poison" to the queue of florists again in case of any waiting thread. Now all threads can continue and break loops by simply looking quit flag. After that, all threads joins to main thread and without printing stats, starts exit routine. Before creating all structures, I registered an atexit function called *cleanUp*. Since I blocked signal, signal do not be delivered until unblock signal and unblocking will occurs only after all structures created. So, cleanUp simply free all allocated resources and exit the program.