GIT Department of Computer Engineering
CSE 344 - Spring 2020
Homework 5 Report

Esra EMIRLI
151044069

June 7, 2020

Problem Solution Approach

First of all, all the variables that the florist will need are gathered under the Florist Struct.

This struct :

Florist Features

- **threadId :** Every florist is actually a thread. Its id is kept here so that it gets access in case of shutdown.
- **name[] :** keeps the florist's name. It is limited to 255 characters.
- **x and y :** hold the florist's coordinates.
- **click :**unit of distance in the grid
- **flowers[][] :** holds the flowers it sells. The names and number of flowers are limited to 255.
- **typeNumber :** keeps how many kinds of flowers the florist sells.

Customer Details

- ****queue :** keeps the name of the customer to whom she/he will respond.
- ****delivery :** keeps the flowers to be delivered to customers.
- ***distance :** keeps customers' distances.
- ***clientNumber :**maintains the total number of customers.
- **limitFull :** has a max capacity (1000) for each thread. It checks it.

Then, mutex and condition variables are creates.

Main 1. Section

First, argument checks were made with **getopt**. Causes of errors:

1. Missing argument
2. Unknown option (except i)
3. More arguments

As a restriction was not given about the use of c libraries, this was not paid attention to the file operations.

If the file is opened without error, *"Florist application initializing from file: data.dat"* output is printed on the screen.

Then, the **file starts to be read line by line.** Causes of errors:

1. If the first line is empty
2. If there is no florist information
3. If the florist name starts with a character other than the alphabet
4. There should be 1 line space between florist and client.

   If there is no space, an error will print and the program will end.

   If there are more than 1, clients are not served.
5. If the customer orders a flower that is not available at any florist, the program ends.

Lines are formatted and separated until the flowers sold. It is saved in the florist struct. The flowers are then read and they are recorded.

Since threads will occur here, mutex is locked. For the queue, delivery, distance and clientNumber variables that will keep the information of the customers, they are stored in the memory using malloc. A thread is created for this florist and the global numOfThread is increased. This process continues for all florists.

After one line of space, clients start reading. It is formatted according to the client format. The **Chebyshev distance** formula is used for the distance between the florist and the client.

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

`https://en.wikipedia.org/wiki/Chebyshev_distance`. is the link from which the formula is taken.

Check whether the flower that the customer wants is sold. If it is not sold in any of them, an error message is given and the program is terminated.

Distances are also compared for the nearest florist. Client information (queue, delivery, distance, clientNumber) is recorded in the nearest florist.

A signal is sent to all threads and mutex is unlocked. The reading process continues until the file is finished.

Thread Function

Florist struct is taken from the parameter and threadCount is increased.For every thread 1 assigned a suankiClient is kept. For the total running time of each thread, totalTime is assigned 0 initially.

threadCount: is a variable that is kept to prevent the thread function from running until all threads are created.

When threadCount is equal to numOfThread, it sends a signal with the cod4thread condition variable. After all threads are created, the queue of logged in clients is checked, if there is no client in the queue, the wait signal is thrown.

preparationTime : Randomly generated between [1,255] if there is a client.
deliveryTime : click * distance;
sleepTime : preparationTime + deliveryTime

The output of the information that it has been delivered is printed. And sleeps until sleep time(with usleep to be ms). suankiClient is increased. And this time is added to totalTime. This process continues in a loop until the queue of this thread is empty.

It exits the loop when the file ends and the queue in the florists is empty.

An object of the Statistic struct created to be returned is created and stored in memory.

In Statistic Struct :

- **name :** keeps the name of florist
- **number :** keeps the client number served to sales
- **totalTime :**service time is recorded and returned to totalTime.

Main 2. Section

An array holding a statistic result and a void * thread_result is created to get the return value.
thread_result gets the information by accessing the address of the statistic array created and returned in the thread thanks to join.

The termination of all threads is expected with join. The return value from the terminated threads is saved in the statistic array and threads close.

Required outputs are printed on the screen.

All statistics are printed on the screen.

thread_result is made free. while there are free places taken for all florists, all threads are closed. Mutex and condition variable are destroyed. At the end, the program is exited.

If the program had to close due to a problem, it calls the exitProgram function for free memory.

exitProgram function is called to make all the remaining frees. And while there are free places taken for all florists, all threads are closed. Mutex and con variable are destroyed. At the end, the program is exited.

CTRL-C

CTRL-C signal is handler. In case of catching, ctrlc variable that is defined globally is set to 1.

If the ctrlc is set to 0 for continuity of the loops in the thread and main, all returns are terminated and the program is exited properly if 1 is made.