CSE 344 SYSTEM PROGRAMMING

HOMEWORK 05 - REPORT

GÖKHAN HAS - 161044067

1 Problem Solutions

In the homework, I got the number of florists by reading the file at first. Then I created my arrays. I kept it all as a global pointer. Actually, I had struct at first. But especially when the ctrlc signal came in, sometimes it wasn't completely free in the memory leak test. In order to prevent this, I went to such a solution.

Then I read the file one by one and initialized everything. I created as many threads as the number of florists. I also created the condition variable and mutex as much as the number of florists and initialized them.

The wish list is created one by one. The distance is calculated before it is created. Here Chebyshev distance is used. Then, the smallest distance is taken by comparing all the distances. This means that it is the closest florist that sells the same flower. This is also required from us in homework.

A problem arises here. Sometimes, no seller can sell the flower that the customer wants. I also checked this. If florists do not sell the flower they want, for example, it cannot be calculated. Since this distance is not calculated, the index does not return in the florist array. So the thread cannot be awakened. Because no florist will be able to serve that customer. Because they don't sell.

Then the calculated florist thread is awakened. This is of course done using the cwait function. One two-dimensional integer array was used before being awakened. This array serves as client queue. All of its elements are initially filled with -1. Then the client in the sequence added to the queue is changed to 1. This variable has been defined globally. A separate mutex has been defined since both threads and the main function can access and change. Mutex lock is used when the client queue variable is interfered with. Thus, anyone can interfere with this variable without creating any trouble.

All resources received must be returned. This process is done at the end of the mains by destroying mutex and condition variables and using the free function. In case of any signal (ctrlc) these operations should be done and done again.

The thread function takes an index as a parameter and returns a struct that contains the information of that thread, that is, the products that the florist sells. In other words, the total sales figure and the total time are returned from the thread. Some control has been done in this function. When the CTRLC signal is received, the thread quits immediately, leaving the requests they are making. If everything goes normally, it makes his requests and leaves.

A global flag is marked in the signal function, the signal handler. This flag is checked continuously in threads. As the CTRLC signal (SIGINT) arrives at any time, the threads terminate because the value of the flags will change. After the threads are terminated, the program returns everything it uses. So there is no memory leak process. The screenshots of the tests related to this are available below. There is no memory leak when there is both normal and signal termination.

2 Used Functions

- void printUsage (): Prints the program's proper usage if it is misused.
- void errorExit (char * error): The program should give an error message and terminate cleanly.
- int getFloristCount (FILE * fptr, int * countMaxFLower): The number of flowers is taken from the file.
- void initializeArrays (int count, int countClient, int maxFlowerCount): Arrays are initialized. Each of them takes place in malloc.
- void freeArrays (int count, int clientCount, int maxFlowerCount): When the program is finished or terminated with ctrl + c, the received places are returned by this function.
- int getClientCount (FILE * fptr): The number of customers is returned.
- void readFile (FILE * fptr, int floristCount, int clientCount): The file is read and the import is done to the arrays. It should not be forgotten that the order of requests will be created one by one again.
- int getChCount (char * line, char ch): If there is a few of the value entered as a parameter in the string, it is returned.
- void * florist (void * index): The function where threads will run.

- void endSignals (int floristCount): If the request list is short or sometimes the request part is made in the main, threads may not work by not entering an interrupt. To prevent this, the condition variable in the threads must be awakened again ...
- double getChebyShevDistance (double x1, double y1, double x2, double y2): Chebyshev distance is calculated, the process is made with the shortest distance in the machine.
- int isQueueHasElement (int index, int clientCount): It is checked if there is any element in the request list, if it is not, thread should be put into action.
- void printInfo (int count, FloristEndInfo * infoArr): The last informative messages are printed.
- void signal_catcher (int sigNo): Signal capture function, SIGINT signal is captured.

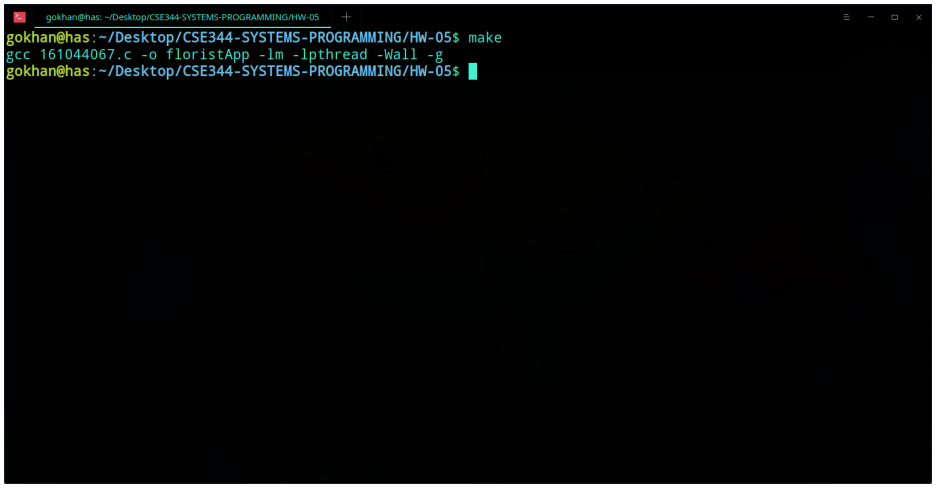

Figure 1: Compile

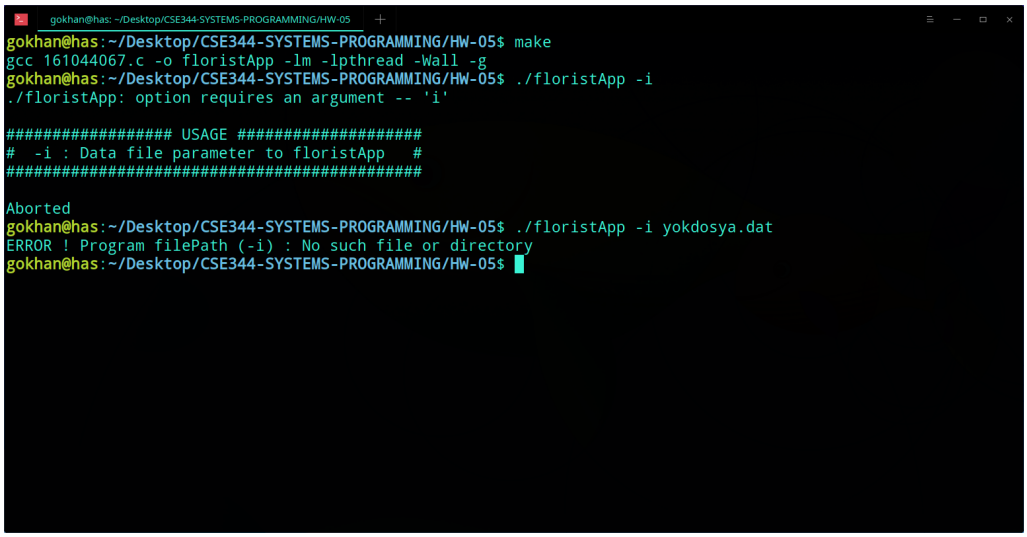

Figure 2: Wrong Input Example

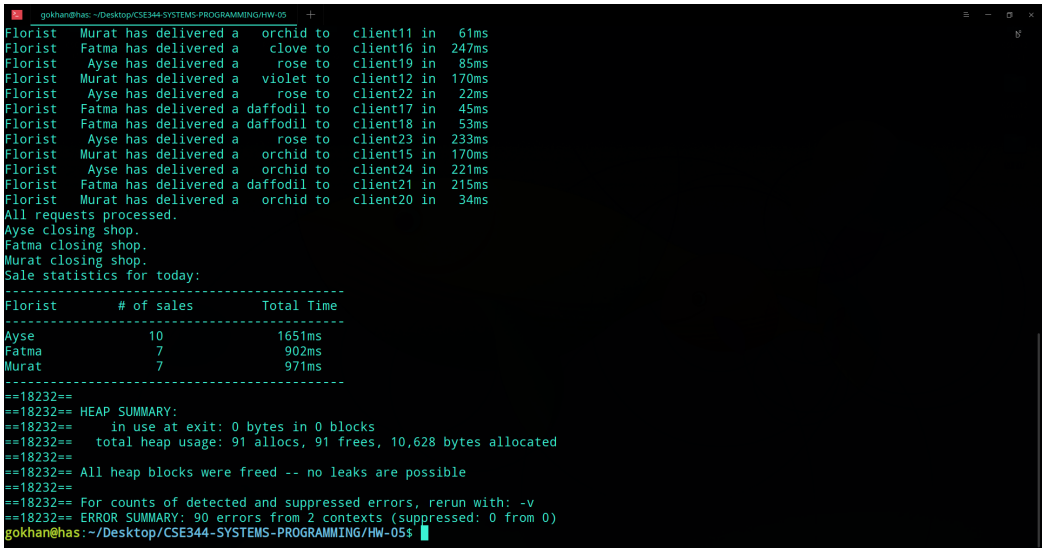

Figure 3: Normal Leak Test

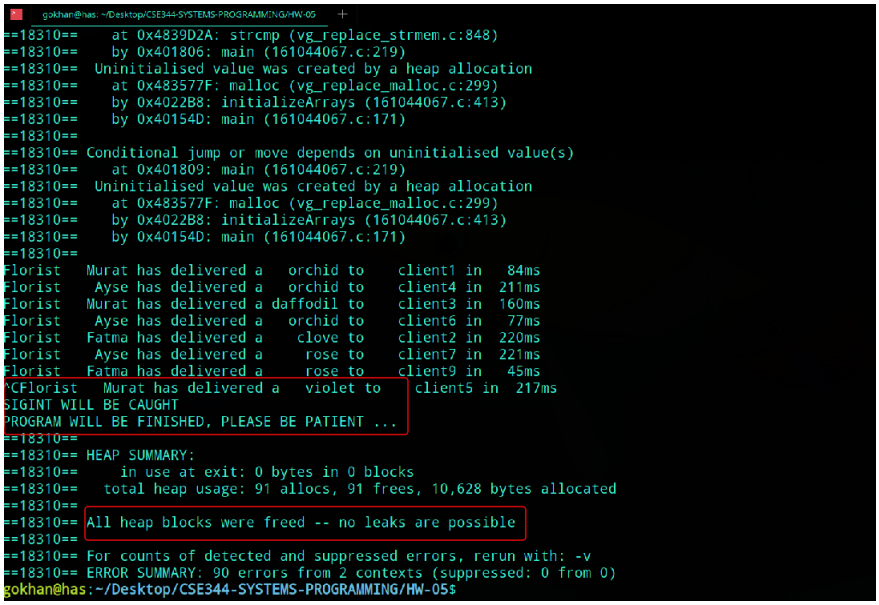

Figure 4: Signal Leak Test

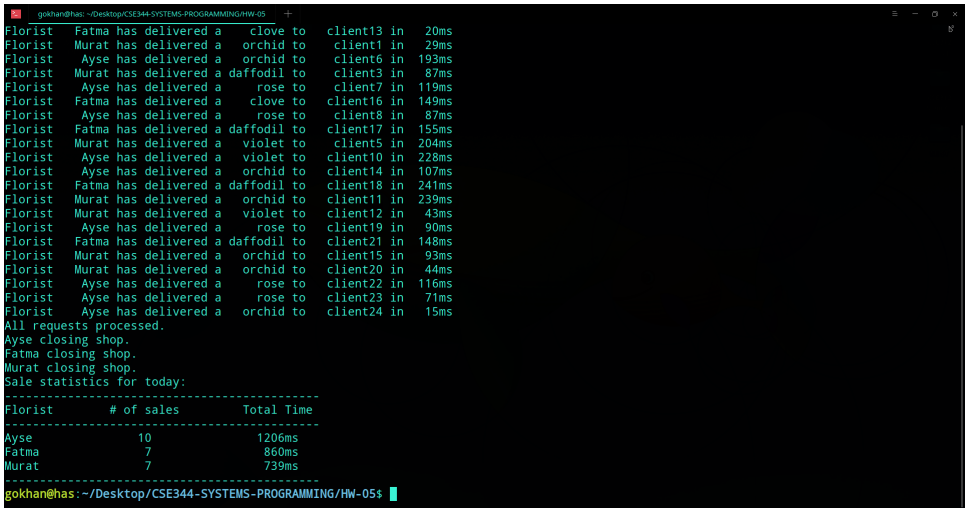

Figure 5: Normal Test

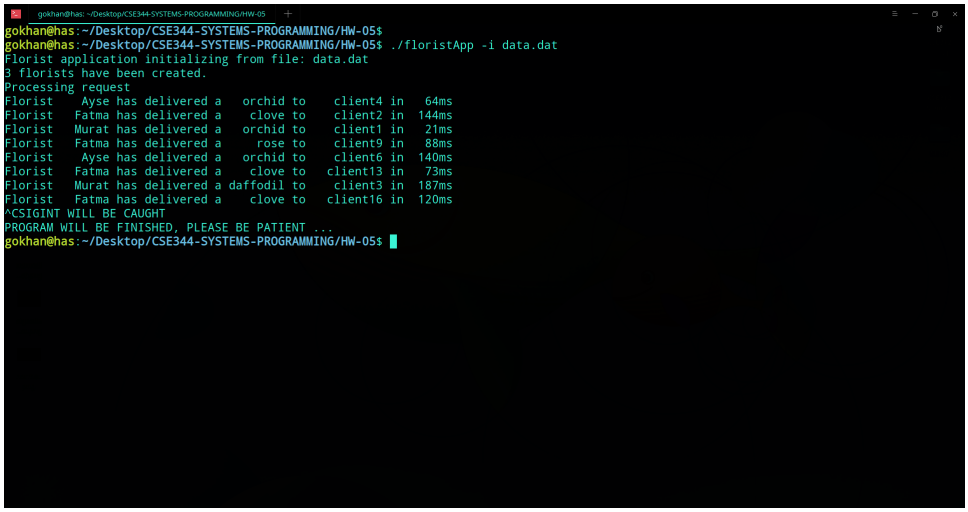

Figure 6: Normal Signal Test