

**Homework - 5 CSE344**  
**System Programming**  
**Hamza YOĞURTCUOĞLU - 171044086**

→ **main.c** : I put the **Usage** at first. I seperated command line argument with **getopt()**.

As stated in the homework pdf file, the input file is thought to be given in the proper format.

- You can assume the file is not empty and that its contents have the proper expected format.

**The Right Format**

**Ayse (10,25; 1.5) : orchid, rose, violet**      #Firstly, the information of florists  
.  
.  
.

**Murat (-10,8; 1.1) : violet, daffodil, orchid**

# just a new line

**client1 (0,4): orchid**      # Secondly, the information of clients

**client2 (1,5): clove**

.  
.  
.

.  
.  
.

**client24 (11,15): orchid**

# If you want, you can put a new line or not. Doesn't matter

**Note :** File must have valid content and at **least 1 florist**.

Flower names and character names of people are unlimited.(Probably 32 bit)

Then Florists are created.

```
struct florist
{
    int X;
    int Y;
    double speed;
    char * name;
    char ** flowerType;
    int flowerTypeSize;
    struct Queue que;
    struct saleStatis sta;
    pthread_mutex_t mutexF;
    pthread_cond_t conditionalV;
    int closingVariable;
};
```

Each florist has a X and Y integer-coordinate and double-speed. I create name and flower types of florist in heap. Flower name or florist name could be 500000 char :) Each one has a queue that has client array. Main thread can put client their queue with

this way.

### HOW THE FLORISTS SEND STATISTICS TO THE MAIN THREAD ?

Each florist has a saleStatis object in its own struct structure. When the florists are closingVariable 1 and there is no client left, this variable `pthread_exit(&floristInfo->sta);` returns the main thread.

The main thread, on the other hand, fills this statistical information into the salesStatis array, respectively, with :

```
pthread_join(floristThreads[i], (void **) &stas [i]);
```

There are total spent preparing time and delivering request in saleStatis struct.

Once all requests have been processed, all florist threads should terminate and return their sale statistics (i.e. how much time each florist spent in total for preparing and delivering the requests) to

**Note :** In statistics, only preparing time has been collected. Ref. Above image. Statistic just **sum time of preparation**

**Florist Fatma has delivered a clove to client2 in 56ms**

If here total = time of preparation + time of delivery

time of preparation = Random(1-250)ms

time of delivery = distance/speed

## CHEBYSHEV DISTANCE

It measures distance between two points (2D vector) by selecting **maximum absolute difference** on 2D vector element absolute difference.

Example : x(10,-10) , y(-1,4)

dif1 = abs(x[0] - y[0]) = abs(10 - (-1)) = 11

dif2 = abs(x[1] - y[1]) = abs(-10 - 4) = 14

**maximum difference = 14**

**equation : distance = max(|x[0]-y[0]|,|x[1]-y[1]|)**

**Reference** : <http://cljavacode.blogspot.com/2017/02/chebyshev-distance-between-two-points.html>

## HOW TO SOLVE SYNCHRONIZATION BETWEEN THREADS ?

```
struct florist
{
    int X;
    int Y;
    double speed;
    char * name;
    char ** flowerType;
    int flowerTypeSize;
    struct Queue que;
    struct saleStatis sta;
    pthread_mutex_t mutexF;
    pthread_cond_t conditionalV;
    int closingVariable;
};
```

Synchronization issues are solved using only mutex and condition variables. Every florist has a **mutex** for the critic section.

If there are no clients and requests are not yet finished (main thread not finished), florist waiting a signal from main thread :

**pthread\_cond\_wait( &floristInfo->conditionalV, &floristInfo-> mutexF);**

florist left **mutexF** then waiting signal.

After reading the client information from the main thread file, it is added to the queue of the nearest florist(florist must have request flower). Then the main thread sends a signal to the florist. If the florist's queue already has a request, it continues to prepare the flowers without waiting.

**NOTE** : I print with write function to console. I used a common mutex for it.

## HOW ALL THREADS CLOSE ?

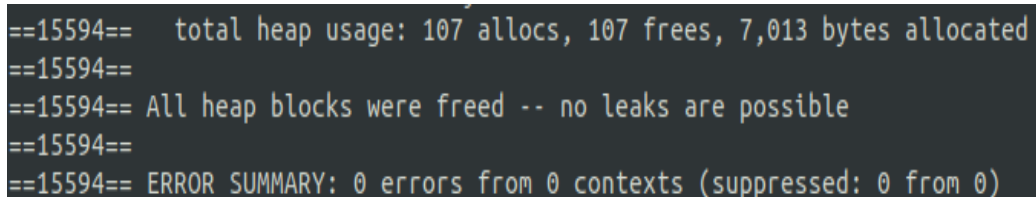
After reading all requests, the main thread sets the closingVariable variable of all threads to 1. Thus, every florist makes all requests and returns statistics to the main thread.

## HOW DOES THE MAIN THREAD DELEGATE THE REQUESTS IN THE MEANTIME TO THREADS?

Each florist has a request queue. If a flower is processing another client request, the main thread can continue by adding a new request queue at the same time.

## → WHEN CTRL + C PRESSING IS DONE, ARE ALL THE RESOURCES BEIGN RELEASED ?

All string arrays in heap memory. In this way, if a memory allocate is made from a method, these resources can be released in SIGINT(CTRL+C) handler. Below you can see the result of **valgrind**:



```
==15594== total heap usage: 107 allocs, 107 frees, 7,013 bytes allocated
==15594==
==15594== All heap blocks were freed -- no leaks are possible
==15594==
==15594== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

**Figure 1 : Freed all resources when ctrl+c pressing and normal execution.**

There is no error or warning from any processes.(This image is just a process)

In SIGINT(CTRL+C) handler function, terminate is converted to 0, so that the program ends **after the last florist delivering process ends.**

Then main thread releases all allocated source and no zombies.

**Test Program Examples:** The following commands write how you can run process.

You can execute with Absolute path or relative path for files.

```
./floristApp -i data.dat
```

```
./floristApp -i (AbsolutePath)/data.dat
```

**Input File Examples:** The file format should be as described on page 1.

```
1  Ayse (10,25; 1.5) : orchid, rose, violet
2  Fatma (-10,-15; 1.3) : clove, rose, daffodil
3  Murat (-10,8; 1.1) : violet, daffodil, orchid
4
5  client1 (0,4): orchid
6  client2 (1,5): clove
7  client3 (2,10): daffodil
8  client4 (4,15): orchid
9  client5 (8,-21): violet
10 client6 (-1,21): orchid
11 client7 (-6,20): rose
12 client8 (-16,18): rose
13 client9 (-12,-3): rose
14 client10 (23,0): violet
15 client11 (5,1): orchid
16 client12 (7,-8): violet
17 client13 (8,-3): clove
18 client14 (9,8): orchid
19 client15 (6,5): orchid
20 client16 (2,6): clove
21 client17 (-6,-4): daffodil
22 client18 (-9,-6): daffodil
23 client19 (-4,16): rose
24 client20 (-9,26): orchid
25 client21 (-4,-12): daffodil
26 client22 (9,13): rose
27 client23 (12,18): rose
28 client24 (11,15): orchid
29
```

a new line between florist and client

**Note :** File must have valid content and at least 1 florist.

→ **Output Example :**

```
Florist application initializing from file: data.dat
3 florists have been created
Processing requests
Florist Fatma has delivered a clove to client2 in 63.385 ms
Florist Ayse has delivered a orchid to client4 in 89.667 ms
Florist Murat has delivered a orchid to client1 in 120.091 ms
Florist Murat has delivered a daffodil to client3 in 14.909 ms
Florist Ayse has delivered a orchid to client6 in 122.333 ms
Florist Murat has delivered a violet to client5 in 130.364 ms
Florist Fatma has delivered a rose to client9 in 217.231 ms
Florist Fatma has delivered a clove to client13 in 79.846 ms
```

...  
...  
...

```
Florist Ayse has delivered a orchid to client24 in 229.667 ms
All requests processed
Ayse closing shop.
Fatma closing shop.
Murat closing shop.
Sale statistics for today:
-----
Florist          # of sales      Total time
-----
Ayse              10             1306.667ms
Fatma              7             764.615ms
Murat              7             672.364ms
```

All florist is terminating. Then Printed statistics

## → Compiler and Run :

Makefiles Commands:

```
make #that command compiler
make clean #that command cleans all object files
```

To run :

```
./floristApp -i data.dat
```

If you want to see outputs. Check above **outputs**.

- Note :
- 1) There is a newline('\n') between client and florist
  - 2) In SIGINT(CTRL+C) handler function, terminate is converted to 0, so that the program ends after the last client delivering process ends.

Thanks :)