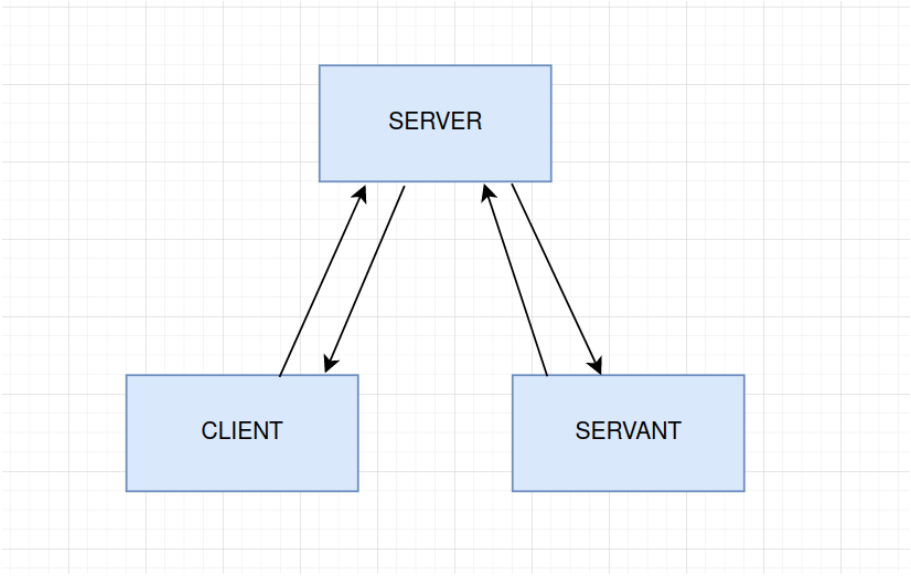
CSE344 – System Programming Final Project Report

Design Approach:

I create three main files which are client, server and servant. Also there is a helper file to increase the reusability.



General communication diagram

1. **Client**

To complete client tasks, first I checked the command line arguments. After that, I start with the reading input file. I checked the whether file is exist or not. I ignore the wrong request of the file. After all requests are read and saved, I create the thread which is count is same with number of proper request. After each thread is created, I joined them. To achieve the sending all request to server same time, I use condition variable. Once any thread is ready, it's waiting for the all threads thanks to cwait

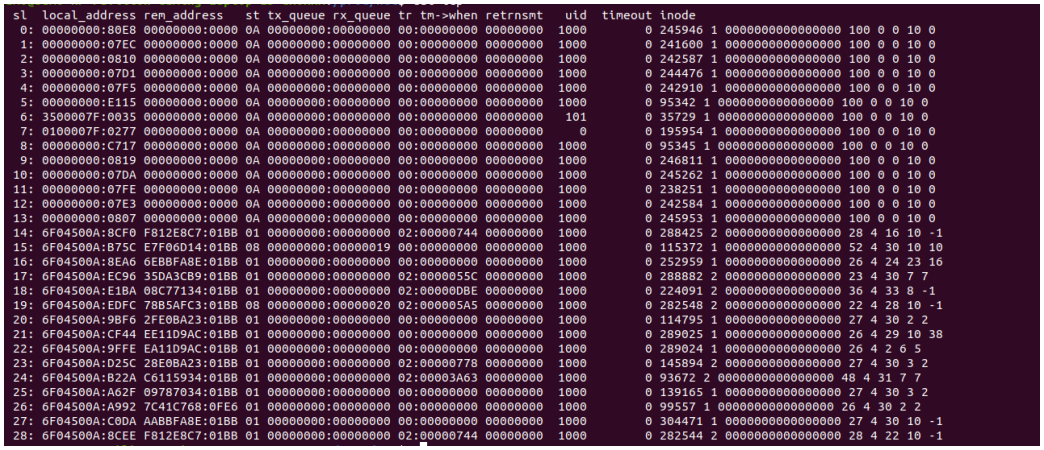
function. Once the condition variable is satisfied, last thread broadcast the waiting threads, and they are sending to request to server. For the purpose of communicating with server, instead of sending line of the file, I create a proper struct and fill the informations. To determine coming request's resource, I use general message type, which contains integer type and buffer. According to integer type, I cast the buffer to proper data structure. After the request is sent to server, client waits for the answer and when It get the information, It's cleans resources and exit.

2. **Server**

To complete server tasks, first I checked the command line arguments. After that, I create the threads and detached them. Also I use the queue data structures to hold incoming requests. To prevent the busy waiting, I use the cwait function If there is no request. Also to synchronize the threads I use mutex. For the purpose of waking up thread, I checked the all status of threads, If any of them free to use, I call the pthread_cond_signal function the wake up only one thread. If the queue is not empty, any thread take the request by using proper descriptor and detects where the request come from. If the request coming from servant and it's type is declaration for the cities and ports, I saved the information to use it for the purpose of handle requests. Servant sends port number, process id and cities which is responsible for. If the request type client request, I checked the cities that are handling by servants and send the request to proper one. After the request sends the servant, I started to wait for response. Once the response is come, I send it to client. If the SIGINT is come, I send the proper message to servants and after than I clear resources and terminate the program.

3. **Servant**

To complete servant tasks, first I checked the command line arguments. If the arguments are okay, I start with finding empty port. To find empty port, I added to 2000 which is recommended from the homework file to start argument which is unique. To be sure, I open the /proc/net/tcp file. File is looking like this:



```
sl  local_address rem_address   st tx_queue rx_queue tr tm->when retrnsmt   uid  timeout inode
 0: 00000000:80E8 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 245946 1 0000000000000000 100 0 0 10 0
 1: 00000000:07EC 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 241600 1 0000000000000000 100 0 0 10 0
 2: 00000000:0810 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 242587 1 0000000000000000 100 0 0 10 0
 3: 00000000:07D1 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 244476 1 0000000000000000 100 0 0 10 0
 4: 00000000:07F5 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 242910 1 0000000000000000 100 0 0 10 0
 5: 00000000:E115 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 95342 1 0000000000000000 100 0 0 10 0
 6: 3500007F:0035 00000000:0000 0A 00000000:00000000 00:00000000 00000000   101        0 35729 1 0000000000000000 100 0 0 10 0
 7: 0100007F:0277 00000000:0000 0A 00000000:00000000 00:00000000 00000000     0        0 195954 1 0000000000000000 100 0 0 10 0
 8: 00000000:C717 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 95345 1 0000000000000000 100 0 0 10 0
 9: 00000000:0819 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 246811 1 0000000000000000 100 0 0 10 0
10: 00000000:07DA 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 245262 1 0000000000000000 100 0 0 10 0
11: 00000000:07FE 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 238251 1 0000000000000000 100 0 0 10 0
12: 00000000:07E3 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 242584 1 0000000000000000 100 0 0 10 0
13: 00000000:0807 00000000:0000 0A 00000000:00000000 00:00000000 00000000  1000        0 245953 1 0000000000000000 100 0 0 10 0
14: 6F04500A:8CF0 F812E8C7:01BB 01 00000000:00000000 02:00000744 00000000  1000        0 288425 2 0000000000000000 28 4 16 10 -1
15: 6F04500A:B75C E7F06D14:01BB 08 00000000:00000019 00:00000000 00000000  1000        0 115372 1 0000000000000000 52 4 30 10 10
16: 6F04500A:8EA6 6EBBFA8E:01BB 01 00000000:00000000 00:00000000 00000000  1000        0 252959 1 0000000000000000 26 4 24 23 16
17: 6F04500A:EC96 35DA3CB9:01BB 01 00000000:00000000 02:0000055C 00000000  1000        0 288882 2 0000000000000000 23 4 30 7 7
18: 6F04500A:E1BA 08C77134:01BB 01 00000000:00000000 02:00000DBE 00000000  1000        0 224091 2 0000000000000000 36 4 33 8 -1
19: 6F04500A:EDFC 78B5AFC3:01BB 08 00000000:00000020 02:000005A5 00000000  1000        0 282548 2 0000000000000000 22 4 28 10 -1
20: 6F04500A:9BF6 2FE0BA23:01BB 01 00000000:00000000 00:00000000 00000000  1000        0 114795 1 0000000000000000 27 4 30 2 2
21: 6F04500A:CF44 EE11D9AC:01BB 01 00000000:00000000 00:00000000 00000000  1000        0 289025 1 0000000000000000 26 4 29 10 38
22: 6F04500A:9FFE EA11D9AC:01BB 01 00000000:00000000 00:00000000 00000000  1000        0 289024 1 0000000000000000 26 4 2 6 5
23: 6F04500A:D25C 28E0BA23:01BB 01 00000000:00000000 02:00000778 00000000  1000        0 145894 2 0000000000000000 27 4 30 3 2
24: 6F04500A:B22A C6115934:01BB 01 00000000:00000000 02:00003A63 00000000  1000        0 93672 2 0000000000000000 48 4 31 7 7
25: 6F04500A:A62F 09787034:01BB 01 00000000:00000000 00:00000000 00000000  1000        0 139165 1 0000000000000000 27 4 30 3 2
26: 6F04500A:A992 7C41C768:0FE6 01 00000000:00000000 00:00000000 00000000  1000        0 99557 1 0000000000000000 26 4 30 2 2
27: 6F04500A:C0DA AABBFA8E:01BB 01 00000000:00000000 00:00000000 00000000  1000        0 304471 1 0000000000000000 27 4 30 10 -1
28: 6F04500A:8CEE F812E8C7:01BB 01 00000000:00000000 02:00000744 00000000  1000        0 282544 2 0000000000000000 28 4 22 10 -1
```

In the local address part, last four digit is showing us to opened ports. I checked them each by each whether is already In used or not. If It's already opened, I increase the port number until to find a proper one. To find the process number, I go through the /proc folder and I traverse the all folders that are showing process ID. I checked the argument file inside the each folder, whether they are equal with our servants argument. After I found the correct one I saved it. After than, I read the folders. As a data structure to save entries, I use hash table to access any city in O(1) time. As a key I use folder name and as a entry I use binary tree which is ordered according to dates. So this way finding proper dates will be fast. Binary tree is a consist on entries inside of the file. After dataset is created, I send the proper informations to server and start to waiting for requests. After each request has come I create another thread to handle with request as It mentioned in homework file. After the entries are counted, each thread sends response the server. Also If request type is SIGINT, thread terminates the program.

Data Structures:

1. **Entry Structure**

This structure represents the content of the files inside the city folders.

```
struct entry // entry represent the inside of the file
{
    date date;
    int transaction_id;
    char real_estate[CHAR_ARRAY_SIZE];
    char street_name[CHAR_ARRAY_SIZE];
    int surface;
    int price;
};
```

2. **Request Structure**

This structure represents the content of the request file.

```
struct request
{
    char transcationCount[ARRAY_SIZE];
    char field_type[ARRAY_SIZE];
    char city[ARRAY_SIZE];
    date start_date;
    date end_date;
};
```

3. **Servant Declaration Structure**

This structure represents the servant informations.

```
struct servantDeclaration
{
    int port;
    int processID;
    int total_city_number;
    char cities[BUFFER_SIZE][CHAR_ARRAY_SIZE];
};
```

4. **Common Message Structure**

Message structure that is used for communication. Type is representing the resource of the message.

```
struct message
{
    enum messageType type;
    uint8_t buffer[MESSAGE_BUFFER_SIZE];
};
```

5. **Message Types**

Message types that are used in server, servant and client.

```
enum messageType
{
    CLIENT_REQUEST,
    RESPONSE,
    EXIT,
    SERVANT_DECLARATION,
    SERVANT_RESPONSE,
    NO_CITY_RESPONSE,
};
```

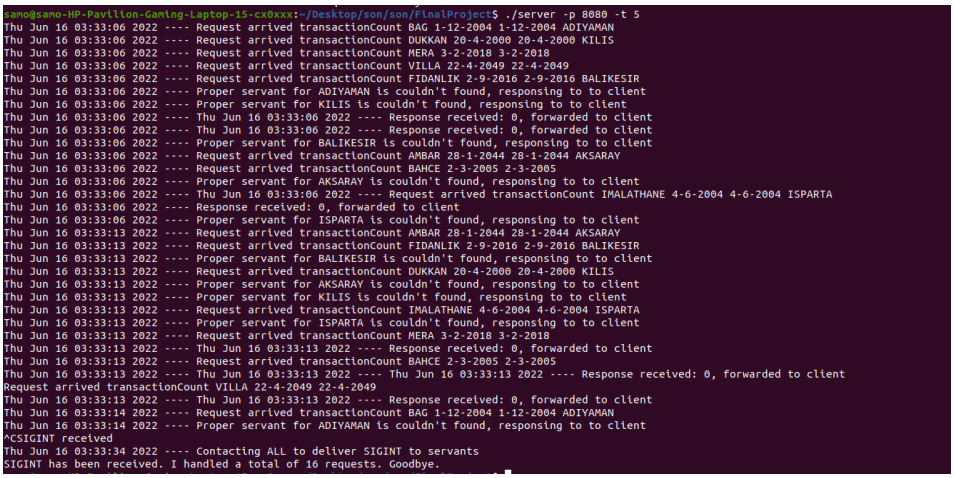
Requirements That Are I Achieved:

I test the several inputs both correct and wrong. Proper ones seem correct and wrong ones are handled. There is no warning and memory leak. I think I achieve the all requirements that are mentioned in the homework file.

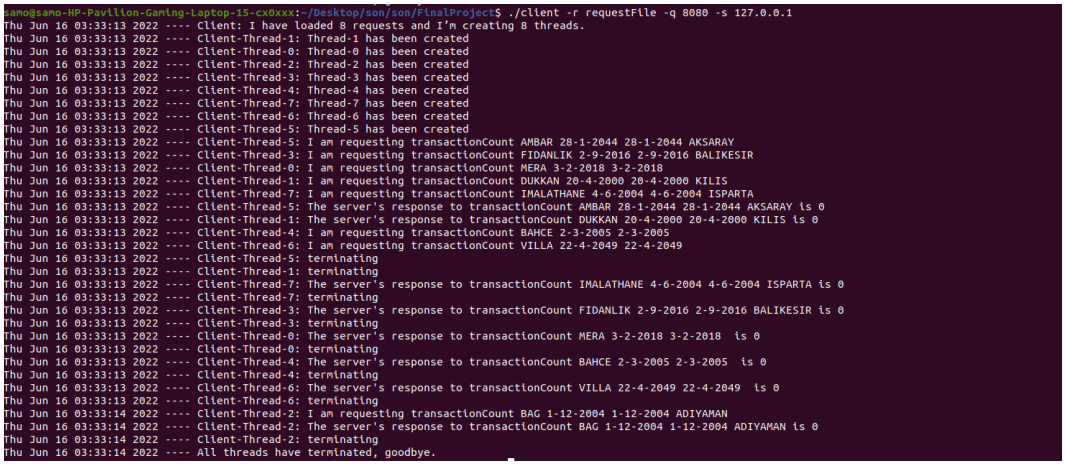
Outputs From The Program:

- **Output 1**

Server output that there is no opened servant. Output also shows the SIGINT is catched properly.

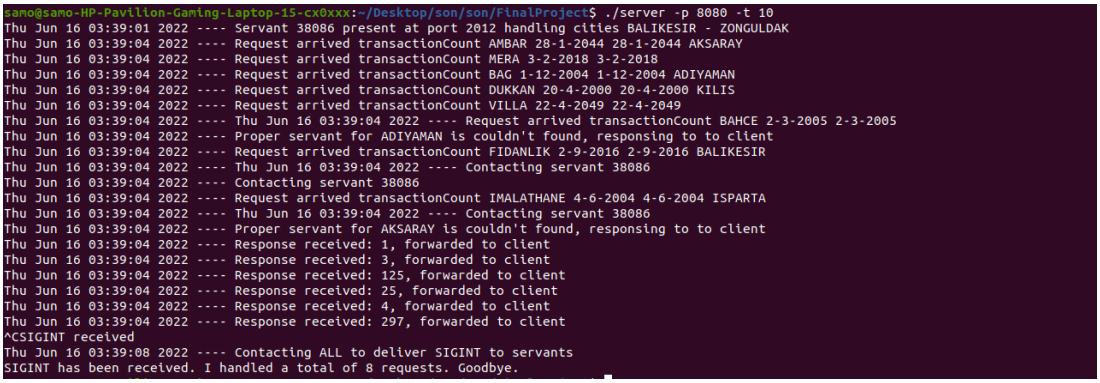


Client output which is sent to mentioned server.



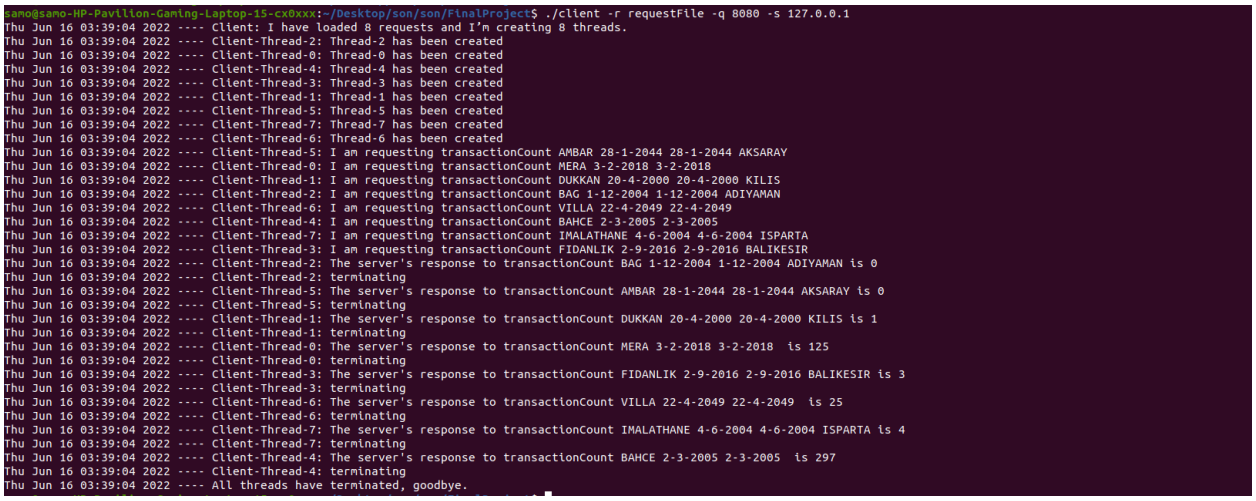
- **Output 2**

Server output with the several servant opened.



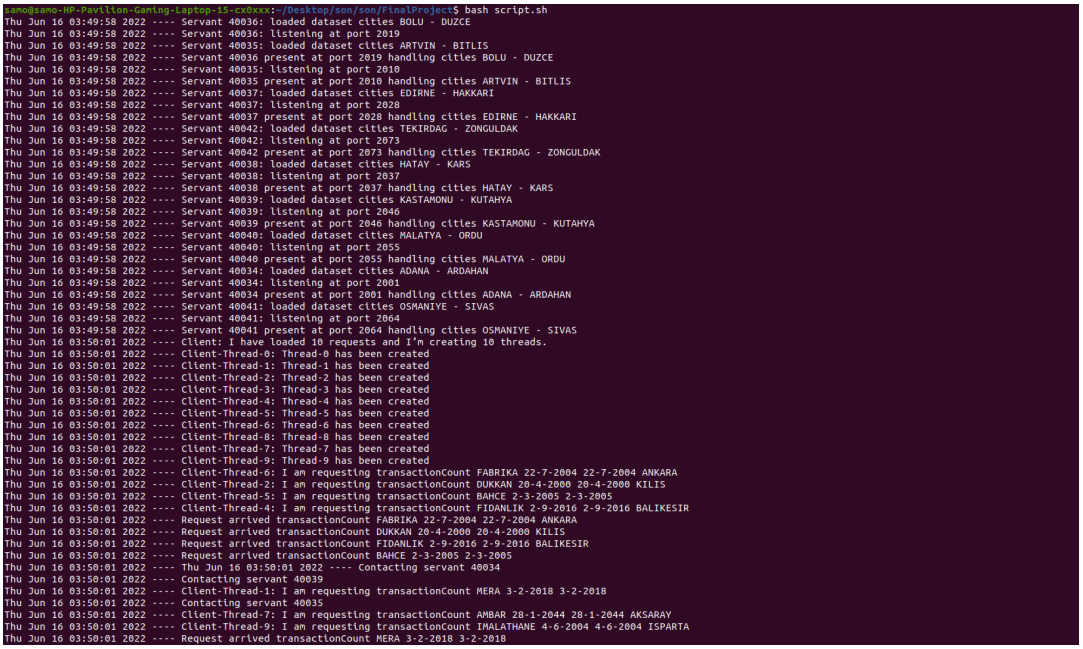
Servant output that show handled requests.

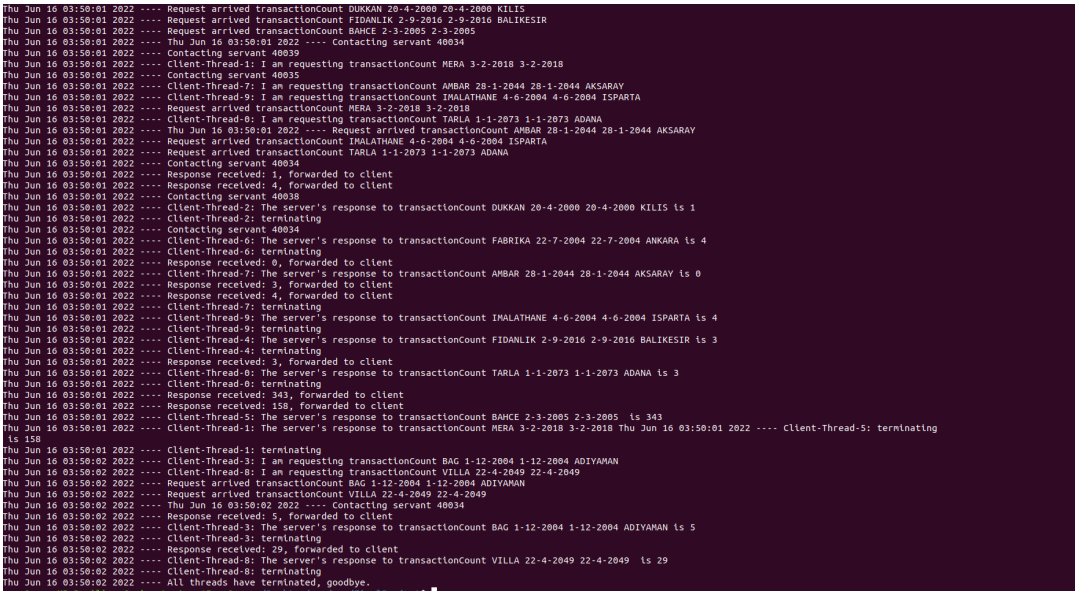
Client output that showed the requests and their response.



- **Output 3**

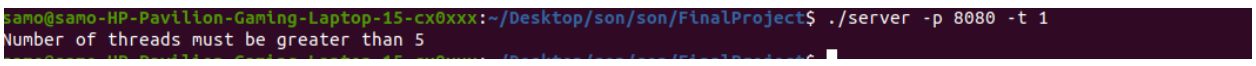
Given script test.



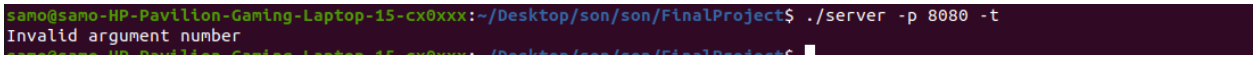


- **Some error outputs**

Thread number that is lower t.



Invalid argument.



Trying to connect non existing server from servant.

```
samo@samo-HP-Pavilion-Gaming-Laptop-15-cx0xxx:~/Desktop/son/son/FinalProject$ ./servant -d dataset -c 12-81 -r 127.0.01 -p 8080
Thu Jun 16 03:59:09 2022 ---- Servant 40608: loaded dataset cities BALIKESIR - ZONGULDAK
Thu Jun 16 03:59:09 2022 ---- Servant 40608: listening at port 2012
Connection failed!
samo@samo-HP-Pavilion-Gaming-Laptop-15-cx0xxx:~/Desktop/son/son/FinalProject$
```

Trying to connect non existing server from client.

```
samo@samo-HP-Pavilion-Gaming-Laptop-15-cx0xxx:~/Desktop/son/son/FinalProject$ ./client -r requestFile -q 8080 -s 127.0.0.1
Thu Jun 16 04:03:52 2022 ---- Client: I have loaded 10 requests and I'm creating 10 threads.
Thu Jun 16 04:03:52 2022 ---- Client-Thread-2: Thread-2 has been created
Thu Jun 16 04:03:52 2022 ---- Client-Thread-1: Thread-1 has been created
Thu Jun 16 04:03:52 2022 ---- Client-Thread-5: Thread-5 has been created
Thu Jun 16 04:03:52 2022 ---- Client-Thread-0: Thread-0 has been created
Thu Jun 16 04:03:52 2022 ---- Client-Thread-4: Thread-4 has been created
Thu Jun 16 04:03:52 2022 ---- Client-Thread-3: Thread-3 has been created
Thu Jun 16 04:03:52 2022 ---- Client-Thread-6: Thread-6 has been created
Thu Jun 16 04:03:52 2022 ---- Client-Thread-7: Thread-7 has been created
Thu Jun 16 04:03:52 2022 ---- Client-Thread-8: Thread-8 has been created
Thu Jun 16 04:03:52 2022 ---- Client-Thread-9: Thread-9 has been created
Error connecting to server
Error connecting to server
Error connecting to server
```

- **Valgrind Results**

1. Server result

```
samo@samo-HP-Pavilion-Gaming-Laptop-15-cx0xxx:~/Desktop/son/son/FinalProject$ valgrind ./server -p 8080 -t 5
==41196== Memcheck, a memory error detector
==41196== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==41196== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==41196== Command: ./server -p 8080 -t 5
==41196==
^CSIGINT received
Thu Jun 16 04:08:35 2022 ---- Contacting ALL to deliver SIGINT to servants
SIGINT has been received. I handled a total of 0 requests. Goodbye.
==41196==
==41196== HEAP SUMMARY:
==41196==     in use at exit: 1,360 bytes in 5 blocks
==41196==   total heap usage: 23 allocs, 18 frees, 3,081,526 bytes allocated
==41196==
==41196== LEAK SUMMARY:
==41196==    definitely lost: 0 bytes in 0 blocks
==41196==    indirectly lost: 0 bytes in 0 blocks
==41196==      possibly lost: 1,360 bytes in 5 blocks
==41196==    still reachable: 0 bytes in 0 blocks
==41196==         suppressed: 0 bytes in 0 blocks
==41196== Rerun with --leak-check=full to see details of leaked memory
==41196==
==41196== For lists of detected and suppressed errors, rerun with: -s
==41196== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
samo@samo-HP-Pavilion-Gaming-Laptop-15-cx0xxx:~/Desktop/son/son/FinalProject$
```

2. Servant result

```
samo@samo-HP-Pavilion-Gaming-Laptop-15-cx0xxx:~/Desktop/son/son/FinalProject$ valgrind ./servant -d dataset -c 12-81 -r 127.0.01 -p 8080
==41272== Memcheck, a memory error detector
==41272== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==41272== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==41272== Command: ./servant -d dataset -c 12-81 -r 127.0.01 -p 8080
==41272==
==41272== Conditional jump or move depends on uninitialised value(s)
==41272==    at 0x10B1E4: findProcessNumber (servant.c:614)
==41272==    by 0x1098E3: main (servant.c:103)
==41272==
==41272== Conditional jump or move depends on uninitialised value(s)
==41272==    at 0x10B218: findProcessNumber (servant.c:614)
==41272==    by 0x1098E3: main (servant.c:103)
==41272==
Thu Jun 16 04:10:53 2022 ---- Servant -1: loaded dataset cities BALIKESIR - ZONGULDAK
Thu Jun 16 04:10:53 2022 ---- Servant -1: listening at port 2012
Connection failed!
==41272==
==41272== HEAP SUMMARY:
==41272==     in use at exit: 0 bytes in 0 blocks
==41272==   total heap usage: 7,394 allocs, 7,394 frees, 4,099,113 bytes allocated
==41272==
==41272== All heap blocks were freed -- no leaks are possible
==41272==
==41272== Use --track-origins=yes to see where uninitialised values come from
==41272== For lists of detected and suppressed errors, rerun with: -s
==41272== ERROR SUMMARY: 123 errors from 2 contexts (suppressed: 0 from 0)
samo@samo-HP-Pavilion-Gaming-Laptop-15-cx0xxx:~/Desktop/son/son/FinalProject$
```

3. Client result