Extra Project 2 Report

Zeynep Hamza Ozcan 151044011

This project consists of 4 processes. As mentioned in the assignment these are Distributor, Client, Server and Servants. Their inner works will be explained in their respective sections.

Basically, Client asks Server some questions, Server learns the answers to those questions from Servants then forwards the answer to Client. All of these processes use threads to handle their connections in order to not block their main threads and work on their jobs in parallel.

It should be noted that the project was developed on an Ubuntu 16.04 machine and was tested on an Ubuntu 16.04, Ubuntu 20.04, Ubuntu 22.04 machines. All three versions generates the consistent outputs which is included in the last three topics.

Data Generator

Data Generator is a simple bash script. It first checks if there are any invalid number of arguments. Then it checks argument#1 cities file, reads it and puts it in “cities_arr” array. Checks the argument#2, puts it in “types_arr” array.

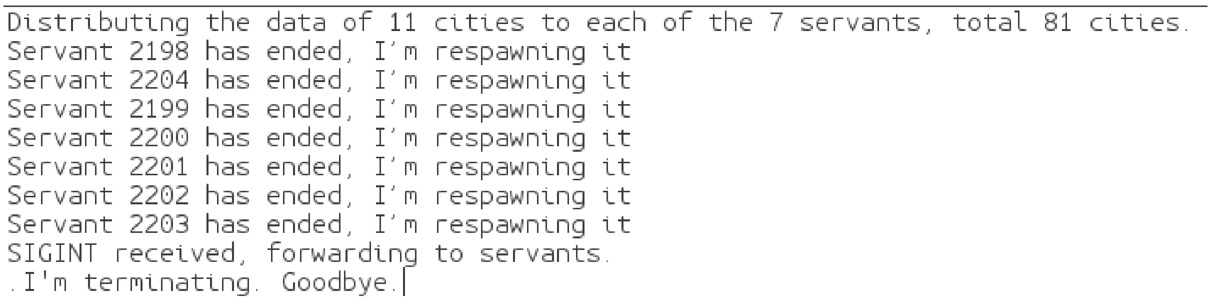
It then creates the argument#3 data directory. Then creates sub directories according to “cities_arr”. After it reads argument#4 and #5 (files per directory and rows per file respectively) the generator now proceeds to generate contents.
For transaction_id it starts from 0 and increases by 1 for each entry to maintain uniqueness easily.
In a for loop that goes through “cities_arr”, “0 to files per directory” then “0 to rows per file”
Days months and years are randomly generated between 01-01-2000-to 01-01-2200.
Real estate types is selected from types_arr randomly.
Name of the street is generated as random Upper case letters with min 3 max 12 letters.
Surface is generated as 35-1000 random number.
Price is generated as 100000 to 10000000 random number.
After the program runs it notifies the user with "Generated files."

The Distributor

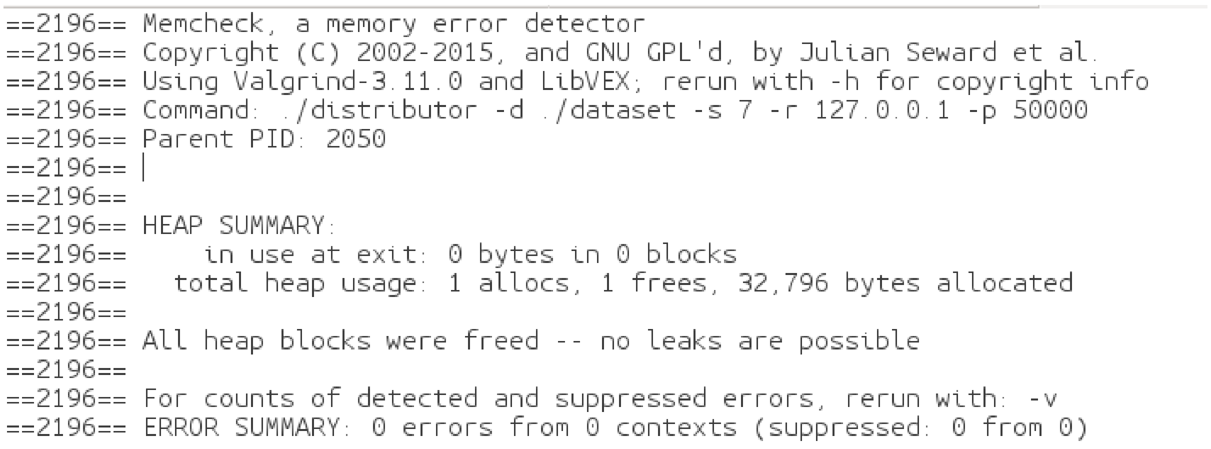
Distributor program reads the contents of the dataset folder determines how many folders are there. After that it uniformly distributes these folders to soon to be created servants. If the division cant be done It tries to distribute the remainder of folders to servants for example 28 folder to 5 servant would look like this; (6 6 6 5 5). Now that it knows which servant will be responsible for which folders it starts to fork() new processes and execv servant program from them. After each fork() but before execv() each child creates a new fifo for that servant to read when it starts. They name the fifo as “fifo1-fifo2….fifo#”. After the servant executes (execv) it reads that fifo# then closes it, this will mentioned again in the servant part.

While Distributor creates new servants it also keeps track of their pids and their respective cities, this is done because after all servants are executed the distributor waits for them to die unexpectedly, if they die it executes them again with the same way before with same cities, and also notifies them to not disturb server when they respawn.

When SIGINT is received the distribitor signals all servants the signal and collects them, then it exits gracefully.



```
Distributing the data of 11 cities to each of the 7 servants, total 81 cities.
Servant 2198 has ended, I'm respawning it
Servant 2204 has ended, I'm respawning it
Servant 2199 has ended, I'm respawning it
Servant 2200 has ended, I'm respawning it
Servant 2201 has ended, I'm respawning it
Servant 2202 has ended, I'm respawning it
Servant 2203 has ended, I'm respawning it
SIGINT received, forwarding to servants.
.I'm terminating. Goodbye.
```

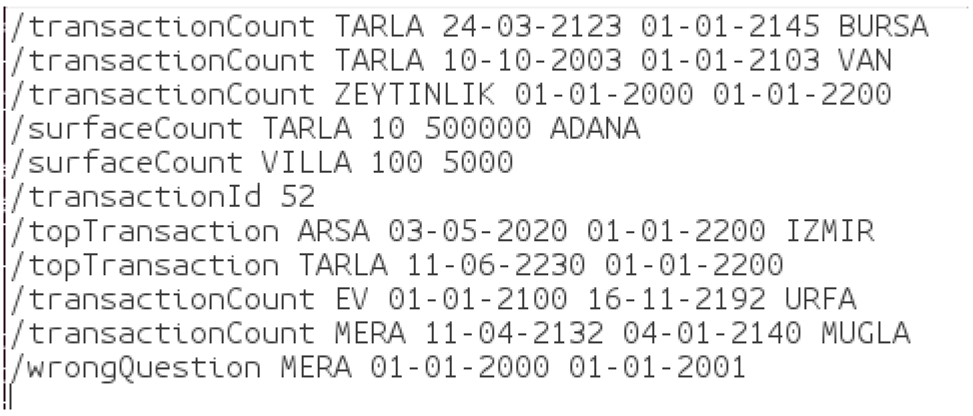


```
==2196== Memcheck, a memory error detector
==2196== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==2196== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==2196== Command: ./distributor -d ./dataset -s 7 -r 127.0.0.1 -p 50000
==2196== Parent PID: 2050
==2196==
==2196==
==2196== HEAP SUMMARY:
==2196==     in use at exit: 0 bytes in 0 blocks
==2196==   total heap usage: 1 allocs, 1 frees, 32,796 bytes allocated
==2196==
==2196== All heap blocks were freed -- no leaks are possible
==2196==
==2196== For counts of detected and suppressed errors, rerun with: -v
==2196== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

The Client

The client first reads the "requestFile" file. In this file each line is a question which will be asked to The Server. As the assignment requires The Client will create BARRIER amount of threads.

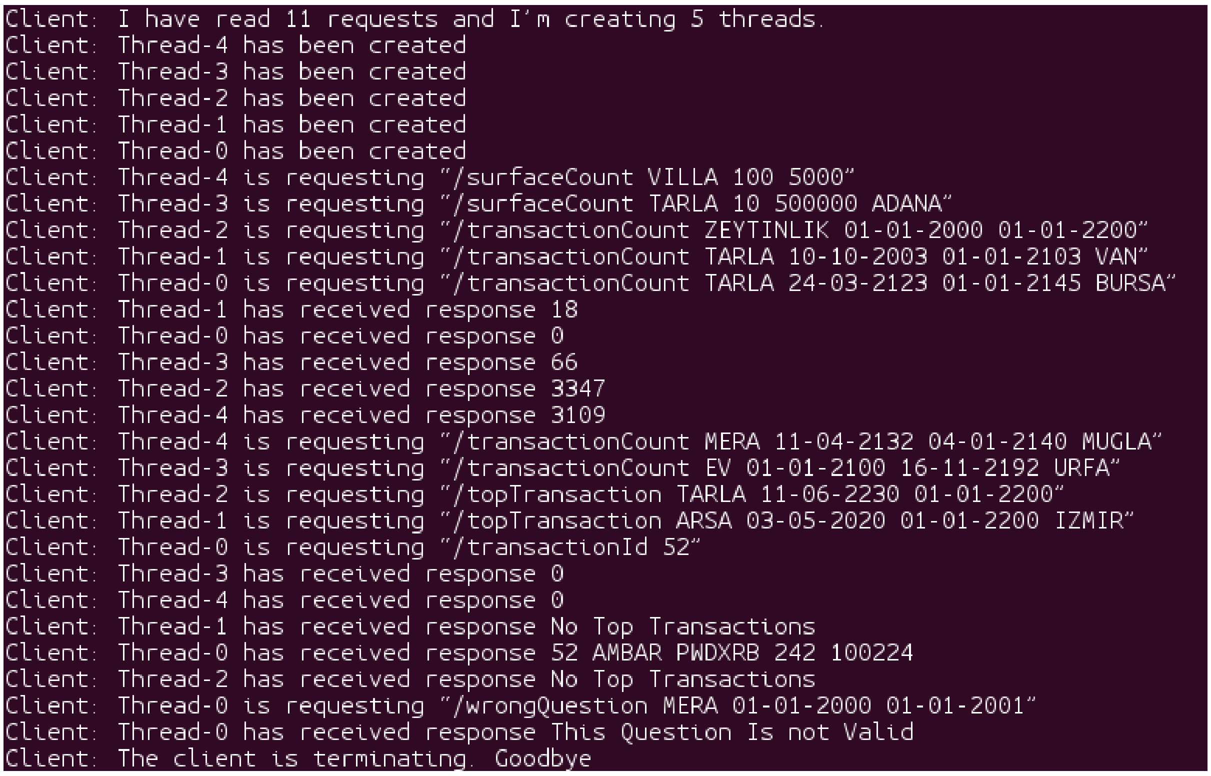
These threads will be sending their questions at the same time with a thread barrier. After every question barrage they will wait for their responses from server. When all threads receive their answer they will ask their new questions in the same manner as before.



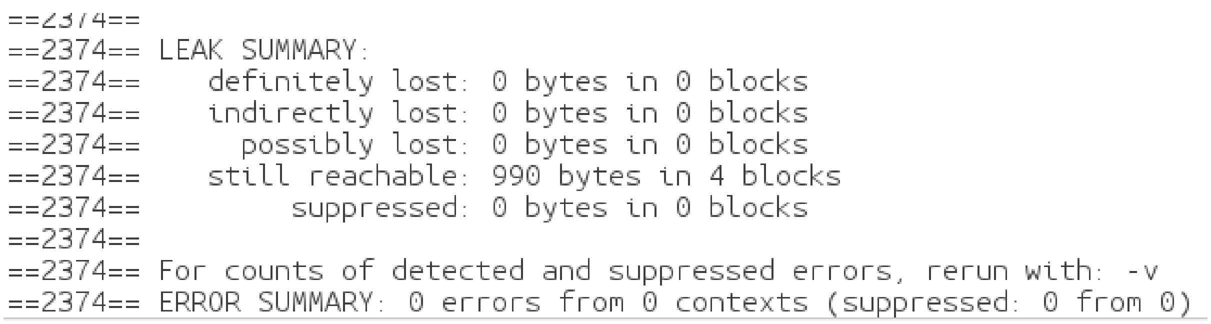
```
/transactionCount TARLA 24-03-2123 01-01-2145 BURSA
/transactionCount TARLA 10-10-2003 01-01-2103 VAN
/transactionCount ZEYTINLIK 01-01-2000 01-01-2200
/surfaceCount TARLA 10 500000 ADANA
/surfaceCount VILLA 100 5000
/transactionId 52
/topTransaction ARSA 03-05-2020 01-01-2200 IZMIR
/topTransaction TARLA 11-06-2230 01-01-2200
/transactionCount EV 01-01-2100 16-11-2192 URFA
/transactionCount MERA 11-04-2132 04-01-2140 MUGLA
/wrongQuestion MERA 01-01-2000 01-01-2001
```

In the case of Request/Barrier is not an integer number, on the last round of sending requests only the threads which will send question will wait their answer while the others wait in the barrier. When all of them arrive to the barrier they will exit gracefully.

At any moment if SIGINT is received The Client will close all of its connections and sockets then terminate.



```
Client: I have read 11 requests and I'm creating 5 threads.
Client: Thread-4 has been created
Client: Thread-3 has been created
Client: Thread-2 has been created
Client: Thread-1 has been created
Client: Thread-0 has been created
Client: Thread-4 is requesting "/surfaceCount VILLA 100 5000"
Client: Thread-3 is requesting "/surfaceCount TARLA 10 500000 ADANA"
Client: Thread-2 is requesting "/transactionCount ZEYTINLIK 01-01-2000 01-01-2200"
Client: Thread-1 is requesting "/transactionCount TARLA 10-10-2003 01-01-2103 VAN"
Client: Thread-0 is requesting "/transactionCount TARLA 24-03-2123 01-01-2145 BURSA"
Client: Thread-1 has received response 18
Client: Thread-0 has received response 0
Client: Thread-3 has received response 66
Client: Thread-2 has received response 3347
Client: Thread-4 has received response 3109
Client: Thread-4 is requesting "/transactionCount MERA 11-04-2132 04-01-2140 MUGLA"
Client: Thread-3 is requesting "/transactionCount EV 01-01-2100 16-11-2192 URFA"
Client: Thread-2 is requesting "/topTransaction TARLA 11-06-2230 01-01-2200"
Client: Thread-1 is requesting "/topTransaction ARSA 03-05-2020 01-01-2200 IZMIR"
Client: Thread-0 is requesting "/transactionId 52"
Client: Thread-3 has received response 0
Client: Thread-4 has received response 0
Client: Thread-1 has received response No Top Transactions
Client: Thread-0 has received response 52 AMBAR PWDXRB 242 100224
Client: Thread-2 has received response No Top Transactions
Client: Thread-0 is requesting "/wrongQuestion MERA 01-01-2000 01-01-2001"
Client: Thread-0 has received response This Question Is not Valid
Client: The client is terminating. Goodbye
```



```
==2374==
==2374== LEAK SUMMARY:
==2374==    definitely lost: 0 bytes in 0 blocks
==2374==    indirectly lost: 0 bytes in 0 blocks
==2374==      possibly lost: 0 bytes in 0 blocks
==2374==    still reachable: 990 bytes in 4 blocks
==2374==         suppressed: 0 bytes in 0 blocks
==2374==
==2374== For counts of detected and suppressed errors, rerun with: -v
==2374== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

The Servants

A Servant will first read from the fifo it was assigned before it was executed and load its data set in to an AVL tree to access it later.

This database is not shared across processes. If the servant was not notified to not disturb server it will create its socket, connect it to the port the server is listening to, then will send the following information to the server;I am Servant, Servant PORT, Starting City Name, Ending City Name.

Servant Port: In the assignment there was a warning about choosing a unique port for each servant. This was achieved by the following formula;

$$Servant\ Port = Server\ Port + Starting\ City\ Number + 100$$

If the each servant has the following starting city numbers (1, 11,21,31 ….) and the Server Port is 40000, their ports will be; 40101, 40111,40121,40131 etc.

Starting City Name: In the input parameters this information was given as an integer number. But Servants will convert this information to the corresponding City Name (Which was found while loading the dataset). Reasoning behind this is the fact that The Server does not have access to the database and city numbers have no meaning for it.

Ending City Name: Same as Starting City Name.

After these informations are sent to The Server, each Servant will close the connection and the socket.

At this point Servants are ready to accept questions. To do this they will create a socket and bind it to the port they will be listening to which will be their unique ports (which was sent to The Server) . Only The Server will connect to servants.

When a Servant successfully accepts an incoming connection it will create a new thread and hand it this fd (which is the return value of accept). Then go back to waiting for a new connection. This connection handling thread will read the contents of the message and find the suitable answer then reply back to The Server through the same fd, then terminate. This way if a servant receives multiple questions at the same time they will not block the main thread.

There are three types of questions;

1) A question which asks for amount of transactions for given realEstate for all cities or a single city between two dates.
2) A question which asks for the amount of transactions with a surface area between given two values for a realEstate type for all cities or a single city
3) A question which asks for the topTransaction with the highest price value for a given realEstate type for all cities or a single city.
4) A question which asks for the transaction with a certain transactionID

For the first two types;

The servant will check if a city name is mentioned in the question to decide which type was asked.

The First Type; If this type of question is asked a Servant will search through its AVL data base. It will go one by one on each node to compare if that node matches the requirements. After that if there are multiple nodes match it will add up all before sending the answer to The Server. If all cities are searched the servant will not check if requested City name matches it's city names since there is no requested City name.

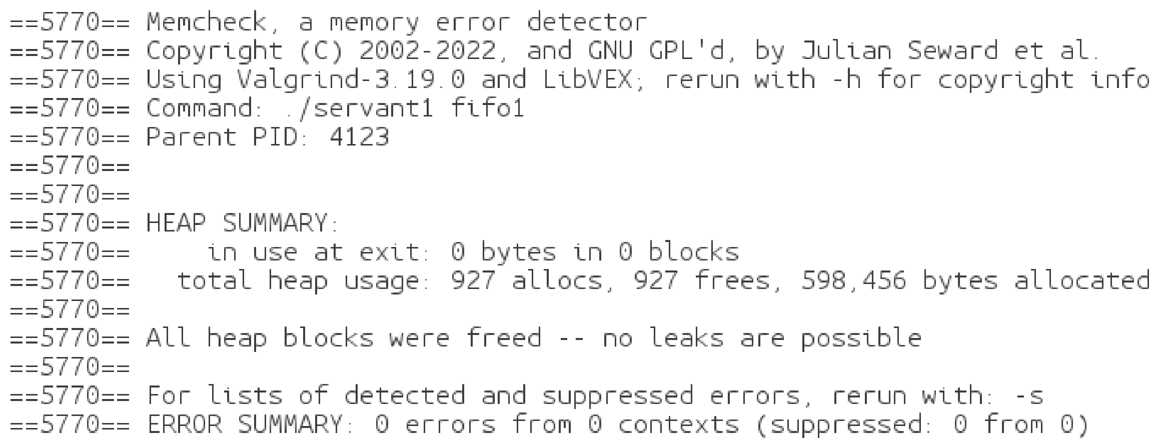
The Second Type; Same idea with the first one, but the servant will now check for surface areas of the real estate instead of date.

The Third Type; Again the servant will go through its database to keep track of current highest transaction price for a given realEstate in mentioned city or all cities. If there are multiple answers they will be appened back to back.

The Fourth Type; This type directly utilizes the AVL structure since AVL nodes have the same IDs as transaction IDS. When the result is found it will put the parameters together to send to server.

After finding the answer to the question the servant thread which was dealing with the question will send The Server the result. This reply is sent through the fd it read the question from.

At any moment if SIGINT is received a servant will close all of its connections and sockets then terminate.



```
==5770== Memcheck, a memory error detector
==5770== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==5770== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==5770== Command: ./servant1 fifo1
==5770== Parent PID: 4123
==5770==
==5770==
==5770== HEAP SUMMARY:
==5770==     in use at exit: 0 bytes in 0 blocks
==5770==   total heap usage: 927 allocs, 927 frees, 598,456 bytes allocated
==5770==
==5770== All heap blocks were freed -- no leaks are possible
==5770==
==5770== For lists of detected and suppressed errors, rerun with: -s
==5770== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

This Valgrind is from a servant program which was run as a separate program. Running distributor with -trace children parameter causes an unwanted behaviour which shifts the arguments of execv.

The Server

The Server will start by creating N amount of thread which is given as an input parameter. These threads will be remain dormant in the threadbarrier#1.

After the threads are created The Server will create and bind it’s two sockets to the Server Port1 and Port2 (which was an input parameter) to listen and accept connections. There is a select() block which detects which socket is the accepted connection from. This will be used to determine the identity of the connection, it will be usefull later.

The server expects all servants to connect to it before the client start sending requests. This is a requirement since the assignment wants “s” amount of threads to handle servant connections, where s is the amount of servant processes. When a client connects the server will finally find out how many servants there are. The rest of the threads will handle connections from clients.

There are two FIFO structures to store incoming connections. Connections from client are stored in to FIFO#1 and connections from servants are stored in FIFO#2. These FIFOs will be also be used to communicate between servant handling threads and client handling threads, this will be explained soon.

These FIFO structures are in a mutex lock, and the thread who are ready to work wait on a lock and a signal above them, when a thread catches the signal it will lock it collect from the FIFO.

If a servant handling thread reads from FIFO#2 and it only includes a FileDescriptor it means it came from a servant who is trying to introduce it self to the server. When this happens the thread will collect the the cities that servant is responsible, the port that servant is listening to, and save the information. When a question arrives servant handling threads will know who to contact this way.

If the message from FIFO#2 contains a FileDescriptor and a Request appended to it, it means it came from a client handling thread who accepted a connection and read the request and wrote it in FIFO#2. When this happens the servant will send the request to the servant who is responsible for it and hold on the FD part for now, after receiving the answer from the servant it will append the reply to the FD which it was holding and put it in FIFO#1, this will be read by a client handling thread soon. Then it goes back to FIFO#2 to collect a new job.

The servant handling thread will act according to the Request it read because answers to them will be in a different format for each of them.

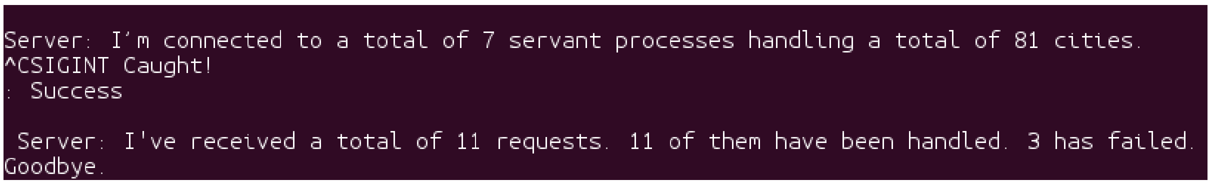
For transactionCount and surfaceCount it will be an integer, and it will be added together for the final count.

For topTransaction it will be one or multiple transactions, and they could be appened if they are equal in price value.

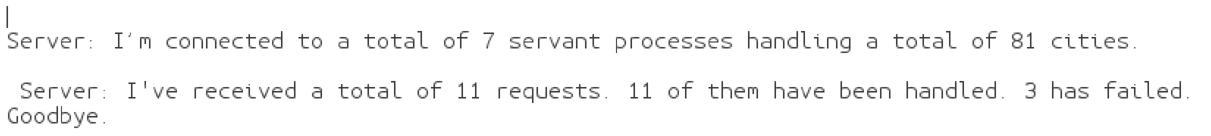
For transactionId it will be a single transaction.

For an invalid question The Server will reply to The Client accordingly.

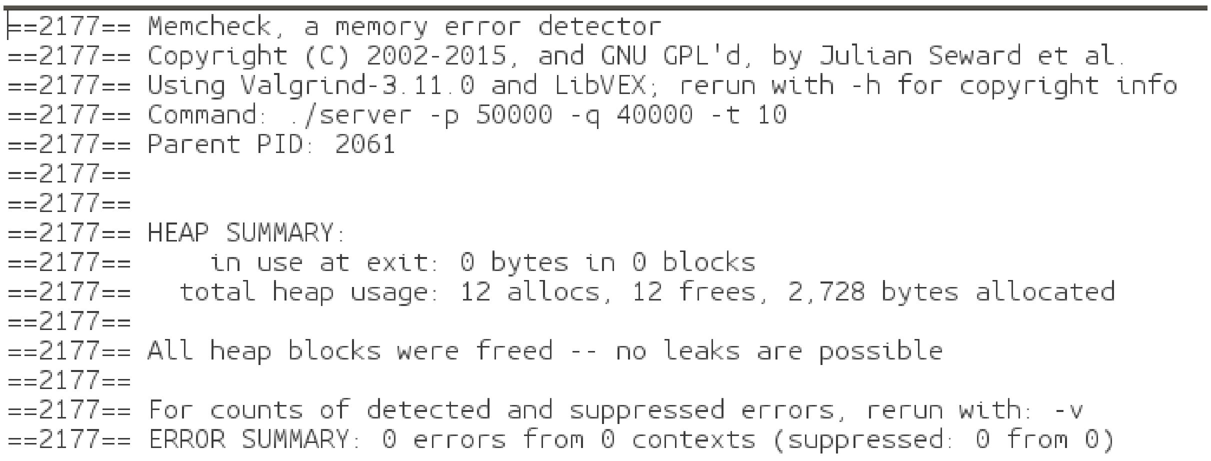
If a client handling thread reads from the FIFO#1 and it only includes a FileDescriptor, it means it came from a client and it contains a request. This thread will collect the Request, Append it to the FD it read from, and put that message to the FIFO#2. If the message from FIFO#1 includes a response next to the FD, it will send that response to that FD. Which will be the client who asked it.



```
Server: I'm connected to a total of 7 servant processes handling a total of 81 cities.
^CSIGINT Caught!
: Success

 Server: I've received a total of 11 requests. 11 of them have been handled. 3 has failed.
Goodbye.
```

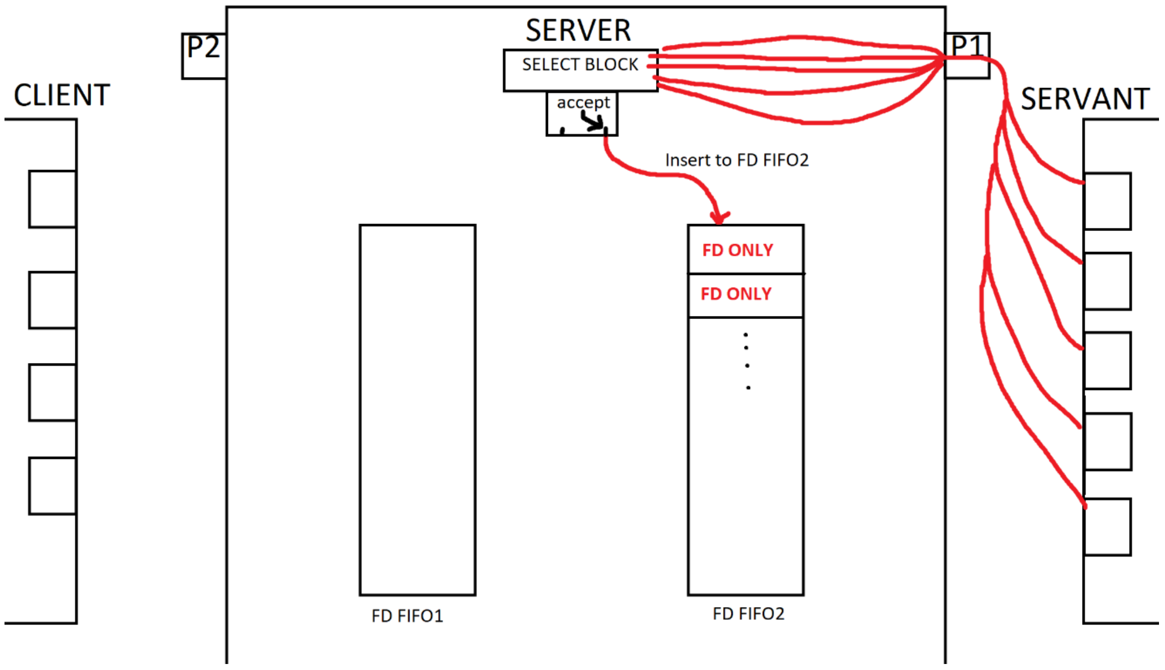


```
|
Server: I'm connected to a total of 7 servant processes handling a total of 81 cities.

 Server: I've received a total of 11 requests. 11 of them have been handled. 3 has failed.
Goodbye.
```

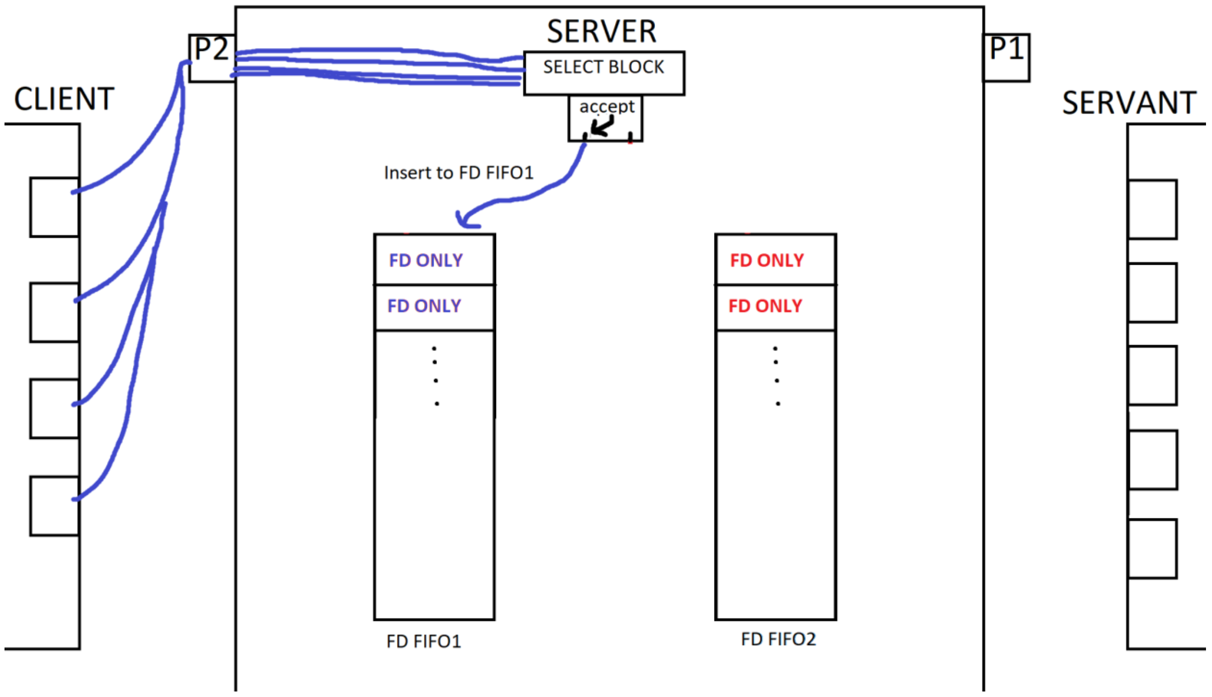


```
==2177== Memcheck, a memory error detector
==2177== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==2177== Using Valgrind-3.11.0 and LibVEX; rerun with -h for copyright info
==2177== Command: ./server -p 50000 -q 40000 -t 10
==2177== Parent PID: 2061
==2177==
==2177==
==2177== HEAP SUMMARY:
==2177==     in use at exit: 0 bytes in 0 blocks
==2177==   total heap usage: 12 allocs, 12 frees, 2,728 bytes allocated
==2177==
==2177== All heap blocks were freed -- no leaks are possible
==2177==
==2177== For counts of detected and suppressed errors, rerun with: -v
==2177== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

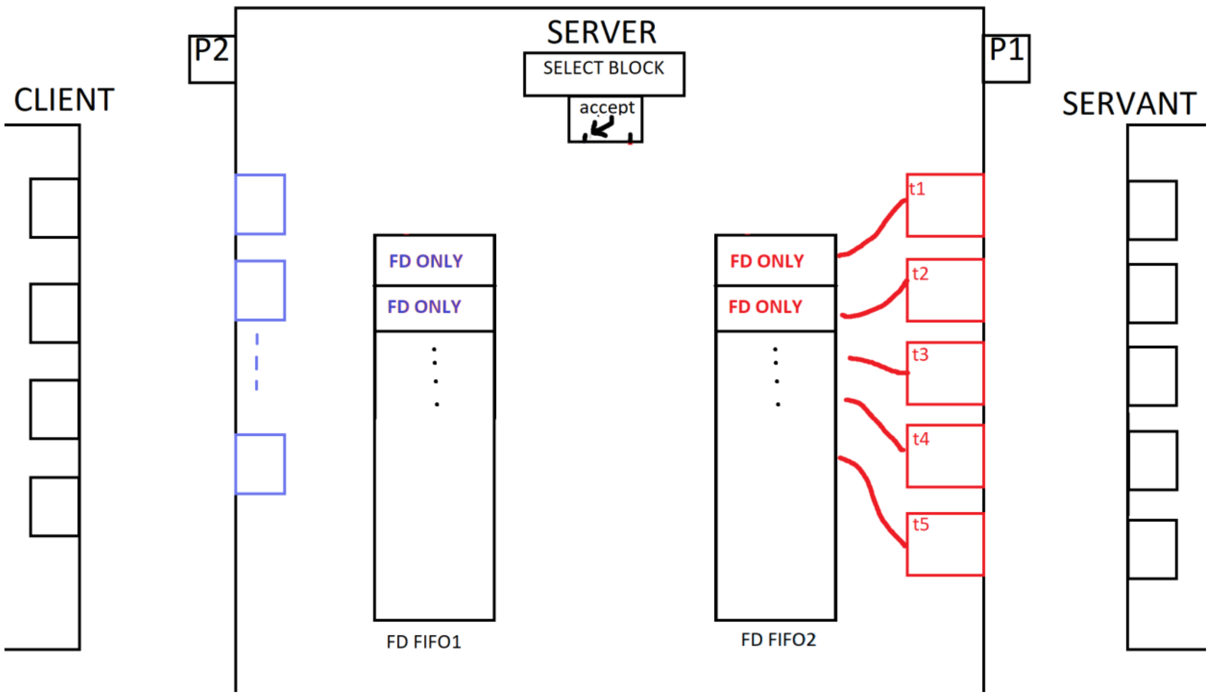
How Server Threads Communicate With Each Other



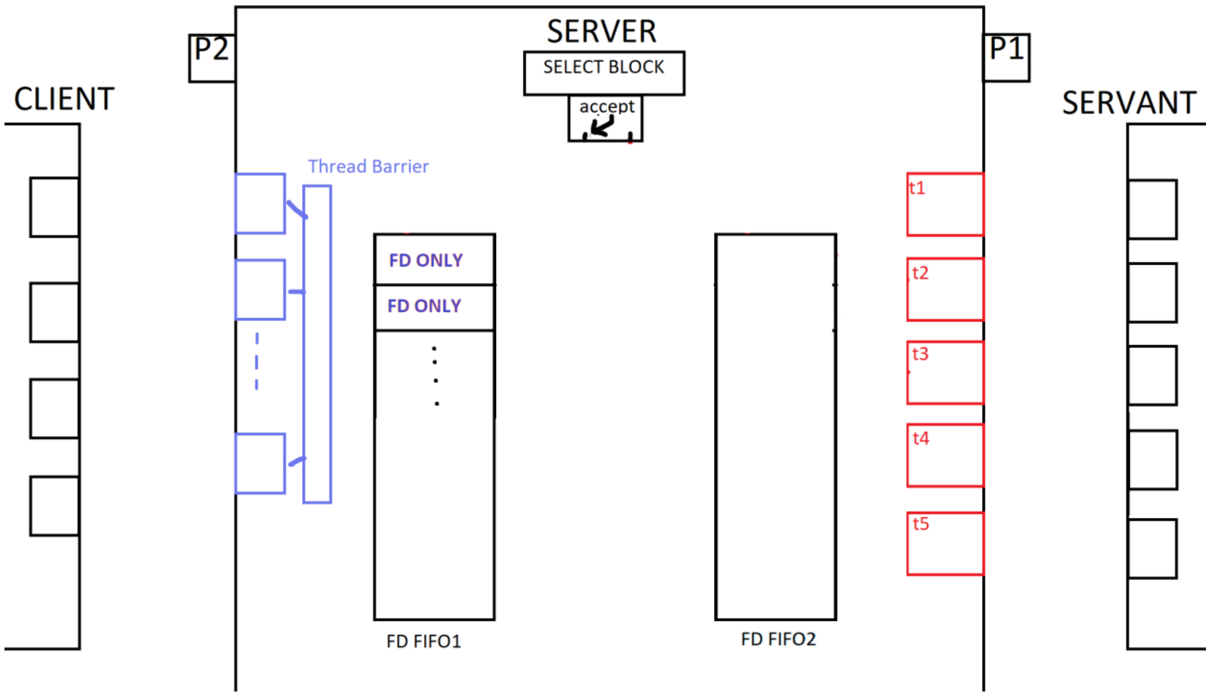
The Servants Introduce themselves.



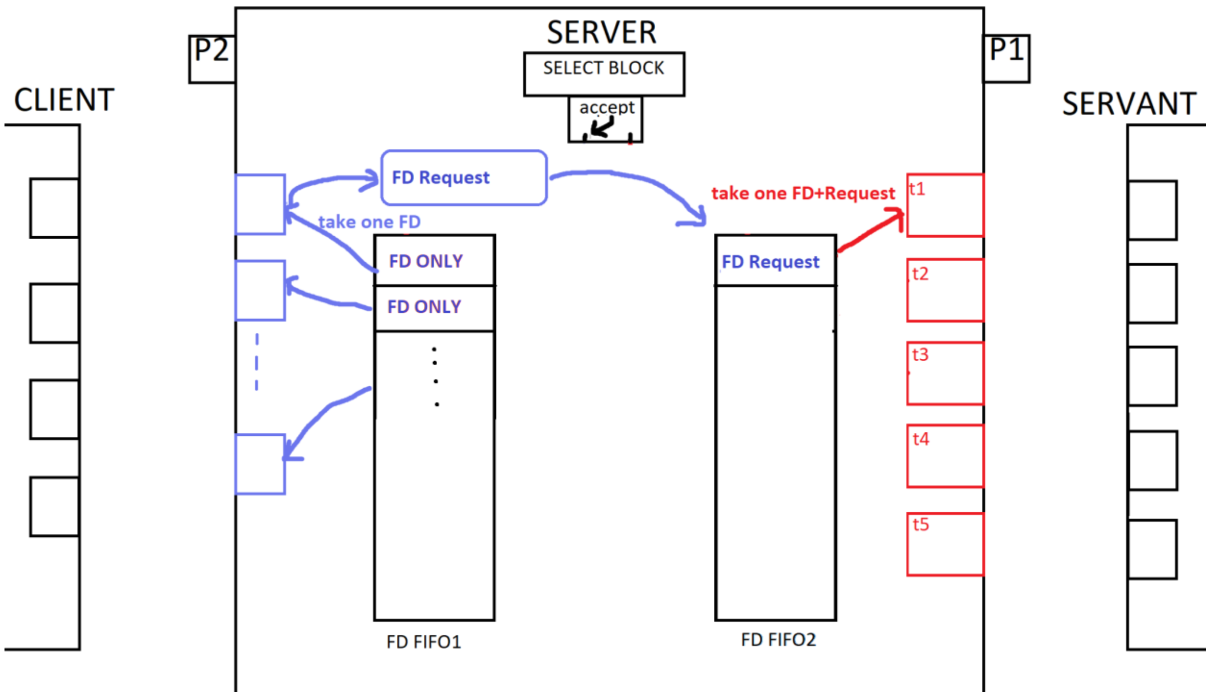
The clients start to connect.



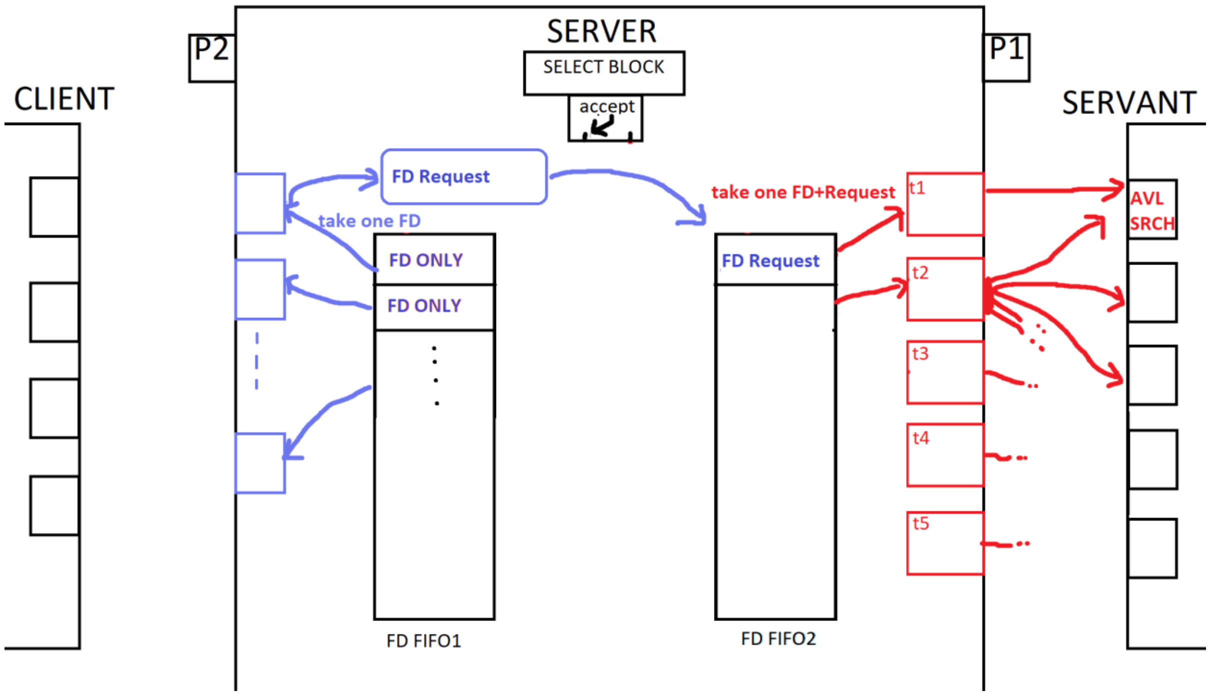
The servants handling threads form and save the informations about them. Then they will lift the barrier.



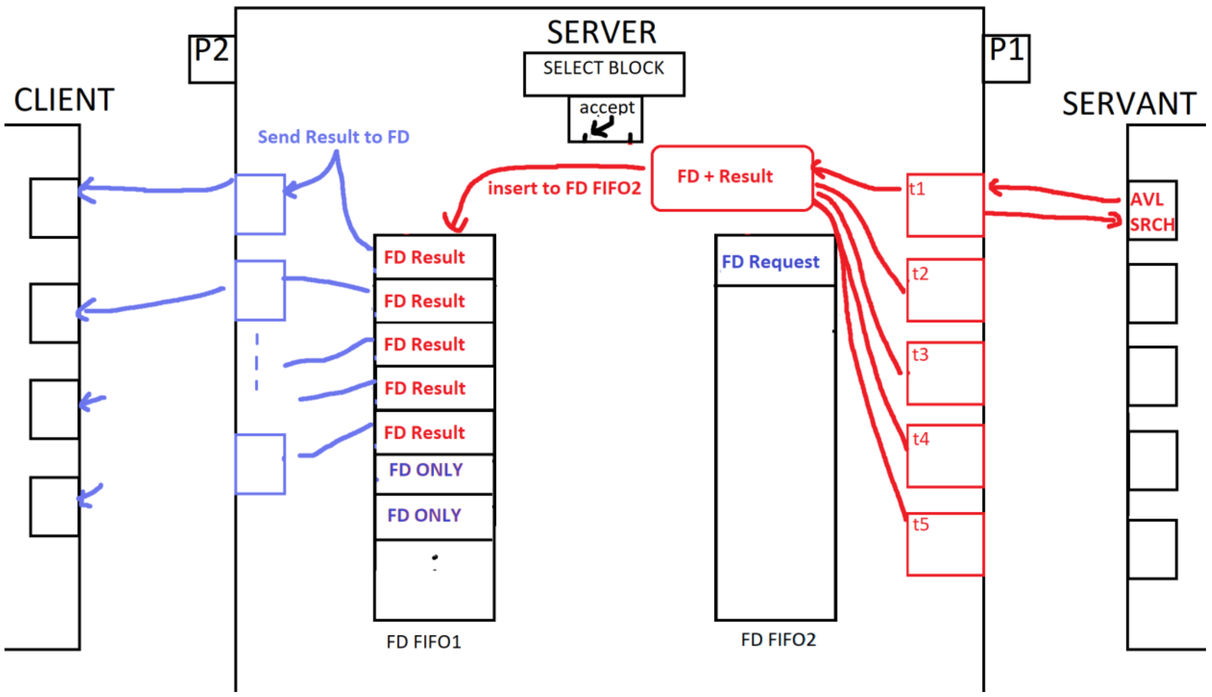
After saving informations servant handling threads lift the barrier of client handling threads.



Now the client handling threads start to read from FIFO1



And servant handling threads from FIFO2



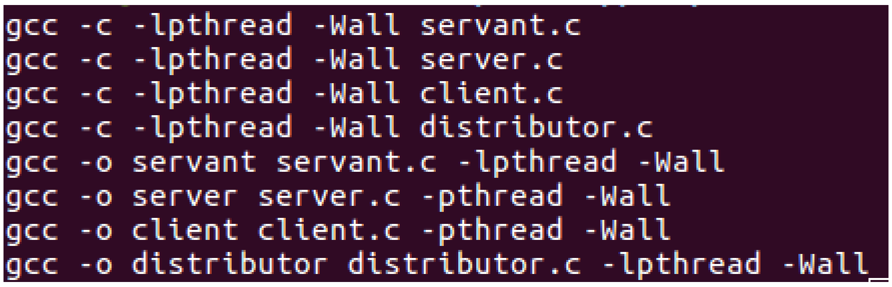
This loops continues until user ends the server.

RESULTS

As it can be seen in the next three parts, both Ubuntu 16.04 and 22.04 versions produce the same outputs. 20.04 uses a different a new randomised dataset.

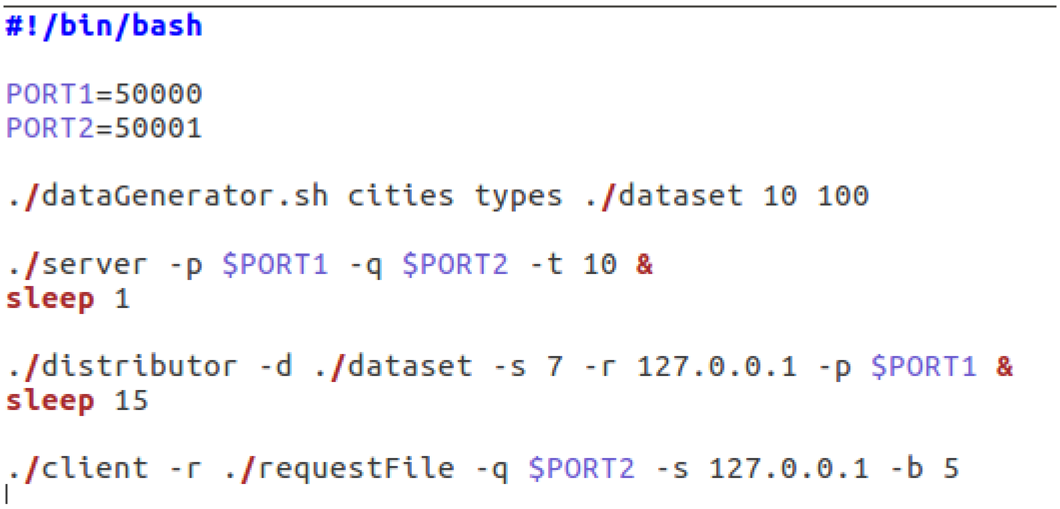
RUNNING THE PROGRAM on Ubuntu 16.04

MakeFile with no -Wall warnings returning. (–Valgrind outputs are in the sections above)



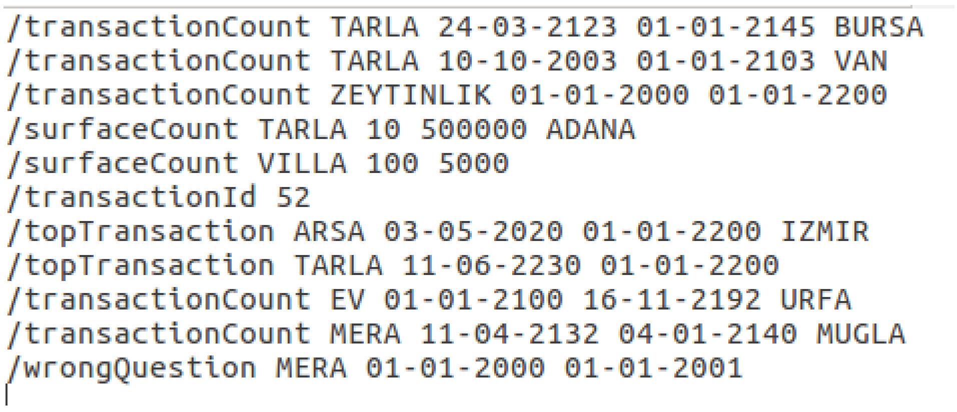
```
gcc -c -lpthread -Wall servant.c
gcc -c -lpthread -Wall server.c
gcc -c -lpthread -Wall client.c
gcc -c -lpthread -Wall distributor.c
gcc -o servant servant.c -lpthread -Wall
gcc -o server server.c -pthread -Wall
gcc -o client client.c -pthread -Wall
gcc -o distributor distributor.c -lpthread -Wall_
```

Running ./script.sh on Ubuntu v16.04



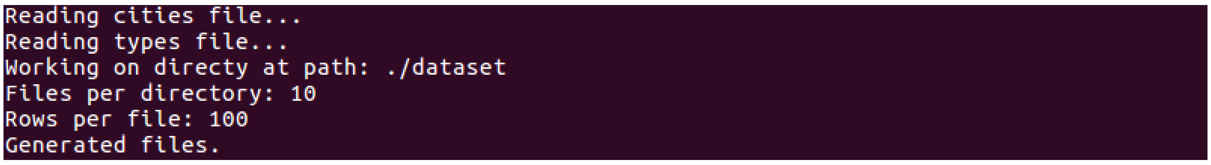
```
#!/bin/bash

PORT1=50000
PORT2=50001

./dataGenerator.sh cities types ./dataset 10 100

./server -p $PORT1 -q $PORT2 -t 10 &
sleep 1

./distributor -d ./dataset -s 7 -r 127.0.0.1 -p $PORT1 &
sleep 15

./client -r ./requestFile -q $PORT2 -s 127.0.0.1 -b 5
```

requestFile



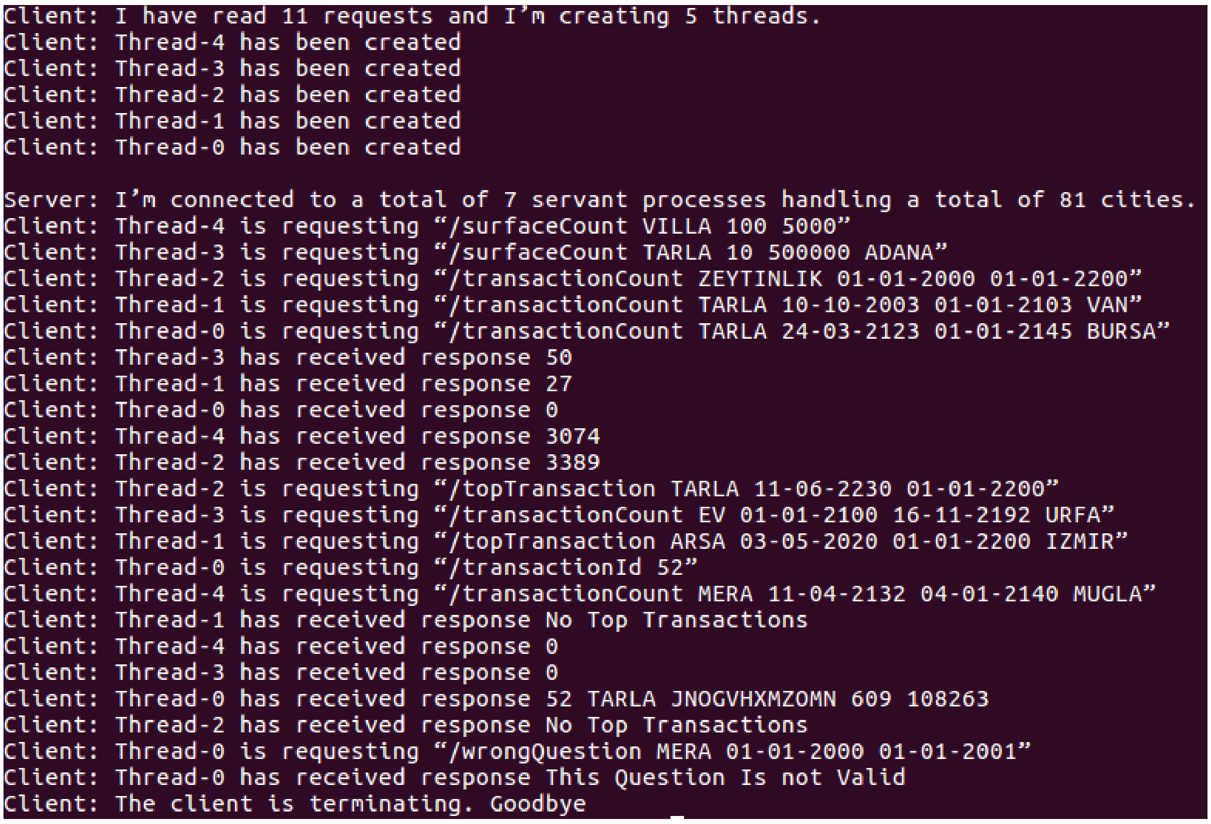
```
/transactionCount TARLA 24-03-2123 01-01-2145 BURSA
/transactionCount TARLA 10-10-2003 01-01-2103 VAN
/transactionCount ZEYTINLIK 01-01-2000 01-01-2200
/surfaceCount TARLA 10 500000 ADANA
/surfaceCount VILLA 100 5000
/transactionId 52
/topTransaction ARSA 03-05-2020 01-01-2200 IZMIR
/topTransaction TARLA 11-06-2230 01-01-2200
/transactionCount EV 01-01-2100 16-11-2192 URFA
/transactionCount MERA 11-04-2132 04-01-2140 MUGLA
/wrongQuestion MERA 01-01-2000 01-01-2001
```

Output of ./script.sh



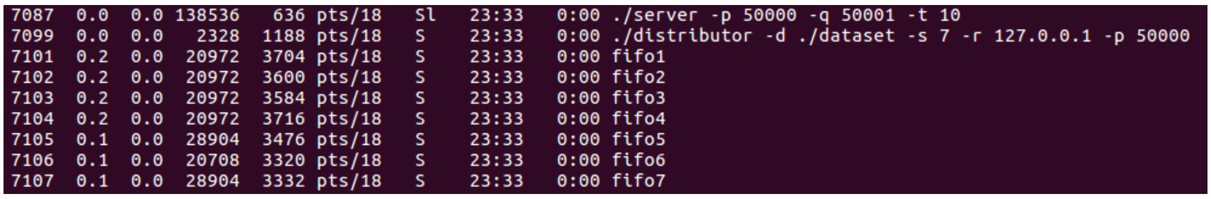
```
Reading cities file...
Reading types file...
Working on directy at path: ./dataset
Files per directory: 10
Rows per file: 100
Generated files.
```

After The Generator is Done Server->Distributor-> Client starts in other with sleep() in between.



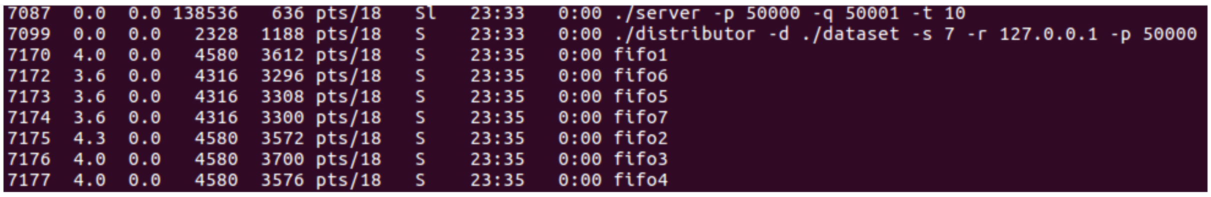
```
Client: I have read 11 requests and I'm creating 5 threads.
Client: Thread-4 has been created
Client: Thread-3 has been created
Client: Thread-2 has been created
Client: Thread-1 has been created
Client: Thread-0 has been created

Server: I'm connected to a total of 7 servant processes handling a total of 81 cities.
Client: Thread-4 is requesting "/surfaceCount VILLA 100 5000"
Client: Thread-3 is requesting "/surfaceCount TARLA 10 500000 ADANA"
Client: Thread-2 is requesting "/transactionCount ZEYTINLIK 01-01-2000 01-01-2200"
Client: Thread-1 is requesting "/transactionCount TARLA 10-10-2003 01-01-2103 VAN"
Client: Thread-0 is requesting "/transactionCount TARLA 24-03-2123 01-01-2145 BURSA"
Client: Thread-3 has received response 50
Client: Thread-1 has received response 27
Client: Thread-0 has received response 0
Client: Thread-4 has received response 3074
Client: Thread-2 has received response 3389
Client: Thread-2 is requesting "/topTransaction TARLA 11-06-2230 01-01-2200"
Client: Thread-3 is requesting "/transactionCount EV 01-01-2100 16-11-2192 URFA"
Client: Thread-1 is requesting "/topTransaction ARSA 03-05-2020 01-01-2200 IZMIR"
Client: Thread-0 is requesting "/transactionId 52"
Client: Thread-4 is requesting "/transactionCount MERA 11-04-2132 04-01-2140 MUGLA"
Client: Thread-1 has received response No Top Transactions
Client: Thread-4 has received response 0
Client: Thread-3 has received response 0
Client: Thread-0 has received response 52 TARLA JNOGVHXMZOMN 609 108263
Client: Thread-2 has received response No Top Transactions
Client: Thread-0 is requesting "/wrongQuestion MERA 01-01-2000 01-01-2001"
Client: Thread-0 has received response This Question Is not Valid
Client: The client is terminating. Goodbye
```

Processes



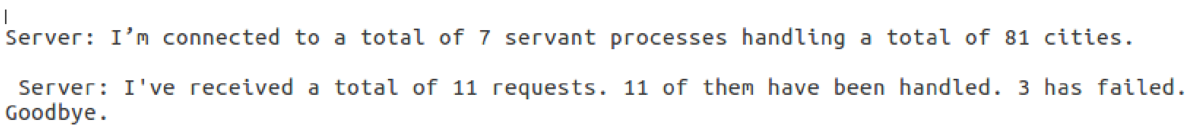
```
 7087  0.0  0.0 138536   636 pts/18   Sl   23:33   0:00 ./server -p 50000 -q 50001 -t 10
 7099  0.0  0.0   2328  1188 pts/18   S    23:33   0:00 ./distributor -d ./dataset -s 7 -r 127.0.0.1 -p 50000
 7101  0.2  0.0  20972  3704 pts/18   S    23:33   0:00 fifo1
 7102  0.2  0.0  20972  3600 pts/18   S    23:33   0:00 fifo2
 7103  0.2  0.0  20972  3584 pts/18   S    23:33   0:00 fifo3
 7104  0.2  0.0  20972  3716 pts/18   S    23:33   0:00 fifo4
 7105  0.1  0.0  28904  3476 pts/18   S    23:33   0:00 fifo5
 7106  0.1  0.0  20708  3320 pts/18   S    23:33   0:00 fifo6
 7107  0.1  0.0  28904  3332 pts/18   S    23:33   0:00 fifo7
```

Processes After Distributor Revives Servants which were killed with "killall servant" command



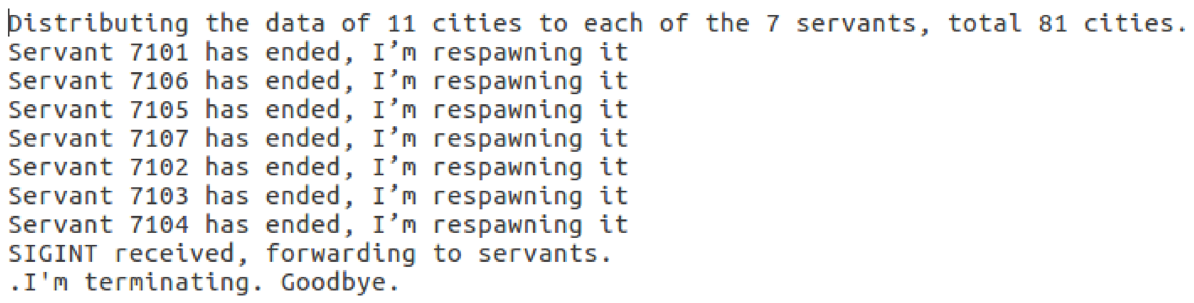
```
7087  0.0  0.0 138536   636 pts/18   Sl   23:33   0:00 ./server -p 50000 -q 50001 -t 10
7099  0.0  0.0   2328  1188 pts/18   S    23:33   0:00 ./distributor -d ./dataset -s 7 -r 127.0.0.1 -p 50000
7170  4.0  0.0   4580  3612 pts/18   S    23:35   0:00 fifo1
7172  3.6  0.0   4316  3296 pts/18   S    23:35   0:00 fifo6
7173  3.6  0.0   4316  3308 pts/18   S    23:35   0:00 fifo5
7174  3.6  0.0   4316  3300 pts/18   S    23:35   0:00 fifo7
7175  4.3  0.0   4580  3572 pts/18   S    23:35   0:00 fifo2
7176  4.0  0.0   4580  3700 pts/18   S    23:35   0:00 fifo3
7177  4.0  0.0   4580  3576 pts/18   S    23:35   0:00 fifo4
```

Output Of Server Log After Sending SIGINT to Server with "kill -2 7087" (pid of server)



```
Server: I'm connected to a total of 7 servant processes handling a total of 81 cities.

 Server: I've received a total of 11 requests. 11 of them have been handled. 3 has failed.
Goodbye.
```

Output Of Distributor Log After sending SIGINT to Distributor with “kill -2 7099” (pid of Distributor)

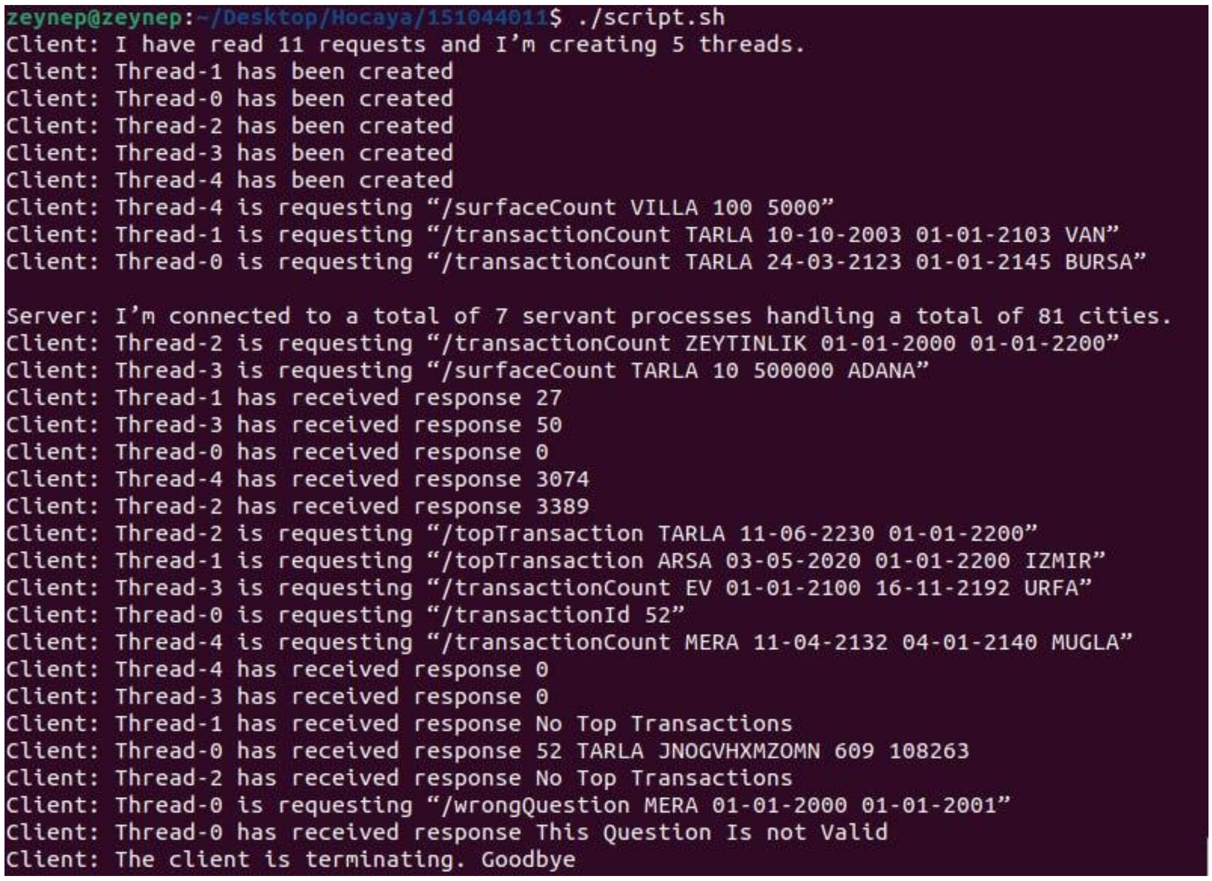


```
Distributing the data of 11 cities to each of the 7 servants, total 81 cities.
Servant 7101 has ended, I’m respawning it
Servant 7106 has ended, I’m respawning it
Servant 7105 has ended, I’m respawning it
Servant 7107 has ended, I’m respawning it
Servant 7102 has ended, I’m respawning it
Servant 7103 has ended, I’m respawning it
Servant 7104 has ended, I’m respawning it
SIGINT received, forwarding to servants.
.I'm terminating. Goodbye.
```

No processes are left at this point.

RUNNING THE PROGRAM on Ubuntu 22.04

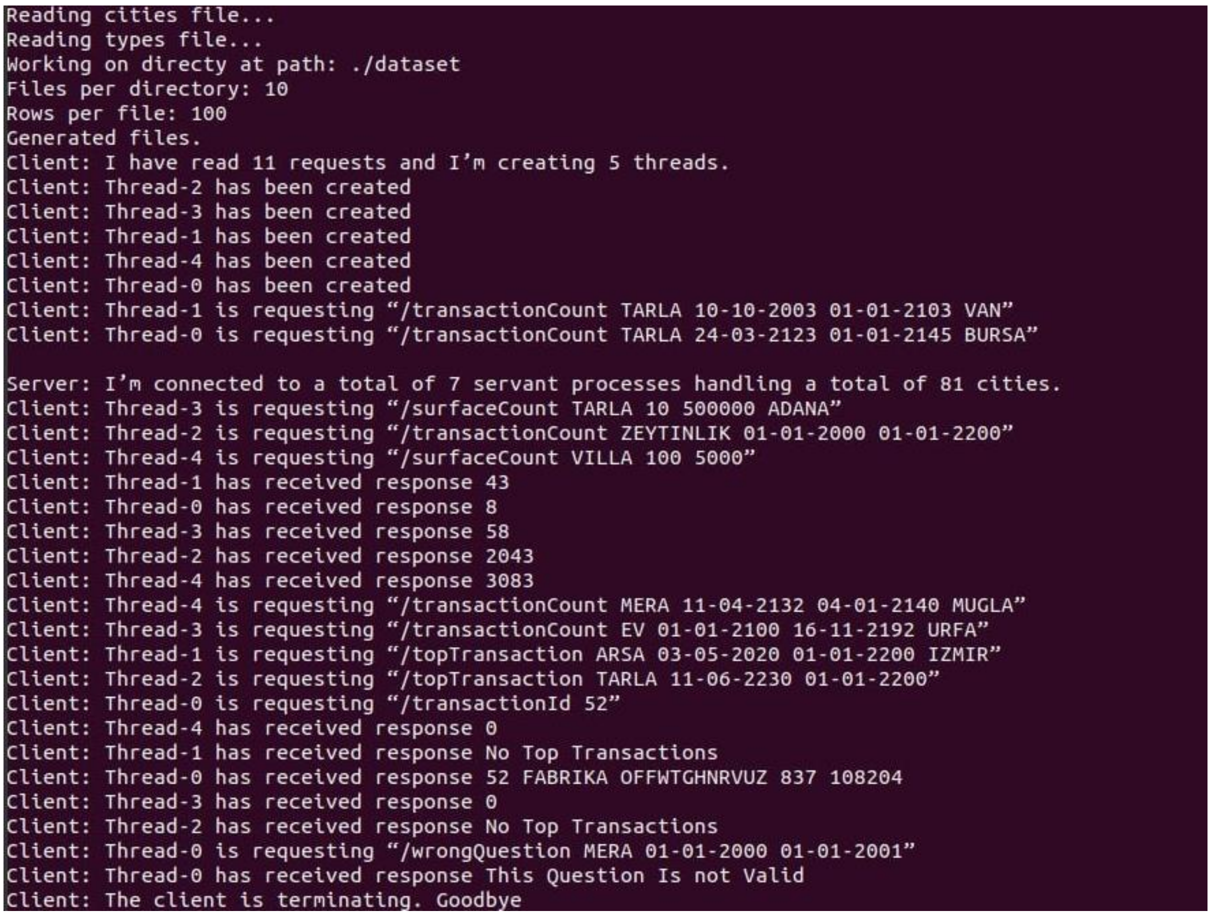
Output of ./script.sh (using same data set as Ubuntu 16.04, skipping datagenerator)



```
zeynep@zeynep:~/Desktop/Hocaya/151044011$ ./script.sh
Client: I have read 11 requests and I’m creating 5 threads.
Client: Thread-1 has been created
Client: Thread-0 has been created
Client: Thread-2 has been created
Client: Thread-3 has been created
Client: Thread-4 has been created
Client: Thread-4 is requesting “/surfaceCount VILLA 100 5000”
Client: Thread-1 is requesting “/transactionCount TARLA 10-10-2003 01-01-2103 VAN”
Client: Thread-0 is requesting “/transactionCount TARLA 24-03-2123 01-01-2145 BURSA”

Server: I’m connected to a total of 7 servant processes handling a total of 81 cities.
Client: Thread-2 is requesting “/transactionCount ZEYTINLIK 01-01-2000 01-01-2200”
Client: Thread-3 is requesting “/surfaceCount TARLA 10 500000 ADANA”
Client: Thread-1 has received response 27
Client: Thread-3 has received response 50
Client: Thread-0 has received response 0
Client: Thread-4 has received response 3074
Client: Thread-2 has received response 3389
Client: Thread-2 is requesting “/topTransaction TARLA 11-06-2230 01-01-2200”
Client: Thread-1 is requesting “/topTransaction ARSA 03-05-2020 01-01-2200 IZMIR”
Client: Thread-3 is requesting “/transactionCount EV 01-01-2100 16-11-2192 URFA”
Client: Thread-0 is requesting “/transactionId 52”
Client: Thread-4 is requesting “/transactionCount MERA 11-04-2132 04-01-2140 MUGLA”
Client: Thread-4 has received response 0
Client: Thread-3 has received response 0
Client: Thread-1 has received response No Top Transactions
Client: Thread-0 has received response 52 TARLA JNOGVHXMZOMN 609 108263
Client: Thread-2 has received response No Top Transactions
Client: Thread-0 is requesting “/wrongQuestion MERA 01-01-2000 01-01-2001”
Client: Thread-0 has received response This Question Is not Valid
Client: The client is terminating. Goodbye
```

RUNNING THE PROGRAM on Ubuntu 20.04

Output of ./script.sh (using a new data set)



```
Reading cities file...
Reading types file...
Working on directy at path: ./dataset
Files per directory: 10
Rows per file: 100
Generated files.
Client: I have read 11 requests and I'm creating 5 threads.
Client: Thread-2 has been created
Client: Thread-3 has been created
Client: Thread-1 has been created
Client: Thread-4 has been created
Client: Thread-0 has been created
Client: Thread-1 is requesting "/transactionCount TARLA 10-10-2003 01-01-2103 VAN"
Client: Thread-0 is requesting "/transactionCount TARLA 24-03-2123 01-01-2145 BURSA"

Server: I'm connected to a total of 7 servant processes handling a total of 81 cities.
Client: Thread-3 is requesting "/surfaceCount TARLA 10 500000 ADANA"
Client: Thread-2 is requesting "/transactionCount ZEYTINLIK 01-01-2000 01-01-2200"
Client: Thread-4 is requesting "/surfaceCount VILLA 100 5000"
Client: Thread-1 has received response 43
Client: Thread-0 has received response 8
Client: Thread-3 has received response 58
Client: Thread-2 has received response 2043
Client: Thread-4 has received response 3083
Client: Thread-4 is requesting "/transactionCount MERA 11-04-2132 04-01-2140 MUGLA"
Client: Thread-3 is requesting "/transactionCount EV 01-01-2100 16-11-2192 URFA"
Client: Thread-1 is requesting "/topTransaction ARSA 03-05-2020 01-01-2200 IZMIR"
Client: Thread-2 is requesting "/topTransaction TARLA 11-06-2230 01-01-2200"
Client: Thread-0 is requesting "/transactionId 52"
Client: Thread-4 has received response 0
Client: Thread-1 has received response No Top Transactions
Client: Thread-0 has received response 52 FABRIKA OFFWTGHNRVUZ 837 108204
Client: Thread-3 has received response 0
Client: Thread-2 has received response No Top Transactions
Client: Thread-0 is requesting "/wrongQuestion MERA 01-01-2000 01-01-2001"
Client: Thread-0 has received response This Question Is not Valid
Client: The client is terminating. Goodbye
```