CSCE 313 Programming Assignment 2

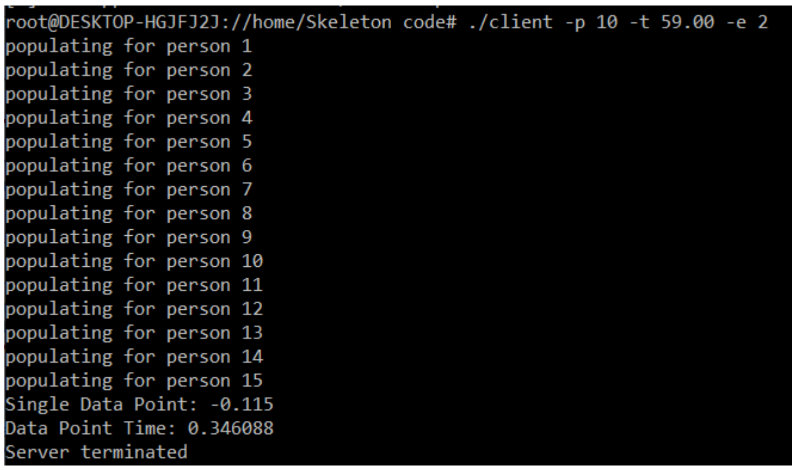
A Client Process Speaking to a Server Process

Aldo Leon Marquez UIN: 326004699

The Objective of the Assignment is to implement a client to a given server, i.e. a connection point to a server with stored information, in this case our program has the page address loaded for 15 patients ecg data points, Our goal is to communicate with this server, extract data in different supported ways and also implement the complete program as a Child-Parent

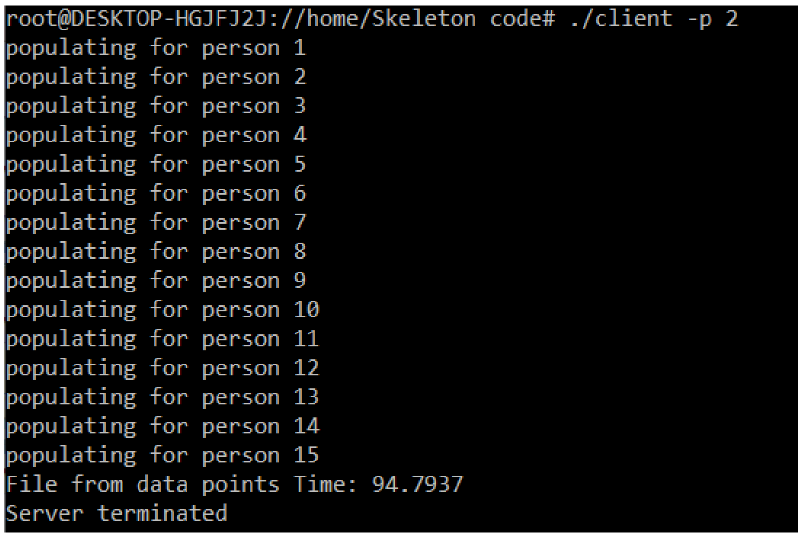
To do the different type of supported server calls we look at the common file and see that the server can: Request a data point, request a file, Create a new channel, and a quit message.

For the first part, we need to correctly format a message to the server with the wanted information from it, the Patient number the time and which ecg value, the individual search might not look too slow but when doing multiple, the time cost, grows quickly int the next two images we can see just that that:



```
root@DESKTOP-HGJFJ2J://home/Skeleton code# ./client -p 10 -t 59.00 -e 2
populating for person 1
populating for person 2
populating for person 3
populating for person 4
populating for person 5
populating for person 6
populating for person 7
populating for person 8
populating for person 9
populating for person 10
populating for person 11
populating for person 12
populating for person 13
populating for person 14
populating for person 15
Single Data Point: -0.115
Data Point Time: 0.346088
Server terminated
```

Single point



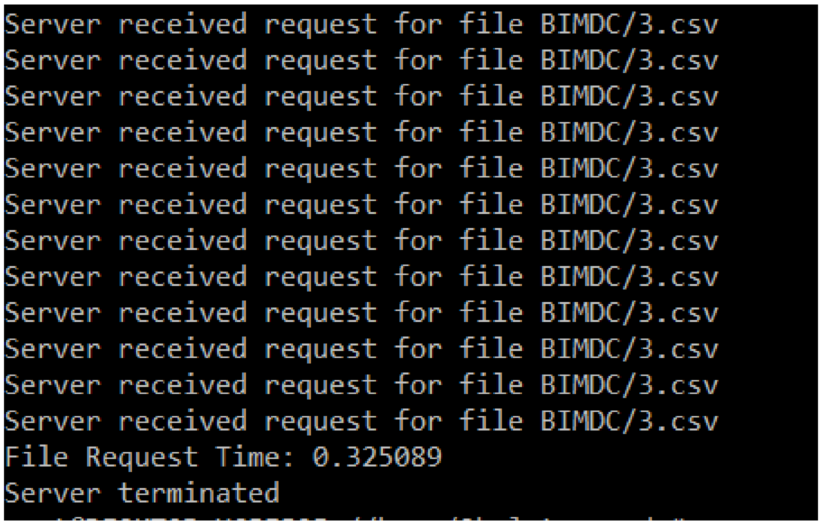
```
root@DESKTOP-HGJFJ2J://home/Skeleton code# ./client -p 2
populating for person 1
populating for person 2
populating for person 3
populating for person 4
populating for person 5
populating for person 6
populating for person 7
populating for person 8
populating for person 9
populating for person 10
populating for person 11
populating for person 12
populating for person 13
populating for person 14
populating for person 15
File from data points Time: 94.7937
Server terminated
```

Point by point

from data 2.csv

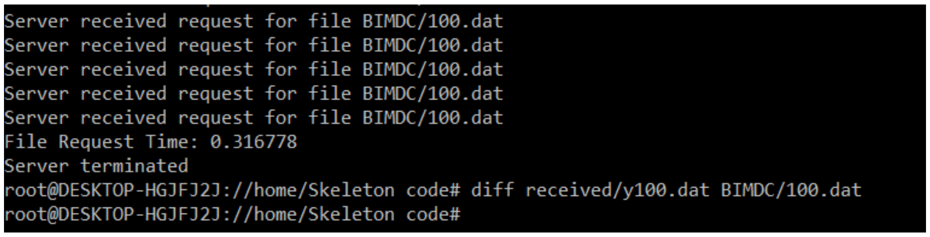
The Server also supports getting the entire file from it, to do so we also pass a data message to the server but this time with some constrains, we limit the unit message size to 256 bytes, to optimize and prevent the buffer from overflows or errors. With the correct format then we repeat as many times as necessary to cover the entire file size with 256 message blocks

As we can tell from the next image the time to get all point with this method is far more efficient

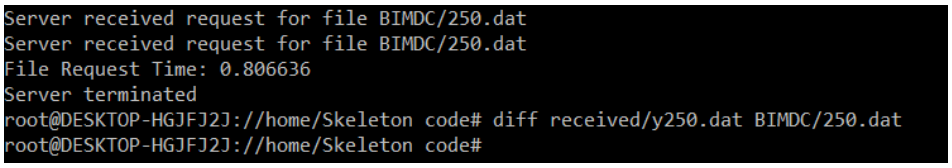


```
Server received request for file BIMDC/3.csv
Server received request for file BIMDC/3.csv
Server received request for file BIMDC/3.csv
Server received request for file BIMDC/3.csv
Server received request for file BIMDC/3.csv
Server received request for file BIMDC/3.csv
Server received request for file BIMDC/3.csv
Server received request for file BIMDC/3.csv
Server received request for file BIMDC/3.csv
Server received request for file BIMDC/3.csv
Server received request for file BIMDC/3.csv
Server received request for file BIMDC/3.csv
File Request Time: 0.325089
Server terminated
```

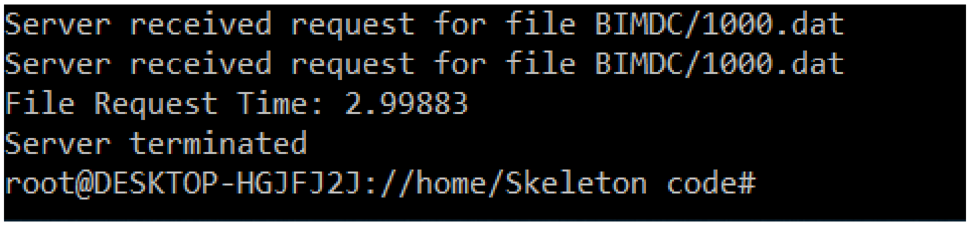
Also Included are the n.data truncated files: 100.dat, 250.dat, 1000.dat and 100MB.dat



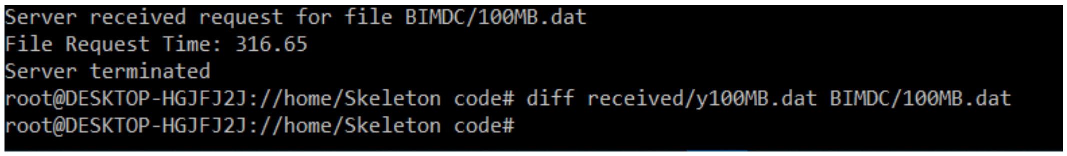
```
Server received request for file BIMDC/100.dat
Server received request for file BIMDC/100.dat
Server received request for file BIMDC/100.dat
Server received request for file BIMDC/100.dat
Server received request for file BIMDC/100.dat
File Request Time: 0.316778
Server terminated
root@DESKTOP-HGJFJ2J://home/Skeleton code# diff received/y100.dat BIMDC/100.dat
root@DESKTOP-HGJFJ2J://home/Skeleton code#
```



```
Server received request for file BIMDC/250.dat
Server received request for file BIMDC/250.dat
File Request Time: 0.806636
Server terminated
root@DESKTOP-HGJFJ2J://home/Skeleton code# diff received/y250.dat BIMDC/250.dat
root@DESKTOP-HGJFJ2J://home/Skeleton code#
```



```
Server received request for file BIMDC/1000.dat
Server received request for file BIMDC/1000.dat
File Request Time: 2.99883
Server terminated
root@DESKTOP-HGJFJ2J://home/Skeleton code#
```

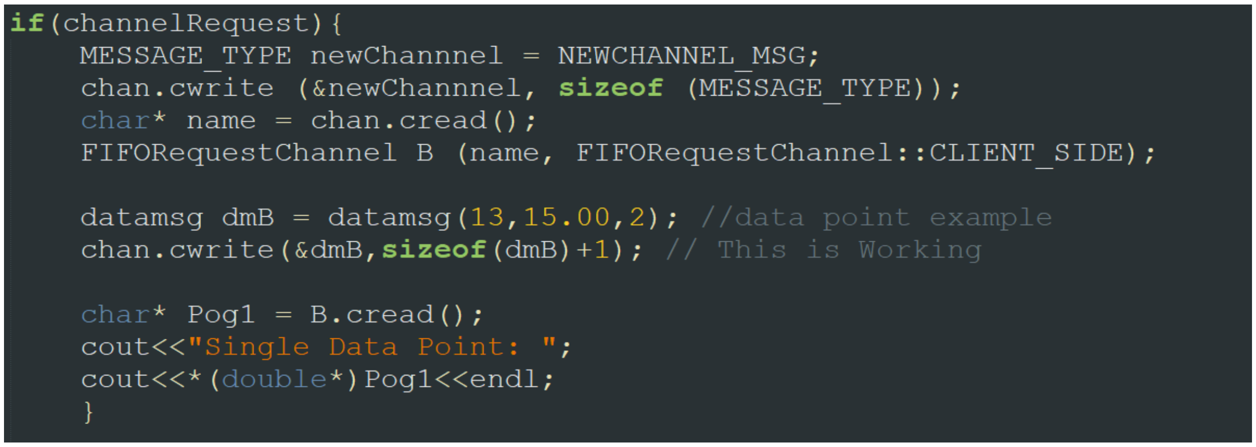


```
Server received request for file BIMDC/100MB.dat
File Request Time: 316.65
Server terminated
root@DESKTOP-HGJFJ2J://home/Skeleton code# diff received/y100MB.dat BIMDC/100MB.dat
root@DESKTOP-HGJFJ2J://home/Skeleton code#
```

Copy of 100MB.dat and diff check

For the .dat files, to ensure that a null character was passed I had to add a check that would output a null char times the msg size to the buffer, before this the program would pass Null as empty not as a character. For this reason, my code ends up being pretty slow for the 100MB File, I could have improved It by somehow creating a message sized data Full of null characters.

The third thing to try was to support a new channel creation, for this we again format a message for the server and get the new channel active, important to close said channel before exiting!



```
if(channelRequest){
    MESSAGE_TYPE newChannnel = NEWCHANNEL_MSG;
    chan.cwrite (&newChannnel, sizeof (MESSAGE_TYPE));
    char* name = chan.cread();
    FIFORequestChannel B (name, FIFORequestChannel::CLIENT_SIDE);

    datamsg dmB = datamsg(13,15.00,2); //data point example
    chan.cwrite(&dmB,sizeof(dmB)+1); // This is Working

    char* Pog1 = B.cread();
    cout<<"Single Data Point: ";
    cout<<*(double*)Pog1<<endl;
    }
```

The Last Part of the Assignment was to implement the whole program in a single executable with parameters, to do so we use the Child-Parent setup, we use the for() function to execute the server from a child process, and finally with getopt() we get the correct request from the terminal. Important to keep sending the Quit Msg to the server otherwise there could be (in my experience) some issues with the directory.

Overall Aside from the workaround for the .dat files, the code executes and performs correctly.