Computer Networks

Programming Assignment 1 - Socket Programming

Student Name :

Yehia Elsayed Mohamed Mohamed ID : **62**

Problem Statement

Berners-Lee and his team are credited for inventing the original HyperText Transfer Protocol (HTTP) along with Hyper Text Markup Language (HTML) and the associated technology for a web server and a text-based web browser. The first version of the protocol had only one method, namely GET, which would request a page from a server. The response from the server was always an HTML page.

What you're about to do is to reinvent the wheel on the motivation of getting a deep understanding of how HTTP works! In this assignment, you will use sockets to implement a simple web client that communicates with a web server using a restricted subset of HTTP. The main objective of this assignment is to give you hands-on experience with UNIX sockets.

System Calls

System Call	Objective Summary
Socket()	creates an endpoint for communication and returns a file descriptor that refers to that endpoint. Takes parameters domain , type and protocol which in our case AF_NET , SOCKET_STREAM and 0 respectively.
bind()	assigning a transport address to the socket (a port number in IP networking).
listen()	Used at the server side , tells a socket that it should be capable of accepting incoming connections
accept()	Used at the server side , grabs the first connection request on the queue of pending connections (set up in listen) and creates a new socket for that connection.
connect()	initiates a connection on a socket. If the parameter s (a socket) is of type SOCK_STREAM, then connect() attempts to make a connection to another socket.

Important DataStructures & Structures

Datastructure / Structure	**objective**
Character arrays	Used in both server and client as the buffer to read & write on the socket.
Vector <string>	Used in parsing the request and split it by white spaces
Struct socket_address	is a generic container that just allows the OS to be able to read the first couple of bytes that identify the address family
struct arg_struct { int client_socket; long long* timer; };	Used in the server side to be attached to each client the struct contains the server_fd and last time that client was active.

Program Organization Summary

- First the server starts with the given port number and initiates its socket then wait for a new connection.
- The client starts with the same port number that is given to the socket to be able to connect to it.
- The client is delegated to its thread and now is ready to send requests to the server (get , post).
- The server takes the request and parses it, extracting the file name and request type.
- Incase of get , If the file not exist will respond by 404 message else it will send 200 OK message followed by the size and the file content.
- Incase of post , it responds by OK message to the client then the client sends the file after receiving it the server saves the file and tells the client it is saved.
- There is another thread to detect the inactive clients for along , if client's elapsed time exceeds the defined threshold the server will close its socket connection.

Major Functions

1. Multi-threaded Web server

a. void * HandleClientConnection(void* P_socketClient)

i. It is the thread function that is attached to each client.

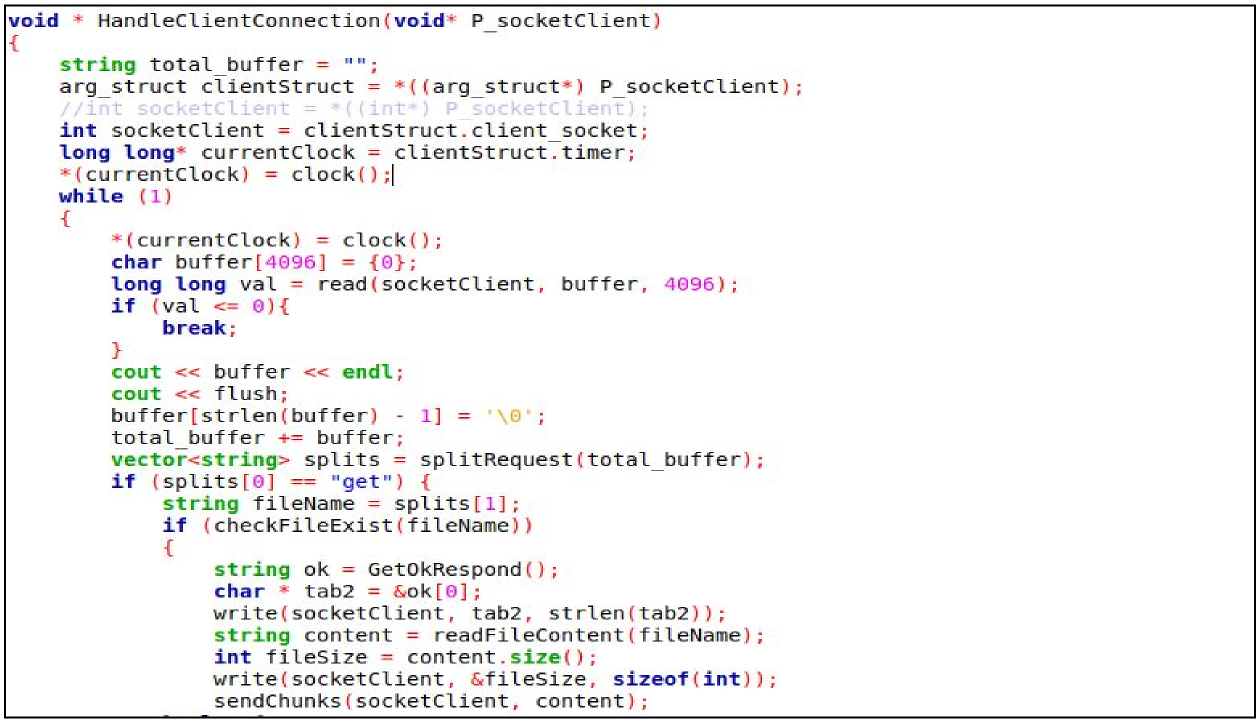
ii. The functions takes the request and passes it to the parser functions to extract file name and request time.

iii. This function handles {get , post} requests by the ways that described above in the last section.

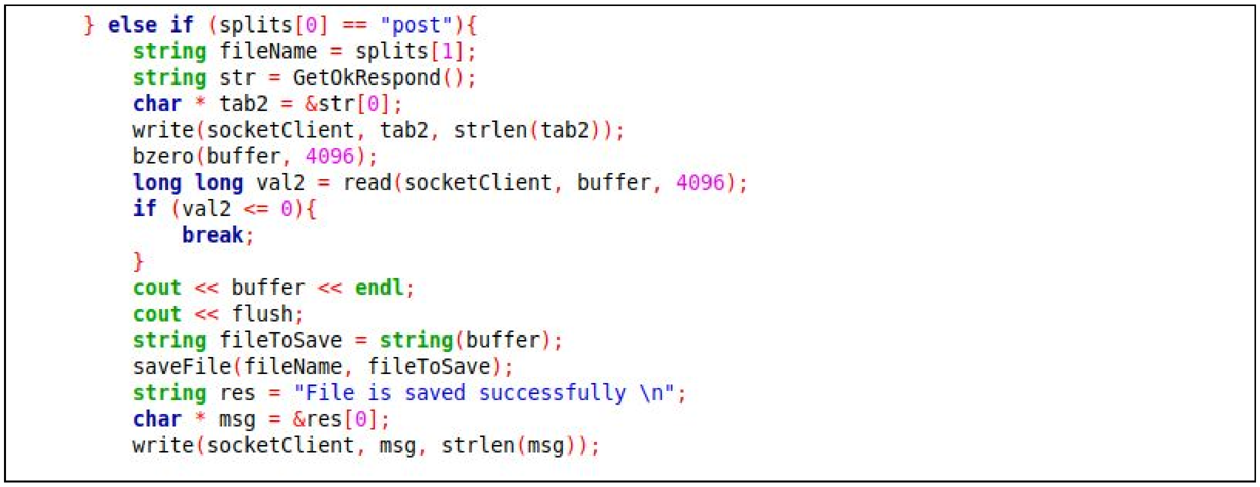
iv. This function handles the requests from a real web browser.

v. Also the function can close the client socket connection.

Code snippets from the function :



```
void * HandleClientConnection(void* P_socketClient)
{
    string total_buffer = "";
    arg_struct clientStruct = *((arg_struct*) P_socketClient);
    //int socketClient = *((int*) P_socketClient);
    int socketClient = clientStruct.client_socket;
    long long* currentClock = clientStruct.timer;
    *(currentClock) = clock();
    while (1)
    {
        *(currentClock) = clock();
        char buffer[4096] = {0};
        long long val = read(socketClient, buffer, 4096);
        if (val <= 0){
            break;
        }
        cout << buffer << endl;
        cout << flush;
        buffer[strlen(buffer) - 1] = '\0';
        total_buffer += buffer;
        vector<string> splits = splitRequest(total_buffer);
        if (splits[0] == "get") {
            string fileName = splits[1];
            if (checkFileExist(fileName))
            {
                string ok = GetOkRespond();
                char * tab2 = &ok[0];
                write(socketClient, tab2, strlen(tab2));
                string content = readFileContent(fileName);
                int fileSize = content.size();
                write(socketClient, &fileSize, sizeof(int));
                sendChunks(socketClient, content);
```

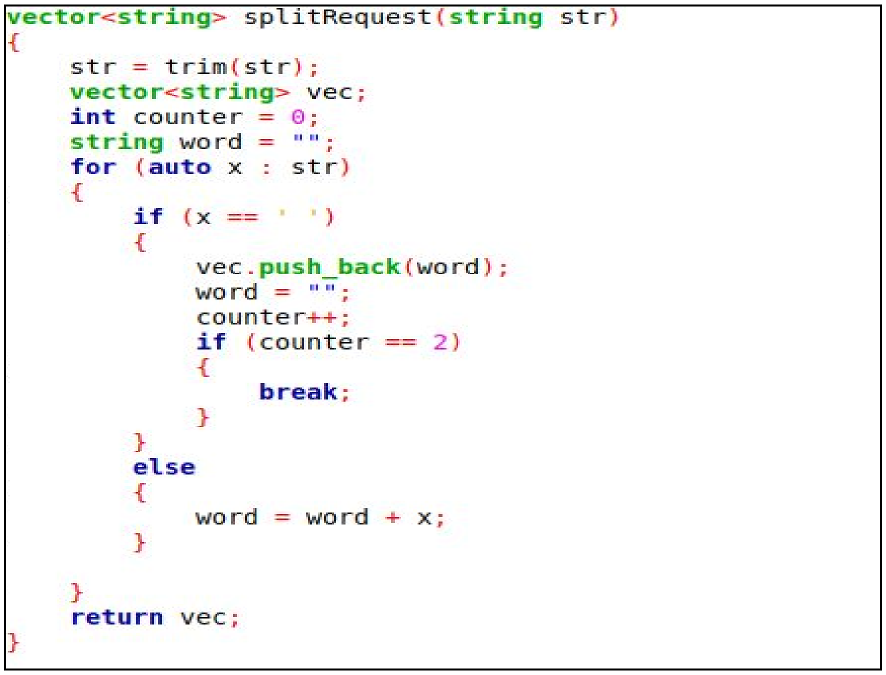


```
    } else if (splits[0] == "post"){
        string fileName = splits[1];
        string str = GetOkRespond();
        char * tab2 = &str[0];
        write(socketClient, tab2, strlen(tab2));
        bzero(buffer, 4096);
        long long val2 = read(socketClient, buffer, 4096);
        if (val2 <= 0){
            break;
        }
        cout << buffer << endl;
        cout << flush;
        string fileToSave = string(buffer);
        saveFile(fileName, fileToSave);
        string res = "File is saved successfully \n";
        char * msg = &res[0];
        write(socketClient, msg, strlen(msg));
```

b. vector<string> splitRequest(string str)

i. This function takes the client's request.

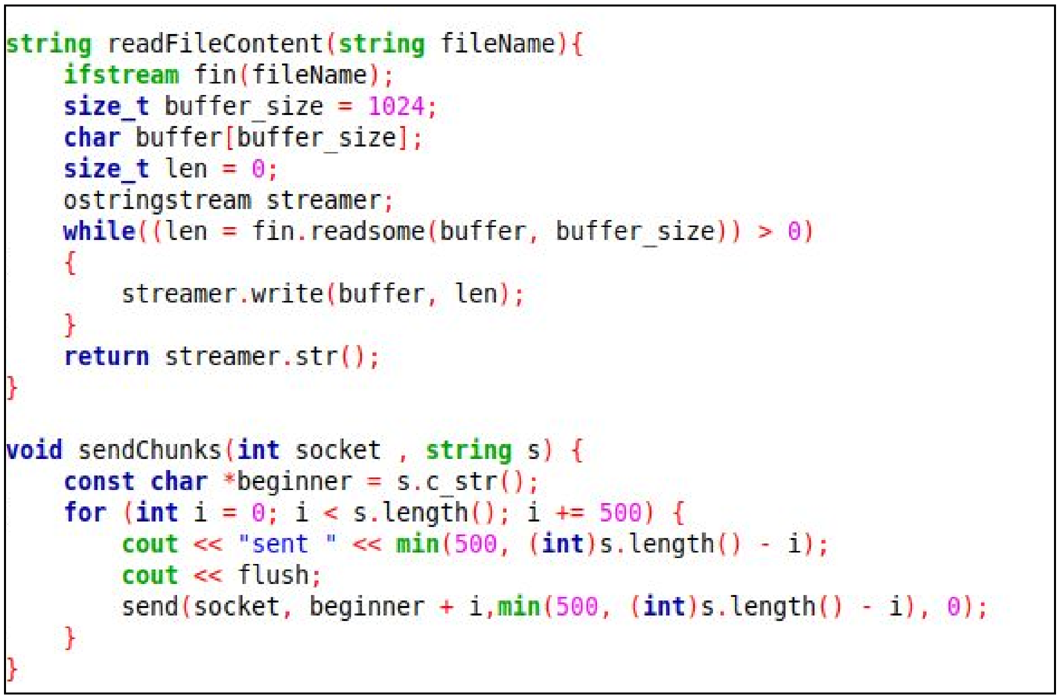
ii. Split it by spaces and return the vector of request type and the file name.



```
vector<string> splitRequest(string str)
{
    str = trim(str);
    vector<string> vec;
    int counter = 0;
    string word = "";
    for (auto x : str)
    {
        if (x == ' ')
        {
            vec.push_back(word);
            word = "";
            counter++;
            if (counter == 2)
            {
                break;
            }
        }
        else
        {
            word = word + x;
        }

    }
    return vec;
}
```

c. **string readFileContent(string fileName)**
   i. This function takes the file name.
   ii. Read the file line by line using string stream and return one string of the file contents.

d. **void sendChunks(int socket , string s)**
   i. This function is used incase of get requests it takes the client socket and the file content.
   ii. It sends this content to the client chunk by chunk until finishing transmitting the file.

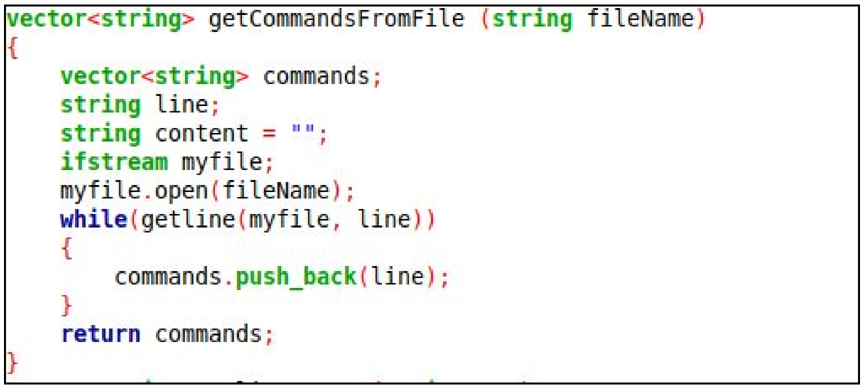


```
string readFileContent(string fileName){
    ifstream fin(fileName);
    size_t buffer_size = 1024;
    char buffer[buffer_size];
    size_t len = 0;
    ostringstream streamer;
    while((len = fin.readsome(buffer, buffer_size)) > 0)
    {
        streamer.write(buffer, len);
    }
    return streamer.str();
}

void sendChunks(int socket , string s) {
    const char *beginner = s.c_str();
    for (int i = 0; i < s.length(); i += 500) {
        cout << "sent " << min(500, (int)s.length() - i);
        cout << flush;
        send(socket, beginner + i,min(500, (int)s.length() - i), 0);
    }
}
```

2. The Client Side

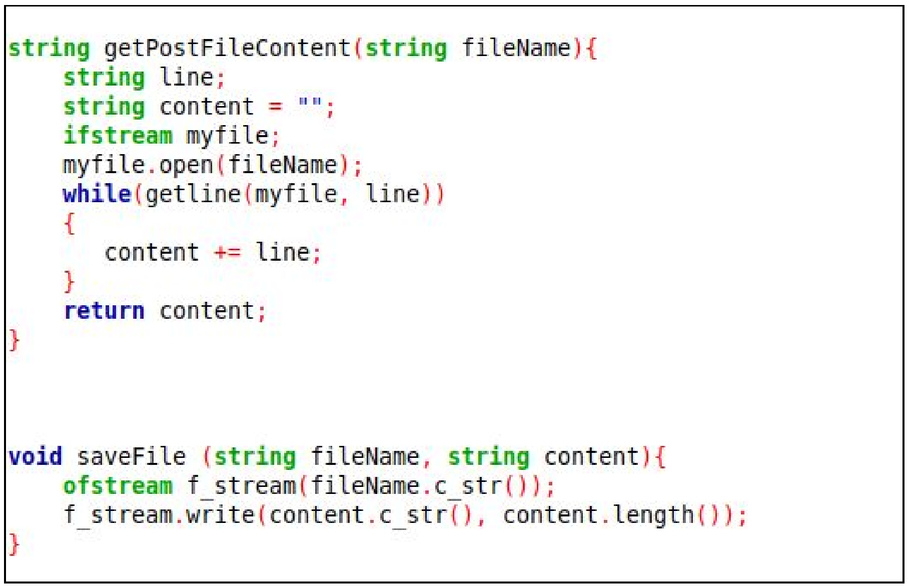
a. **vector<string> getCommandsFromFile (string fileName)**

    i. Read commands line by line and return the vector of strings of them.



```
vector<string> getCommandsFromFile (string fileName)
{
    vector<string> commands;
    string line;
    string content = "";
    ifstream myfile;
    myfile.open(fileName);
    while(getline(myfile, line))
    {
        commands.push_back(line);
    }
    return commands;
}
```

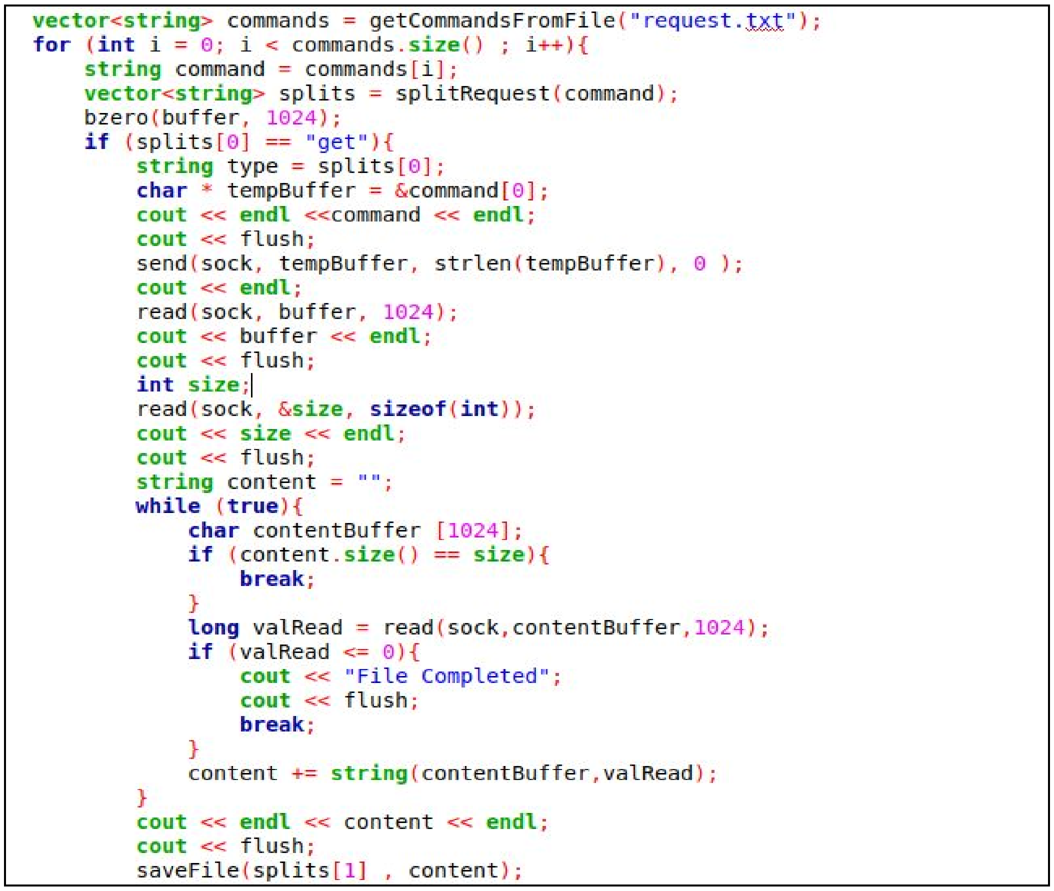
b. Function to readFileContent in case of POST request and another one write into a file in case of GET Request.



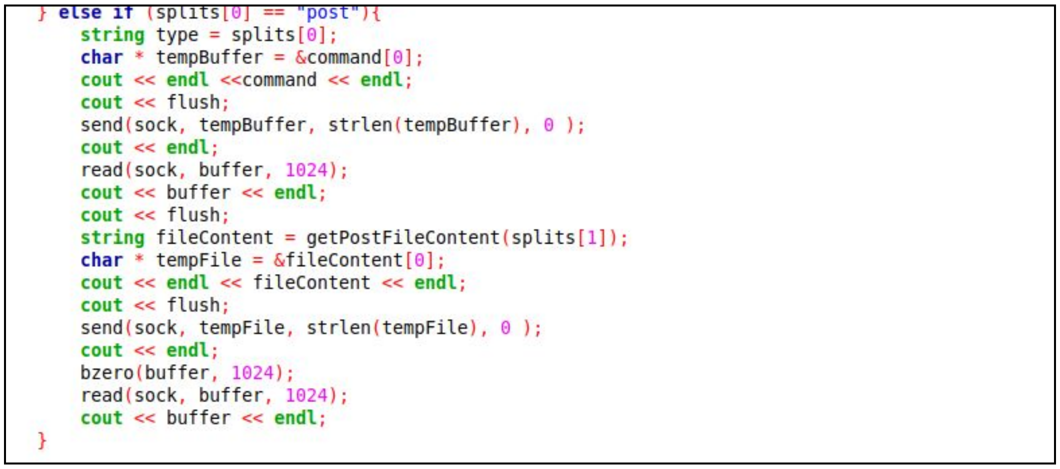
```
string getPostFileContent(string fileName){
    string line;
    string content = "";
    ifstream myfile;
    myfile.open(fileName);
    while(getline(myfile, line))
    {
        content += line;
    }
    return content;
}



void saveFile (string fileName, string content){
    ofstream f_stream(fileName.c_str());
    f_stream.write(content.c_str(), content.length());
}
```

c. Also used the same function to split request type and file name like above.

d. Loop through the request parse each one check the request type and handle each of them as declared in previous sections.



```cpp
vector<string> commands = getCommandsFromFile("request.txt");
for (int i = 0; i < commands.size() ; i++){
    string command = commands[i];
    vector<string> splits = splitRequest(command);
    bzero(buffer, 1024);
    if (splits[0] == "get"){
        string type = splits[0];
        char * tempBuffer = &command[0];
        cout << endl <<command << endl;
        cout << flush;
        send(sock, tempBuffer, strlen(tempBuffer), 0 );
        cout << endl;
        read(sock, buffer, 1024);
        cout << buffer << endl;
        cout << flush;
        int size;
        read(sock, &size, sizeof(int));
        cout << size << endl;
        cout << flush;
        string content = "";
        while (true){
            char contentBuffer [1024];
            if (content.size() == size){
                break;
            }
            long valRead = read(sock,contentBuffer,1024);
            if (valRead <= 0){
                cout << "File Completed";
                cout << flush;
                break;
            }
            content += string(contentBuffer,valRead);
        }
        cout << endl << content << endl;
        cout << flush;
        saveFile(splits[1] , content);
```

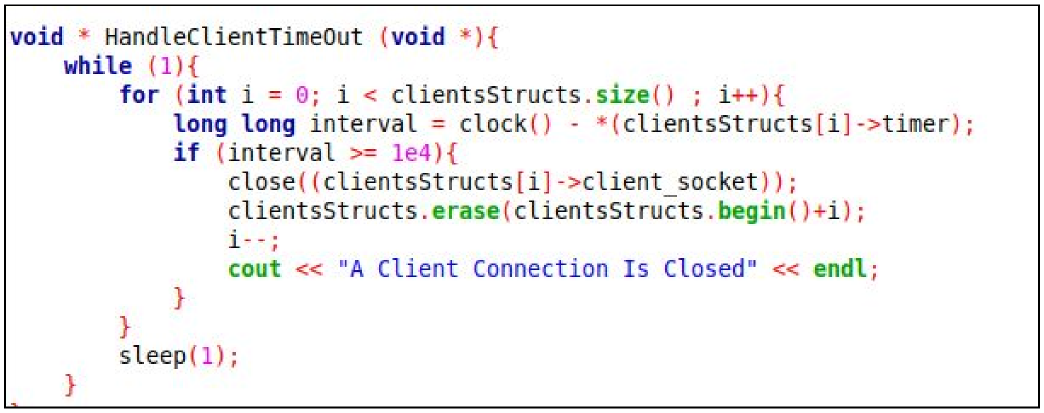


```cpp
} else if (splits[0] == "post"){
    string type = splits[0];
    char * tempBuffer = &command[0];
    cout << endl <<command << endl;
    cout << flush;
    send(sock, tempBuffer, strlen(tempBuffer), 0 );
    cout << endl;
    read(sock, buffer, 1024);
    cout << buffer << endl;
    cout << flush;
    string fileContent = getPostFileContent(splits[1]);
    char * tempFile = &fileContent[0];
    cout << endl << fileContent << endl;
    cout << flush;
    send(sock, tempFile, strlen(tempFile), 0 );
    cout << endl;
    bzero(buffer, 1024);
    read(sock, buffer, 1024);
    cout << buffer << endl;
}
```

3. HTTP 1.1

a. For Persistent connection the client makes multiple requests on the same connection socket without needing to open a new socket for each connection.

b. For timeout idle clients I introduced a new thread that is working in the background watching the activity of each client when the time from the last activity of a client exceeded the threshold this client will be timed out and its socket will be closed . the last active time is updated at each request.

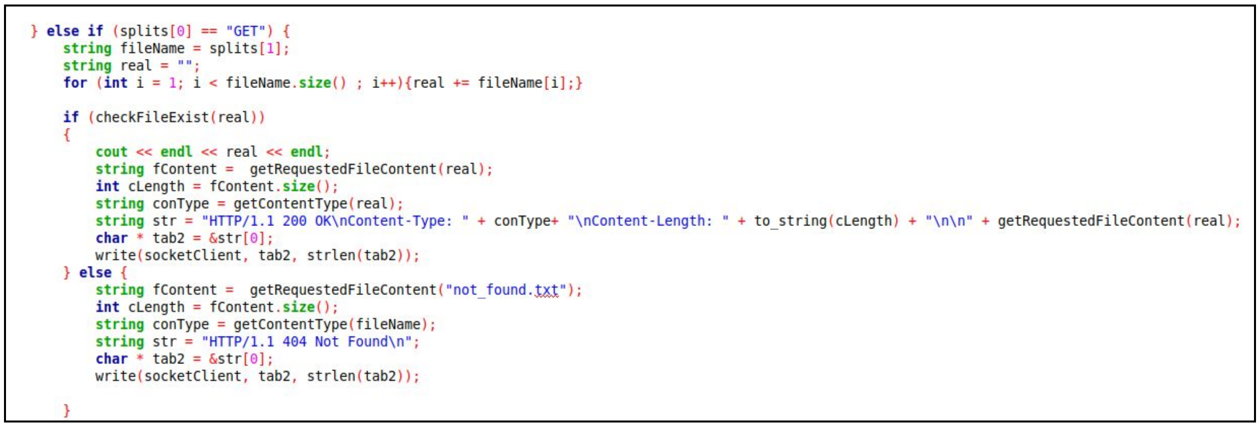


```
void * HandleClientTimeOut (void *){
    while (1){
        for (int i = 0; i < clientsStructs.size() ; i++){
            long long interval = clock() - *(clientsStructs[i]->timer);
            if (interval >= 1e4){
                close((clientsStructs[i]->client_socket));
                clientsStructs.erase(clientsStructs.begin()+i);
                i--;
                cout << "A Client Connection Is Closed" << endl;
            }
        }
        sleep(1);
    }
```

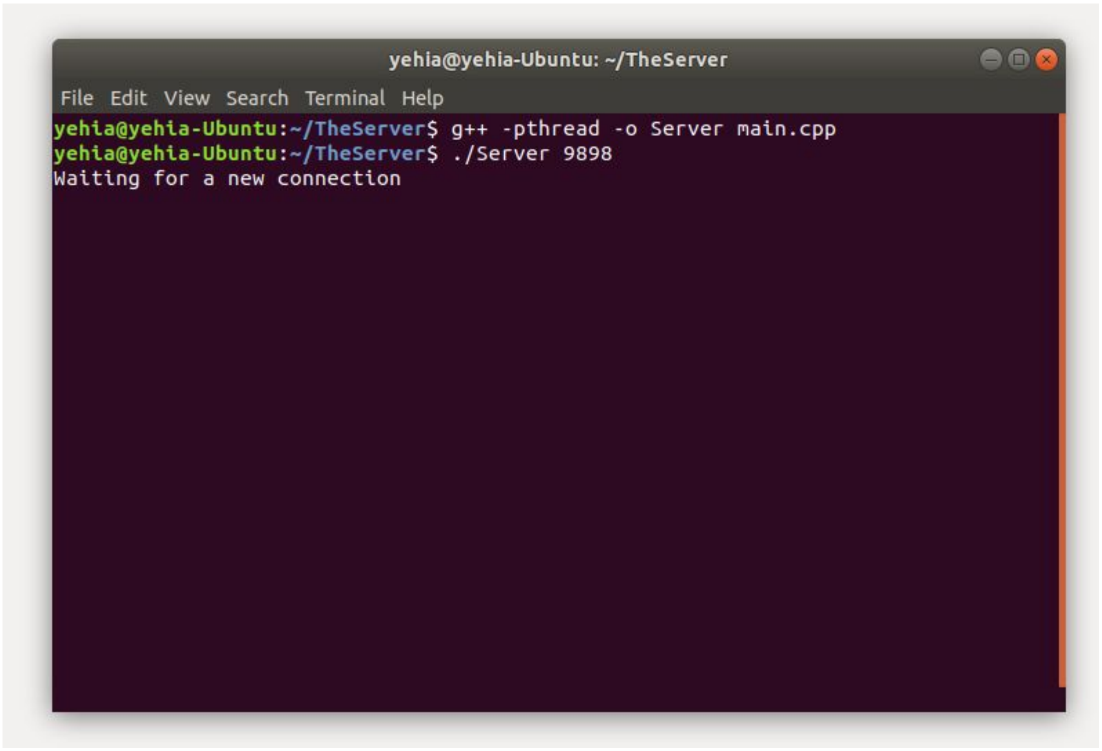
4. Bonus Part

Implemented this part : 5.1 Test your server with a real web browser.

- Handle the request and extract the file name & request type.
- Build the response message format in each case.
- Write it back to the browser.

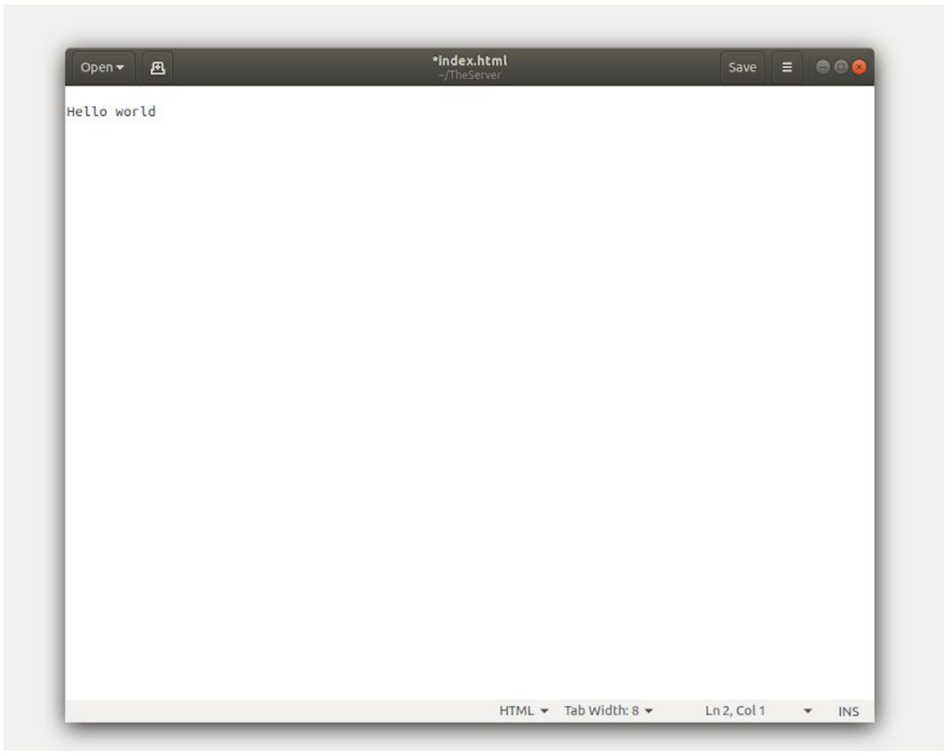


```
} else if (splits[0] == "GET") {
    string fileName = splits[1];
    string real = "";
    for (int i = 1; i < fileName.size() ; i++){real += fileName[i];}

    if (checkFileExist(real))
    {
        cout << endl << real << endl;
        string fContent =  getRequestedFileContent(real);
        int cLength = fContent.size();
        string conType = getContentType(real);
        string str = "HTTP/1.1 200 OK\nContent-Type: " + conType+ "\nContent-Length: " + to_string(cLength) + "\n\n" + getRequestedFileContent(real);
        char * tab2 = &str[0];
        write(socketClient, tab2, strlen(tab2));
    } else {
        string fContent =  getRequestedFileContent("not_found.txt");
        int cLength = fContent.size();
        string conType = getContentType(fileName);
        string str = "HTTP/1.1 404 Not Found\n";
        char * tab2 = &str[0];
        write(socketClient, tab2, strlen(tab2));

    }
```

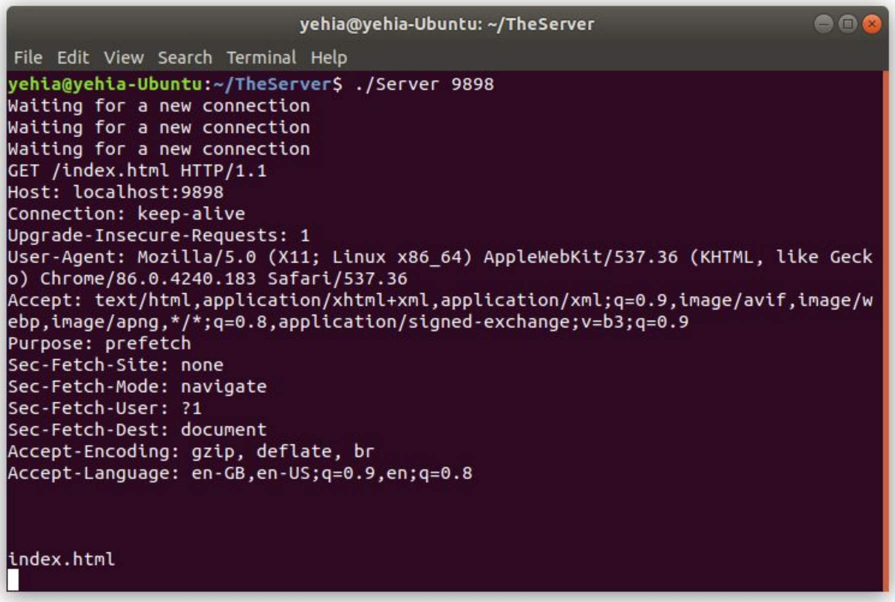


yehia@yehia-Ubuntu: ~/TheServer
File Edit View Search Terminal Help

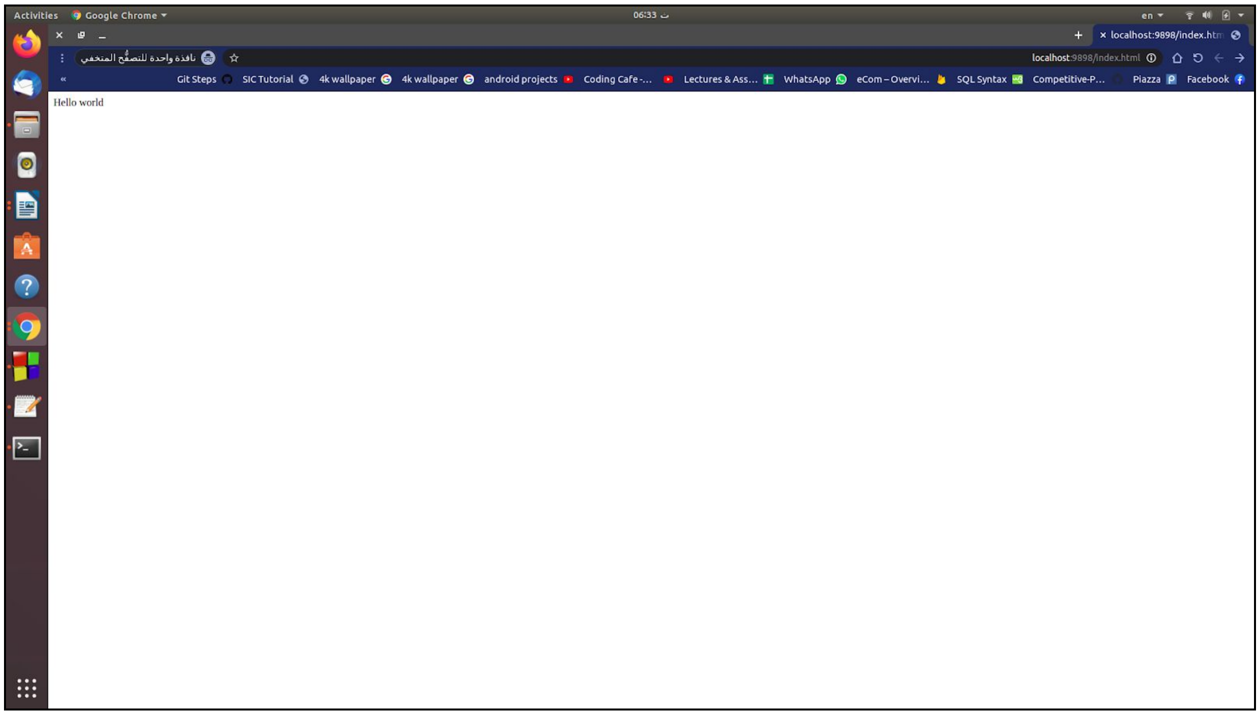
```
yehia@yehia-Ubuntu:~/TheServer$ g++ -pthread -o Server main.cpp
yehia@yehia-Ubuntu:~/TheServer$ ./Server 9898
Waiting for a new connection
```



```
Hello world
```


yehia@yehia-Ubuntu: ~/TheServer
File Edit View Search Terminal Help

```
yehia@yehia-Ubuntu:~/TheServer$ ./Server 9898
Waiting for a new connection
Waiting for a new connection
Waiting for a new connection
GET /index.html HTTP/1.1
Host: localhost:9898
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Geck
o) Chrome/86.0.4240.183 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Purpose: prefetch
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8



index.html
```

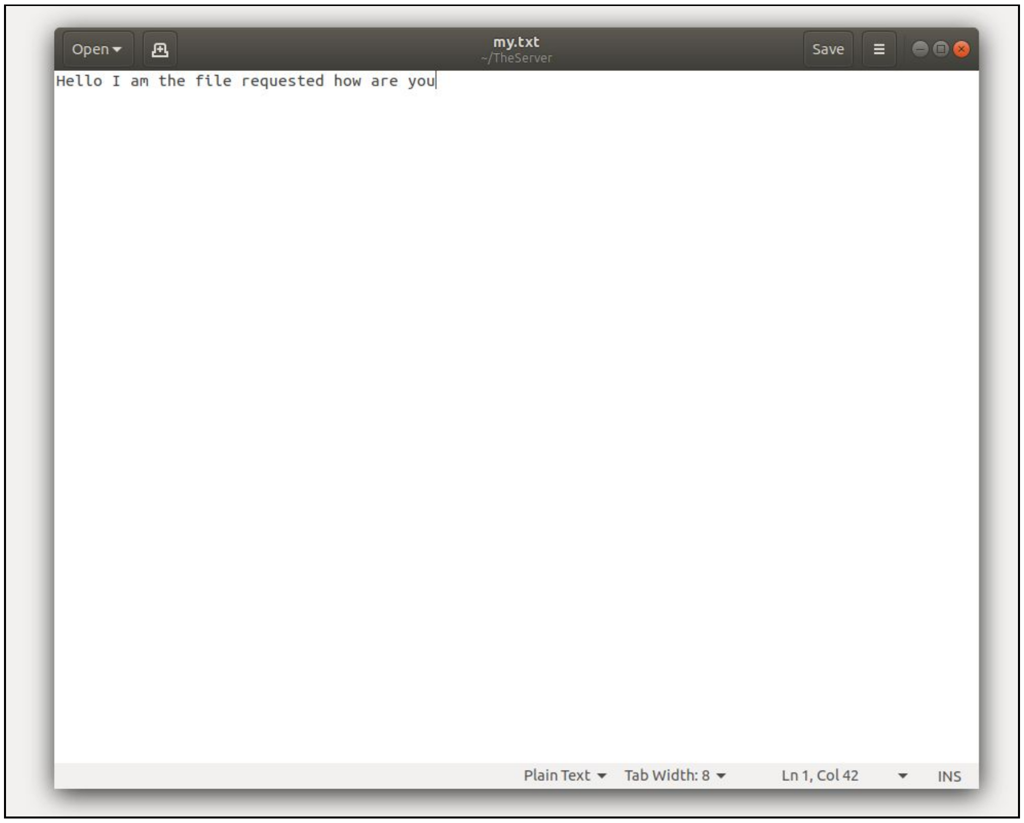


Hello world

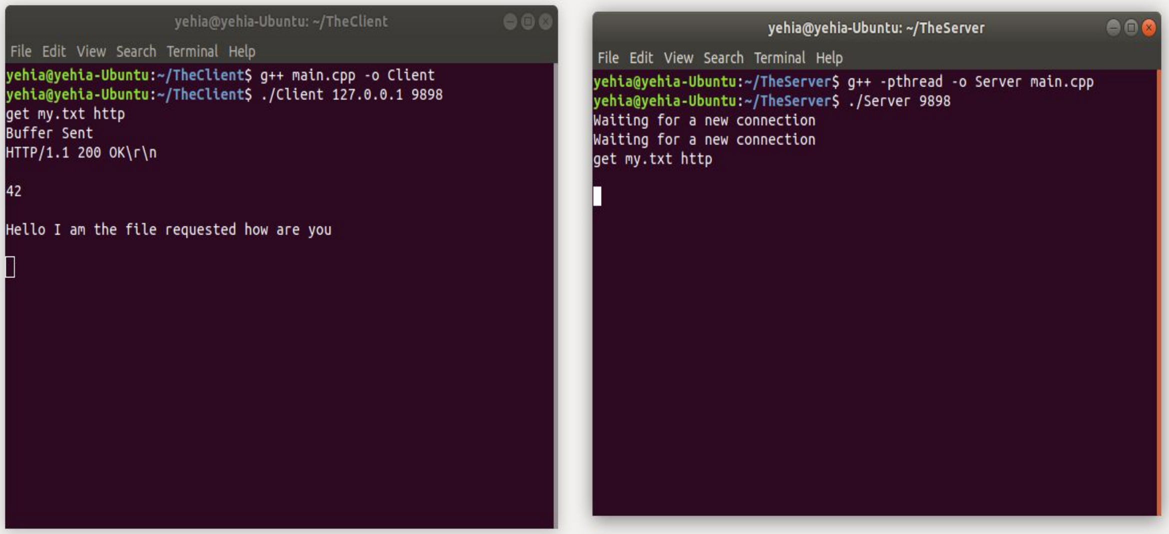
Sample Runs

- **GET Request**

  The requested file



```
Hello I am the file requested how are you
```



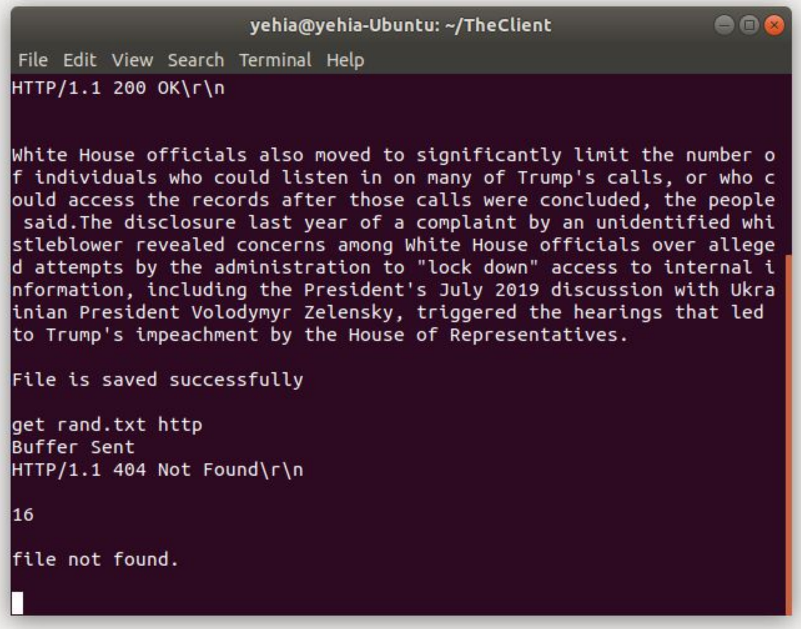
yehia@yehia-Ubuntu: ~/TheClient
File Edit View Search Terminal Help

```
yehia@yehia-Ubuntu:~/TheClient$ g++ main.cpp -o Client
yehia@yehia-Ubuntu:~/TheClient$ ./Client 127.0.0.1 9898
get my.txt http
Buffer Sent
HTTP/1.1 200 OK\r\n

42

Hello I am the file requested how are you
```

yehia@yehia-Ubuntu: ~/TheServer
File Edit View Search Terminal Help

```
yehia@yehia-Ubuntu:~/TheServer$ g++ -pthread -o Server main.cpp
yehia@yehia-Ubuntu:~/TheServer$ ./Server 9898
Waiting for a new connection
Waiting for a new connection
get my.txt http
```

- **Get not existing file**

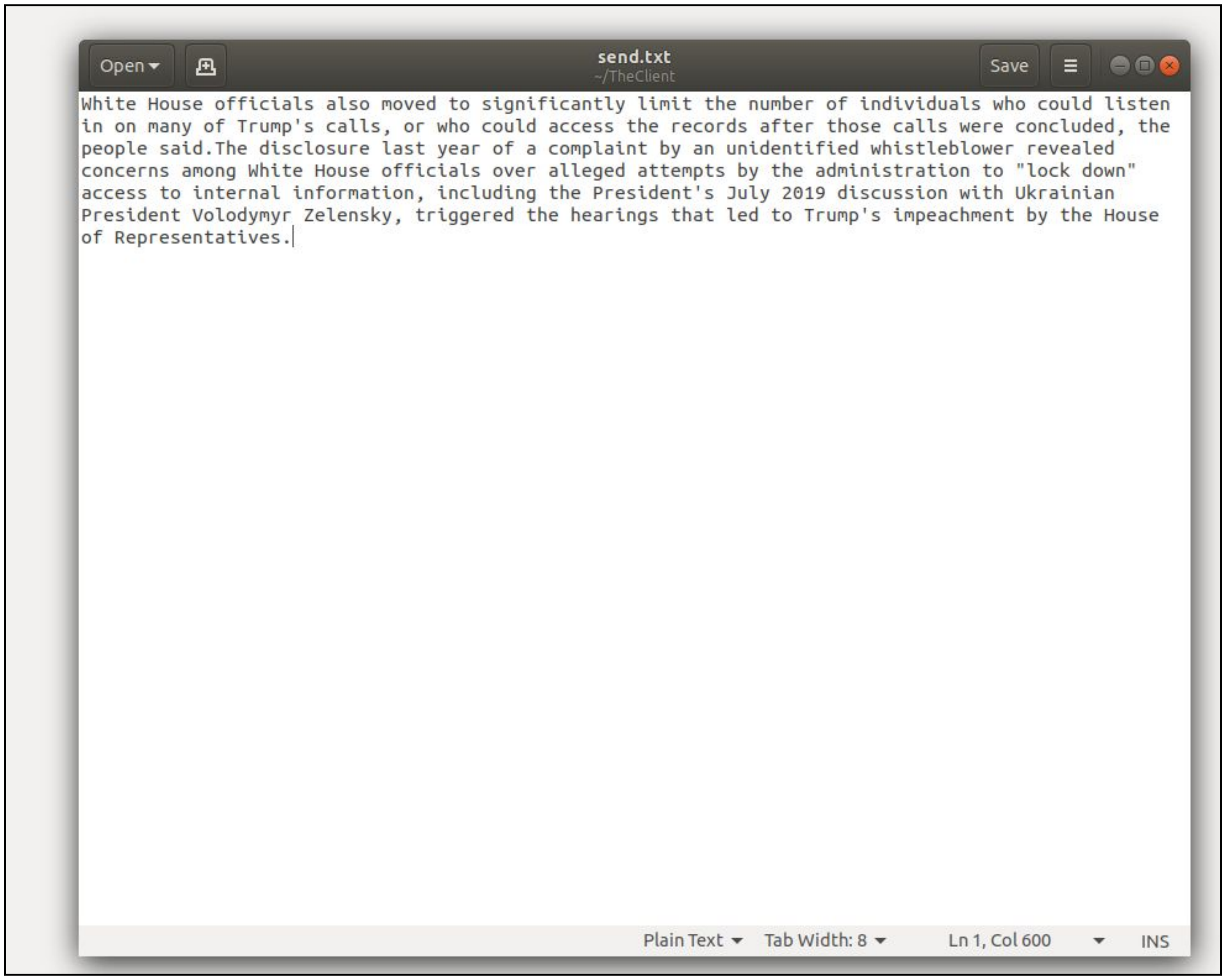


yehia@yehia-Ubuntu: ~/TheClient
File Edit View Search Terminal Help

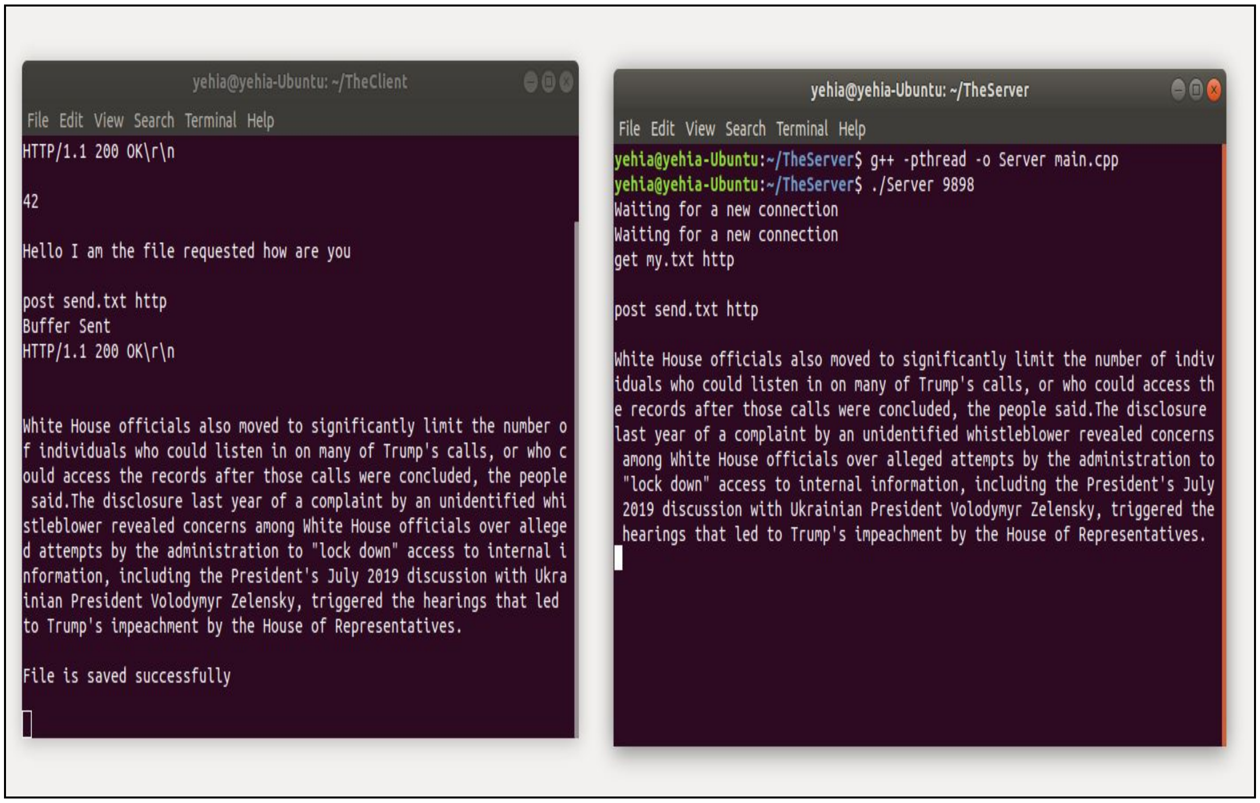
```
HTTP/1.1 200 OK\r\n


White House officials also moved to significantly limit the number o
f individuals who could listen in on many of Trump's calls, or who c
ould access the records after those calls were concluded, the people
 said.The disclosure last year of a complaint by an unidentified whi
stleblower revealed concerns among White House officials over allege
d attempts by the administration to "lock down" access to internal i
nformation, including the President's July 2019 discussion with Ukra
inian President Volodymyr Zelensky, triggered the hearings that led
to Trump's impeachment by the House of Representatives.

File is saved successfully

get rand.txt http
Buffer Sent
HTTP/1.1 404 Not Found\r\n

16

file not found.
```

- **Post Request**

  The posted file :



```
White House officials also moved to significantly limit the number of individuals who could listen
in on many of Trump's calls, or who could access the records after those calls were concluded, the
people said.The disclosure last year of a complaint by an unidentified whistleblower revealed
concerns among White House officials over alleged attempts by the administration to "lock down"
access to internal information, including the President's July 2019 discussion with Ukrainian
President Volodymyr Zelensky, triggered the hearings that led to Trump's impeachment by the House
of Representatives.
```

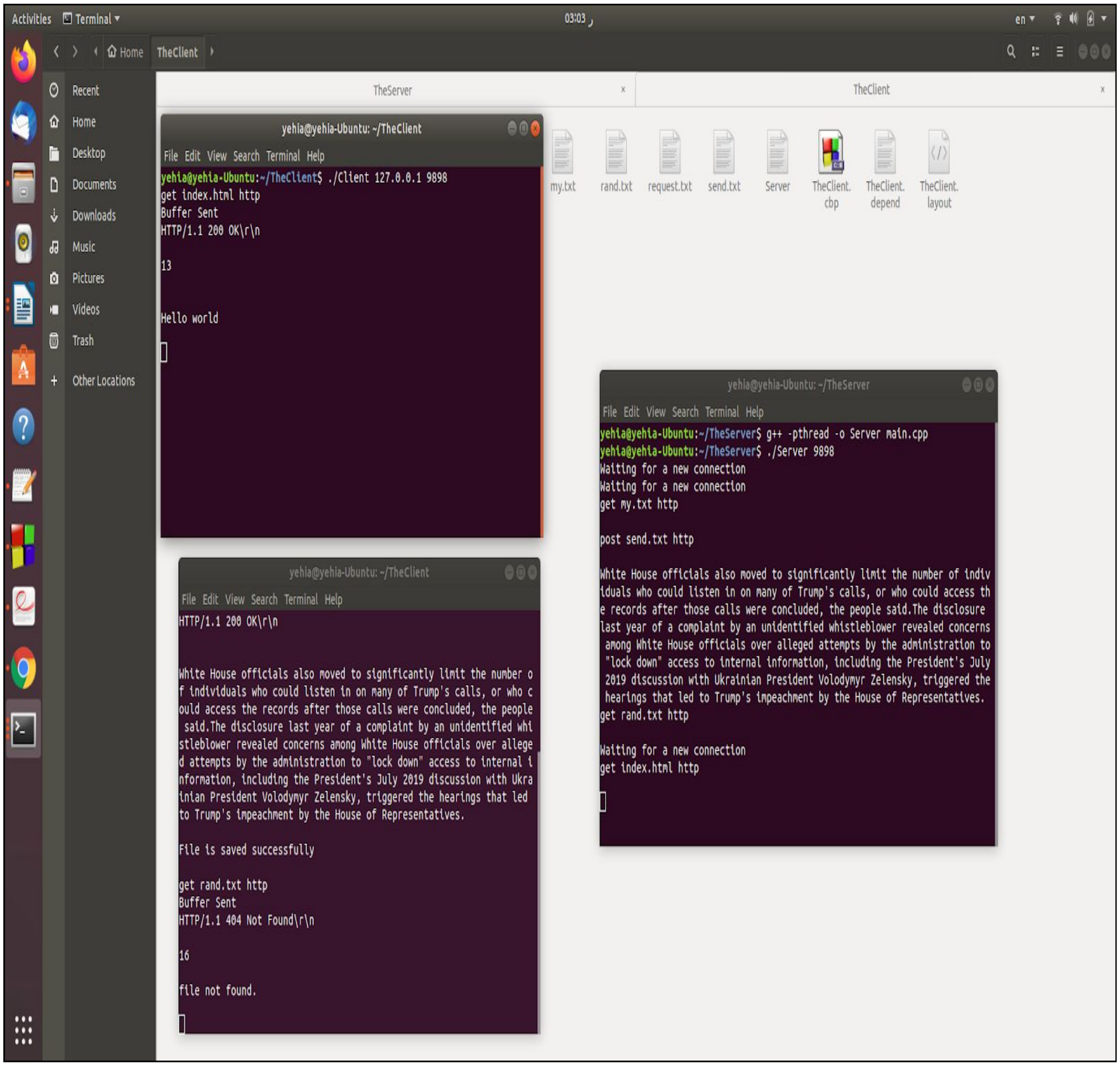

yehia@yehia-Ubuntu: ~/TheClient
File Edit View Search Terminal Help

```
HTTP/1.1 200 OK\r\n

42

Hello I am the file requested how are you

post send.txt http
Buffer Sent
HTTP/1.1 200 OK\r\n


White House officials also moved to significantly limit the number o
f individuals who could listen in on many of Trump's calls, or who c
ould access the records after those calls were concluded, the people
 said.The disclosure last year of a complaint by an unidentified whi
stleblower revealed concerns among White House officials over allege
d attempts by the administration to "lock down" access to internal i
nformation, including the President's July 2019 discussion with Ukra
inian President Volodymyr Zelensky, triggered the hearings that led
to Trump's impeachment by the House of Representatives.

File is saved successfully
```

yehia@yehia-Ubuntu: ~/TheServer
File Edit View Search Terminal Help

```
yehia@yehia-Ubuntu:~/TheServer$ g++ -pthread -o Server main.cpp
yehia@yehia-Ubuntu:~/TheServer$ ./Server 9898
Waiting for a new connection
Waiting for a new connection
get my.txt http

post send.txt http

White House officials also moved to significantly limit the number of indiv
iduals who could listen in on many of Trump's calls, or who could access th
e records after those calls were concluded, the people said.The disclosure
last year of a complaint by an unidentified whistleblower revealed concerns
 among White House officials over alleged attempts by the administration to
 "lock down" access to internal information, including the President's July
 2019 discussion with Ukrainian President Volodymyr Zelensky, triggered the
 hearings that led to Trump's impeachment by the House of Representatives.
```

- **Handle multiple clients at a time**

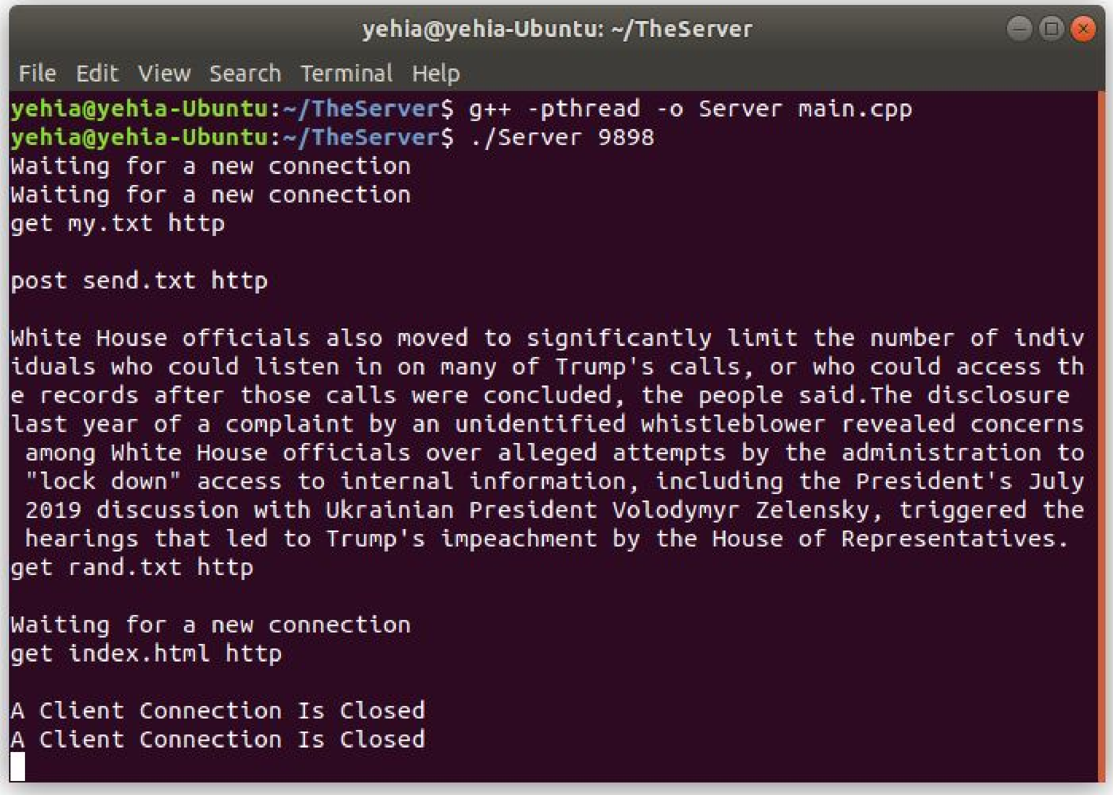


```
yehia@yehia-Ubuntu:~/TheClient$ ./Client 127.0.0.1 9898
get index.html http
Buffer Sent
HTTP/1.1 200 OK\r\n

13


Hello world
```

```
HTTP/1.1 200 OK\r\n


White House officials also moved to significantly limit the number o
f individuals who could listen in on many of Trump's calls, or who c
ould access the records after those calls were concluded, the people
 said.The disclosure last year of a complaint by an unidentified whi
stleblower revealed concerns among White House officials over allege
d attempts by the administration to "lock down" access to internal i
nformation, including the President's July 2019 discussion with Ukra
inian President Volodymyr Zelensky, triggered the hearings that led
to Trump's impeachment by the House of Representatives.

File is saved successfully

get rand.txt http
Buffer Sent
HTTP/1.1 404 Not Found\r\n

16

file not found.
```

```
yehia@yehia-Ubuntu:~/TheServer$ g++ -pthread -o Server main.cpp
yehia@yehia-Ubuntu:~/TheServer$ ./Server 9898
Waiting for a new connection
Waiting for a new connection
get my.txt http

post send.txt http

White House officials also moved to significantly limit the number of indiv
iduals who could listen in on many of Trump's calls, or who could access th
e records after those calls were concluded, the people said.The disclosure
last year of a complaint by an unidentified whistleblower revealed concerns
 among White House officials over alleged attempts by the administration to
 "lock down" access to internal information, including the President's July
 2019 discussion with Ukrainian President Volodymyr Zelensky, triggered the
 hearings that led to Trump's impeachment by the House of Representatives.
get rand.txt http

Waiting for a new connection
get index.html http
```

- Closing idle clients connections:


yehia@yehia-Ubuntu: ~/TheServer
File Edit View Search Terminal Help

```
yehia@yehia-Ubuntu:~/TheServer$ g++ -pthread -o Server main.cpp
yehia@yehia-Ubuntu:~/TheServer$ ./Server 9898
Waiting for a new connection
Waiting for a new connection
get my.txt http

post send.txt http

White House officials also moved to significantly limit the number of indiv
iduals who could listen in on many of Trump's calls, or who could access th
e records after those calls were concluded, the people said.The disclosure 
last year of a complaint by an unidentified whistleblower revealed concerns
 among White House officials over alleged attempts by the administration to
 "lock down" access to internal information, including the President's July
 2019 discussion with Ukrainian President Volodymyr Zelensky, triggered the
 hearings that led to Trump's impeachment by the House of Representatives.
get rand.txt http

Waiting for a new connection
get index.html http

A Client Connection Is Closed
A Client Connection Is Closed
```