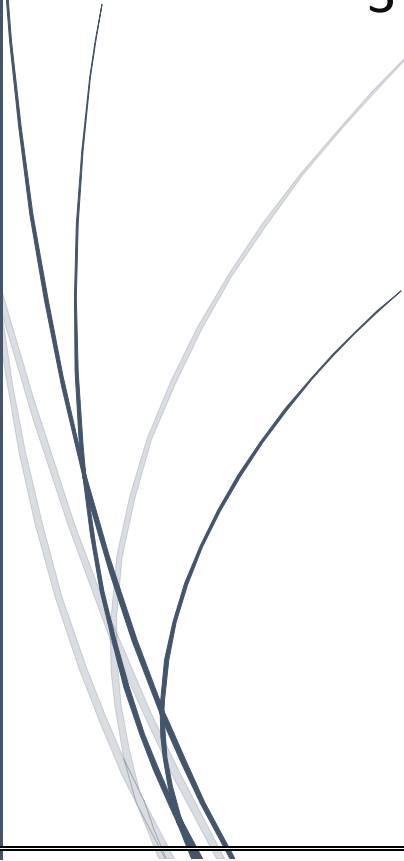




# **Programming Assignment 1- Introduction to Socket Programming in C/C++**

Group:

- 1- Ebtehal Elaraby (2)
  - 2- Remon Hanna (21)
  - 3- Shaimaa Mousa (26)
- 

## **Contents:**

- 1)Abstract.**
- 2)Introduction.**
- 3)System Model.**
- 4)Assumptions.**
- 5)Evaluation.**
- 6)Conclusion.**
- 7)Reference.**

## 1) Abstract :

This program is a simple web client that communicates with a web server using a restricted subset of HTTP. Implementing this web server is done by using socket programming with C++ and it's used multi-threading to achieve communication with more than one client at same time. Using socket programming, the server could accept incoming connection requests and look for the GET (to transmit/download html , png , txt , e.g. Files from server to client) /POST(to upload html , png , txt , e.g. Files on server from client ) and Transmit contents of file (in case of get) and serve many clients at same time. The program also offers communicate with real browser. Results show that Communicating server with client and posting and getting files for one client is faster than more than one.

## **2) Introduction :**

### **2.1) Purpose:**

This assignment studies how use sockets to create server/client protocol and using multi-threading to serve many clients by using TCP connection and implementing GET (to send files,images,html)from server to client and POST (to upload files ) from client to server and experience with UNIX sockets.

### **2.2) background:**

Berners-Lee and his team are credited for inventing the original Hyper Text 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!

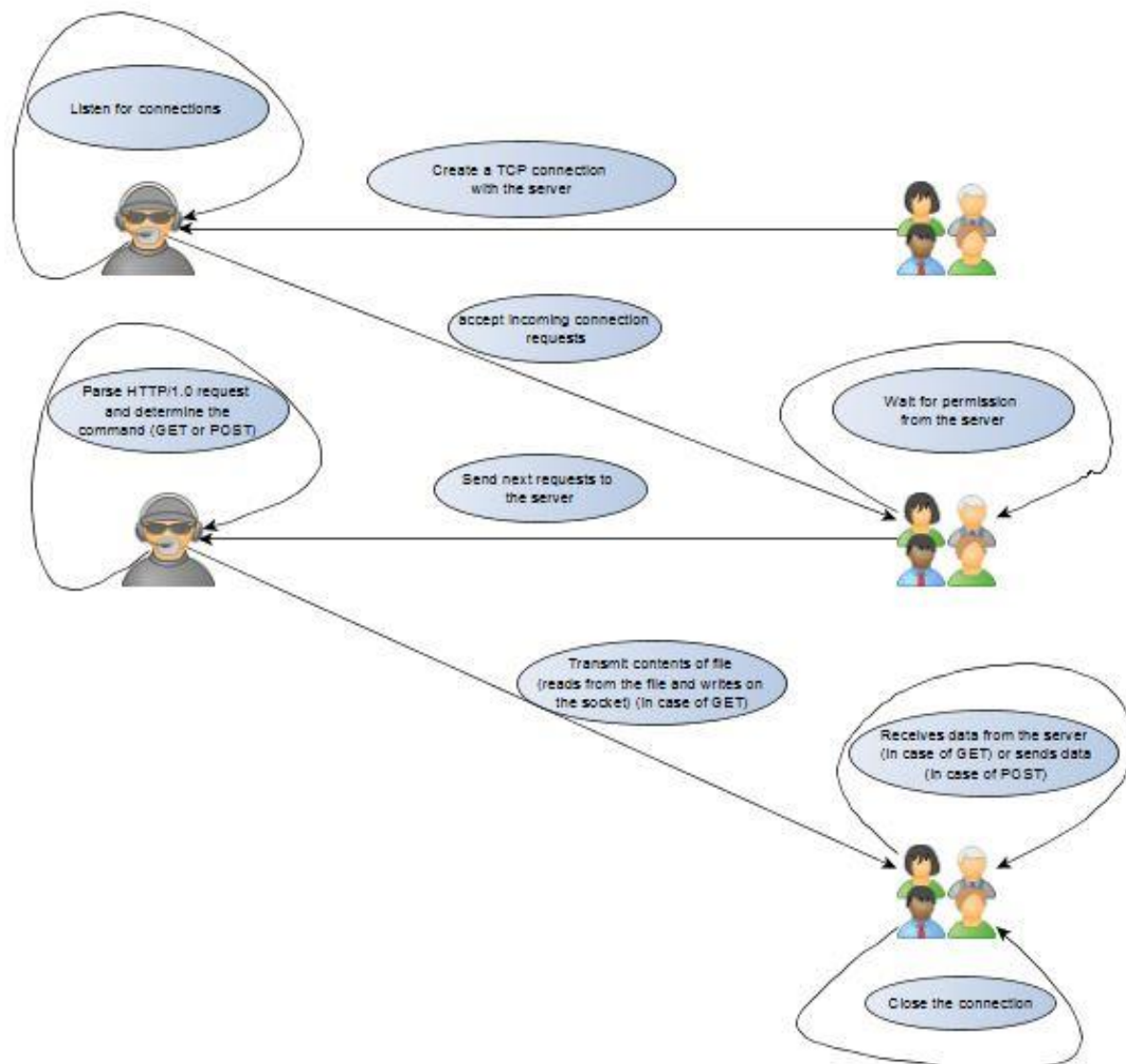
### **2.3) Method of Investigation:**

We used socket programming to create the server client as following:

- 1- First we tried to bind the server and one client can connect it.
- 2- We added get and post methods to client and server.
- 3- We added multithreaded technique to server to handle parallel connections

### 3) System model :

Here is a simple model of the system. The client starts the TCP connection, the server listens all the time of connection then accept the client's connection and send a message , once the client receives the acceptance message from the server he continues sending his requests to server and the server parses these requests and execute them.



## External Documentation:

### 1- For Server

Just one class main.cpp that has the following:

- **Methods**

1- main method:

First, it initializes the server sockaddr\_in struct and create a socket then use it to bind. Then it loop forever to listen to requests, accept coming connections and then create a thread for them .

2- task method:

This is the function that any thread must execute it. First, take the request, parse it and then determine whether it's get or post.

3- get method:

This method takes the file name the user request as input then it searched for it.

- If found then it send "HTTP/1.0 200 OK\r\n" then send the files in chunks every chunk is 1024 character.
- If not found then it sends "HTTP/1.0 404 not found /r/n" to indicate that the file doesn't exist.

4- post method:

This method takes the file name , which the user requests to post, as input then it sends OK message for client to begin transmitting the file while receiving it from server and write it to the specified file.

5- split and clean\_string methods:

They are used to help in parsing the command string coming from user.

- Data Structures

- socklen\_t len; //for host addr length
- int MAX\_NO\_OF\_CONNECTIONS=300;
  - for the max number of connection that server can listen
- struct sockaddr\_in svrAdd, clntAdd;
  - 2 structs one for the server and the other for the client to hold their information
- pthread\_t thread;
  - threads used to execute every request in parallel
- struct PARAMS
  - {  
int clientSocket;
    - this int to hold the client socket.
- };
- vector<string>;
  - vectors used to help parsing the command line.

## 2- For Client

Just one class main.cpp that has the following:

- **Methods**

1- main method:

First, it initialize the server sockaddr\_in struct and create a connection with it from the arguments that it takes from the user which is server ip and port number.

Then it calles the method read\_file\_commands().

2- read\_file\_commandsmethod:

This is the function that read the input file that contain the requests after reading them, it sends them to execute method.

3- execute method:

It parses the command line determine the server information then try to connect to it and parse the command line to determine whether it's get or post then it sends this command to the server.

4- get method:

This method takes the file name ,that the server is going to transmit, as input then it receive first the replay of the server for the request:

- If found then it receive the files in chunks every chunk is 1024 character.
- If not found then do nothing.



#### 5- post method:

This method takes the file name that it's required to post as input then it waits for the OK message from server then it begins transmitting the file while receiving it from server and write it to the specified file.

#### 6- split and clean\_string methods:

They are used to help in parsing the command string coming from user.

- **Data Structure**

- socklen\_t len; //for host addr length
- int MAX\_NO\_OF\_CONNECTIONS=300;
  - for the max number of connection that server can listen
- struct sockaddr\_in svrAdd;
  - struct for the server to hold its information
- struct hostent \*server;
  - struct to help in finding the server.
- vector<string>;
  - vectors used to help parsing the command line.

#### 4) Assumptions :

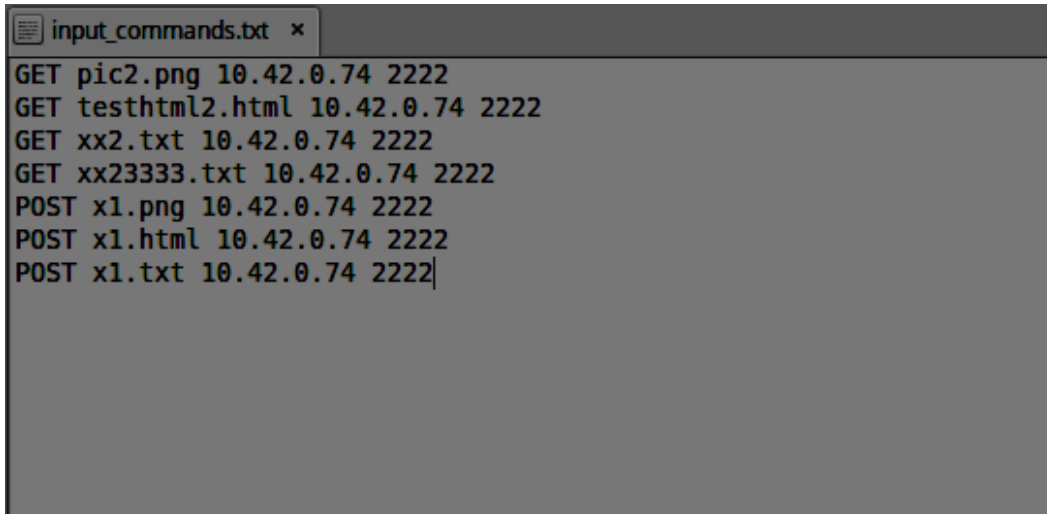
We assumed the following:

- The input user commands are valid on these format  
<GET/POST file\_name host\_name port\_no>.
- The only HTTP status codes we handled are 404 and 200.

## 5) Evaluation :

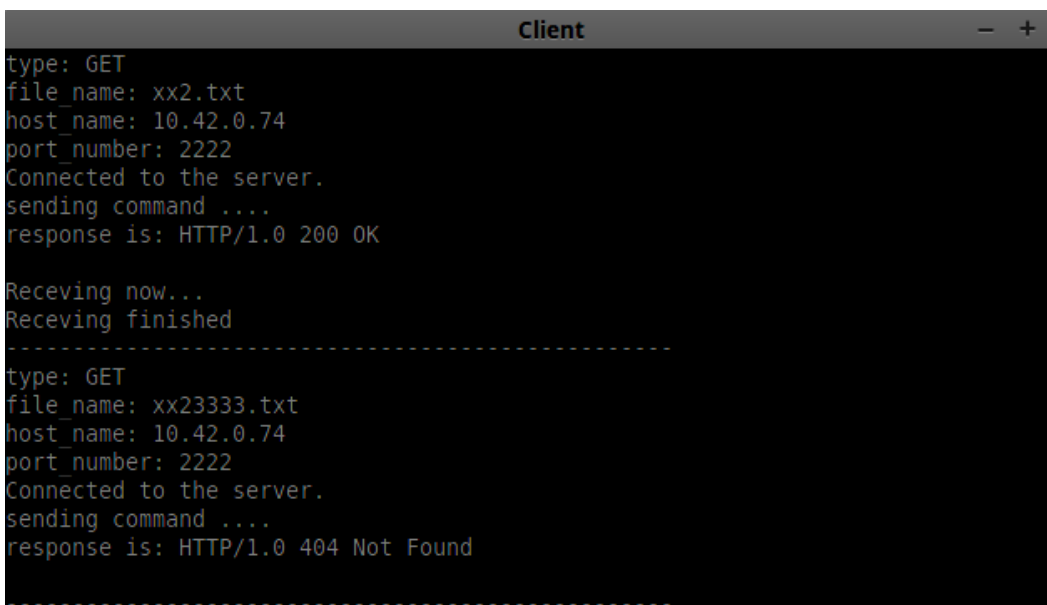
Now , the server/client will be evaluated using two Different ways  
“get” and “post” from client to server and test more than one client  
at the same time .Here are some sample runs from our web :

the input requests example :



```
input_commands.txt x
GET pic2.png 10.42.0.74 2222
GET testhtml2.html 10.42.0.74 2222
GET xx2.txt 10.42.0.74 2222
GET xx23333.txt 10.42.0.74 2222
POST x1.png 10.42.0.74 2222
POST x1.html 10.42.0.74 2222
POST x1.txt 10.42.0.74 2222|
```

1) Get example :



```
Client
type: GET
file_name: xx2.txt
host_name: 10.42.0.74
port_number: 2222
Connected to the server.
sending command ....
response is: HTTP/1.0 200 OK

Receiving now...
Receiving finished
-----
type: GET
file_name: xx23333.txt
host_name: 10.42.0.74
port_number: 2222
Connected to the server.
sending command ....
response is: HTTP/1.0 404 Not Found
-----
```

### 1) Post example :

```
-----  
type: POST  
file_name: x1.png  
host_name: 10.42.0.74  
port_number: 2222  
Connected to the server.  
sending command ....  
Response from server is  OK  
Posting now...  
Posting finished.  
-----  
type: POST  
file_name: x1.html  
host_name: 10.42.0.74  
port_number: 2222  
Connected to the server.  
sending command ....  
Response from server is  OK  
Posting now...
```

## Bonus Part

### 1- Testing our server from real web browser:

We choose Firefox from which we send requests the response that reached the server is like this:

/filename/HTTP/1.1

Then we parse it and get the file name, make an html file for it and then sending it to the browser.

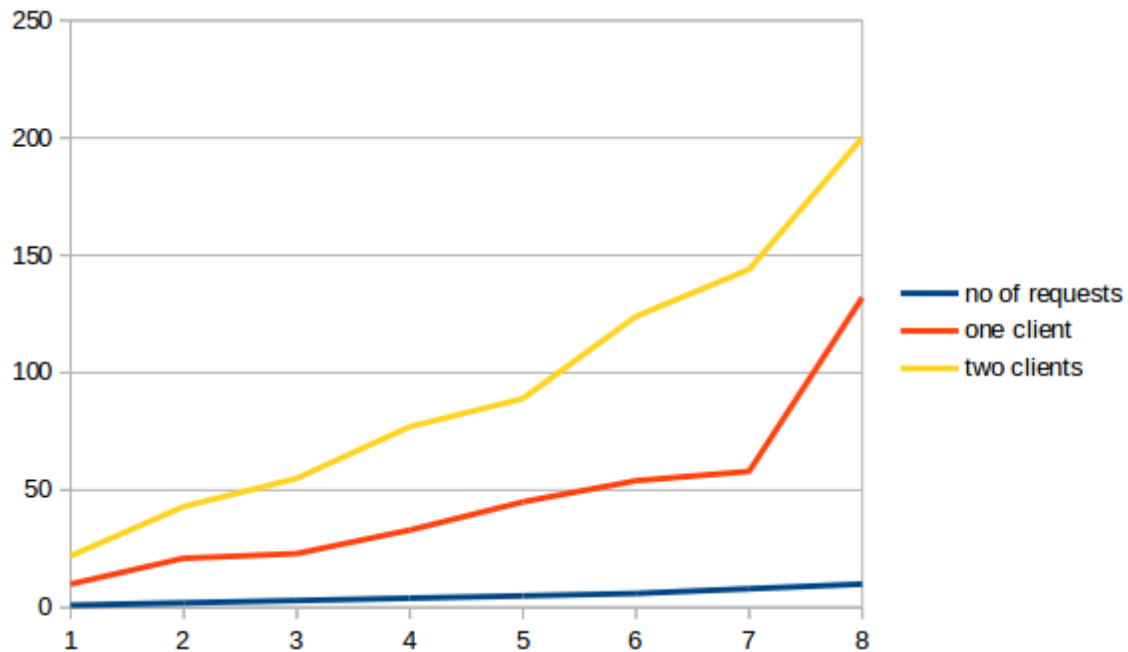
To run it follow these instructions:

1- Open Firefox tab.

2- Write localhost:port\_number/file\_name

Then the data you requested should have been showed.

2- we run many client at the same time then plot the relation between number of requests and delay time.



## 6) Conclusion :

- We can use Socket programming to create simple client server protocol.
- Http/1.0 supports non persistent connections which create a single socket for each request.

## 7) References:

- 1) <http://beej.us/guide/bgnet/output/html/multipage/index.html>
- 2) "TCP\IP Sockets in C" book