CN Assignment1

# Introduction to Socket Programming in C/C++

Linh Ahmed Mohamed Salah

ID: 48

## Problem Statements:

### Part 1: Server

- Your web server should accept incoming connection requests. It should then look for the GET request and pick out the name of the requested file. If the request is POST then it sends an OK message and waits for the uploaded file from the client.
- Your server should first print out the received command, then the server should then respond.
- After finishing the transmission, the server should keep the connection open waiting for new requests from the same client. If the document is not found (in case of GET), the server should respond with Not found Message

### Part 2: Client

- Your web client must read and parse a series of commands from the input file. If it is not specified, use the default HTTP port number, 80.
- In response to the specified operation (GET or POST), the client must open a connection to an HTTP server on the specified host listening on the specified (or default) port number.
- The receiver must display the file and then store it in the local directory. The client should shut down when reaching the end of the file.

### Part 3: Timeout

- You are required to add simple HTTP/1.1 support to your web server,.
- You will also need to add some heuristic to your web server

## Code Design:

### In Server Code:

- First, Opening a Socket and binding with a local address.
- Loop While forever
    - do Listen for connections.
    - Accept new connection from incoming client and delegate it to worker thread/process.
    - In the worker thread set time begin for timeout and start looping on a duration of 5 seconds
    - When receiving a new HTTP/1.1 request ,Parse it , determine the command (GET or POST) and restart the beginning time.
    - Determine if the target file exists (in case of GET) and return 404 Not Found otherwise.
    - If it exists, handle the response message as 200 ok response.
    - Then, transmit contents of file reads from the file and writes on the socket in case of GET or writes the file received in case of POST.
    - Wait for new requests (persistent connection)
    - Close if connection timed out

### In Client Code:

- Parse the commands.txt file to send the requests to the Server.
- Then, create a TCP connection with the server.
- While more operations exist do
    - Send next requests to the server
    - Receives data from the server in case of GET or sends data in case of POST

    - Parse the response message to get the data received in case of GET.
- end while
- Close the connection after finishing requests.

## Data Structure:

### In Server Code:

- Struct Request contains:
    - Vector request of strings that has the details of the Request received from the client.
    - Vector postData of strings that has the data received from the client in the post request to write it.

### In Client Code:

- Struct Request contains:
    - Vector request of strings that has the details of the Request received from the client.
    - Vector headers of strings to add headers of the request message to send to the server.

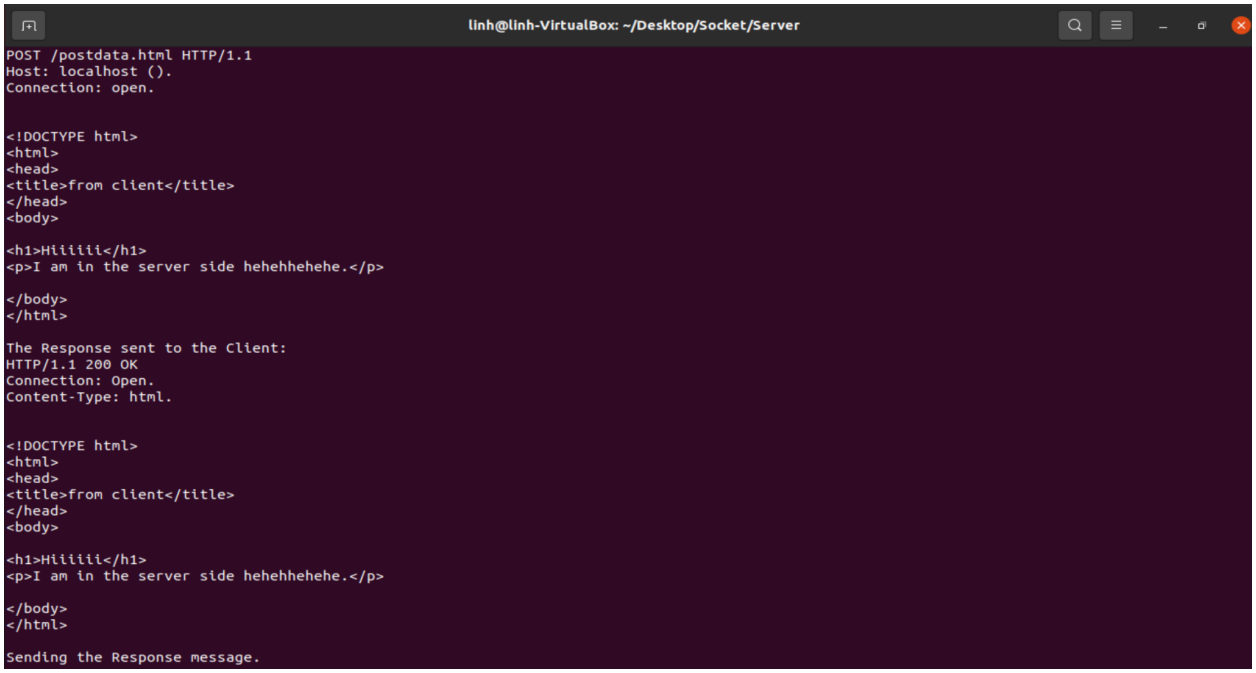
## Sample Runs:

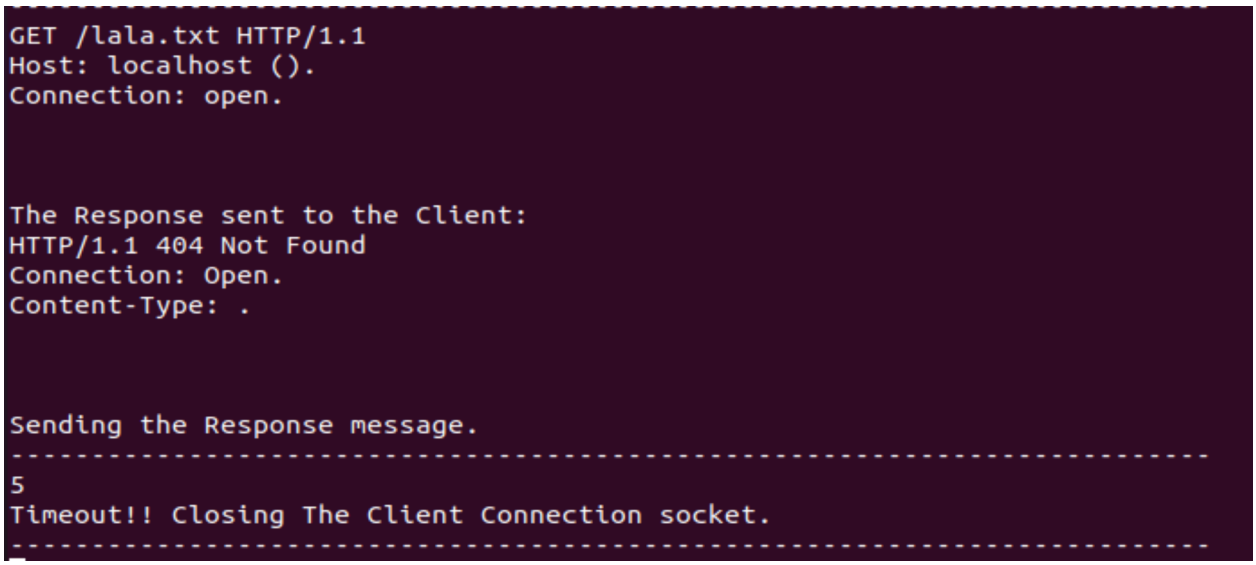
### Content of Commands.txt

```
client_get /hehe.txt localhost
client_post /postdata.html localhost
client_get /lala.txt localhost

```

### Server Terminal

```
linh@linh-VirtualBox:~/Desktop/Socket/Server$ g++ main.cpp -o exec -pthread
linh@linh-VirtualBox:~/Desktop/Socket/Server$ ./exec 2000
serverSocket: 3
exec Listening on port 2000
Accepting connection from client 127.0.0.1 and the new socket to communicate with it: 4
exec Listening on port 2000
GET /hehe.txt HTTP/1.1
Host: localhost ().
Connection: open.



The Response sent to the Client:
HTTP/1.1 200 OK
Connection: Open.
Content-Type: txt.


hehehehehehehehehehe
hahahaha
111111

2222222

Sending the Response message.
--------------------------------------------------------------------------
```


linh@linh-VirtualBox: ~/Desktop/Socket/Server
POST /postdata.html HTTP/1.1
Host: localhost ().
Connection: open.
<!DOCTYPE html>
<html>
<head>
<title>from client</title>
</head>
<body>
<h1>Hiiiiii</h1>
I am in the server side hehehhehehe.
</body>
</html>
The Response sent to the Client:
HTTP/1.1 200 OK
Connection: Open.
Content-Type: html.
<!DOCTYPE html>
<html>
<head>
<title>from client</title>
</head>
<body>
<h1>Hiiiiii</h1>
I am in the server side hehehhehehe.
</body>
</html>
Sending the Response message.


GET /lala.txt HTTP/1.1
Host: localhost ().
Connection: open.
The Response sent to the Client:
HTTP/1.1 404 Not Found
Connection: Open.
Content-Type: .
Sending the Response message.
5
Timeout!! Closing The Client Connection socket.

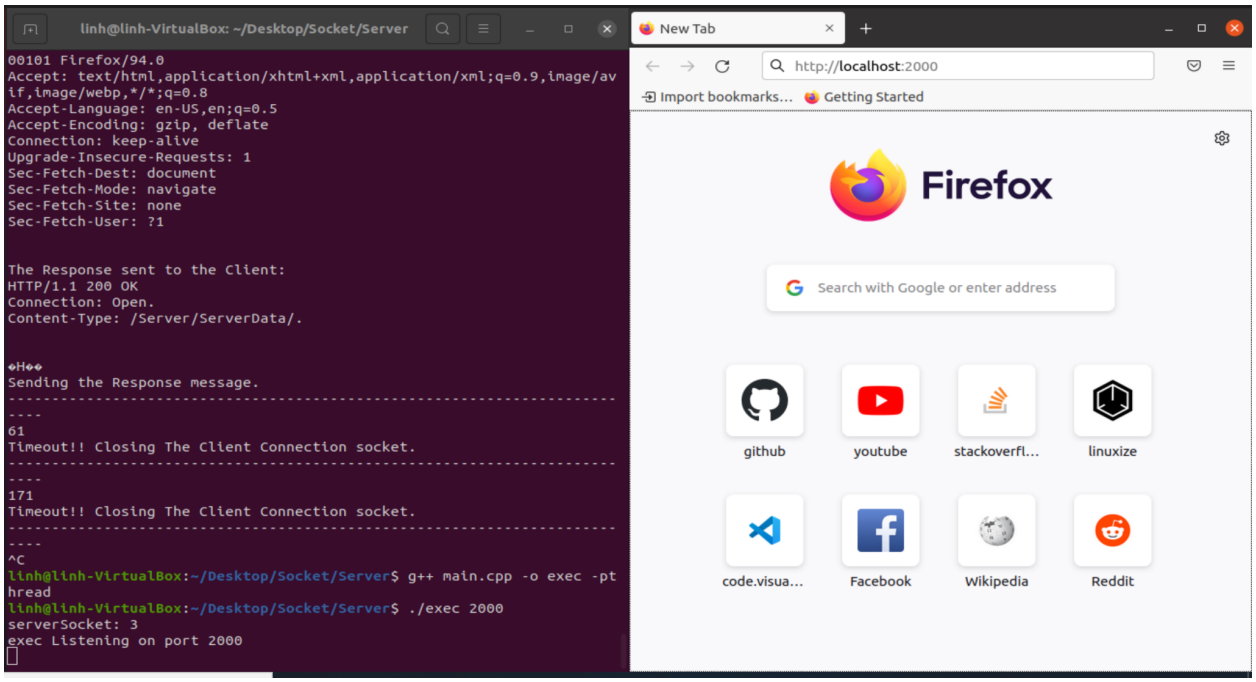
## Client Terminal

```
linh@linh-VirtualBox:~/Desktop/Socket/Client$ g++ main.cpp -o client -pthread
linh@linh-VirtualBox:~/Desktop/Socket/Client$ ./client localhost 2000
clientSocket: 3
Connection estabilished with server 127.0.0.1
--------------------------------------------------------------------------------
The Request message from client to the server.
GET /hehe.txt HTTP/1.1
Host: localhost ().
Connection: open.



Sending the Request message to the Server.
The Response recieved from the Server:
HTTP/1.1 200 OK
Connection: Open.
Content-Type: txt.


hehehehehehehehehehe
hahahaha
111111

2222222

--------------------------------------------------------------------------------
```

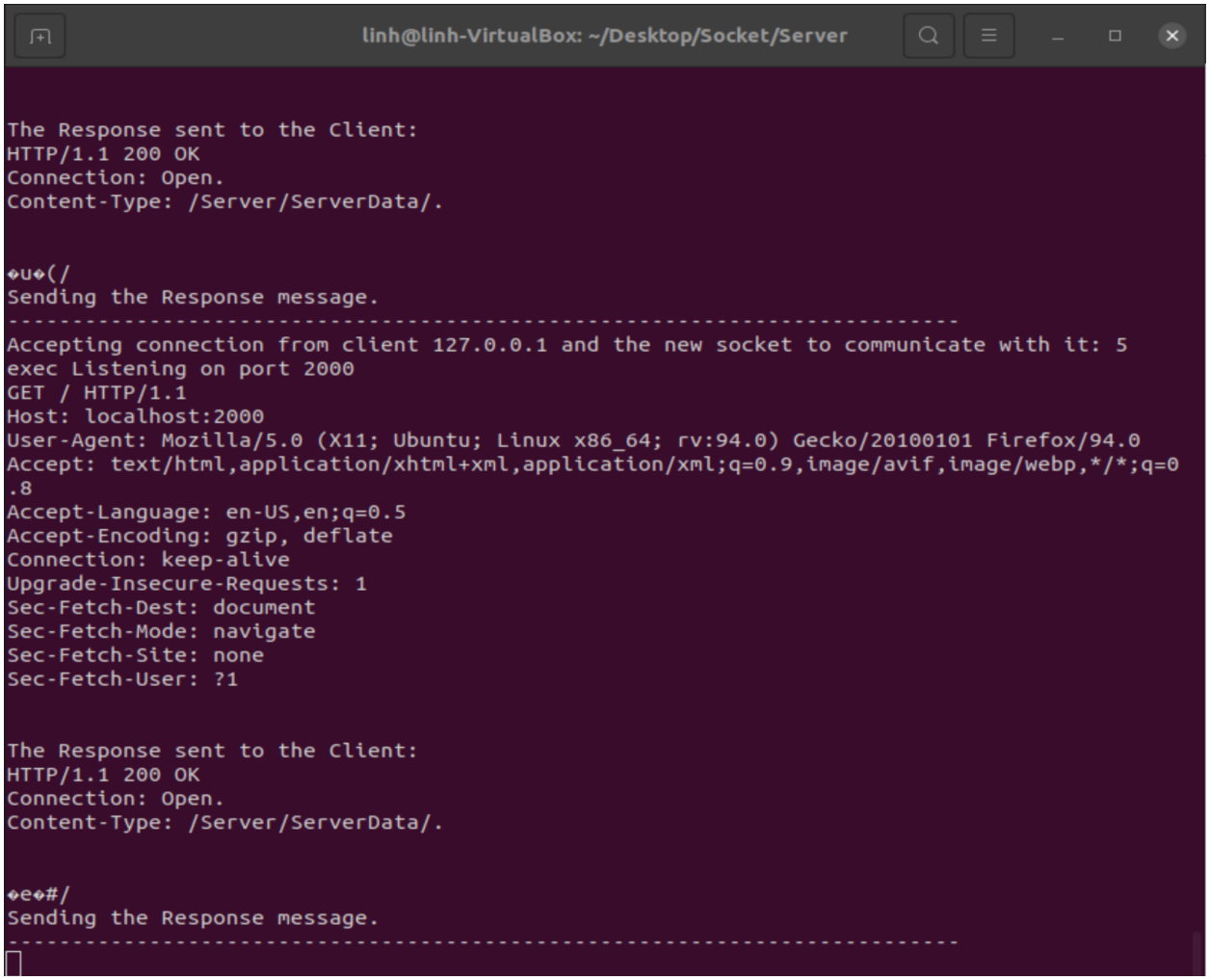
```
The Request message from client to the server.
POST /postdata.html HTTP/1.1
Host: localhost ().
Connection: open.


<!DOCTYPE html>
<html>
<head>
<title>from client</title>
</head>
<body>

<h1>Hiiiiii</h1>
<p>I am in the server side hehehhehehe.</p>

</body>
</html>

Sending the Request message to the Server.
The Response recieved from the Server:
HTTP/1.1 200 OK
Connection: Open.
Content-Type: html.


<!DOCTYPE html>
<html>
<head>
<title>from client</title>
</head>
<body>

<h1>Hiiiiii</h1>
<p>I am in the server side hehehhehehe.</p>

</body>
</html>
```

```
--------------------------------------------------------------------------------
The Request message from client to the server.
GET /lala.txt HTTP/1.1
Host: localhost ().
Connection: open.



Sending the Request message to the Server.
The Response recieved from the Server:
HTTP/1.1 404 Not Found
Connection: Open.
Content-Type: .



--------------------------------------------------------------------------------
linh@linh-VirtualBox:~/Desktop/Socket/Client$
```

## Bonus Testing my server with a real web browser http://localhost:2000



```
linh@linh-VirtualBox:~/Desktop/Socket/Server$ ./exec 2000
serverSocket: 3
exec Listening on port 2000
Accepting connection from client 127.0.0.1 and the new socket to commu
nicate with it: 4
exec Listening on port 2000
GET / HTTP/1.1
Host: localhost:2000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:94.0) Gecko/201
00101 Firefox/94.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/av
if,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


The Response sent to the Client:
HTTP/1.1 200 OK
Connection: Open.
Content-Type: /Server/ServerData/.


u(/
Sending the Response message.
```




## Code Source:

[Code]