

# HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

## Introduction

The HTTP i.e. Hypertext Transfer Protocol, the web's application layer protocol, has been defined in RFC 1945 and RFC 2616 and utilizes TCP as its underlying transport protocol. It defines the structure of messages exchanged between Web Clients and Web Servers as the formers requests the Web pages which basically consist of a base HTML file and several referenced objects from the latter. Comprising of two types of messages, HTTP request, sent by the client over TCP and HTTP response, sent by the server on receiving a HTTP request message from the client.

## Implementing HTTP Client and Server

The process of establishing a connection between a Web Client and Web server through HTTP is as follows: -

The client application / process initiates a TCP connection with the server which shall have a socket associated with it at the client and server side. With the connection established on receiving TCP segment from the server, the client acknowledges the same and sends an HTTP request message to the server.

The server on receiving the message through its socket retrieves the object requested (if it is GET command in the method field of the request) or stores the object in one of its directories (if command in the method field is PUT). The object, if retrieved, is then encapsulated in an HTTP response message and sent over to the client through the server side of the socket.

The client on receiving the response message from the server, extracts the file requested from it.

While HTTP Client and Server can be implemented using various programming languages, we have implemented the same using the C Language. The codes for both Client and Server have been written, implemented and tested on the Desktop version of the Linux Operating System's 64-bit Ubuntu 17.10.1.

Having completed writing the codes for both client and server in accordance with the requirements of the project and the useful tips provided for the same, the output, when the Server program was compiled and executed over the Linux terminal followed by the client, for the project requirements is as follows: -

# HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

1. As per the guidance provided on tackling the project we created the client program and attempted to connect it to “google.com” through the GET command from the command line. The request was successful and the server sent “200 OK” status message indicating that the request was successful. The headers and body of the message follow the status message as per the response format of HTTP

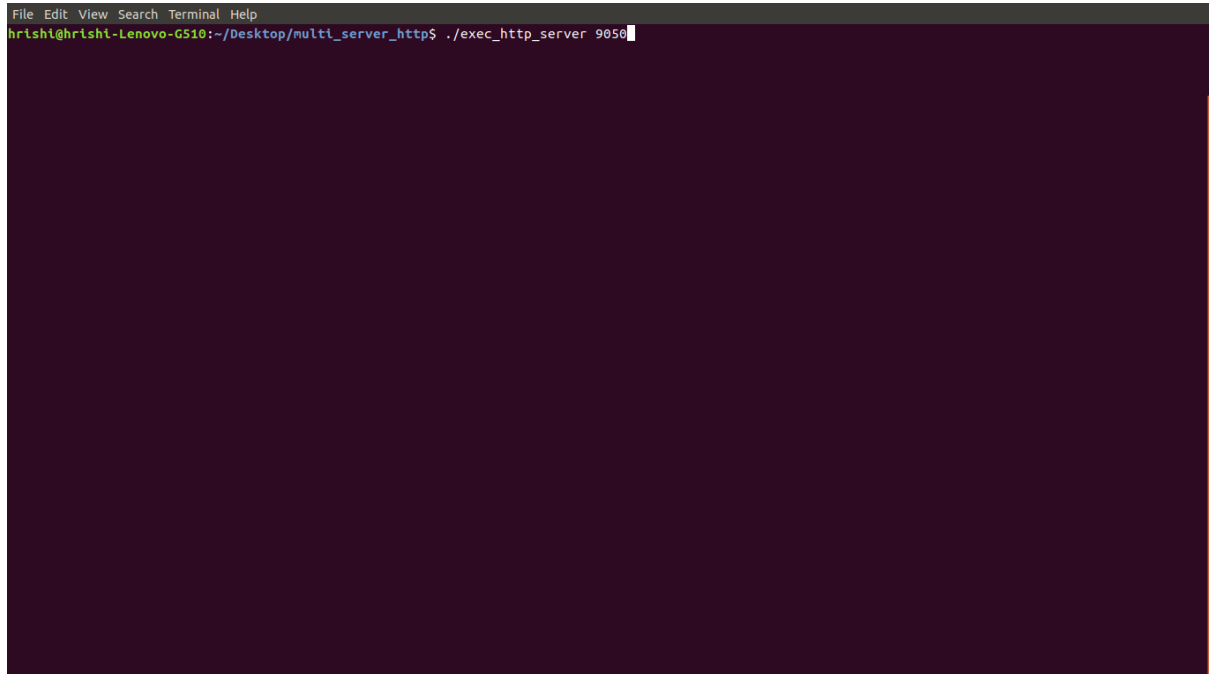
```
File Edit View Search Terminal Help
hrishighrishi-Lenovo-G510:~/Desktop/safe_quit/client_http$ ./exec_http_client www.google.com 80 GET /index.html
hostname :- www.google.com
port :- 80
command :- GET
file_path :- /index.html
Response from the server: HTTP/1.1 200 OK
Date: Sat, 10 Feb 2018 20:32:49 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See g.co/p3phelp for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: 1P_JAR=2018-02-10-20; expires=Mon, 12-Mar-2018 20:32:49 GMT; path=/; domain=.google.com
Set-Cookie: NID=123=bVMKFM3h2bLm5tWV7ZH5bo-nLYsycOBHaQdtuIREu8vS3C8sXkf13lyrULDP0U5SloFZ-aprb4dc1UfRmrqEQ0n0gx0sA1XaXkYs8CsZNNf2awd34jrr0Tbsv0ge30i3B; expires=Sun, 12-Aug-2018 20:32:49 GMT; path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
Transfer-Encoding: chunked

7548
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en"><head><meta content="Search the world's information, including webpages, images, videos and more. Google has many special features to help you find exactly what you're looking for." name="description"><meta content="noopen" names="robots"><meta content="text/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/logos/doodles/2018/2018-doodle-snow-games-day-4-6-210713008734208-1.png" itemprop="image"><meta content="Day 2 of the Doodle Snow Games!" property="twitter:title"><meta content="Day 2 of the Doodle Snow Games! &#10052; #GoogleDoodle" property="twitter:description"><meta content="Day 2 of the Doodle Snow Games! &#10052; #GoogleDoodle" property="twitter:description"><meta content="Day 2 of the Doodle Snow Games! &#10052; #GoogleDoodle" property="twitter:description">
hrishighrishi-Lenovo-G510:~/Desktop/safe_quit/client_http$
```

## HTTP Client and Server Project

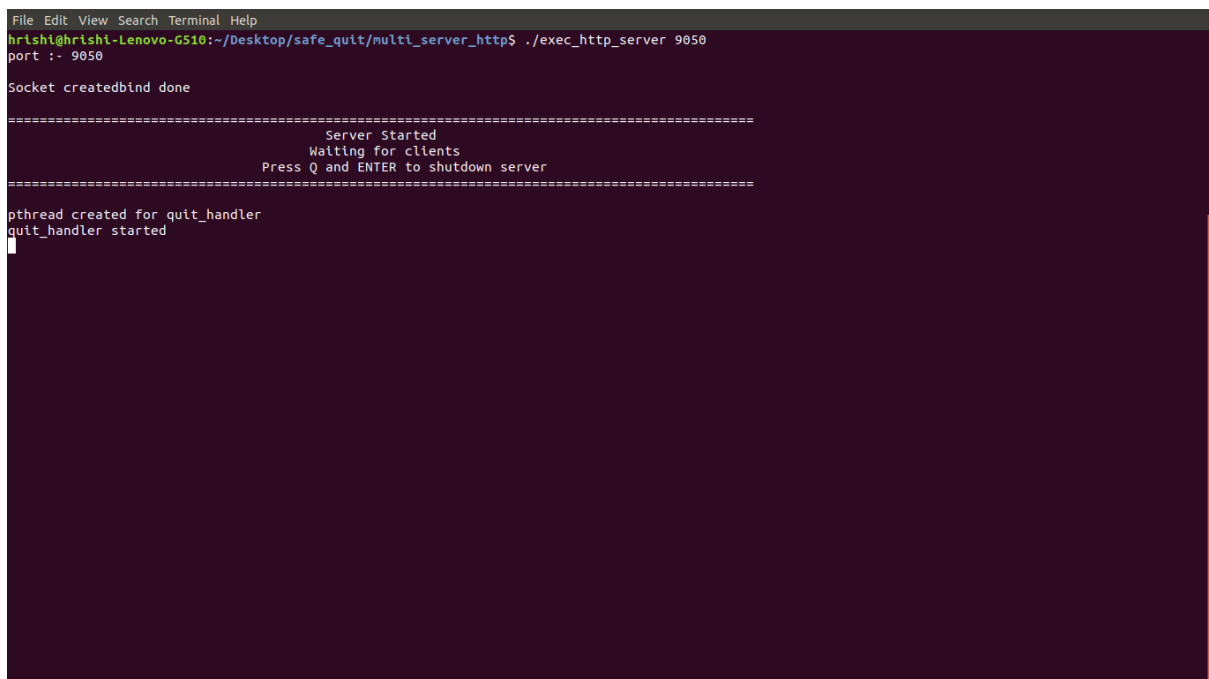
BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

- Below, we have attempted to start the server by executing the program for the same. As is required by the project, executable is followed by the port number on which the server shall listen for connections.

A terminal window with a dark purple background and a light green title bar. The title bar contains the text 'File Edit View Search Terminal Help'. The terminal shows a command prompt 'hrishi@hrishi-Lenovo-G510:~/Desktop/multi\_server\_http\$' followed by the command './exec\_http\_server 9050'. The cursor is at the end of the command.

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/multi_server_http$ ./exec_http_server 9050
```

- The server has successfully started and a socket has been created for it to receive and respond to requests from clients.

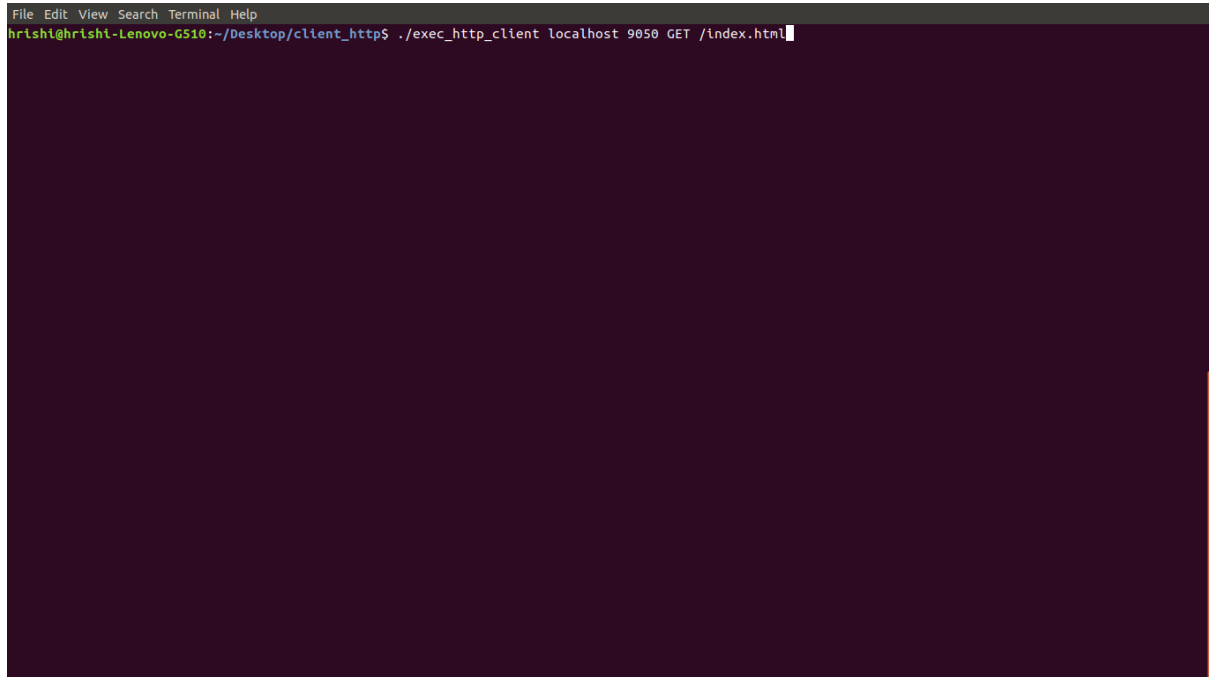
A terminal window with a dark purple background and a light green title bar. The title bar contains the text 'File Edit View Search Terminal Help'. The terminal shows the command './exec\_http\_server 9050' being executed. The output includes 'port :- 9050', 'Socket createdbind done', a separator line of equals signs, 'Server Started', 'Waiting for clients', 'Press Q and ENTER to shutdown server', another separator line of equals signs, 'pthread created for quit\_handler', and 'quit\_handler started'. The cursor is at the end of the last line.

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/safe_quit/multi_server_http$ ./exec_http_server 9050
port :- 9050
Socket createdbind done
=====
                Server Started
                Waiting for clients
                Press Q and ENTER to shutdown server
=====
pthread created for quit_handler
quit_handler started
```

## HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

4. Now, we have attempted to execute the client program and implement the GET command of HTTP. As per project prerequisites, the name of the executable is followed by the hostname, the port number on which it should connect to the server on, command name i.e. GET in this case and the name of the file which is requested from the server.

A terminal window with a dark purple background and a light green title bar. The title bar contains the text "File Edit View Search Terminal Help". The terminal shows the command prompt "hrishi@hrishi-Lenovo-G510:~/Desktop/client\_http\$" followed by the command "./exec\_http\_client localhost 9050 GET /index.html". The cursor is at the end of the command line.

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$ ./exec_http_client localhost 9050 GET /index.html
```

# HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

5. The server has received the GET command from the client and sends the requested file. The client thereafter closes the socket connection with the server.

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/multi_server_http$ ./exec_http_server 9050
port :- 9050
Socket created
bind done
Server Started. Waiting for clients to connect
Client connection accepted
Client Service Thread created

Received GET command
./index.html
Client disconnected

=====Thread returning =====
```

6. The file was received by the client and its contents displayed over the command line along with the message of HTTP/1.1 200 OK from the server, indicating that the request was successful.

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$ ./exec_http_client localhost 9050 GET /index.html
hostname :- localhost
port :- 9050
command :- GET
file_path :- /index.html
Response from the server: HTTP/1.1 200 OK

<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>
    <h1>This is a Heading</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

```
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$
```

# HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

7. Further testing on the client and server programs was implemented by giving the wrong command.

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$ ./exec_http_client localhost 9050 GETP /index.html
```

8. With GETP being given as command instead of GET, a message of “Invalid Command” flashes over the command line on the server side. The same message is sent over to the client with the status message of “400 Bad Request” which indicates prevalence of a generic error code and that the server doesn’t understand the request.

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/multi_server_http$ ./exec_http_server 9050
port :- 9050
Socket created
bind done
Server Started. Waiting for clients to connect
Client connection accepted
Client Service Thread created

Invalid Command
Client disconnected

=====Thread returning =====
```

# HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$ ./exec_http_client localhost 9050 GETP /index.html
hostname :- localhost
port :- 9050
command :- GETP
file_path :- /index.html
Response from the server: HTTP/1.1 400 Bad Request

Invalid Command
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$
```

9. Further, we tested the Client and Server by requesting for a wrong file or a file that the server does not have in its directory.

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$ ./exec_http_client localhost 9050 GET /unknown.html
```

## HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

10. The server in such case, on looking through its directory, displays the message of “No such file” on the command line and closes the connection. The same message is sent over to the client with the status message of “404 Not Found” indicates that the requested object is not present on the server.

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/multi_server_http$ ./exec_http_server 9050
port :- 9050
Socket created
bind done
Server Started, Waiting for clients to connect
client connection accepted
Client Service Thread created

Received GET command
./unknown.html
No such file

Client disconnected

=====Thread returning =====
```

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$ ./exec_http_client localhost 9050 GET /unknown.html
hostname :- localhost
port :- 9050
command :- GET
file_path :- /unknown.html
Response from the server: HTTP/1.1 404 Not Found

No such File

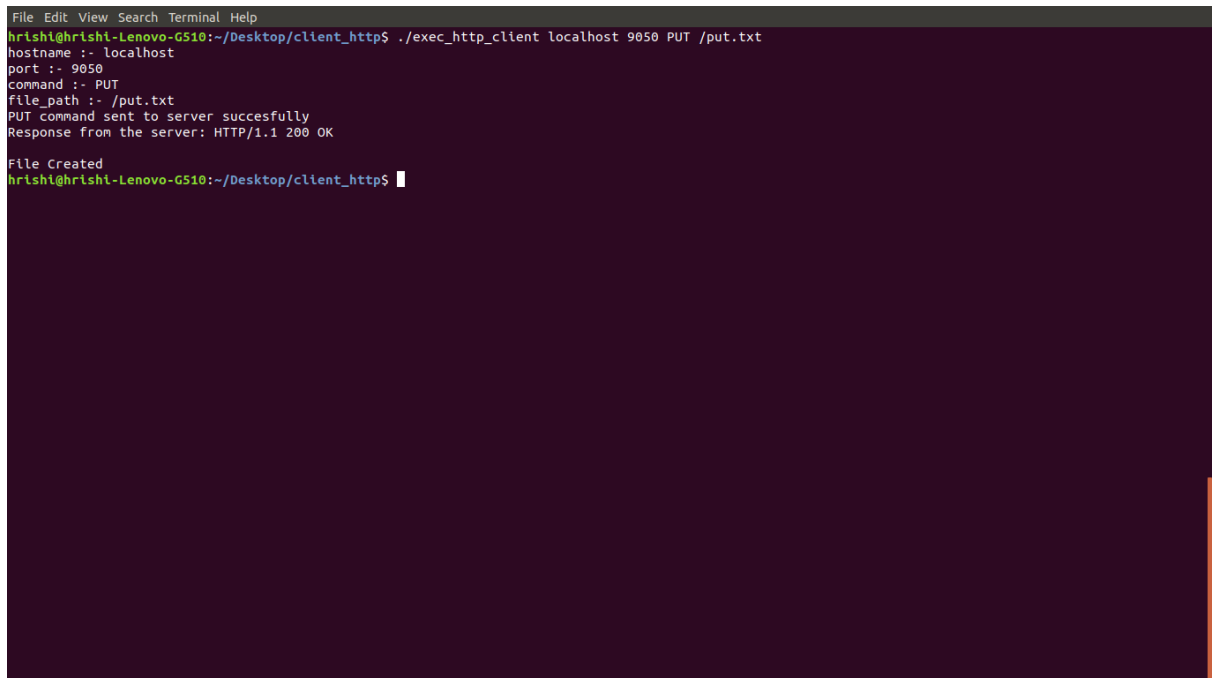
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$
```



## HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

11. Then, we attempted to execute the PUT command by sending an HTTP request to the server which took the command line parameters of the hostname followed by the port number on which to send the request, the PUT command and the file which we want to send to the server and wish for it to be saved locally in one of its directories. On execution, we noticed the command was successfully sent to the server along with the file which responded back with the status message of “200 OK” which implied that the request had succeeded.



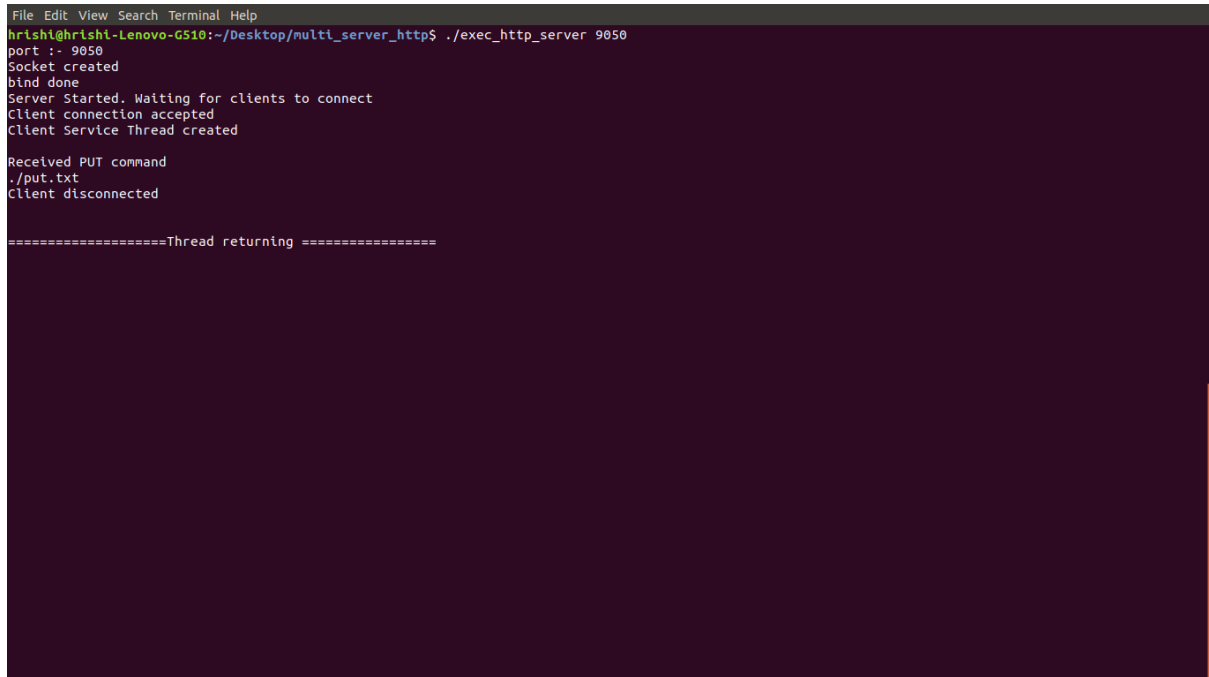
```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$ ./exec_http_client localhost 9050 PUT /put.txt
hostname :- localhost
port :- 9050
command :- PUT
file_path :- /put.txt
PUT command sent to server succesfully
Response from the server: HTTP/1.1 200 OK

File Created
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$
```

## HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

12. On the server side of the terminal, we noticed that it had successfully received the PUT command along with the file which was saved locally in its directory. The client was then disconnected.

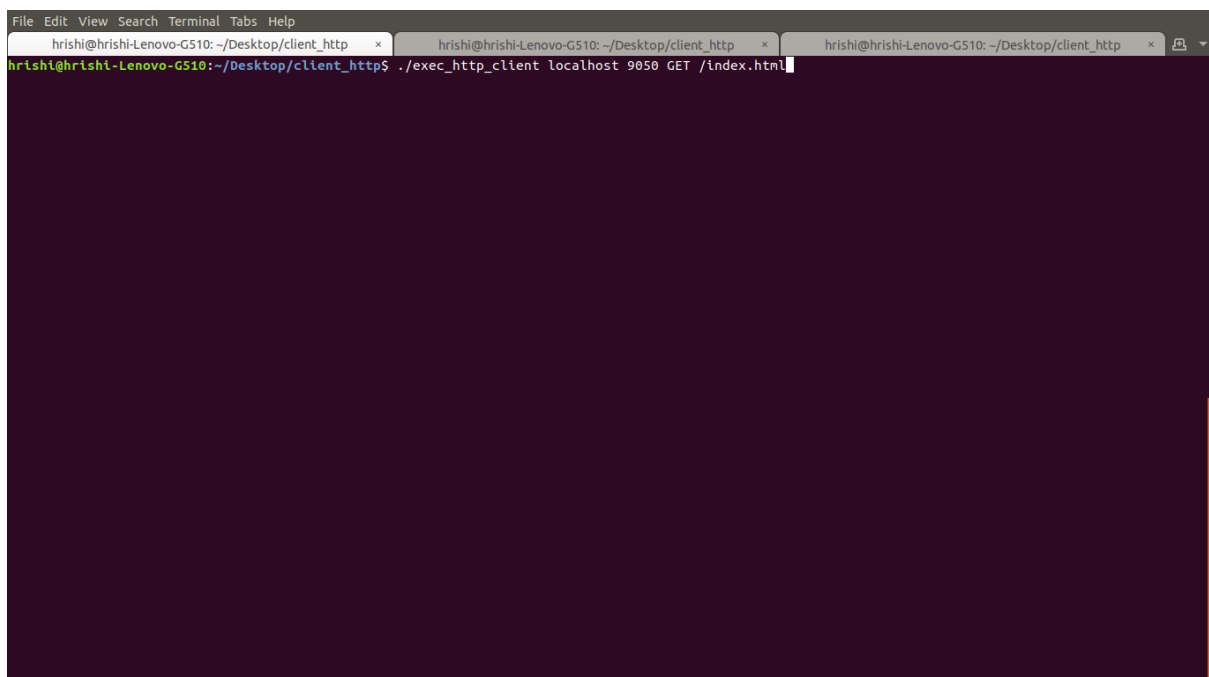
A terminal window with a dark purple background and green text. The window title is "File Edit View Search Terminal Help". The prompt is "hrishi@hrishi-Lenovo-G510: ~/Desktop/multi\_server\_http\$". The command entered is "./exec\_http\_server 9050". The output shows the server starting on port 9050, creating a socket, binding, and waiting for clients. It then receives a PUT command for a file named ".put.txt", saves it, and disconnects the client. A separator line "=====Thread returning =====" is shown at the bottom.

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510: ~/Desktop/multi_server_http$ ./exec_http_server 9050
port :- 9050
Socket created
bind done
Server Started. Waiting for clients to connect
Client connection accepted
Client Service Thread created

Received PUT command
./put.txt
Client disconnected

=====Thread returning =====
```

13. Further, to test the multithreaded capability of our server, we ran three instances of the client almost simultaneously on different terminal windows and sent HTTP request messages of GET command to the server.

A terminal window with a dark purple background and green text. The window title is "File Edit View Search Terminal Tabs Help". There are three tabs open, all titled "hrishi@hrishi-Lenovo-G510: ~/Desktop/client\_http". The prompt is "hrishi@hrishi-Lenovo-G510: ~/Desktop/client\_http\$". The command entered is "./exec\_http\_client localhost 9050 GET /index.html".

```
File Edit View Search Terminal Tabs Help
hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x
hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http$ ./exec_http_client localhost 9050 GET /index.html
```

## HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

14. The server responded to the three clients by sending the files they requested through their GET commands on their terminals. The connections with the clients was closed thereafter.

```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/multi_server_http$ ./exec_http_server 9050
port :- 9050
Socket created
bind done
Server Started. Waiting for clients to connect
Client connection accepted
Client Service Thread created

Received GET command
./index.html
Client disconnected

=====Thread returning =====
Client connection accepted
Client Service Thread created

Received GET command
./index.html
Client disconnected

=====Thread returning =====
Client connection accepted
Client Service Thread created

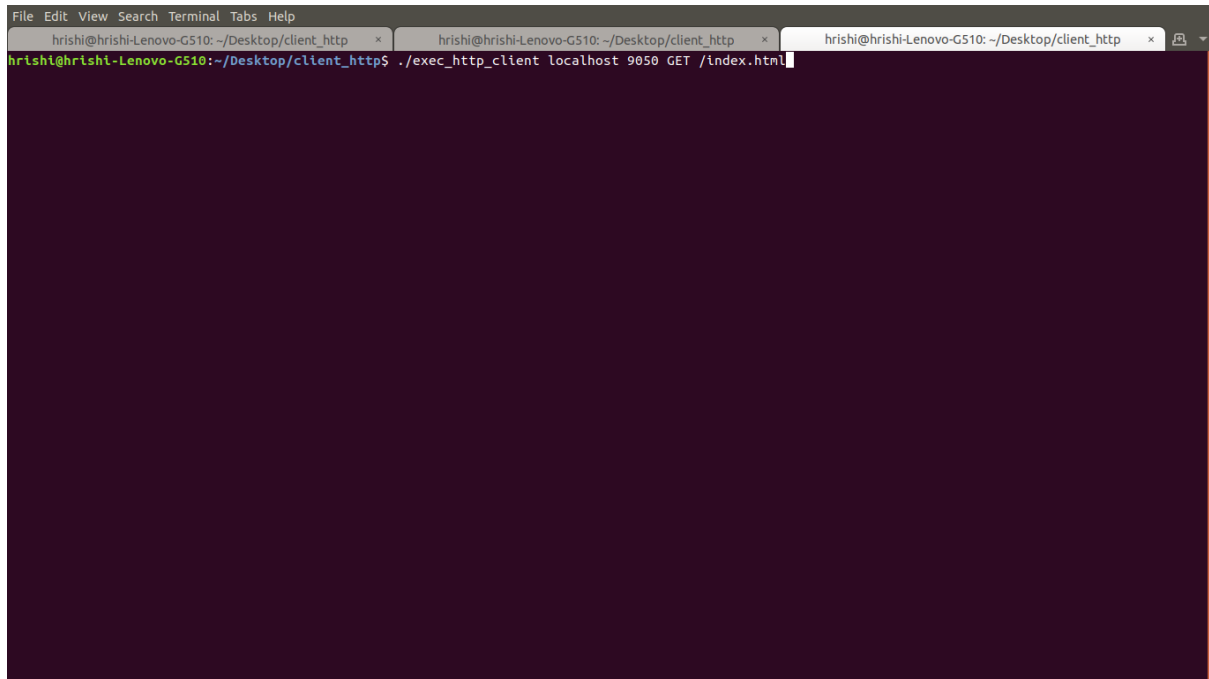
Received GET command
./index.html
Client disconnected

=====Thread returning =====
```

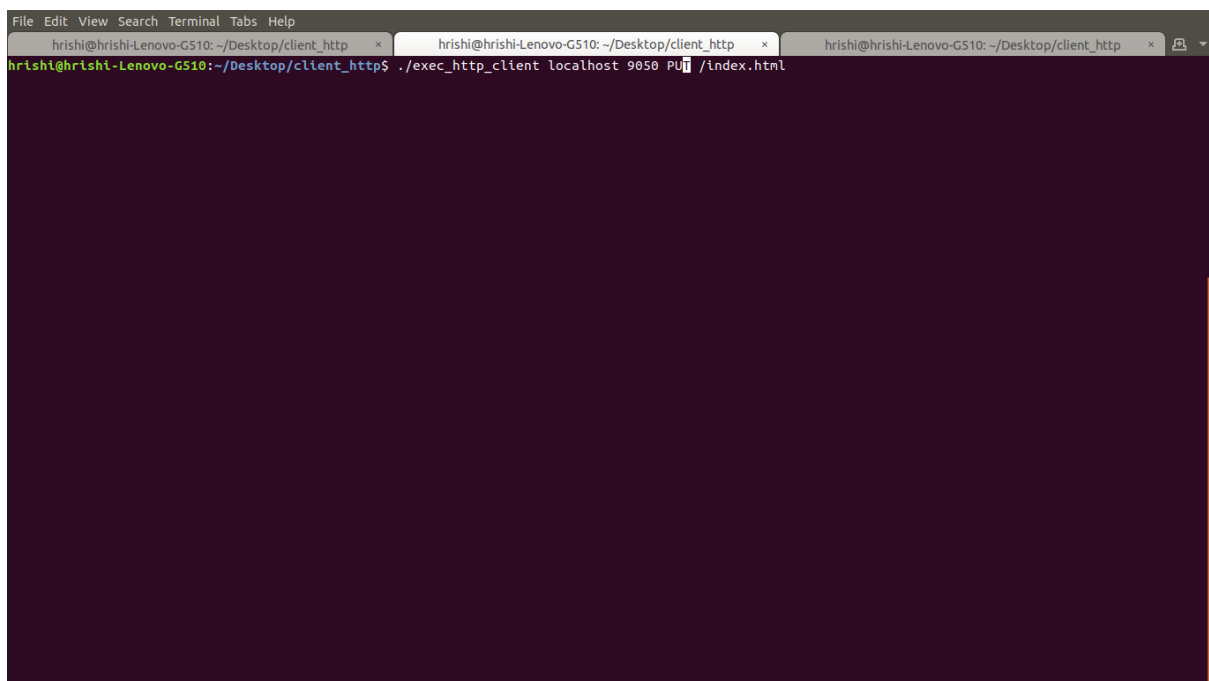
## HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

15. Finally, to test our server's ability to serve different clients with different commands, we again ran three instances of client. While one instance sent an HTTP request using the GET command, the other send an HTTP request using the PUT command. The last instance gave an invalid command to the server.



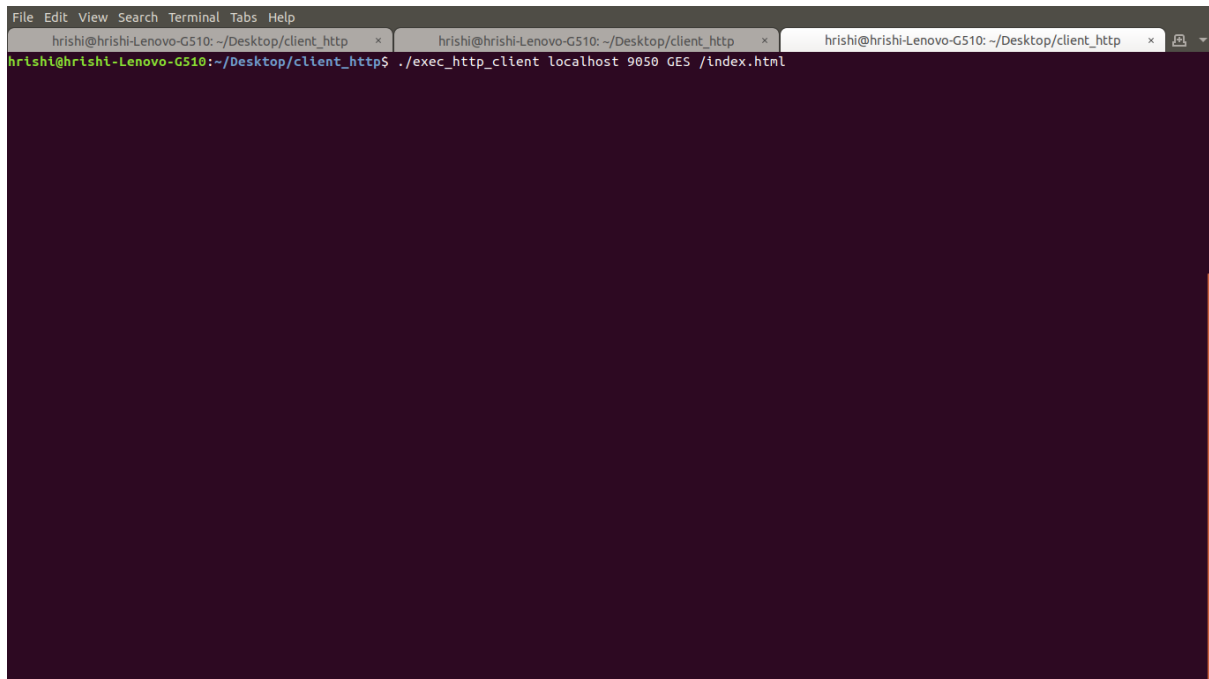
```
File Edit View Search Terminal Tabs Help
hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$ ./exec_http_client localhost 9050 GET /index.html
```



```
File Edit View Search Terminal Tabs Help
hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$ ./exec_http_client localhost 9050 PUT /index.html
```

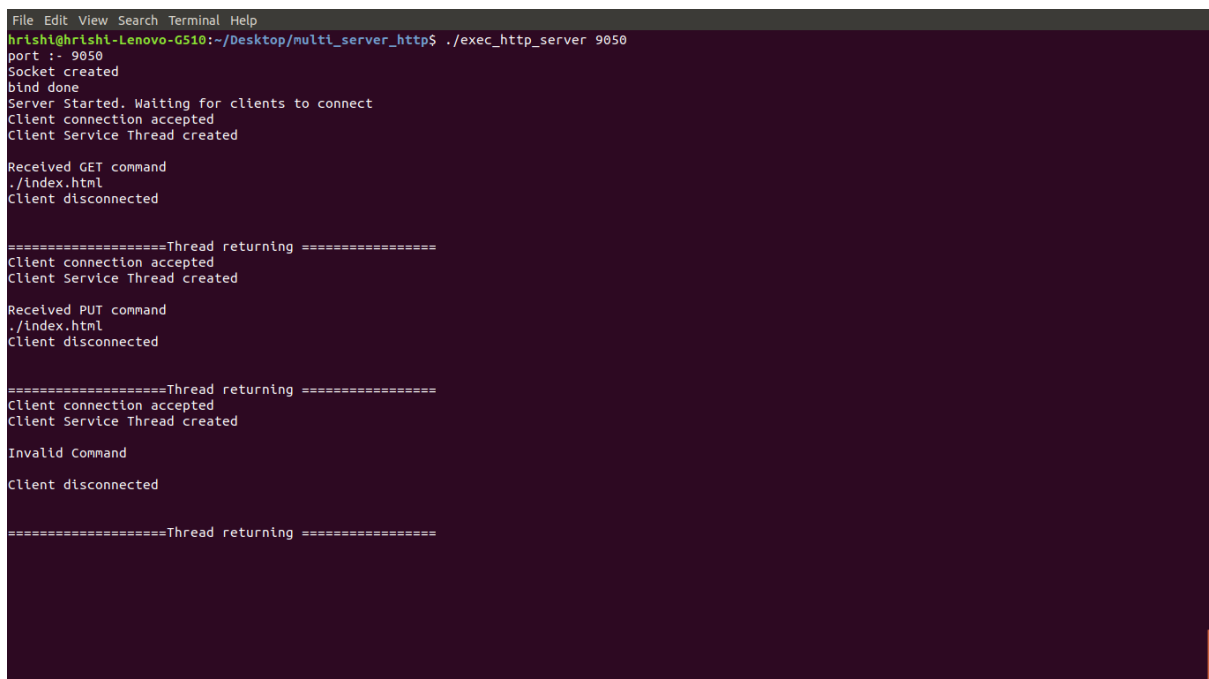
# HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)



```
File Edit View Search Terminal Tabs Help
hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x
hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x
hrishi@hrishi-Lenovo-G510: ~/Desktop/client_http x
hrishi@hrishi-Lenovo-G510:~/Desktop/client_http$ ./exec_http_client localhost 9050 GET /index.html
```

16. The server implemented multithreading neatly as the client sending an HTTP request with GET was sent the appropriate file. The PUT command too was successfully executed and the file sent by the client was saved by the server in its local directory. In regard of the client sending an invalid command through its request, the server displayed the message of Invalid command.



```
File Edit View Search Terminal Help
hrishi@hrishi-Lenovo-G510:~/Desktop/multi_server_http$ ./exec_http_server 9050
port :- 9050
Socket created
bind done
Server Started. Waiting for clients to connect
Client connection accepted
Client Service Thread created

Received GET command
./index.html
Client disconnected

=====Thread returning =====
Client connection accepted
Client Service Thread created

Received PUT command
./index.html
Client disconnected

=====Thread returning =====
Client connection accepted
Client Service Thread created

Invalid Command

Client disconnected

=====Thread returning =====
```

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

17. To further test the multithreading capability of our server, we developed a client program which sends request continuously and ran two instances of the same, one instance sending HTTP requests with GET command and the other sending HTTP requests with PUT command.

```
File Edit View Search Terminal Help
hritshi@hritshi-Lenovo-G510:~/Desktop/safe_quit/client_http$ ./exec_multithreaded_http_client localhost 9050 GET /index.html
hostname :- localhost
port :- 9050
command :- GET
file_path :- /index.html

pthread created for client
client started
Response from the server: HTTP/1.1 200 OK

i<!DOCTYPE html>
<html>
  <head>
    <title>PUT Data</title>
  </head>
  <body>
    <h1>index.html</h1>
    <p>This is the file sent in response to PUT request from client.</p>
  </body>
</html>

Response from the server: HTTP/1.1 200 OK

i<!DOCTYPE html>
<html>
  <head>
    <title>PUT Data</title>
  </head>
  <body>
    <h1>index.html</h1>
    <p>This is the file sent in response to PUT request from client.</p>
  </body>
</html>
```

[illegible]

## HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

18. The server, as expected, serves requests from the two instances of multithreaded client program by responding appropriately to GET and PUT requests sent through HTTP requests.

```
File Edit View Search Terminal Help
hrishik@hrishi-Lenovo-G510:~/Desktop/safe_quit/multi_server_http$ ./exec_http_server 9050
port :- 9050

Socket createdbind done

=====
                        Server Started
                        Waiting for clients
                        Press Q and ENTER to shutdown server
=====

pthread created for quit_handler
quit_handler started
Client connection accepted
Client Service Thread created

Received PUT command
./put.txt
Received PUT command
./put.txt
Received PUT command
./put.txt
Client connection accepted
Client Service Thread created

Received GET command
./index.html
Received PUT command
./put.txt
Received GET command
./index.html
Received PUT command
./put.txt
Received GET command
./index.html
Received PUT command
./put.txt
Received GET command
./index.html
Received PUT command
./put.txt
```

# HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

19. Finally, to test if the server quits and shuts down from command line, we once again ran the multithreaded client program sending PUT command through its HTTP requests, sending command for server to shut-down after receiving a few requests. The server, as expected shuts down when all threads running are closed.

Note: Since the client implemented is a multithreaded client running in infinite “while (1)” loop, the command to quit i.e. “Q” has to be given to the multithreaded client as well.

```
File Edit View Search Terminal Help
hrishik@hrishi-Lenovo-G510:~/Desktop/safe_quit/multi_server_http$ ./exec_http_server 9050
port :- 9050

Socket createdbind done

=====
                        Server Started
                        Waiting for clients
                        Press Q and ENTER to shutdown server
=====

pthread created for quit_handler
quit_handler started
Client connection accepted
Client Service Thread created

Received PUT command
./put.txt
Received PUT command
./put.txt
Received PUT command
./put.txt
Received PUT command
./put.txt
Received PUT command
./put.txt
Received PUT command
Q
./put.txt
Received PUT command
./put.txt
=====Waiting for all threads to close=====
=====Waiting for all threads to close=====

Received PUT command
```

```
File Edit View Search Terminal Help

Received PUT command
./put.txt
=====Waiting for all threads to close=====
=====Waiting for all threads to close=====

Received PUT command
./put.txt
=====Waiting for all threads to close=====
=====Waiting for all threads to close=====

Received PUT command
./put.txt
=====Waiting for all threads to close=====
=====Waiting for all threads to close=====

Received PUT command
./put.txt
=====Waiting for all threads to close=====
=====Waiting for all threads to close=====

Received PUT command
./put.txt
=====Waiting for all threads to close=====
=====Waiting for all threads to close=====

Client disconnected

=====Thread returning =====
=====Waiting for all threads to close=====
=====All service threads closed=====
=====Server Shutting Down=====
hrishik@hrishi-Lenovo-G510:~/Desktop/safe_quit/multi_server_http$
```



## HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

### Instructions for Execution of Program

- 1) Heading to the directory where the server program is stored from the terminal of Linux, execute the make file for the server, to compile the program.  
`$ make`
- 2) Execute and start the server (http\_server.c) and give the port number for client to connect to. For our testing, we have used port number 9050.  
`$ ./exec_http_server 9050`
- 3) On a new terminal window, heading to the directory of client program, execute the make file to compile the client program.  
`$ make`
- 4) Execute and run the client (http\_client.c) and give the host name i.e. localhost in our case followed by the port number (9050), the command to be sent through the request (GET or PUT) and the file to be requested from the server i.e. "index.html" or sent to the server i.e. "put.txt".  
`$ ./exec_http_client localhost 9050 GET /index.html`  
`$ ./exec_http_client localhost 9050 PUT /put.txt`
- 5) The request being received by the server and the response sent to the client can be viewed over the terminal windows of server and client respectively.

Note: We wrote the program of multithreaded client to test our multithreaded server only and the former is not a project requirement. The code for the program however, has been provided within the HTTP client folder (client\_http) with the file name of "multithreaded\_http\_client.c". The make file for both the clients remains the same along with the parameters taken on command line at the time of execution.

## HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

### References: -

- 1) James F. Kurose and Keith W. Ross COMPUTER NETWORKING: A Top-Down Approach
- 2) Learn Socket Programming in C from scratch: <https://www.udemy.com/learn-socket-programming-in-c-from-scratch/learn/v4/content>

# HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

## Implementing HTTP Client and Server

While HTTP Client and Server can be implemented in various languages, we have implemented the same in C Language.

The server shall continuously listen for connections from clients on port number 9223 which has been given from the command screen along with its name i.e. "Http\_Server". On receiving the connection request on the port, the server shall connect to the client and receive the HTTP request sent from the client. The request message shall house the socket number of the client to which the server has to send its response message.

If the request consists of the GET method in its request line, the server shall retrieve the requested file namely "index.html" from its directory and send the same to the client through its socket.

If the request received from the client has PUT method in its request line, the server shall save the file received in its directory.

On the client side, TCP connection shall be established with the server when its name, port number, method to be executed and the name of the file are entered through the command line.

## HTTP Client and Server Project

BY: Avadhut Naik (801045233) and Hrishikesh Kasar (801053462)

In case of GET method being in the request line, the file requested shall be sent from the server and its contents displayed over the command line.

In case of PUT command, the file entered over the command line shall be sent to the server for it to save it in its directory.

In the instance of a wrong command given at command line, the server and client shall both display “Invalid command” over the command line.

In the instance of an unknown file inputted at command line for either GET or PUT commands, unknown file message shall be inputted over the Command Line of both server and client.

The server being multithreaded can serve more than one client at the same time. For the sake of testing, we have implemented and successfully accessed the server from three different clients.