**WEB CLIENT AND SERVER PROGRAM- SERVER PART**

**OVERVIEW:**  In this program we have designed a server that handles http request from the web client such as Chrome or Firefox. The function of the server is to fetch the file that that the client requests and return the corresponding output. First step is that we have to establish a connection with the web server and web client. So the server has to create a socket and bind the socket with the particular port number. Then the server starts listening at the particular port number waiting for the new connection from the client. Then by calling accept() function the connection is established between the client and the server. The server would search for the file requested and if found it will return it to the client. All the data is put in the send buffer and is sent to the client side.

--------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------

**MAIN**

In the main function of the program it receives 2 arguments from the users. First argument would take the Directory path where the file is stored in the local system. The Second one would be the port number through which the server is configured to listen for the new connection. Then creating the new socket by taking its family name, type of the of the data being send like SOCK\_STREAM or SOCK\_DGRAM. Then after that socket is created successfully. Input all the necessary corresponding values. In this procedure, client would call connect() function in order to establish a communication between the client and the server. After a successful establishment connection with the server. Client would request for a file that is present in the server. The GET request from the server is parsed to get the file name. Then the corresponding file is opened using the fopen() function by the sever and transfer all the contents from the file to the buffer and transfer the contents from buffer to the client. As it is a http server it will have a header part. So if the connection is successful then it would print "HTTP/1.1 200 OK" else the error is handled otherwise. Then send the body which contains the contents of the file that the client asked for along with the header. The first part of the message in http is considered to be header, which contains information like the Connection length, connection status etc. We have used the gettimeofday() function to get the values at 2 points in the program to record the time taken for the serve to fetch the file and send it the server.

--------------------------------------------------------------------------------------------------------------------------

--------------------------------------------------------------------------------------------------------------------------

**OUTPUT:**

Arguments to be given while executing the program.

**Positive case**

**Test Case 1:**

Server: My Server

Client: My Client.

**Command Line**

**Server: The server is at port 8080**

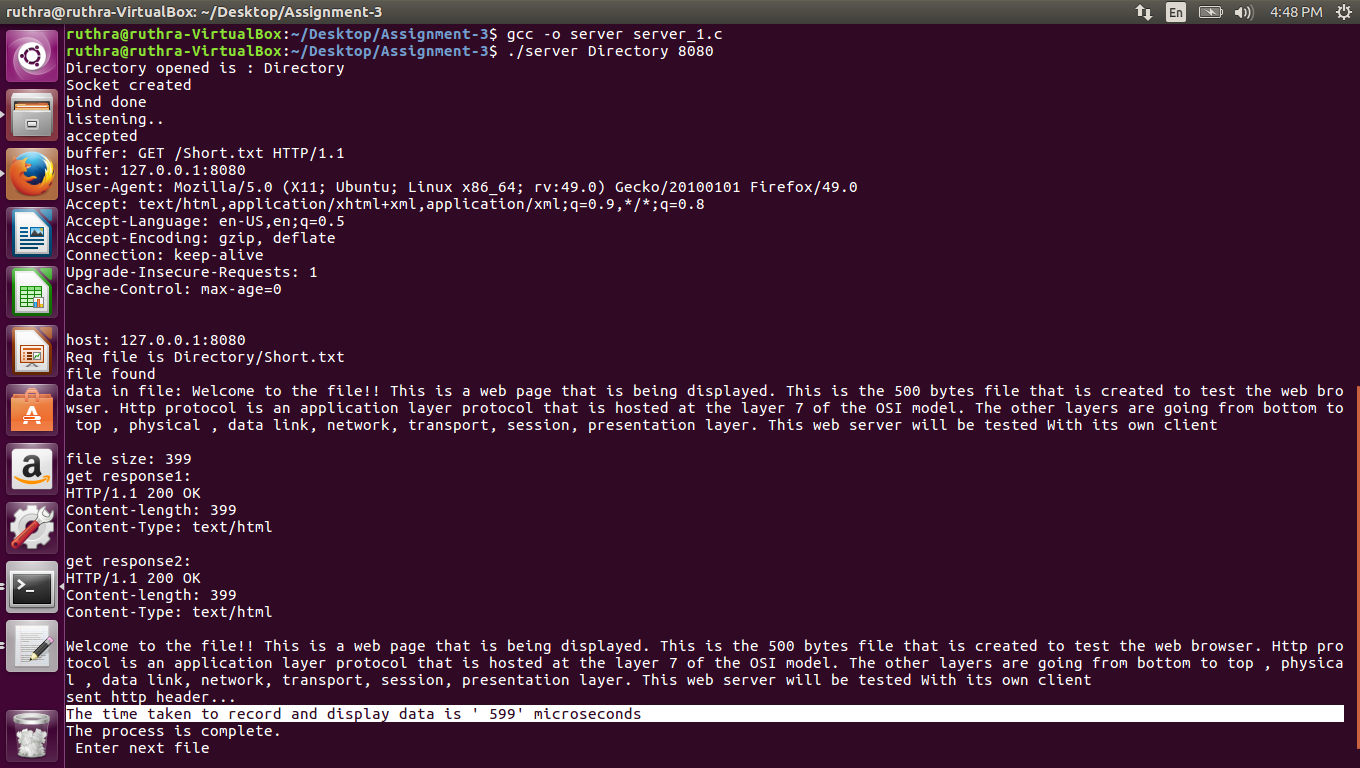
**gcc -o server server.c**

**./server Directory 8080**

1. This the output when we have localhost as a SERVER and designed program as CLIENT

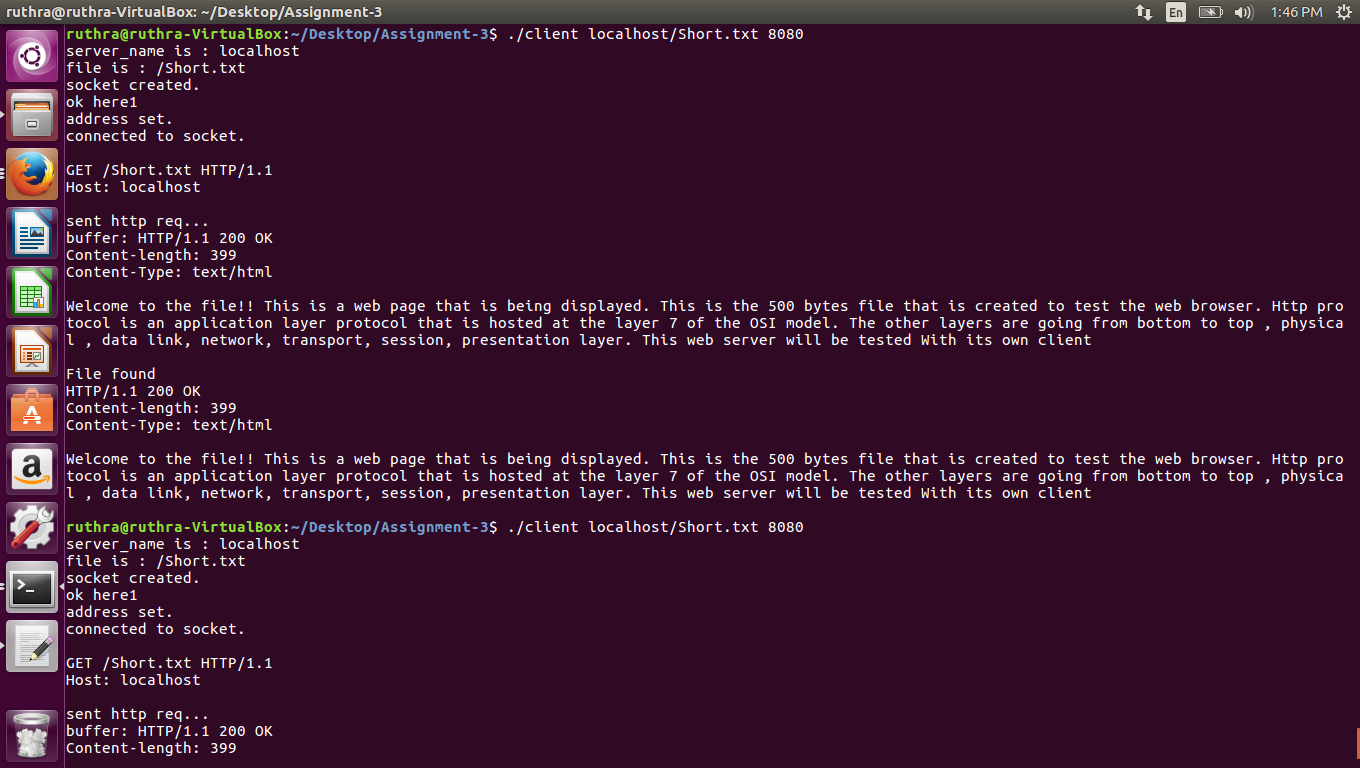
The server will also show the time taken to fetch the file. Please not that the file is present inside the path (**Home/Desktop/Assignment3/Directory**), but since we are running the command in the terminal from (**Home/Desktop/Assignment3**) we will mention only the “**Directory**” directory name in the command line. If the file is present inside (**Home/Desktop/Assignment3/Directory/Some\_directory**), then we should provide the directory name as “**Directory/Some\_directory**”

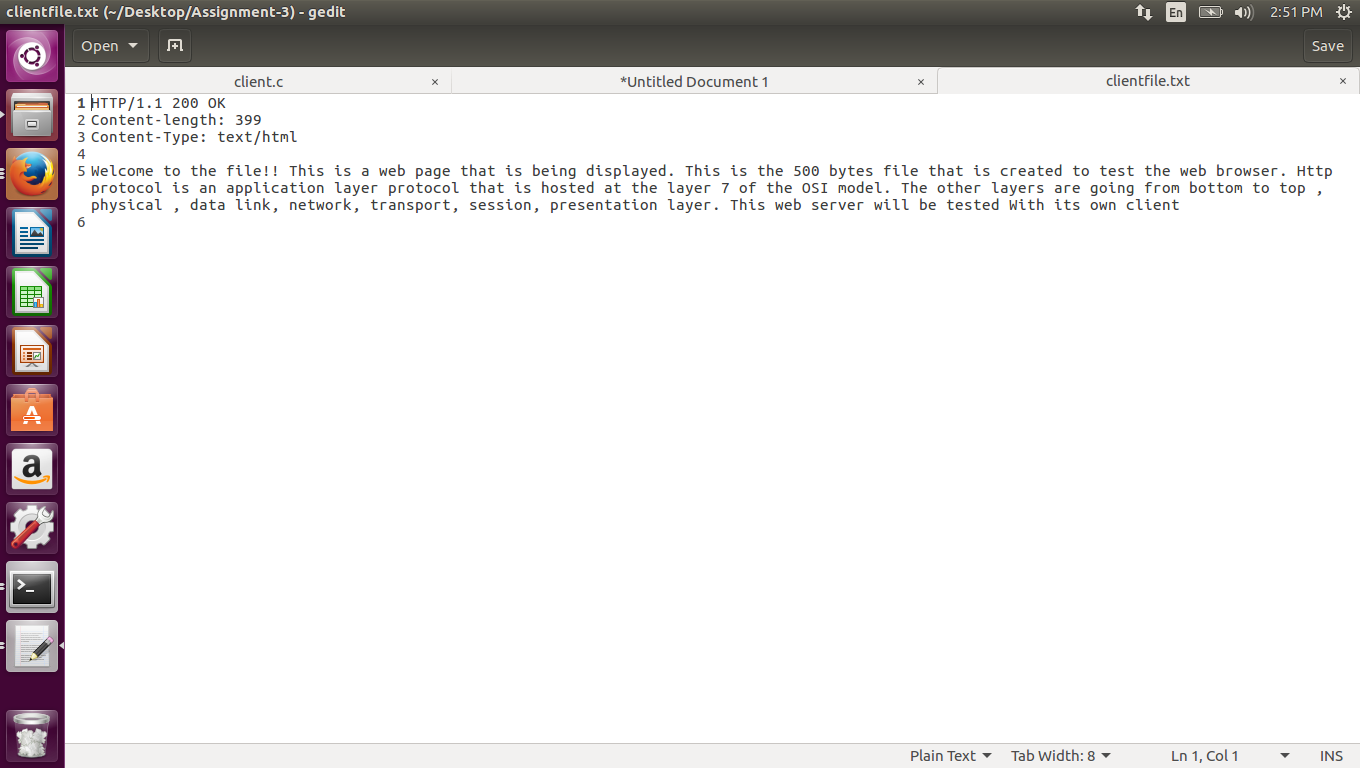
**Server:**

****

**Client:**

Command line output:





**Test Case 2:**

Server: My server (port number 8080)

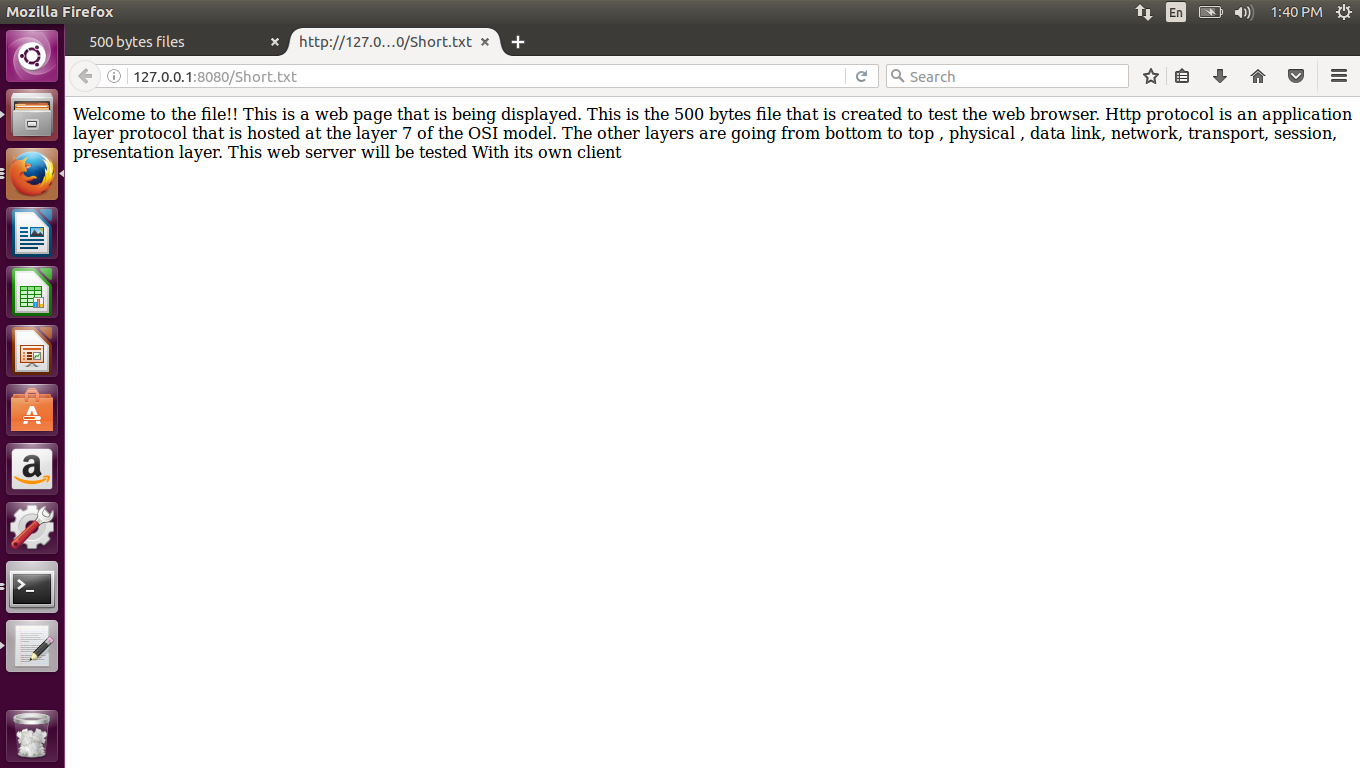
Client: Http client (we have used Firefox Browser)

Command to be input in the browser search line:

**Localhost:8080/Short.txt**

**Or**

**127.0.0.1:8080/Short.txt**



**Error Handling:**

The server gives 404-File Not Found error when the browser (client) asks for a file that is not with the server.

