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

**OVERVIEW:**  In this program we have designed a client that sends a http request to the web server requesting to fetch a file that is present in the server. 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. And then the client would request for the connection with the server. Then by calling accept() function the connection is established between the client and the server. Now the client would request for the specific file. The server would search for the file and if found it will return the file to the client along with the header.

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

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

**MAIN**

In the main function of the program it receives 2 arguments from the users. First argument would take the server name (eg., my.sjsu.edu) followed by a “/” and the name of the file to be fetched from the server. The Second one would be the port number through which the server is configured to listen for the new connection and the client wants to connect. Then using a for loop we would separate the IP- address and the file that has to be searched. 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 the socket is created successfully, Input all the necessary values to fill the structure. 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. As it is a http web client (HTTP 1.1) it has to have two parts -one is the header part which would contain all the information such as accept, connection - to keep-alive, content length etc. Then comes the body, where the actual content is being sent. So this information is passed onto the web server using GET function. In the GET function we would be sending only the file name that is required along with the header. Then the corresponding file is opened using the fopen() function by the sever and all the contents from the file are transferred to the buffer from which the contents are sent to the client.

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

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

**OUTPUT:**

Arguments to be given while executing the program.

**Test Case 1**

Server: My Server **(Port number 8080)**

Client: My client

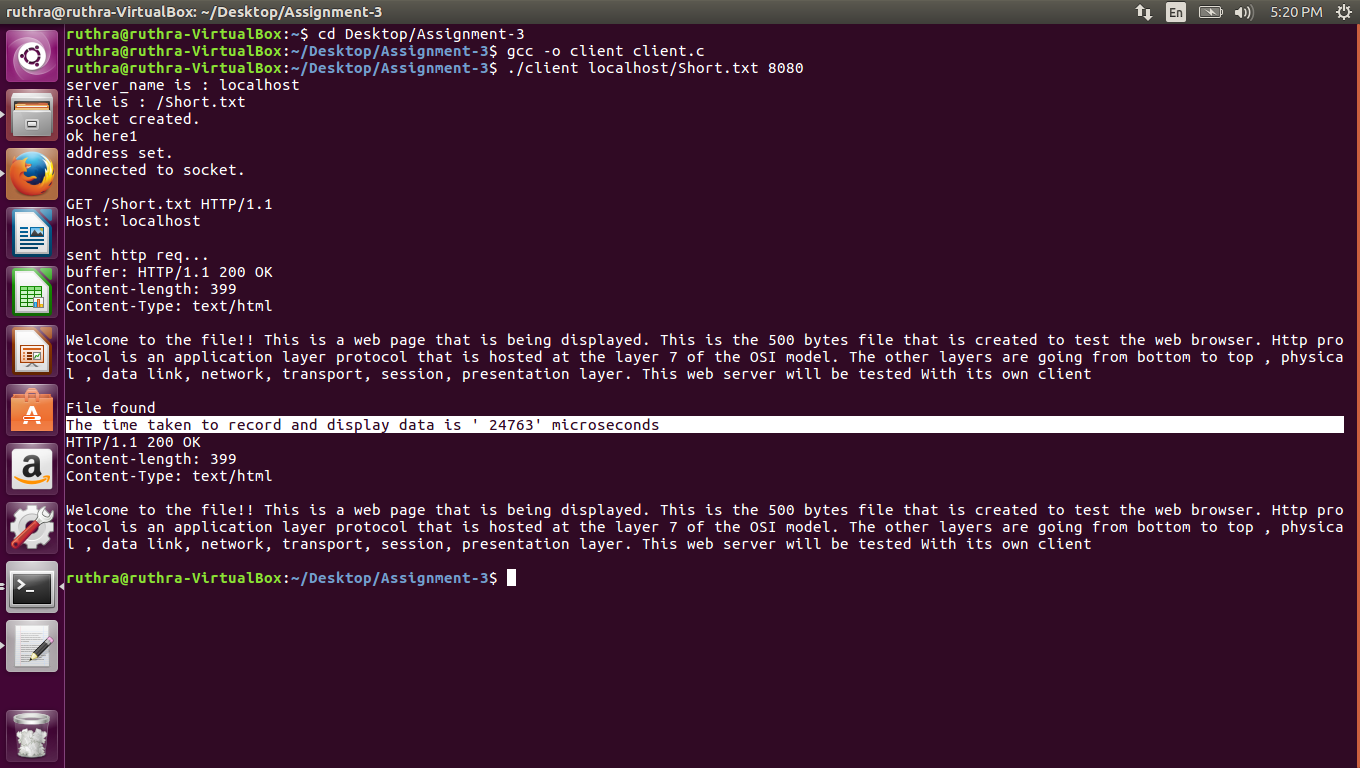
**Command Line on the server**

**gcc -o server server.c**

**./server Directory 8080**

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

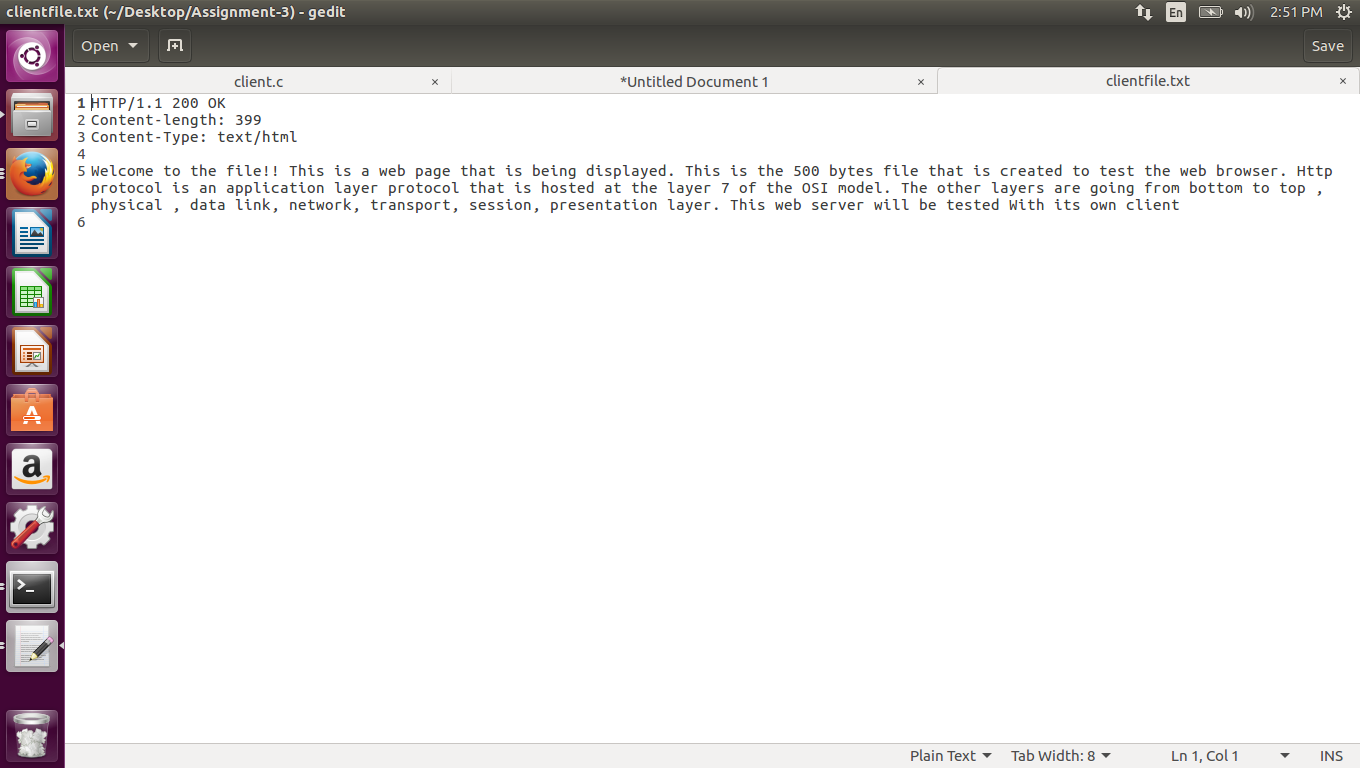
The client will be able to display the time taken to record and display data.



**Command Line on the client**

**gcc -o client client.c**

**./client localhost/Short.txt 8080**



**Test Case 2:**

Server: HTTP Web Server (name provided by the user)

Client: My client (**Port number 80)**

2. This is the output when we have localhost as CLIENT as well as a SERVER

**Command Line on the client**

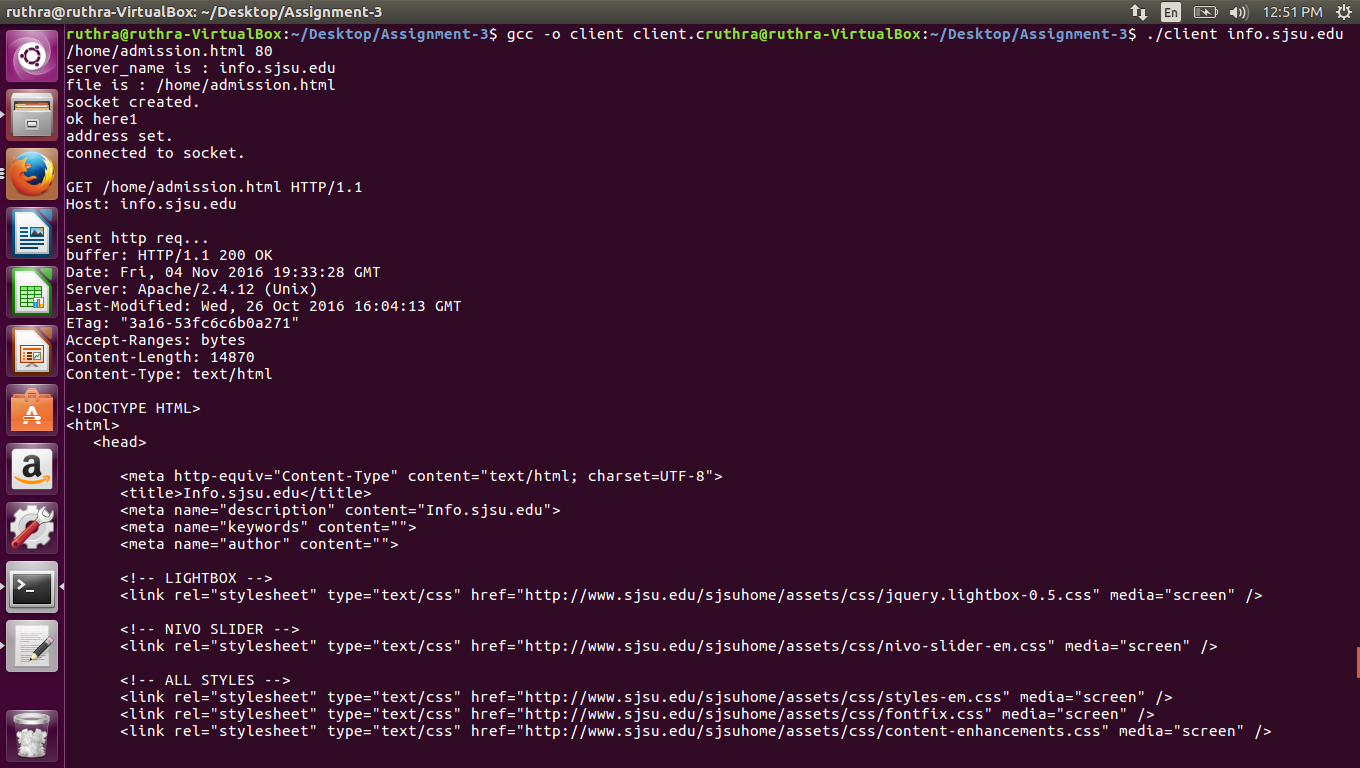
**Page 1**

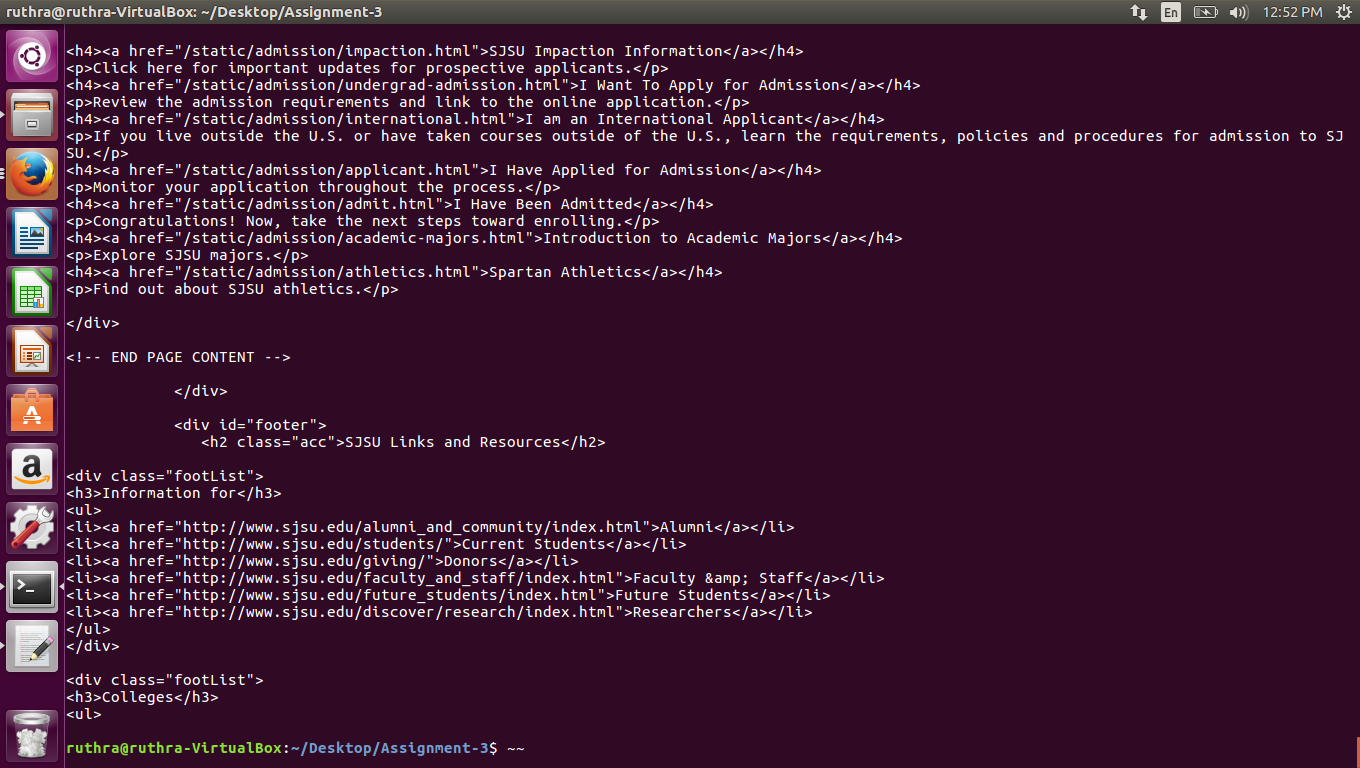
**gcc -o client client.c**

**./client info.sjsu.edu/home/admission.html 80**

In order to explicitly show that the client and server is communicating properly we have designed the client in such a way to display the header received from the server and the data in the client-side html file (onto which the data from the buffer is filled).

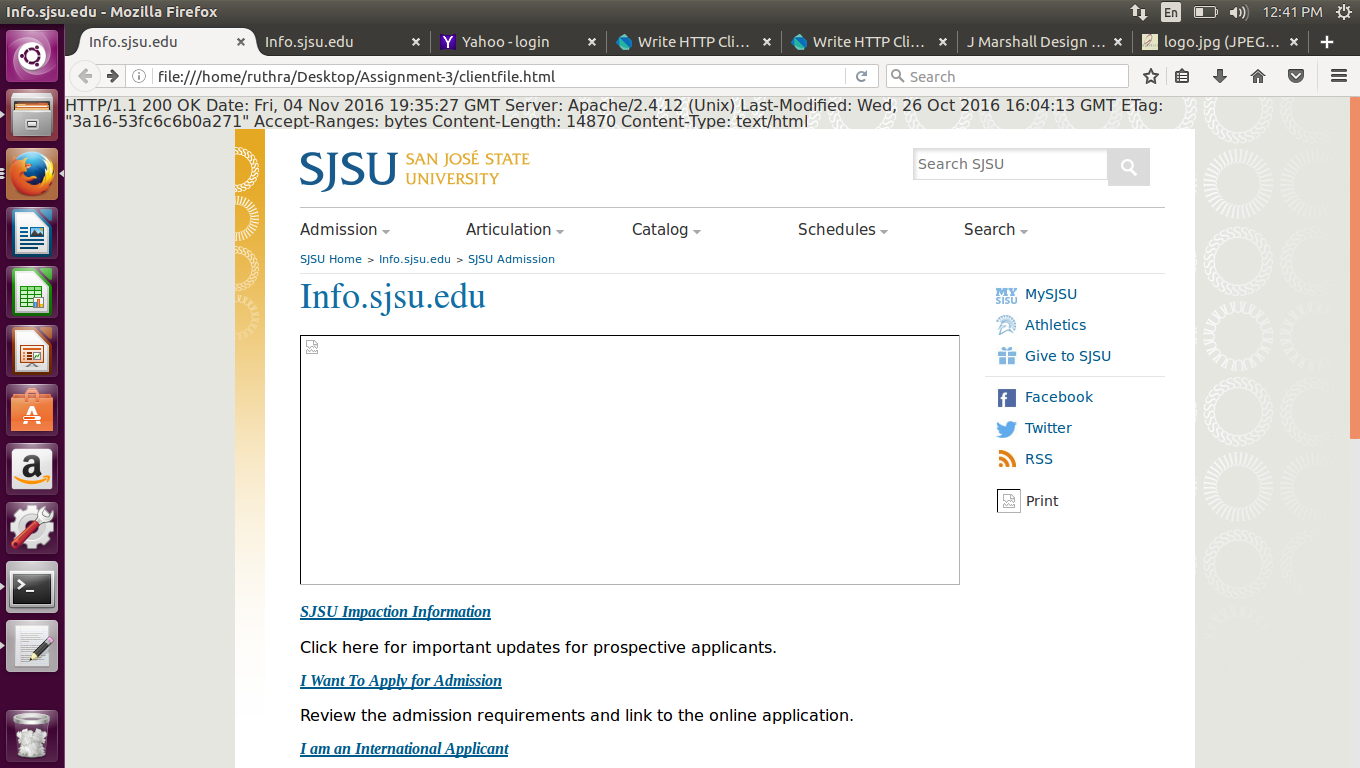
**Command Line Outputs:**

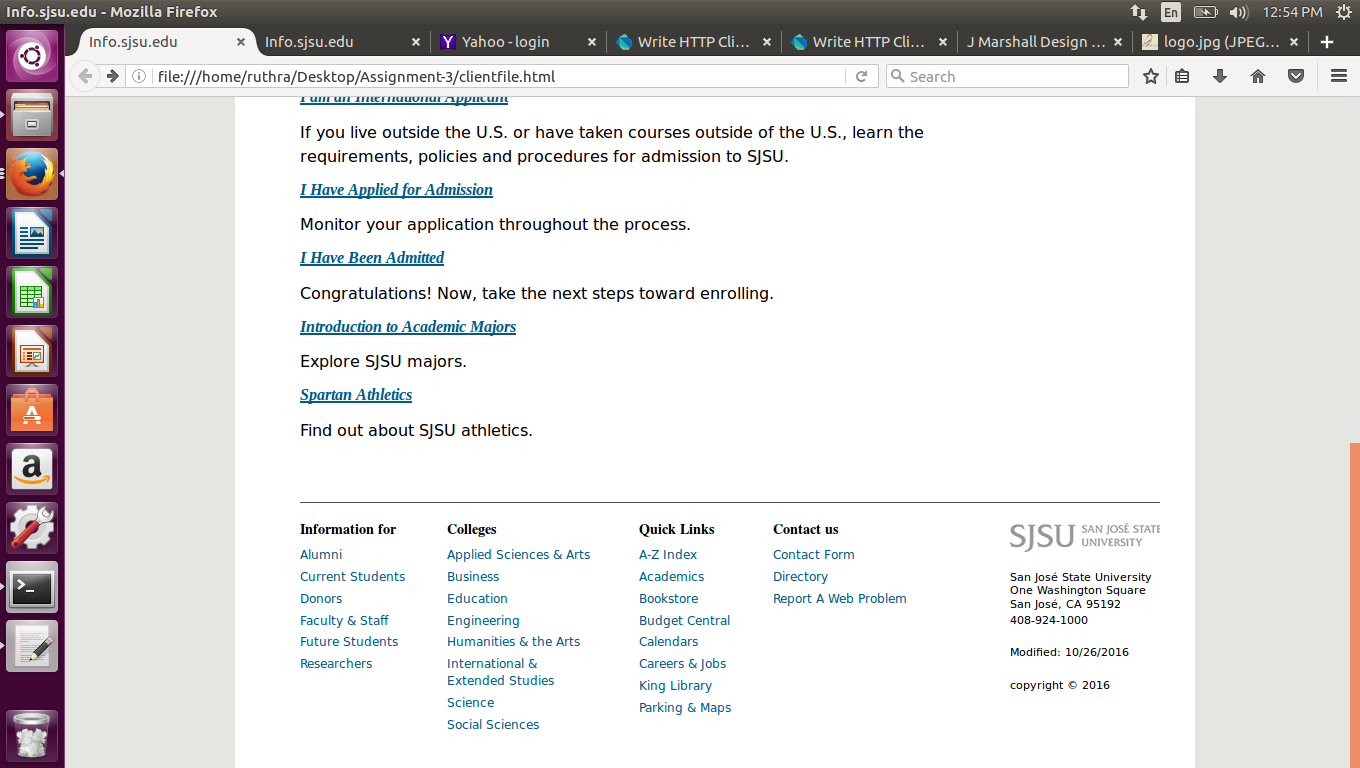




**HTML file Outputs:**

As said above, please note the Header file content displayed on top of the page.





**Command Line on the client**

**Page 2**

**gcc -o client client.c**

**./client its.sjsu.edu/ 80**

