Computer Networks Assignment

Web server using socket programming

Q) Build a Web Server in C/Python using Socket programming. The web server should be able to respond to simple HTTP commands like GET, POST etc. When a GET request is sent, the server should respond with a page containing your name and roll number.

Code And Explanation

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <string.h>
#include <fcntl.h>
#include <sys/sendfile.h>
#include <unistd.h>
#include <netinet/in.h>

#define BUFFER_SIZE 4096

void send_http_ok(int client_socket,off_t file_size) {
    // Construct the HTTP OK response
    char response[512];
    snprintf(response, sizeof(response), "HTTP/1.1 200 OK\r\n"
                                         "Content-Type: text/html\r\n"
                                         "Content-Length: %ld\r\n"
                                         "\r\n", (long)file_size);
    send(client_socket, response, strlen(response), 0);
}

int main() {
    int s = socket(AF_INET, SOCK_STREAM, 0);
    if (s == -1)
    {
        perror("Socket creation failed");
        exit(EXIT_FAILURE);
    }

    struct sockaddr_in addr =
    {
        AF_INET,
        0x901f,
        0
    };
```

- Here I have created a function for giving HTTP OK response.
- After that moving to the main part of the code I have initialesed a socket by using the socker() function in C.
- And I also created the address data structure for using in the bind function.

```
36
37      if (bind(s, (struct sockaddr *)&addr, sizeof(addr)) == -1)
38      {
39          perror("Socket binding failed");
40          exit(EXIT_FAILURE);
41      }
42
43      if (listen(s, 5) == -1)
44      {
45          perror("Socket listening failed");
46          exit(EXIT_FAILURE);
47      }
48
49      printf("Listening...\n");
50
```

- Here I bind this address to a socket and hence that socket will have a port number, this is done using bind function in C.
- After that our server is listening for the connection to occur.
- This is the done using the listen() function in C.
- In this condition the code below the listen function will be executed only after we get a connection from listening.

```
50
51      while (1)
52      {
53          int client_fd = accept(s, 0, 0);
54          if (client_fd == -1)
55          {
56              perror("Accept failed");
57              exit(EXIT_FAILURE);
58          }
59
60          printf("Client connected\n");
61
62          char buffer[4096] = {0};
63          recv(client_fd, buffer, sizeof(buffer), 0);
64          printf("%s",buffer);
65
66          //Code for responding to GET
67
68          if(buffer[0]=='G')
69          {
70              char* f = buffer + 5;
71              *strchr(f, ' ') = 0;
72              int opened_fd = open(f, O_RDONLY);
73
74              off_t file_size = lseek(opened_fd, 0, SEEK_END);
75              lseek(opened_fd, 0, SEEK_SET);
76
77              send_http_ok(client_fd,file_size);
78              sendfile(client_fd, opened_fd, 0, 512);
79              close(opened_fd);
80              close(client_fd);
81          }
82
```

- Here when a connection comes, we accept it using the accept() function in C.
- After that we receive the request from the client using the receive function in C.
- And whatever data that we receive is stored in the charecter array called buffer.
- After that we classify it as a GET Command if the HTTP message starts with G, in that case we read the file name that the GET request is asking for using string manipulation techniques.
- The requested file is opened if it is found, and then its length is also found.
- And first we send the HTTP OK response and related header files using the function that we defined earlier.

- After that we send the file that was requested by the server using the sendfile() function in C.

```
        // Code for responding to POST

        else
        {
            printf("\nPOST received\n");
            char *body_start = strstr(buffer, "\r\n\r\n");
            char response[] = "HTTP/1.1 200 OK\r\nContent-Type: text/plain\r\n\r\nPOST received\n";
            send(client_fd, response, strlen(response), 0);
        }
    }

    return 0;
}
```

- Here the code for POST command is given.

- In this we receive the POST message and then send an OK response to the browser using HTTP 200 message.

```
<html>
  <head>
    <title>adwayithks</title>
  </head>
  <body>
    <br>
    <br>
    <h1 align="center">
      <font color="cyan">Adwayith K S</font>
    </h1>
    <h1 align="center">
      <font color="lime">Roll.no: B210664EC</font>
    </h1>
    <br>
    <br>
    <div align="center">
      <form method="post">
        <label for="textbox">Enter Text:</label>
        <input type="text" id="textbox" name="textbox">
        <button type="submit">Submit</button>
      </form>
    </div>

  </body>
</html>
```

This is the code for the index.html file which is going to be shown in the web browser.

Outputs :

Output for get command :

```
adwayithks@jarvis:~/Desktop/temp files$ gcc websocket3.c
adwayithks@jarvis:~/Desktop/temp files$ ./a.out
Listening...
Client connected
GET /index.html HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

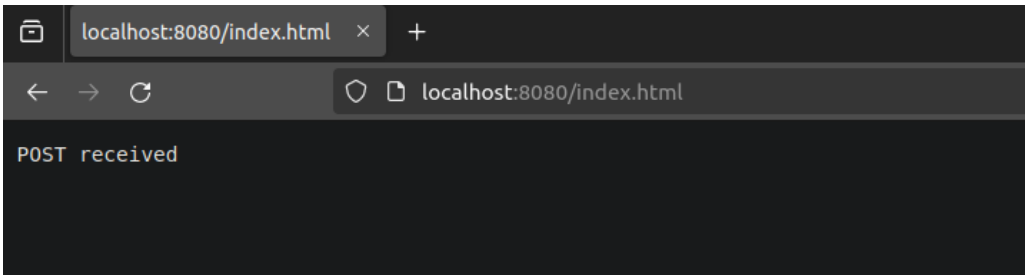
Browser Output :

Output for POST command :

```
Client connected
POST /index.html HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:123.0) Gecko/20100101 Firefox/123.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 13
Origin: http://localhost:8080
DNT: 1
Connection: keep-alive
Referer: http://localhost:8080/index.html
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

textbox=hello
POST received
```

Browser Output After POST received :



Flow Chart :