CSE3027: Computer Networks

Project 1:

Concurrent Web Server using BSD Sockets

2020064911 Seong surib

Decribtion of my server's design :

1. Server Setup and Initialization:
    - Determine the port to use and create the server socket.
    - Initialize and bind the server address structure.
    - Put the socket in a listening state to accept client connections.
2. Waiting for Client Connections:
    - The server waits for incoming client connections on a specific port.
3. Accepting Client Connections:
    - When a client sends a connection request to the server, the server accepts that connection using the `accept()` function.
    - It creates a new socket for communication with the accepted client.
4. Receiving Client Requests:
    - The server reads data from the accepted client socket. It waits until an HTTP request arrives from the client.
5. Handling Requests and Sending Responses:
    - Upon receiving an HTTP request, the server processes it and performs the requested operation. This might involve reading a requested file, querying a database, or other operations.
    - The server then constructs an appropriate response, including the requested content or an error message, and sends it back to the client as an HTTP response.
6. Closing the Connection:
    - Once the communication with the client is completed, the server closes the connection with that client.
    - It closes the socket used for the connection.
7. Repeating Connection Waiting:
    - The server goes back to waiting for the next client connection by returning to the listening state on the socket.
    - This process repeats until the server is shut down.

Part A

First you create an index.html file and then activate the server, enter the address and port number to the address, and enter /index.html, the following screen is output from the web page.

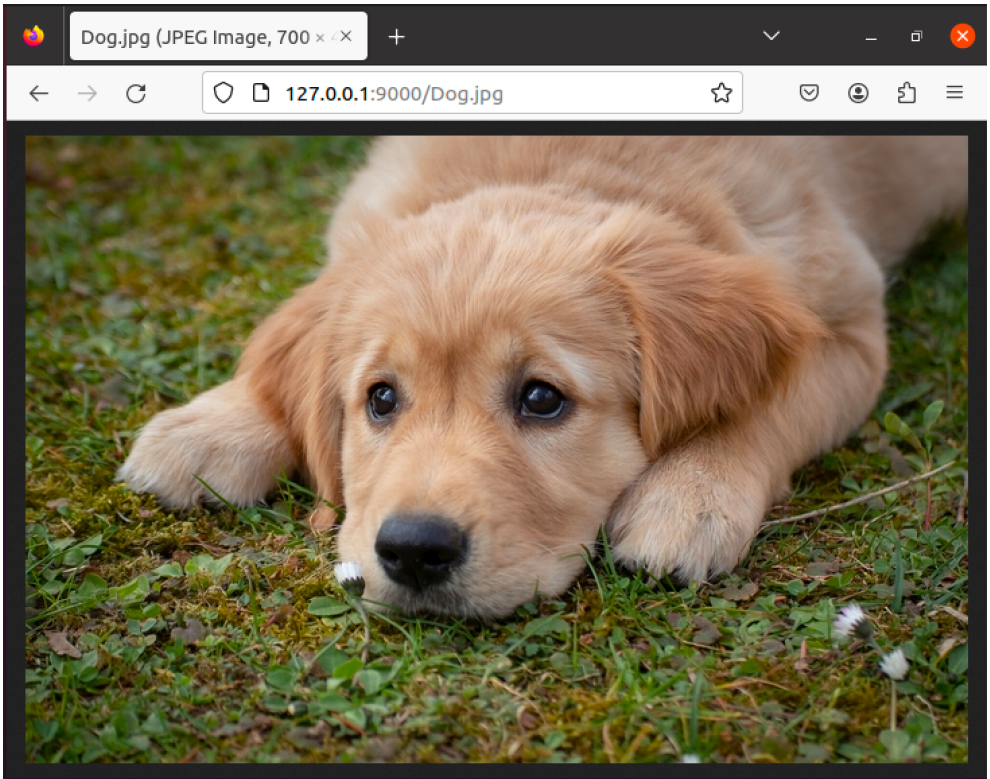
```
vboxuser@ubuntu:~/Documents/Web$ ./server 9000
Server: Waiting for client's connection on port 9000...
HTTP Request is GET /index.html HTTP/1.1
Host: 127.0.0.1:9000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Fi
refox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Response message: HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 90
```

The Html Request used the GET method. The GET method is mainly used to read or retrieve data. If the GET request is successful, it returns a 200 OK HTTP response code.
Host represents the port currently used (the number after localhost)
User-agent contains identification information of user software, such as devices and browsers that send HTTP requests.
Accept tells clients what types of content they can understand.

Part B

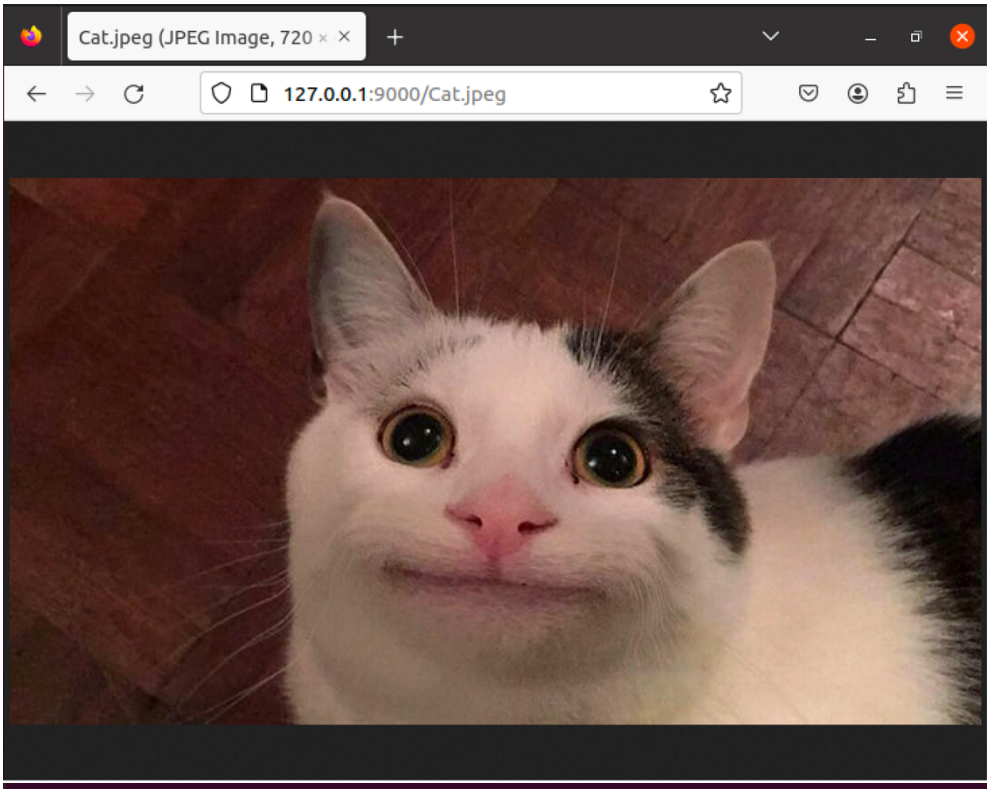
Implementation example :

I. JPG




```
HTTP Request is GET /Dog.jpg HTTP/1.1
Host: 127.0.0.1:9000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Fi
refox/111.0
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
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Response message: HTTP/1.1 200 OK
Content-Type: (null)
Content-Length: 264776
```

II. JPEG

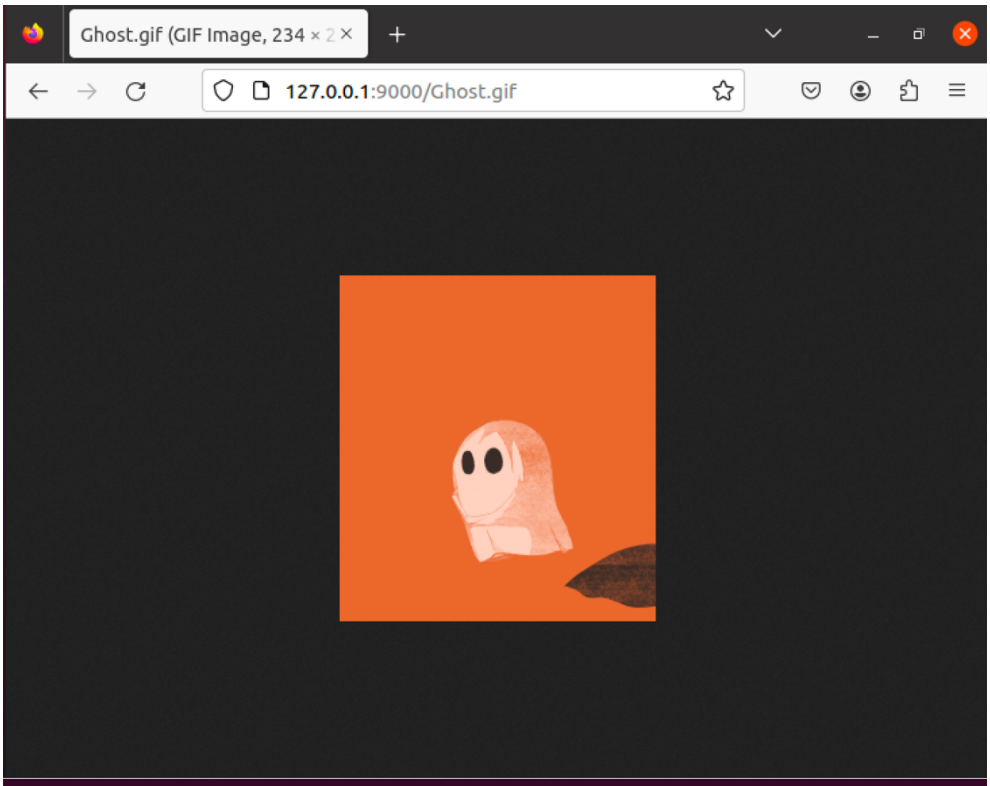



```
HTTP Request is GET /Cat.jpeg HTTP/1.1
Host: 127.0.0.1:9000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Fi
refox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Response message: HTTP/1.1 200 OK
Content-Type: image/jpeg
Content-Length: 40220
```

III. PNG

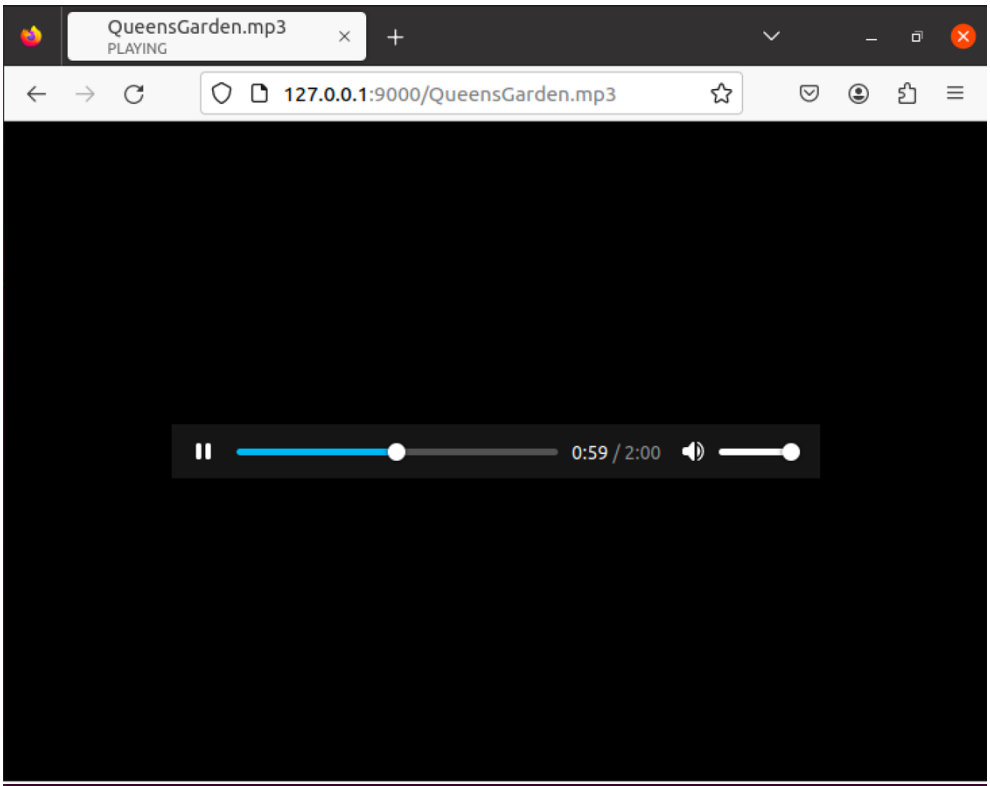

Ghibli.png (PNG Image, 600
127.0.0.1:9000/Ghibli.png
HTTP Request is GET /Ghibli.png HTTP/1.1
Host: 127.0.0.1:9000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
Response message: HTTP/1.1 200 OK
Content-Type: image/png
Content-Length: 363494

IV. GIF



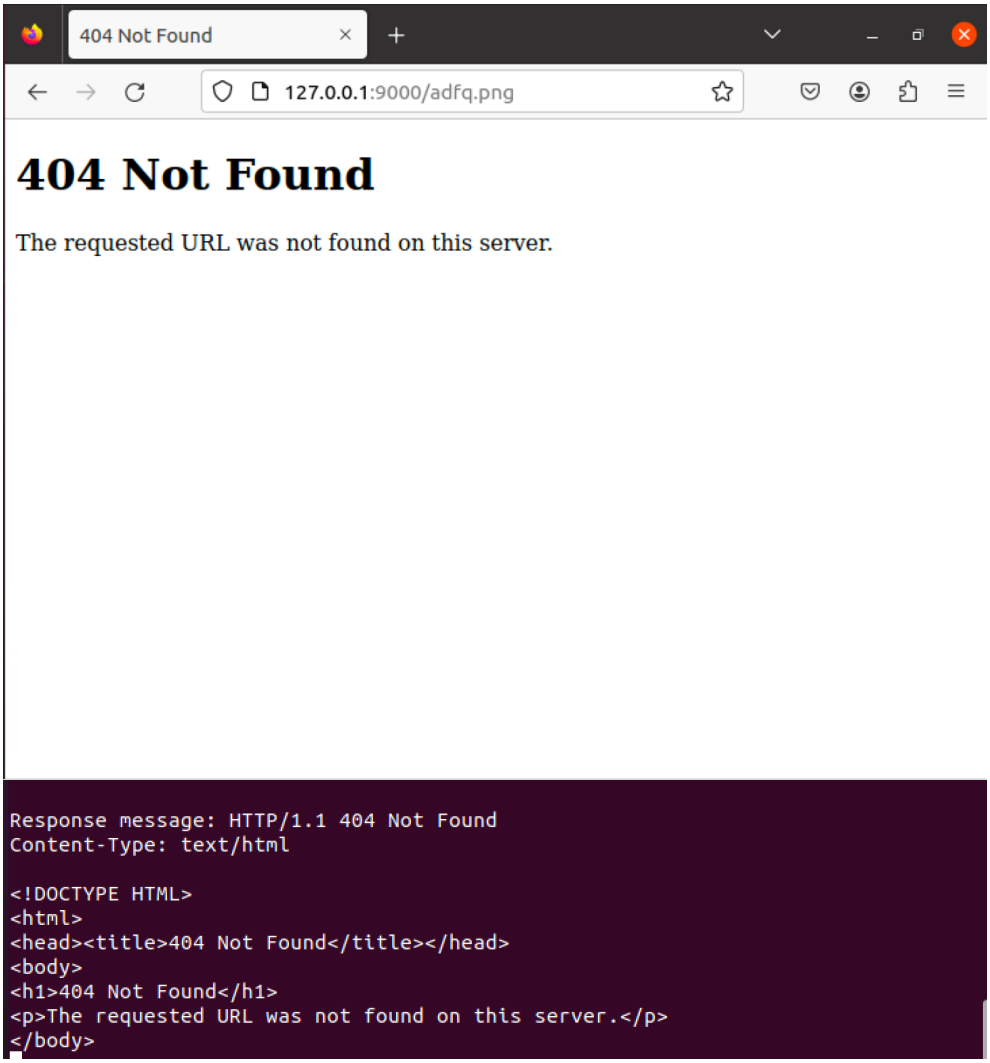

```
HTTP Request is GET /Ghost.gif HTTP/1.1
Host: 127.0.0.1:9000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Fi
refox/111.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Response message: HTTP/1.1 200 OK
Content-Type: image/gif
Content-Length: 89660
```

V. MP3




```
HTTP Request is GET /QueensGarden.mp3 HTTP/1.1
Host: 127.0.0.1:9000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:109.0) Gecko/20100101 Fi
refox/111.0
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
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1


Response message: HTTP/1.1 200 OK
Content-Type: audio/mpeg
Content-Length: 1195886
```

VI. ERROR (404 NOT FOUND)



Difficulties that I faced :

Due to the lack of knowledge of pointer variables in language c, it was difficult to declare functions and variables. In addition, due to the lack of knowledge of various string functions, it was difficult to know how to use the function. From the basics of socket programming, I was able to overcome it by slowly studying how the function works in language c and how to use the variables one by one.