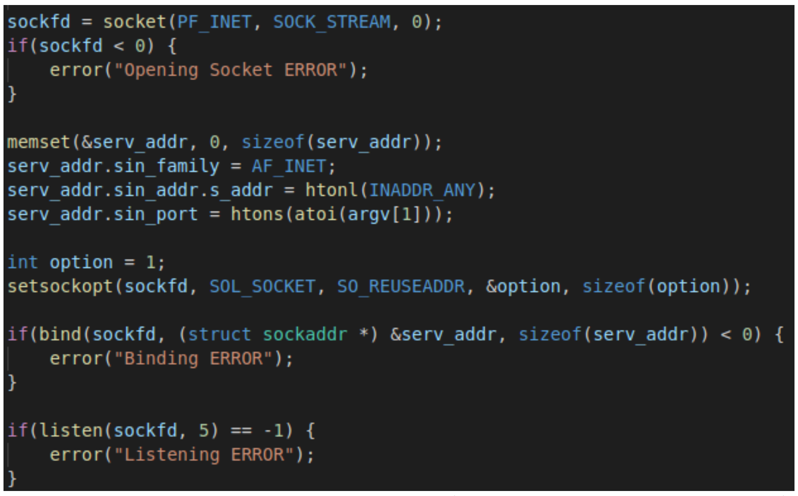
Computer Network Project1

2019040519 Taehyung Kim

1. High-level description

- socket(), bind(), listen(), accept()

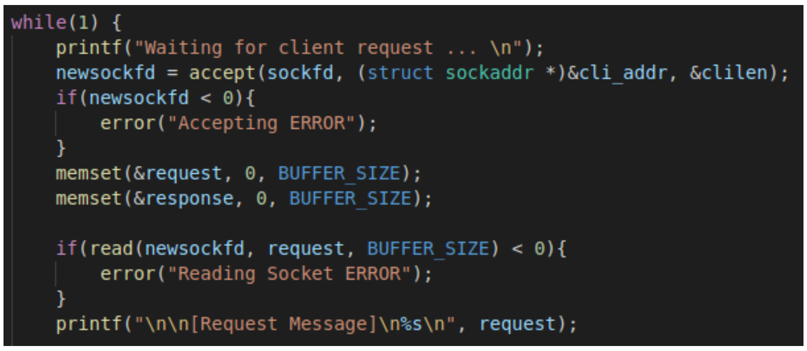


```c
sockfd = socket(PF_INET, SOCK_STREAM, 0);
if(sockfd < 0) {
    error("Opening Socket ERROR");
}

memset(&serv_addr, 0, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
serv_addr.sin_port = htons(atoi(argv[1]));

int option = 1;
setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &option, sizeof(option));

if(bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
    error("Binding ERROR");
}

if(listen(sockfd, 5) == -1) {
    error("Listening ERROR");
}
```

A TCP socket is generated by a socket(PF_INET, SOCK_STREAM, 0).

Then, bind() the server socket to assign the IP address and port number.

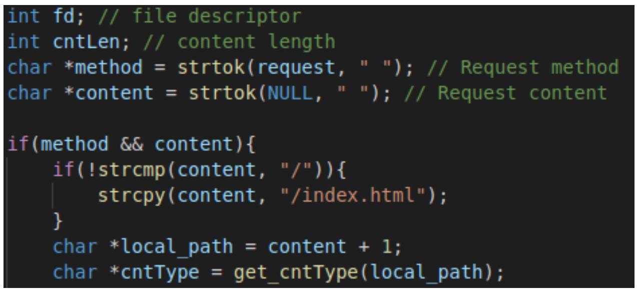
The function listen(sockfd, 5) puts the socket in a waiting state to receive a connection request.



```c
while(1) {
    printf("Waiting for client request ... \n");
    newsockfd = accept(sockfd, (struct sockaddr *)&cli_addr, &clilen);
    if(newsockfd < 0){
        error("Accepting ERROR");
    }
    memset(&request, 0, BUFFER_SIZE);
    memset(&response, 0, BUFFER_SIZE);

    if(read(newsockfd, request, BUFFER_SIZE) < 0){
        error("Reading Socket ERROR");
    }
    printf("\n\n[Request Message]\n%s\n", request);
```

If a connection request is received from the client, it is allowed with the accept() function.

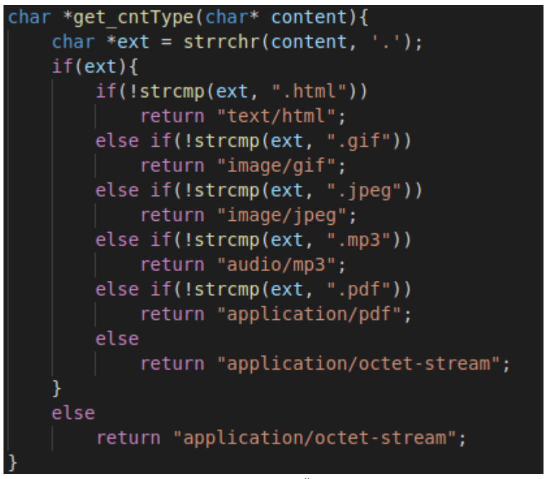
- Request Message Parsing



```c
int fd; // file descriptor
int cntLen; // content length
char *method = strtok(request, " "); // Request method
char *content = strtok(NULL, " "); // Request content

if(method && content){
    if(!strcmp(content, "/")){
        strcpy(content, "/index.html");
    }
    char *local_path = content + 1;
    char *cntType = get_cntType(local_path);
```

Client request message is parsed based on the space to create a pointer of method and content.

The content pointer is passed as a parameter of the function **get_cntType()**.



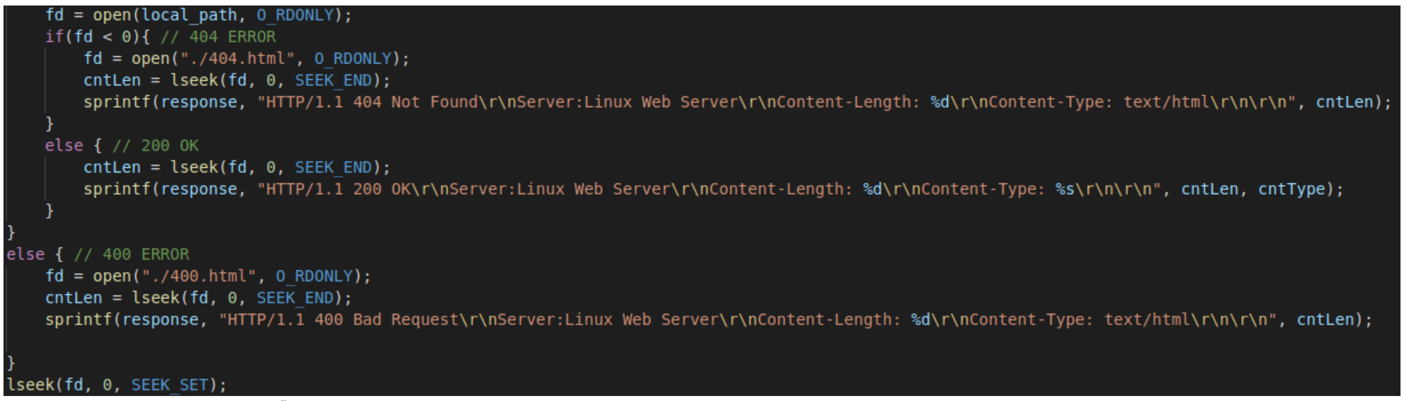
```c
char *get_cntType(char* content){
    char *ext = strrchr(content, '.');
    if(ext){
        if(!strcmp(ext, ".html"))
            return "text/html";
        else if(!strcmp(ext, ".gif"))
            return "image/gif";
        else if(!strcmp(ext, ".jpeg"))
            return "image/jpeg";
        else if(!strcmp(ext, ".mp3"))
            return "audio/mp3";
        else if(!strcmp(ext, ".pdf"))
            return "application/pdf";
        else
            return "application/octet-stream";
    }
    else
        return "application/octet-stream";
}
```

▲ Function get_cntType()

The **get_cntType()** is a function that returns the content type requested by the client.

If it is not the content type required by the project, return "application/octet-stream" to the default type.

- Make Response Message



```c
    fd = open(local_path, O_RDONLY);
    if(fd < 0){ // 404 ERROR
        fd = open("./404.html", O_RDONLY);
        cntLen = lseek(fd, 0, SEEK_END);
        sprintf(response, "HTTP/1.1 404 Not Found\r\nServer:Linux Web Server\r\nContent-Length: %d\r\nContent-Type: text/html\r\n\r\n", cntLen);
    }
    else { // 200 OK
        cntLen = lseek(fd, 0, SEEK_END);
        sprintf(response, "HTTP/1.1 200 OK\r\nServer:Linux Web Server\r\nContent-Length: %d\r\nContent-Type: %s\r\n\r\n", cntLen, cntType);
    }
}
else { // 400 ERROR
    fd = open("./400.html", O_RDONLY);
    cntLen = lseek(fd, 0, SEEK_END);
    sprintf(response, "HTTP/1.1 400 Bad Request\r\nServer:Linux Web Server\r\nContent-Length: %d\r\nContent-Type: text/html\r\n\r\n", cntLen);

}
lseek(fd, 0, SEEK_SET);
```

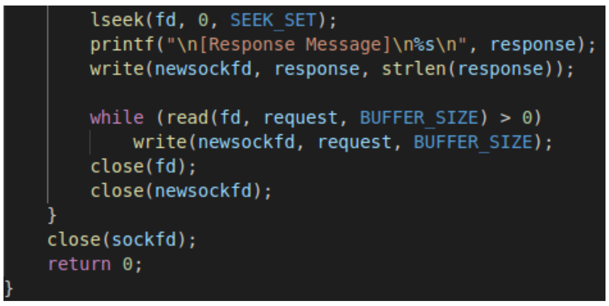
Then, attempts to open() the client request file.

In the case of a request without a problem, write the 200 OK message, content type, and content length in the response message format to the response buffer.

If the requested file is not on the server, open() the 404.html file and fill with a 404 error message in the response buffer.

If the request message does not contain method or content, open() the 400.html file and fill with a 400 Error message in the response buffer.

- Send Response Message

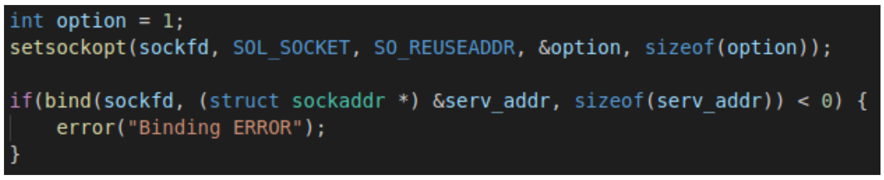


```c
        lseek(fd, 0, SEEK_SET);
        printf("\n[Response Message]\n%s\n", response);
        write(newsockfd, response, strlen(response));

        while (read(fd, request, BUFFER_SIZE) > 0)
            write(newsockfd, request, BUFFER_SIZE);
        close(fd);
        close(newsockfd);
    }
    close(sockfd);
    return 0;
}
```

Finally, the response message is write() to the client socket.

2. Difficulties

- If the server was forced to shut down using ctrl+c in the terminal, the port could not be reused.

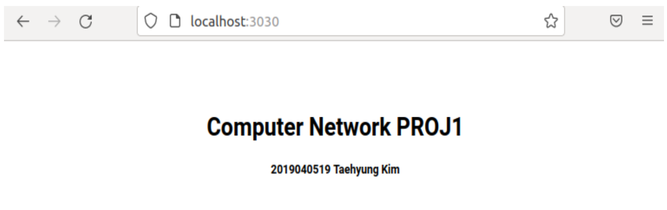


```
int option = 1;
setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &option, sizeof(option));

if(bind(sockfd, (struct sockaddr *) &serv_addr, sizeof(serv_addr)) < 0) {
    error("Binding ERROR");
}
```

-> Before binding the socket, setsockopt(sockfd, SOL_SOCKET, SO_REUSEEDDR, &option, sizeof(option)) commands were added to reuse the port number even when the server was forcibly terminated.

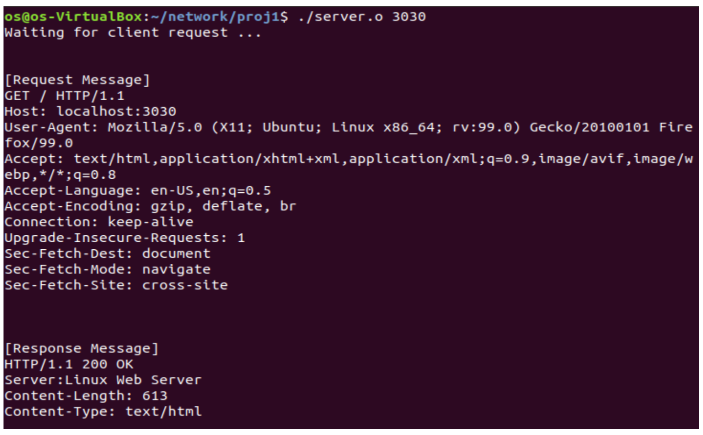
3. Sample outputs

- index.html



Computer Network PROJ1

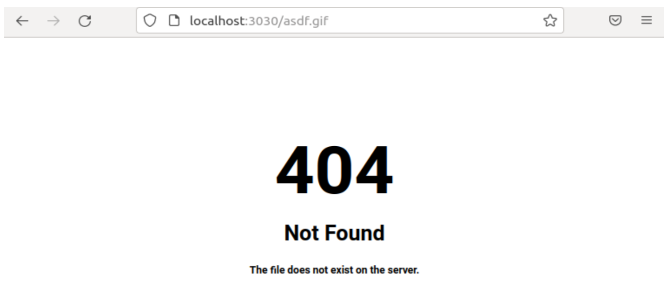
2019040519 Taehyung Kim



```
os@os-VirtualBox:~/network/proj1$ ./server.o 3030
Waiting for client request ...

[Request Message]
GET / HTTP/1.1
Host: localhost:3030
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:99.0) Gecko/20100101 Fire
fox/99.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;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: cross-site


[Response Message]
HTTP/1.1 200 OK
Server:Linux Web Server
Content-Length: 613
Content-Type: text/html
```

When the client requests index.html, the server responds to the 200 OK message with the index.html page.

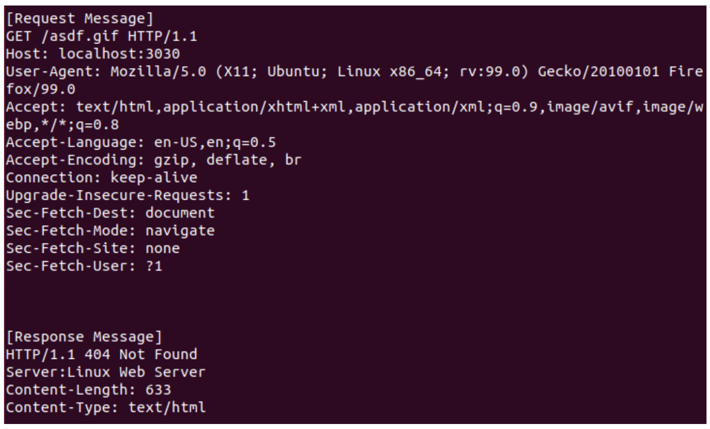
- 404 Error Page



404

Not Found

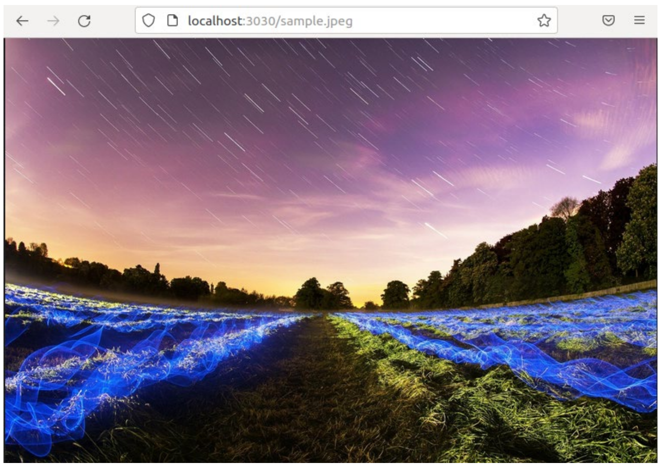
The file does not exist on the server.

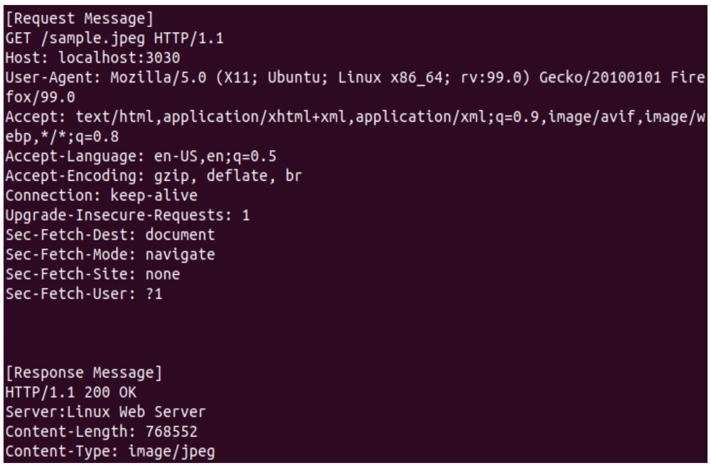


```
[Request Message]
GET /asdf.gif HTTP/1.1
Host: localhost:3030
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:99.0) Gecko/20100101 Fire
fox/99.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, 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/1.1 404 Not Found
Server:Linux Web Server
Content-Length: 633
Content-Type: text/html
```

When the client requests a file that does not exist in the server, the server responds to the 404 Not Found error response message with the 404 error.html page.

- sample.jpeg

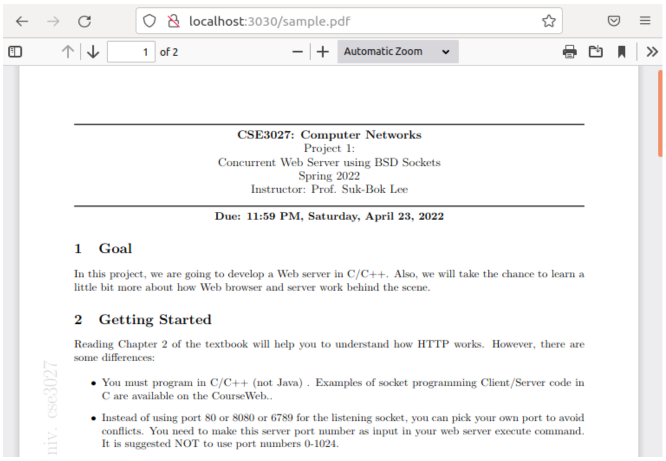




```
[Request Message]
GET /sample.jpeg HTTP/1.1
Host: localhost:3030
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:99.0) Gecko/20100101 Fire
fox/99.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: 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
Server:Linux Web Server
Content-Length: 768552
Content-Type: image/jpeg
```

When the client requests sample.jpeg, the server responds to the 200 OK message with the sample.jpeg file.

- sample.pdf



CSE3027: Computer Networks
Project 1:
Concurrent Web Server using BSD Sockets
Spring 2022
Instructor: Prof. Suk-Bok Lee

Due: 11:59 PM, Saturday, April 23, 2022

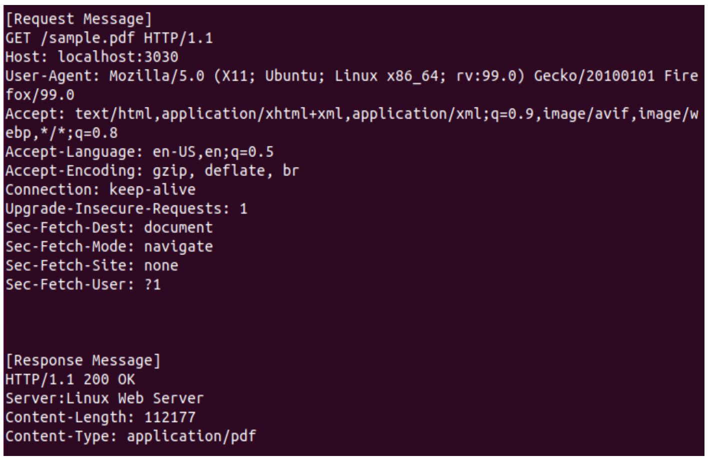
1 Goal

In this project, we are going to develop a Web server in C/C++. Also, we will take the chance to learn a little bit more about how Web browser and server work behind the scene.

2 Getting Started

Reading Chapter 2 of the textbook will help you to understand how HTTP works. However, there are some differences:

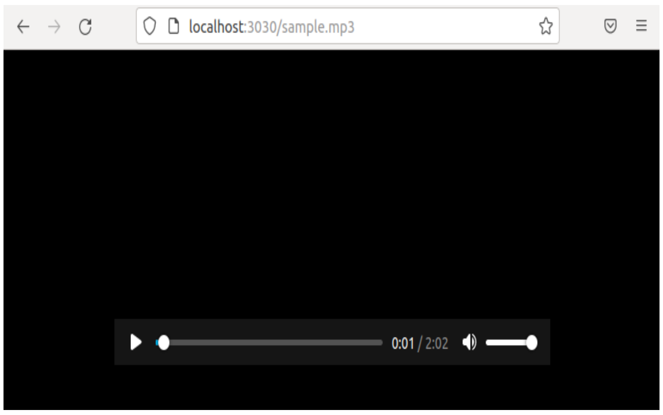
- You must program in C/C++ (not Java) . Examples of socket programming Client/Server code in C are available on the CourseWeb..
- Instead of using port 80 or 8080 or 6789 for the listening socket, you can pick your own port to avoid conflicts. You need to make this server port number as input in your web server execute command. It is suggested NOT to use port numbers 0-1024.

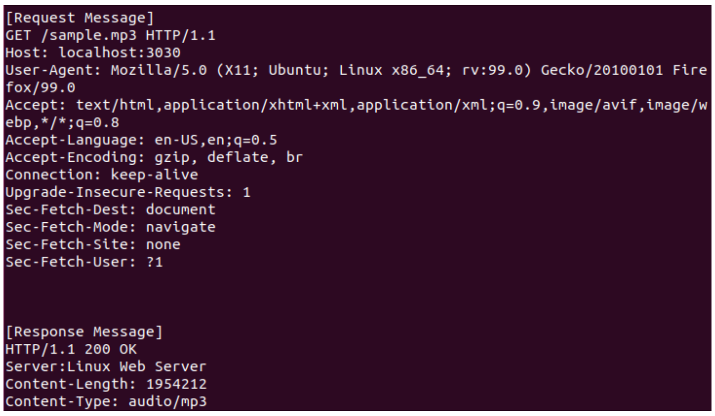


```
[Request Message]
GET /sample.pdf HTTP/1.1
Host: localhost:3030
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:99.0) Gecko/20100101 Fire
fox/99.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: 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
Server:Linux Web Server
Content-Length: 112177
Content-Type: application/pdf
```

When the client requests sample.pdf, the server responds to the 200 OK message with the sample.pdf file.

- sample.mp3

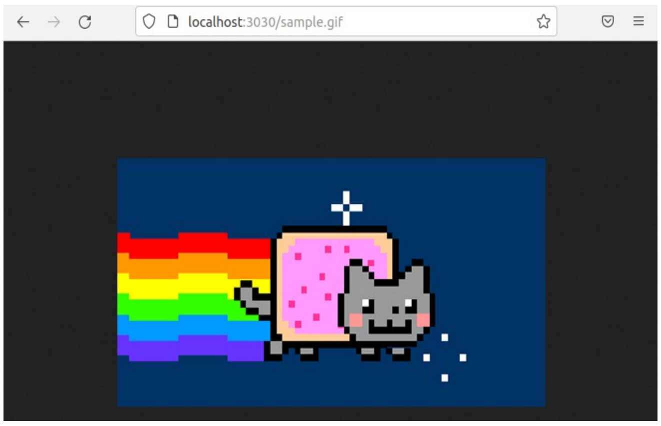


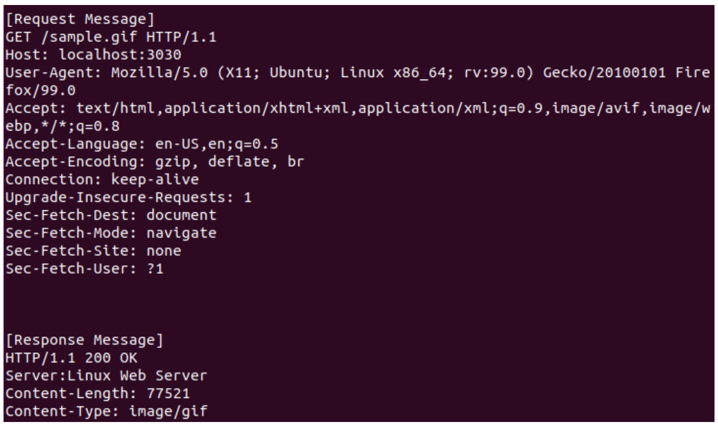


```
[Request Message]
GET /sample.mp3 HTTP/1.1
Host: localhost:3030
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:99.0) Gecko/20100101 Fire
fox/99.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: 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
Server:Linux Web Server
Content-Length: 1954212
Content-Type: audio/mp3
```

When the client requests sample.mp3, the server responds to the 200 OK message with the sample.mp3 file.

- sample.gif





```
[Request Message]
GET /sample.gif HTTP/1.1
Host: localhost:3030
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:99.0) Gecko/20100101 Fire
fox/99.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
ebp,*/*;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
Server:Linux Web Server
Content-Length: 77521
Content-Type: image/gif
```

When the client requests sample.gif, the server responds to the 200 OK message with the sample.gif file.