

Computer Networks Project 1

Socket Programming

By HONGGEUN JI [2019042633], Division of Computer Science, HYU ERICA

22.04.21



Introduction

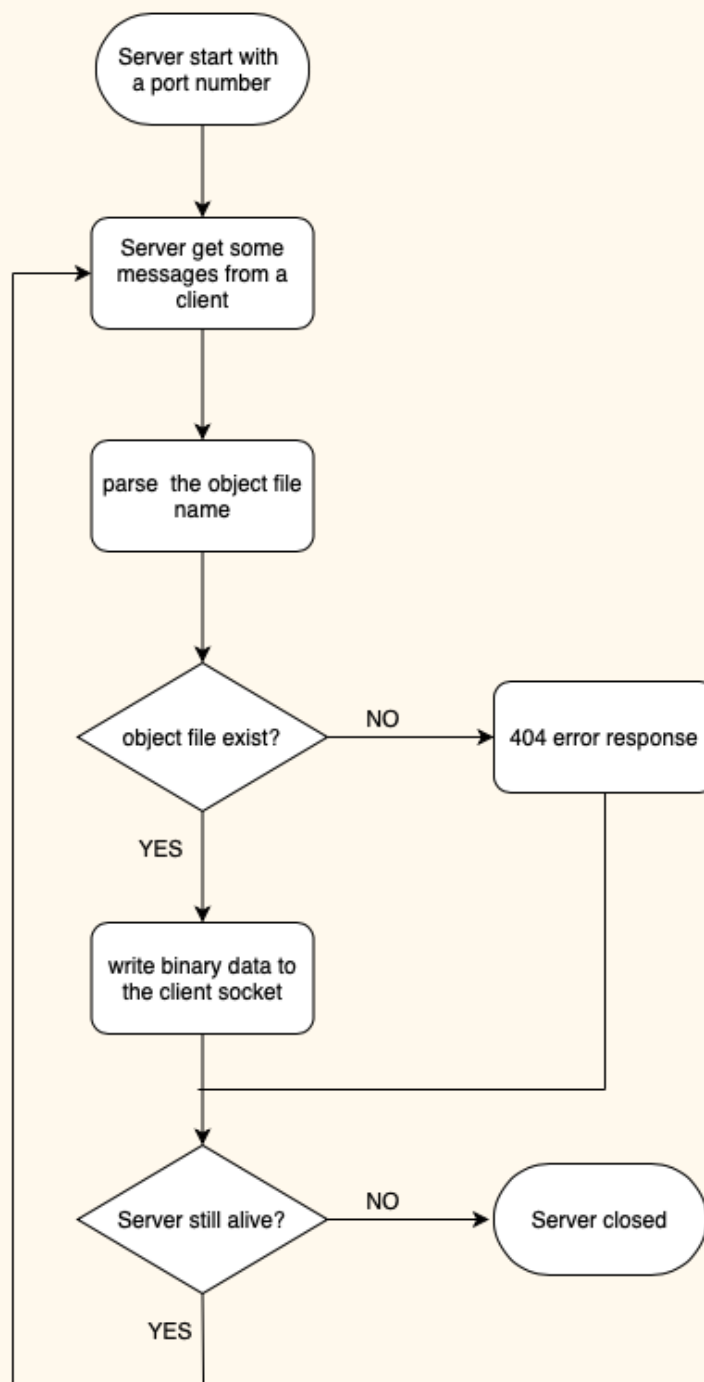
This report is dealing with the socket programming project given by Dr. SUKBOK LEE. The project was to develop a Web server with C/C++. We do not have to do the client-side socket programming, but need to implement HTTP server-side. The HTTP server will parse the client messages and try to respond to the client with an object file.

How does it work?

From this assignment, we could get some insights into the socket communication workflow. There are some way to design the HTTP server, but my server was done like this,

1. HTTP server creates a socket and ready to receive from clients
2. Clients request the object file with an HTTP message.
3. The server will take the message and parse the requested object file name.
4. The server will find the object with the parsed name.
5. If there is an HTML or other files, respond to the client with the file. Otherwise, give the client a 404 error. The response will be done by using a write system call which will write the binary data to the client socket.

This shows a simple diagram of the HTTP server.



My Opinions

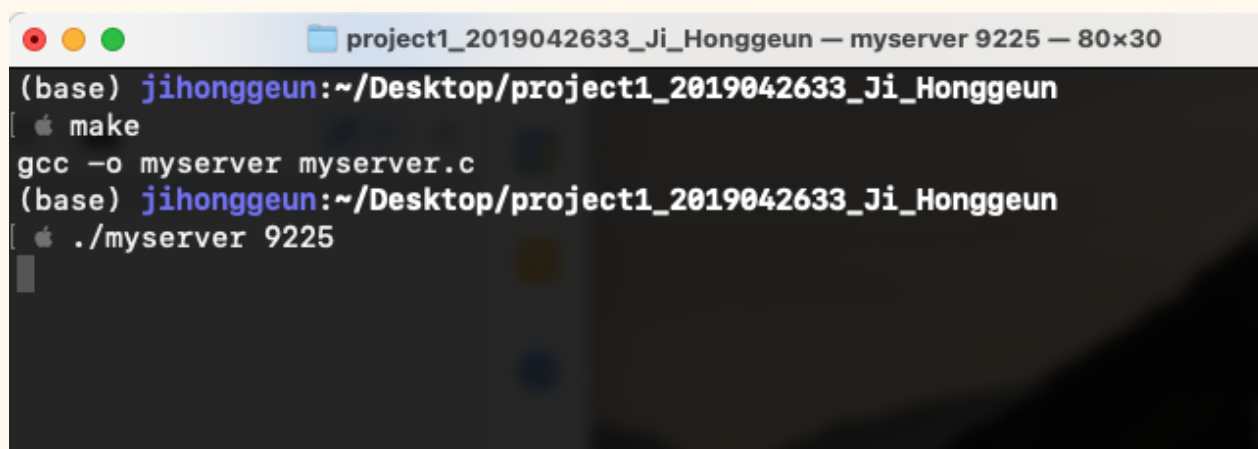
Designing and implementing my web server was not easy; however, it was a great opportunity to understand what is going on when we as a client request some object file. Initially, I did not fully understand the system calls dealing with those sockets, but after I read some documents and the Linux manual, I now can somehow deal with those socket system calls.

Parsing and using the object file was not that difficult, but as I mentioned above, using the system calls to communicate between a client and an HTTP server was the hard part for me as I am not familiar with it.

I would highly recommend reading the Linux manual (`$ man socket`) to other colleagues as I could build my own socket communication with it.

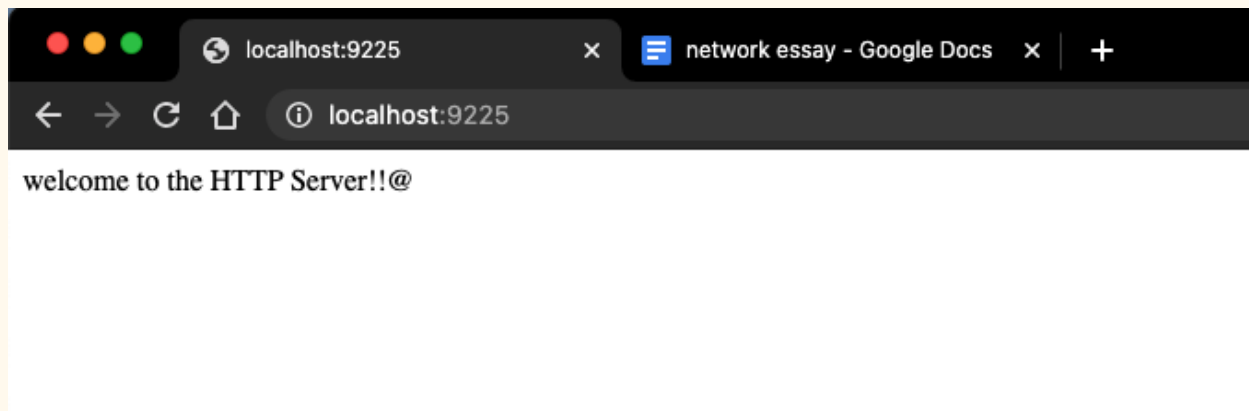
The Results

Startup the *myserver* with a port number.

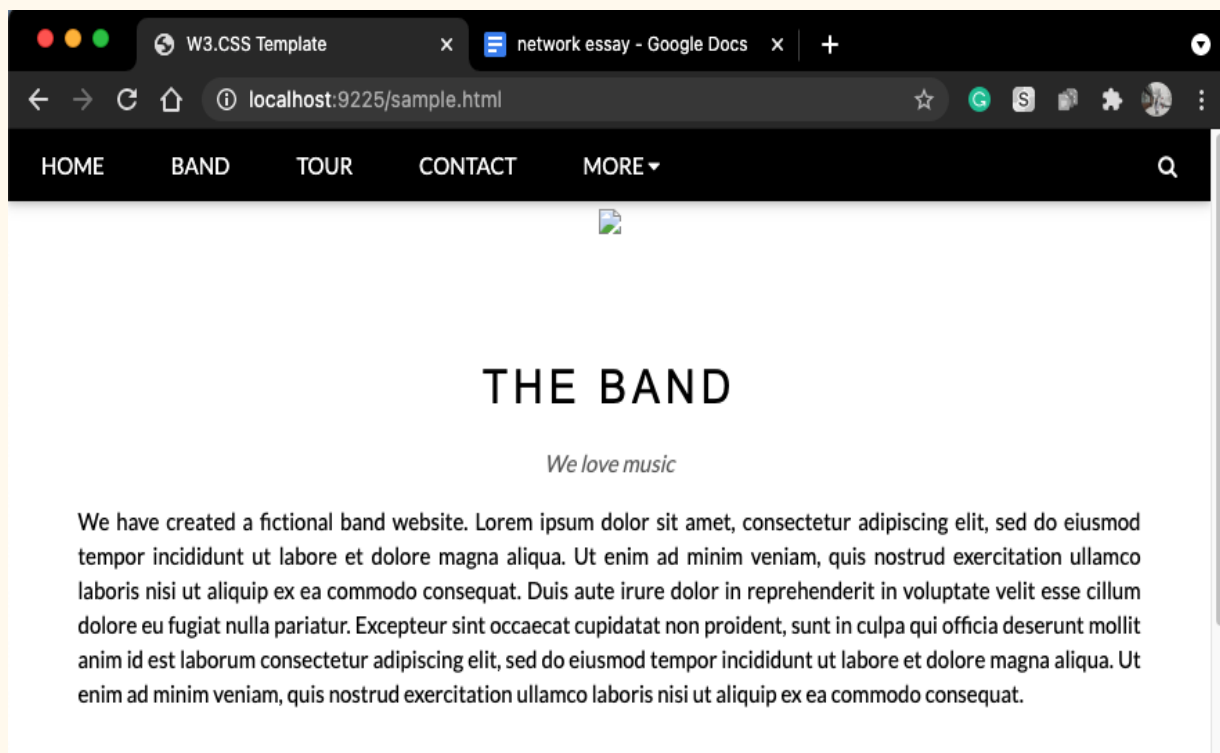
A screenshot of a macOS terminal window. The title bar at the top reads "project1_2019042633_Ji_Honggeun — myserver 9225 — 80x30". The terminal content shows the following commands and their outputs:

```
(base) jihonggeun:~/Desktop/project1_2019042633_Ji_Honggeun
[ 🍏 make
gcc -o myserver myserver.c
(base) jihonggeun:~/Desktop/project1_2019042633_Ji_Honggeun
[ 🍏 ./myserver 9225
```

The client-side(web browser) with the default given html file. (*index.html*)



You may try another html file. (*sample.html*)



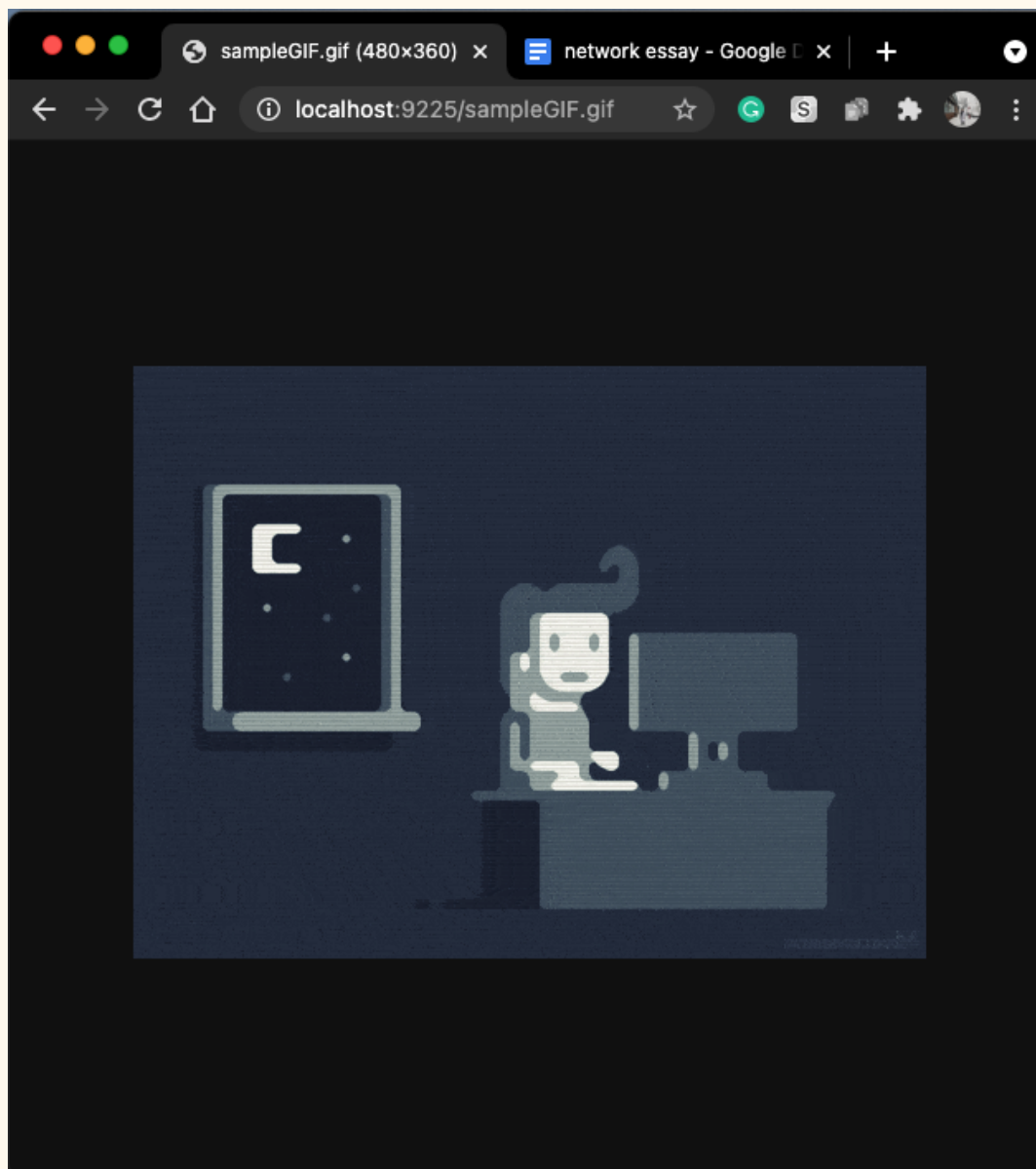
Here is the console with a client message. *(default is the index.html)*

```
GET / HTTP/1.1
Host: localhost:9225
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

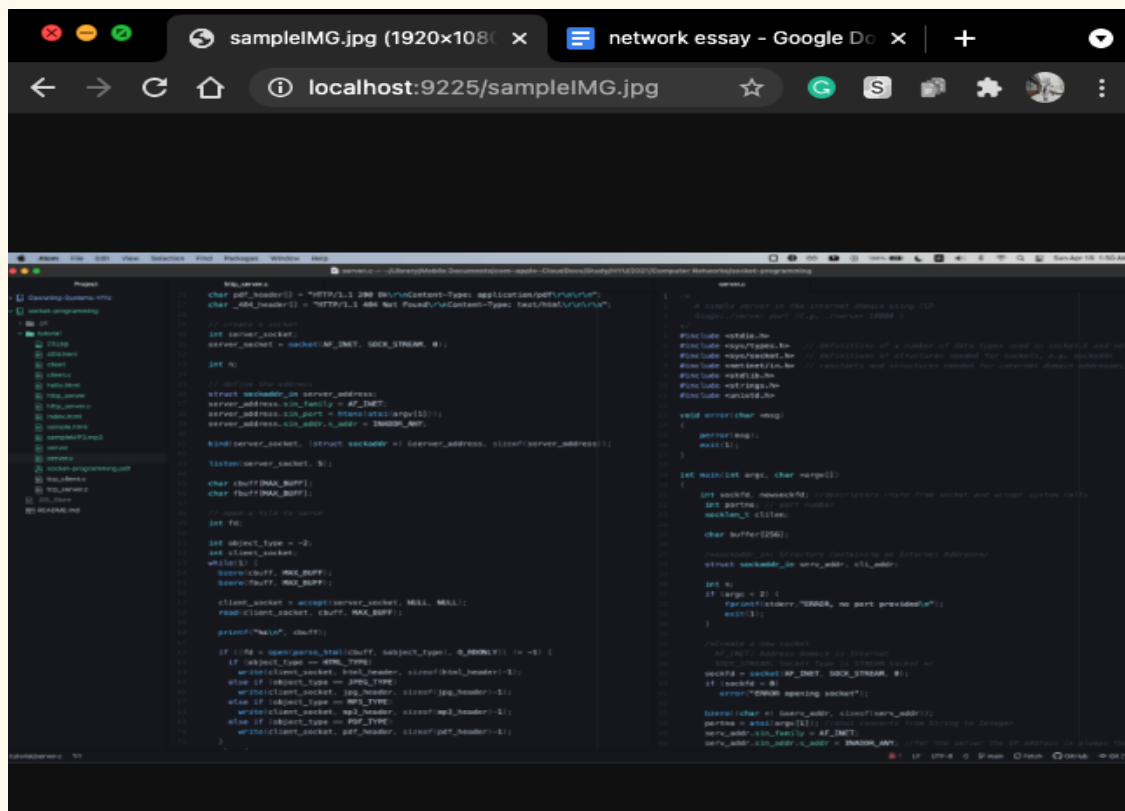
Client message with the sample HTML object.

```
GET /sample.html HTTP/1.1
Host: localhost:9225
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

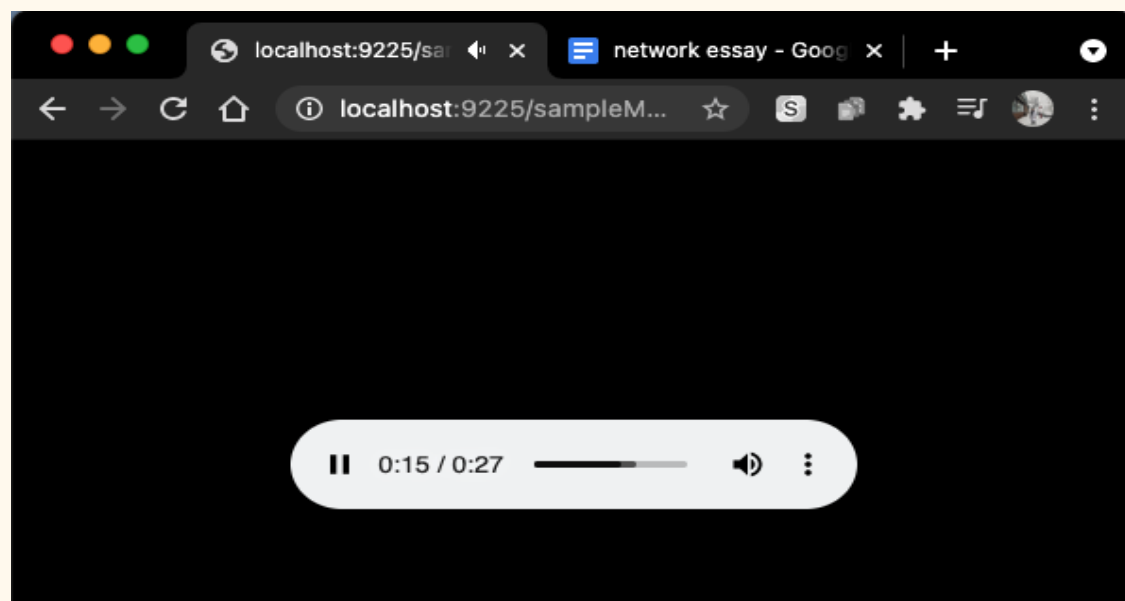
sample GIF



Sample JPG



Sample MP3



Sample PDF

Ch2-basic-socket-programming x network essay - Google Docs x +

localhost:9225/samplePDF.pdf

Ch2-basic-socket-programming 1 / 42 50%

Announcement

- Project 1 has been posted on the course website
- Due: 4/23

Individual project!

- Students can discuss each other, but must write their own codes

Submission

- All commented source codes
- Makefile
- Report (1-3 pages)

Introduction 1-1

Socket Programming

- What is a socket?
- Using sockets
 - Types (Protocols)
 - Associated functions
 - Styles
- Socket programming tutorial video:
 - <https://youtu.be/LtXEMwSG5-8>

2

Socket programming

What does it do? It allows applications that communicate using sockets to communicate with each other. It also allows applications to communicate with each other using sockets.

3

Socket programming

The socket type for the transport services is TCP. It is a reliable, byte stream-oriented communication service. It is used for applications that require a reliable, byte stream-oriented communication service. It is used for applications that require a reliable, byte stream-oriented communication service.

4

Sockets API

Creation and Setup

- Establishing a Connection
- Sending and Receiving Data
- Terminating a Connection

5

Big picture: Socket functions (TCP case)

1. Create a socket

2. Bind the socket to a local address and port

3. Listen for incoming connections

4. Accept an incoming connection

5. Create a new socket for the client

6. Send data to the client

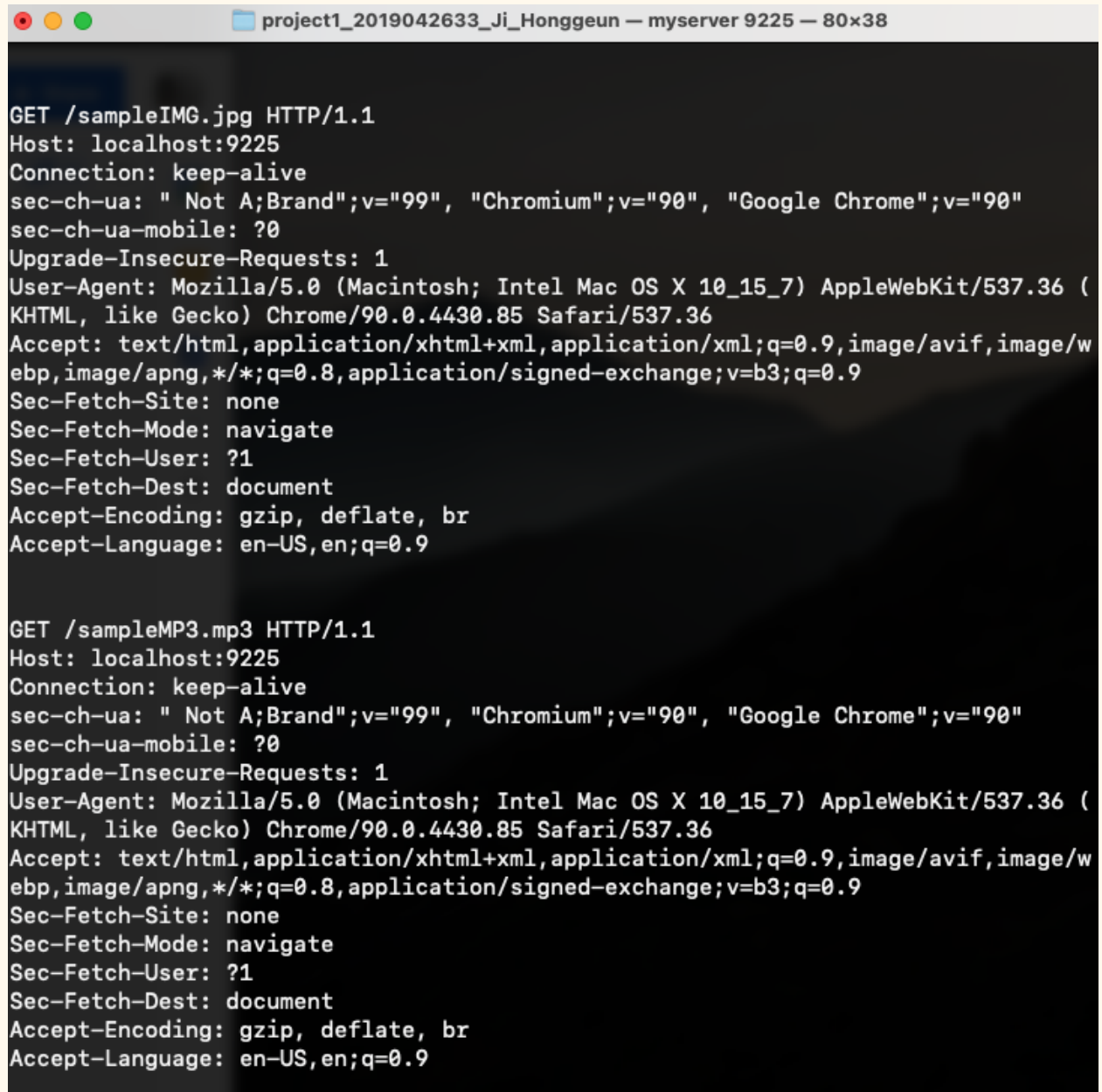
7. Receive data from the client

8. Close the client socket

9. Close the server socket

2

You still can check the client message with the different types of object files.



The screenshot shows a terminal window titled "project1_2019042633_Ji_Honggeun — myserver 9225 — 80x38". It displays two identical HTTP client messages, one for a JPEG image and one for an MP3 audio file. Both messages are from a Chrome browser on a Macintosh, using the same headers and body. The only difference is the requested file path: "/sampleIMG.jpg" and "/sampleMP3.mp3".

```
GET /sampleIMG.jpg HTTP/1.1
Host: localhost:9225
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9

GET /sampleMP3.mp3 HTTP/1.1
Host: localhost:9225
Connection: keep-alive
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.85 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```