

시스템프로그래밍

과제이름: Assignment 2-2

- 담당교수: 김 태 석 교수님
- 학 과: 컴퓨터정보공학부
- 학 번: 2019202021
- 이 름: 정 성 업
- 제출일: 2023/5/3

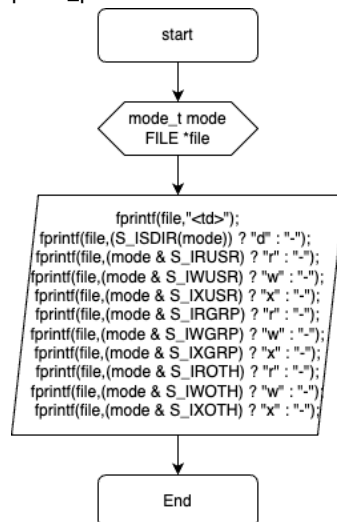
1. Introduction

A. 과제 소개

이번 Assignment 2-2 과제는 이전 Assignment 2-1에서 설계하였던 C언어로 html 파일을 브라우저 즉 client에게 보내는 것이다. 이 때 terminal은 server의 역할을 하며 socket으로 client의 request를 accept하여 원하는 동작을 client에게 보낸다. Root directory는 current working directory이고 Root directory의 parent directory로 갈 수 없으며 오직 하위 directory로만 이동할 수 있다. Root directory에서는 ls -l의 결과를 하위 directory에서는 ls -al의 결과를 출력하도록 한다. directory를 클릭한 경우 해당 directory로 이동하며, image 파일 또는 binary 파일인 경우 이미지를 다운로드하며, text 파일 및 소스 코드 클릭 시 브라우저에 display한다. 이때 링크 파일은 원본 파일 내용을 display 한다.

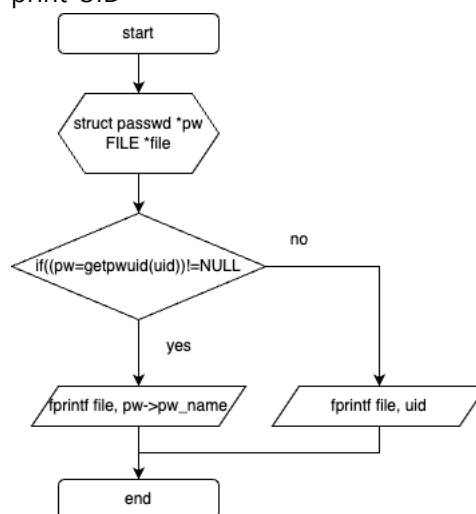
2. Flow chart

A. print_permission



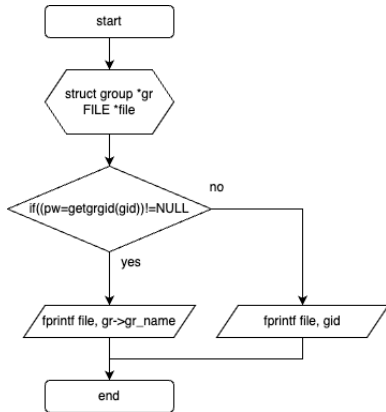
파일의 허용정보를 출력하는 함수이다.

B. print UID



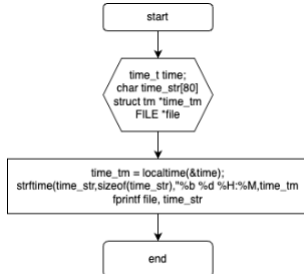
파일의 UID 정보를 출력하는 함수이다.

C. print GID



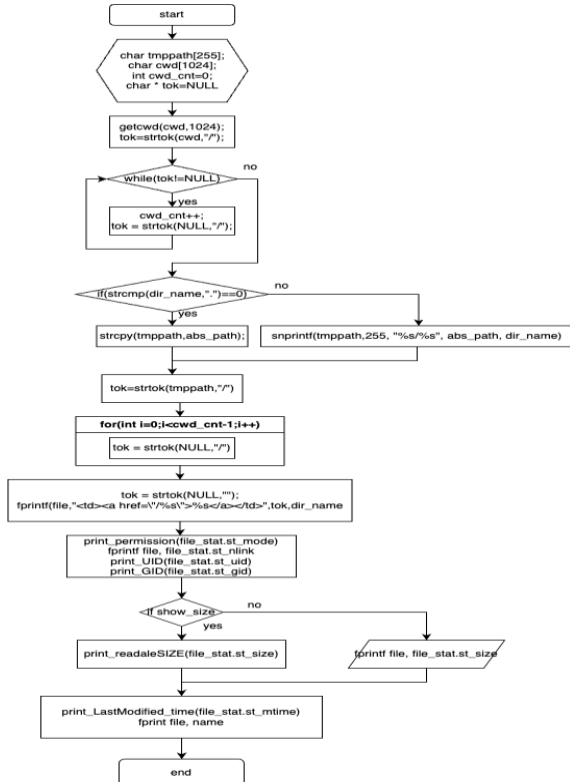
파일의 GID 정보를 출력하는 함수이다.

D. `print_LastModified_time`



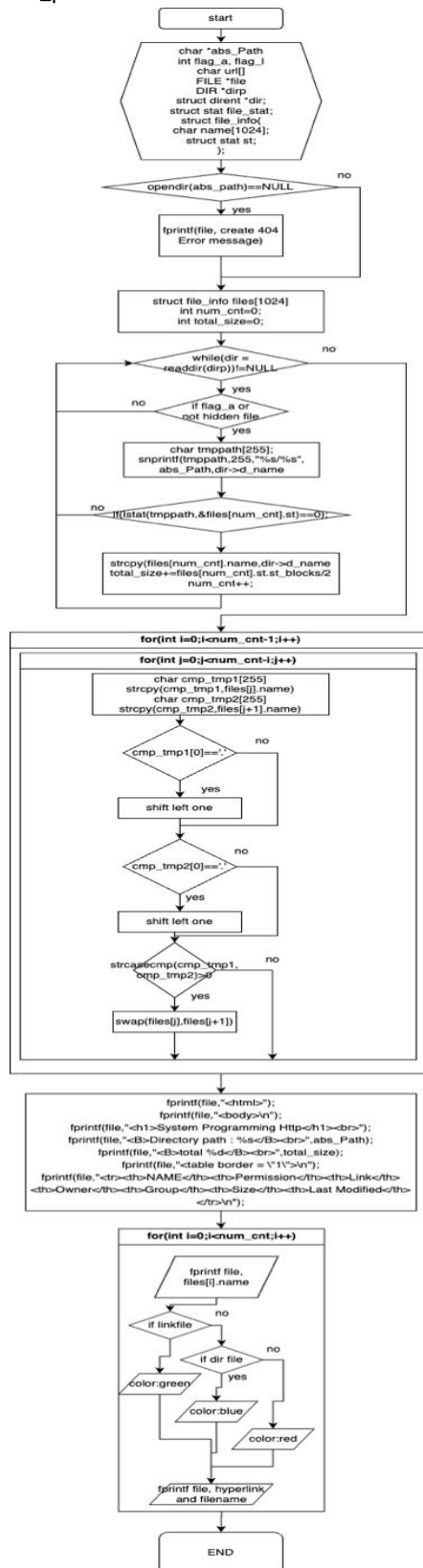
파일의 마지막 수정 일자와 시간을 출력하는 함수이다.

E. print_long



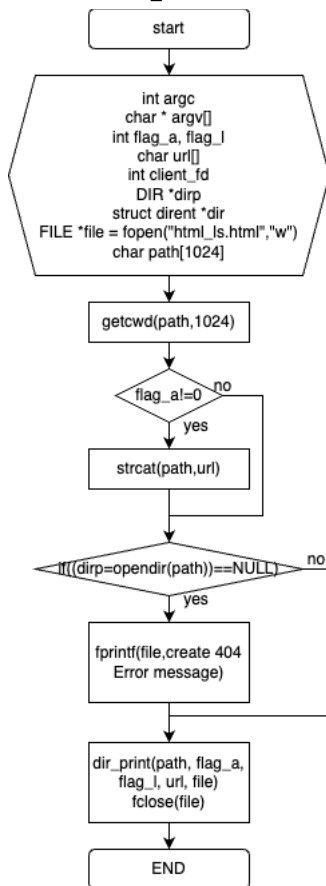
파일의 정보를 long format에 맞춰서 출력하는 함수이다.현재 working directory와 url 정보를 합쳐서 파일에 접근한다.

F. dir_print



입력받은 인자가 경로(directory)인 경우 실행하는 함수이다. 없는 경로인 경우 404 error html을 작성하고 정상 경로인 경우 ls -l or -al의 결과를 html_ls.html에 작성한다.

G. create_ls



ls를 구현하기 위한 중간 단계의 함수로 directory가 존재하지 않는 경우 404 error message를 작성하고 아니라면 flag_a와 flag_l에 따라 ls의 결과를 작성하는 dir_print를 호출한다.

[illegible]

3. Pseudo code

A. print_permission

```
fprint file, <td>
if file is dir, print "d" else "-"
if have read permission by user, fprint file, "r" else "-"
if have write permission by user, fprint file, "w" else "-"
if have execute permission by user, fprint file, "x" else "-"
if have read permission by group, fprint file, "r" else "-"
if have write permission by group, fprint file, "w" else "-"
if have execute permission by group, fprint file, "x" else "-"
if have read permission by other, fprint file, "r" else "-"
if have write permission by other, fprint file, "w" else "-"
if have execute permission by other, fprint file, "x" else "-"
```

B. print UID

```
get struct passwd pointer pw
if pw=getpwuid(uid) is not NULL
    fprint file, pw->pw_name
else
    fprint file, uid
```

C. print GID

```
get struct group pointer gr
if gr=getgrgid(gid) is not NULL
    fprint file, pw -> pw_name
else
    fprint file, gid
```

D. print_LastModified_time

```
make array 80
get struct tm pointer time_tm
time_tm is localtime(&time)
    run strftime to get date and time format
fprint file, time
```

E. print_long

```
getcurrent working directory to cwd
count cwd's slash
if dir_name == "."
    tmppath=abs_path
else
    tmppath = abs_path,dir_name
```

```
get relative directory about root directory
```

```
use print_permission function
```

```
fprint file, file_stat, st_nlink
```

```
use print_UID function
```

```
use print_GID function
```

```
if -h option activated
```

```
    use print_readableSIZE function
```

```
else
```

```
    just fprint file, file_stat.st_size
```

```
use print_LastModified_time function
```

```
fprint file, file name
```

F. dir_print

```
struct file_info{
```

```
    char name[1024];
```

```
    struct stat st;
```

```
};
```

```
int num_cnt=0
```

```
int total_size=0
```

```
struct file_info files[1024]
```

```
if dirp=opendir(abs_Path) ==NULL
```

```
    fprintf file, 404 error message
```

```
while dir = readdir(dirp)!=NULL
```

```
    if(flag_a || !hidden file)
```

```
        files[num_cnt].name = dir->d_name;
```

```
        total_size+= files[num_cnt++].st_blocks/2
```

```
sort files ascending order by name
```

```
fprintf file, html_ls.html result
```

```
for(int i=0; i<num_cnt;i++){
```

```
    if linkfile
```

```
        style ="color:green"
```

```
    else if dir file
```

```
        style ="color:blue"
```

```
    else
```

```
        style="color:red"
```


G. create_ls

```
get current working directory to path
if not root path
    strcat(path,url)
if dirp=opendir(path)==NULL
    if checkFile=open(path,O_RDONLY)-1{
        fprintf file, 404 error message html
    }
dir_print(path, flag_a, flag_l, url, file)
fclose(file)
```

H. main

```
struct sockaddr_in server_addr, client_addr
int socket_fd, client_fd
unsigned int len, len_out;

socket_fd=socket(PF_INET, SOCK_STREAM,0)

setsockopt socket_fd, SOL_SOCKET, SO_REUSEPORT|SO_REUSEADDR, &opt, sizeof(opt)

memset(&server_addr,0,sizeof(server_addr))
server_addr.sin_family = AF_INET
server_addr.sin_addr.s_addr = htonl(INADDR_ANY)
server_addr.sin_port = htons(PORTNO)

bind socket_fd, (struct sockaddr *)&server_addr,sizeof(server_addr)
listen socket_fd 5
while(1)
    struct in_addr inet_client_address
    len = sizeof(client_addr)
    client_fd = accept socket_fd, (struct sockaddr*)&client_addr, &len
    inet_client_address.s_addr = client_addr.sin_addr.s_addr
    ssize_t num_bytes = read(client_fd, buf, BUFSIZE)
    if num_bytes <=0
        continue
    else if num_bytes > BUFSIZE
        continue

    printf request form
    if method == "GET"
        tok = strtok(tmp," ")
```

```

        strcpy(url, tok)
    if url == '/'
        create_ls(argc, argv, 0, 1, url, client_fd)
    else
        create_ls(argc, argv, 1, 1, url, client_fd)

    if not directory
        if not file
            make 404 error text/html response header
            write(client_fd, response_header, strlen(response_header))
            send(client_fd, fd, &offset, filesize)
            continue
        else
            if image file
                make 200 OK image/* response header
                write(client_fd, response_header, strlen(response_header))
                send(client_fd, fd, &offset, filesize)
                continue
            else
                if link file
                    if original of linkfile == NULL
                        make 404 error text/html response header
                        write(client_fd, response_header, strlen(response_header))
                        send(client_fd, fd, &offset, filesize)
                        continue
                    else
                        if image file
                            make 200 OK image/* response header
                            write(client_fd, response_header, strlen(response_header))
                            send(client_fd, fd, &offset, filesize)
                            continue
                        else
                            make 200 OK text/plain response header
                            write(client_fd, response_header, strlen(response_header))
                            send(client_fd, fd, &offset, filesize)
                            continue
                        else
                            make 200 OK text/plain response header
                            write(client_fd, response_header, strlen(response_header))
                            send(client_fd, fd, &offset, filesize)
                            continue
                    else
                        make 200 OK text/html response header

```

```

write(client_fd, response_header, strlen(response_header)
send(client_fd,fd,&offset, filesize)
continue

```

각 함수에 대한 설명은 위의 flow chart에서 설명한 것과 동일하다.

4. 결과화면

A. Web_server 실행

```

kw2019202021@ubuntu: ~/CS_2_2
kw2019202021@ubuntu:~$ cd CS_2_2
kw2019202021@ubuntu:~/CS_2_2$ make
gcc -o web_server 2019202021_web_server.c
kw2019202021@ubuntu:~/CS_2_2$ ./web_server

```

code를 gcc -o web_server 2019202021_web_server.c로 컴파일 후 ./web_server를 실행하였다.

B. Ip 확인

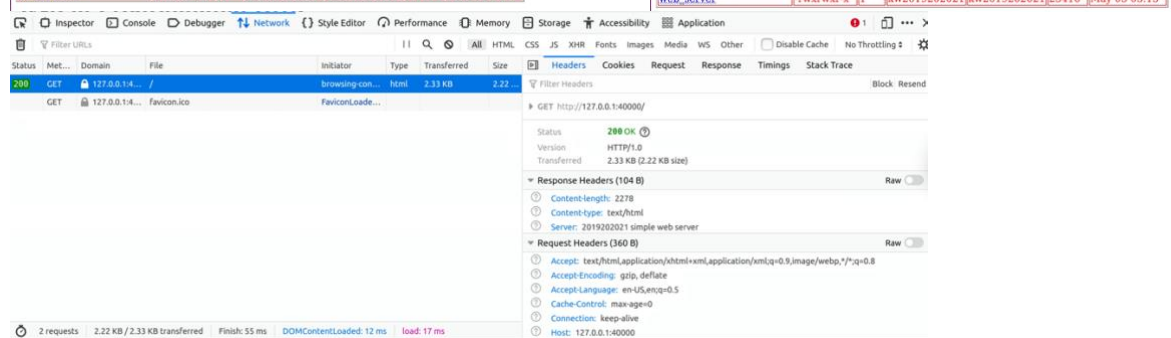
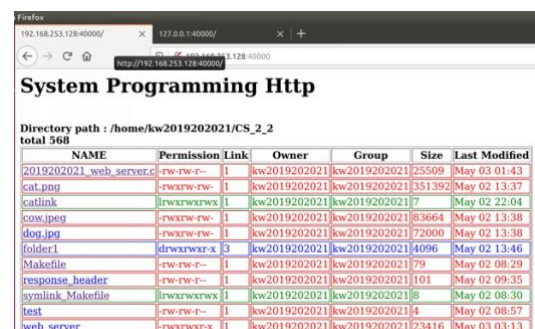
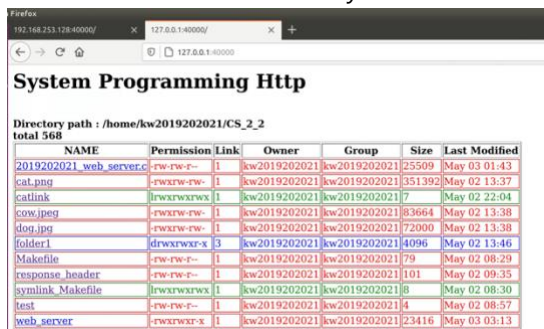
```

kw2019202021@ubuntu:~$ ifconfig | grep inet
inet addr:192.168.253.128 Bcast:192.168.253.255 Mask:255.255.255.0
inet6 addr: fe80::1e03:d80f:bbd9:81a9/64 Scope:Link
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host

```

hostname(192.168.253.128)과 ip address(127.0.0.1)이고 이 둘에 대해 모두 접속해볼 것이다.

C. Root Directory



hostname(192.168.253.128)과 ip address(127.0.0.1)에서 40000포트에 접속하였을 때 모두 정상적으로 작동하였다. 개발자모드를 통해 response header에서 type이 text/html임을 확인하였다. directory는 파란색, linkfile은 초록색, 다른 것은 빨간 색임을 확인할 수 있다. html_js.html는 출력하지 않도록 하였다.

D. 하위 Directory 클릭

NAME	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 02 13:46
..	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 03 03:13
cat.png	-rwxrw-rw-	1	kw2019202021	kw2019202021	351392	May 02 13:37
doc1	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:44
doc2	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:44
doc3	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:45
folder1-1	drwxrwxr-x	2	kw2019202021	kw2019202021	4096	May 02 13:46

NAME	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 02 13:46
..	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 03 03:13
cat.png	-rwxrw-rw-	1	kw2019202021	kw2019202021	351392	May 02 13:37
doc1	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:44
doc2	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:44
doc3	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:45
folder1-1	drwxrwxr-x	2	kw2019202021	kw2019202021	4096	May 02 13:46

hostname(192.168.253.128)과 ip address(127.0.0.1)에서 40000/folder1에 접근했을 때 모두 정상적으로 이동하였다.

E. 다음 하위 Directory 클릭

NAME	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	2	kw2019202021	kw2019202021	4096	May 02 13:46
..	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 02 13:46
dog.jpg	-rwxrw-rw-	1	kw2019202021	kw2019202021	72000	May 02 13:38
txt1	-rw-rw-r--	1	kw2019202021	kw2019202021	12	May 02 13:46

NAME	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	2	kw2019202021	kw2019202021	4096	May 02 13:46
..	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 02 13:46
dog.jpg	-rwxrw-rw-	1	kw2019202021	kw2019202021	72000	May 02 13:38
txt1	-rw-rw-r--	1	kw2019202021	kw2019202021	12	May 02 13:46

hostname(192.168.253.128)과 ip address(127.0.0.1)에서 40000/folder1/folder1-1에 접근했을 때 모두 정상적으로 이동하였다.

F. 현재 디렉토리(.) 클릭

NAME	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	2	kw2019202021	kw2019202021	4096	May 02 13:46
..	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 02 13:46
dog.jpg	-rwxrw-rw-	1	kw2019202021	kw2019202021	72000	May 02 13:38
txt1	-rw-rw-r--	1	kw2019202021	kw2019202021	12	May 02 13:46

NAME	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	2	kw2019202021	kw2019202021	4096	May 02 13:46
..	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 02 13:46
dog.jpg	-rwxrw-rw-	1	kw2019202021	kw2019202021	72000	May 02 13:38
txt1	-rw-rw-r--	1	kw2019202021	kw2019202021	12	May 02 13:46

..을 클릭하였을 때 dir_name을 제외한 경로를 하이퍼링크로 두었기 때문에 제자리임을 확인할 수 있다.

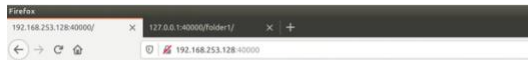
G. 상위 디렉토리(..) 클릭

NAME	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 02 13:46
..	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 03 03:13
cat.png	-rwxrw-rw-	1	kw2019202021	kw2019202021	351392	May 02 13:37
doc1	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:44
doc2	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:44
doc3	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:45
folder1-1	drwxrwxr-x	2	kw2019202021	kw2019202021	4096	May 02 13:46

NAME	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 02 13:46
..	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 03 03:13
cat.png	-rwxrw-rw-	1	kw2019202021	kw2019202021	351392	May 02 13:37
doc1	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:44
doc2	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:44
doc3	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 02 10:45
folder1-1	drwxrwxr-x	2	kw2019202021	kw2019202021	4096	May 02 13:46

..의 상위 디렉토리를 클릭하였을 때 40000/folder1에 접근하는 것을 확인할 수 있다.

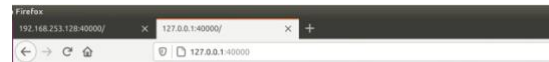
H. 상위 디렉토리(..) 클릭



System Programming Http

Directory path : /home/kw2019202021/CS_2_2
total 568

NAME	Permission	Link	Owner	Group	Size	Last Modified
2019202021_web_server.c	-rw-rw-r--	1	kw2019202021	kw2019202021	25509	May 03 01:43
cat.png	-rw-rw-r--	1	kw2019202021	kw2019202021	351392	May 02 13:37
catlink	lrwxrwxrwx	1	kw2019202021	kw2019202021	7	May 02 22:04
cow.jpeg	-rw-rw-r--	1	kw2019202021	kw2019202021	83664	May 02 13:38
dog.jpg	-rw-rw-r--	1	kw2019202021	kw2019202021	72000	May 02 13:38
folder1	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 02 13:46
Makefile	-rw-rw-r--	1	kw2019202021	kw2019202021	79	May 02 08:29
response_header	-rw-rw-r--	1	kw2019202021	kw2019202021	101	May 02 09:35
symlink_Makefile	lrwxrwxrwx	1	kw2019202021	kw2019202021	8	May 02 08:30
test	-rw-rw-r--	1	kw2019202021	kw2019202021	4	May 02 08:57
web_server	-rwxrwxr-x	1	kw2019202021	kw2019202021	23416	May 03 03:13



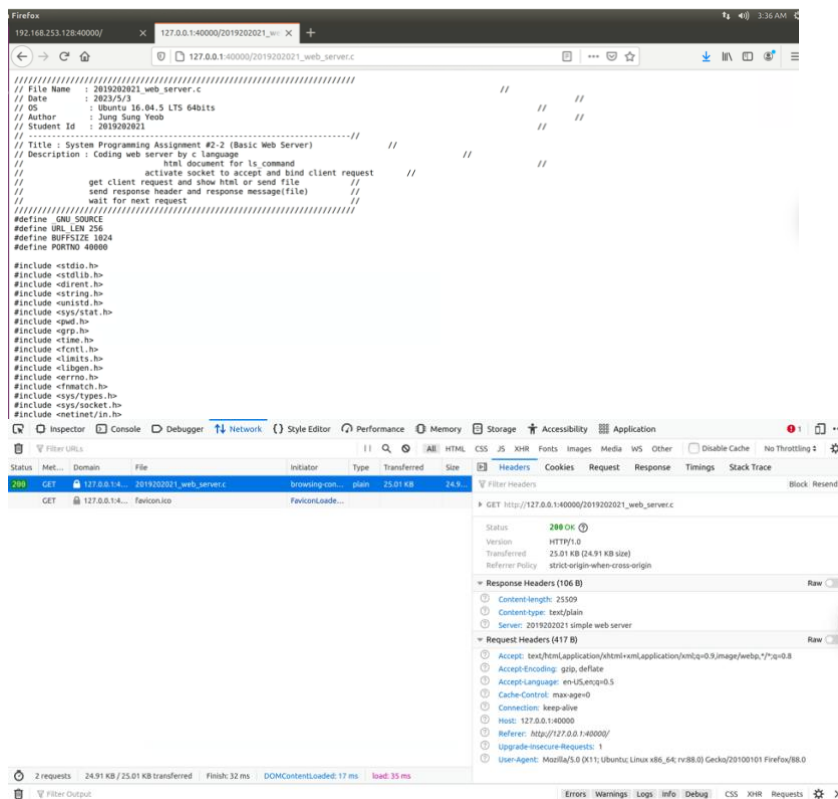
System Programming Http

Directory path : /home/kw2019202021/CS_2_2
total 568

NAME	Permission	Link	Owner	Group	Size	Last Modified
2019202021_web_server.c	-rw-rw-r--	1	kw2019202021	kw2019202021	25509	May 03 01:43
cat.png	-rw-rw-r--	1	kw2019202021	kw2019202021	351392	May 02 13:37
catlink	lrwxrwxrwx	1	kw2019202021	kw2019202021	7	May 02 22:04
cow.jpeg	-rw-rw-r--	1	kw2019202021	kw2019202021	83664	May 02 13:38
dog.jpg	-rw-rw-r--	1	kw2019202021	kw2019202021	72000	May 02 13:38
folder1	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 02 13:46
Makefile	-rw-rw-r--	1	kw2019202021	kw2019202021	79	May 02 08:29
response_header	-rw-rw-r--	1	kw2019202021	kw2019202021	101	May 02 09:35
symlink_Makefile	lrwxrwxrwx	1	kw2019202021	kw2019202021	8	May 02 08:30
test	-rw-rw-r--	1	kw2019202021	kw2019202021	4	May 02 08:57
web_server	-rwxrwxr-x	1	kw2019202021	kw2019202021	23416	May 03 03:13

또 ..의 상위 디렉토리를 연속으로 클릭하여도 상위 디렉토리를 정상적으로 접근하는 것을 확인할 수 있다. 이때 포트는 40000임 또한 확인할 수 있다.

I. text or source code 파일 클릭



C언어 코드를 클릭하였을 때 해당 content-type이 text/plain으로 들어와 브라우저에 display하는 것을 확인할 수 있다. Status 또한 200으로 헤더가 정상적으로 전달되었다.

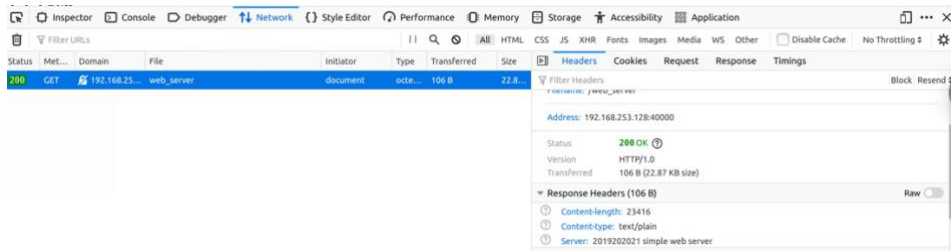
J. 실행 파일(바이너리 파일) 클릭



System Programming Http

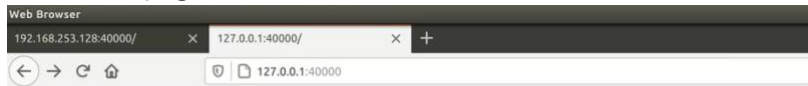
Directory path : /home/kw2019202021/CS_2_2
total 568

NAME	Permission	Link	Owner	Group	Size	Last Modified
2019202021_web_server.c	-rw-rw-r--	1	kw2019202021	kw2019202021	25509	May 03 01:43
cat.png	-rw-rw-r--	1	kw2019202021	kw2019202021	351392	May 02 13:37
catlink	lrwxrwxrwx	1	kw2019202021	kw2019202021	7	May 02 22:04
cow.jpeg	-rw-rw-r--	1	kw2019202021	kw2019202021	83664	May 02 13:38
dog.jpg	-rw-rw-r--	1	kw2019202021	kw2019202021	72000	May 02 13:38
folder1	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 02 13:46
Makefile	-rw-rw-r--	1	kw2019202021	kw2019202021	79	May 02 08:29
response_header	-rw-rw-r--	1	kw2019202021	kw2019202021	101	May 02 09:35
symlink_Makefile	lrwxrwxrwx	1	kw2019202021	kw2019202021	8	May 02 08:30
test	-rw-rw-r--	1	kw2019202021	kw2019202021	4	May 02 08:57
web_server	-rwxrwxr-x	1	kw2019202021	kw2019202021	23416	May 03 03:13



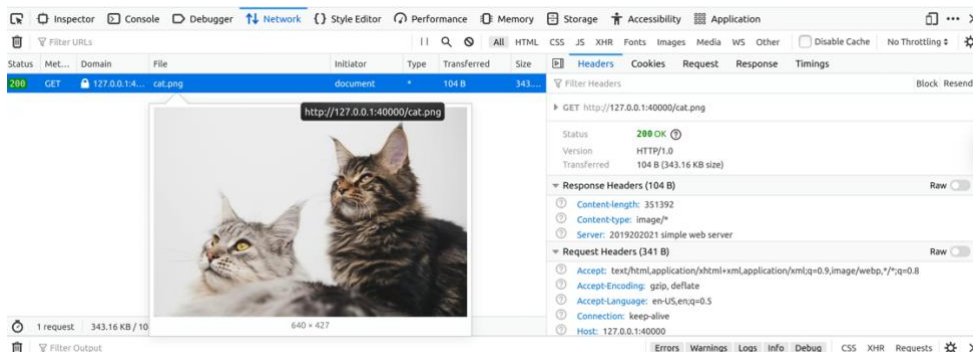
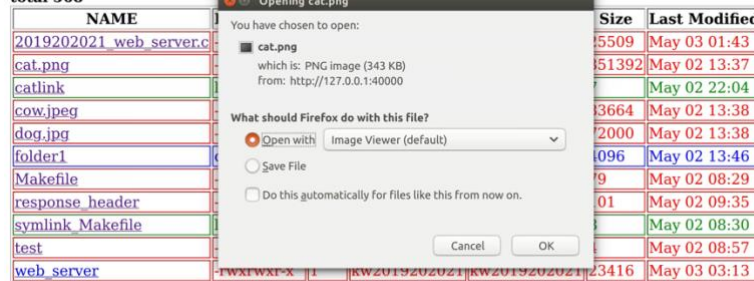
바이너리 파일을 클릭했을 때 Content-type이 text/plain이지만 실행 파일임으로 브라우저에 display가 되는 것이 아닌 download를 진행하는 것을 확인할 수 있다.

K. 사진 png 클릭



System Programming Http

Directory path : /home/kw2019202021/CS_2_2
total 568



System Programmin

Directory path : /home/kw2019202021/C
total 568

NAME	Permission	Lin
2019202021_web_server.c	-rw-rw-r--	1
cat.png	-rwxrwx-rw-	1
catlink	lrwxrwxrwx	1
cow.jpeg	-rwxrwx-rw-	1
dog.jpg	-rwxrwx-rw-	1
folder1	drwxrwxr-x	3
Makefile	-rw-rw-r--	1
response_header	-rw-rw-r--	1
symlink_Makefile	lrwxrwxrwx	1
test	-rw-rw-r--	1
web_server	-rwxrwxr-x	1



png 파일을 클릭했을 때 png image임을 구별하여 opening option을 주고 open with image viewer(default)로 진행하면 아래와 같이 사진이 출력됨을 확인할 수 있다. status는 200 OK이고 image/*로 content-type이 헤더로 전달됨을 확인할 수 있다.

L. 사진 jpeg 클릭

Web Browser

192.168.253.128:40000/ x 127.0.0.1:40000/ x +

192.168.253.128:40000

System Programming Http

Directory path : /home/kw2019202021/CS_2_2
total 568

NAME	Permissions
2019202021_web_server.c	-rw-rw-r--
cat.png	-rw-rw-r--
catlink	lrwxrwxrwx
cow.jpeg	-rw-rw-r--
dog.jpg	-rw-rw-r--
folder1	drwxrwxr-x
Makefile	-rw-rw-r--
response_header	-rw-rw-r--
symlink_Makefile	lrwxrwxrwx
test	-rw-rw-r--
web_server	-rw-rw-r--

Opening cow.jpeg

You have chosen to open:


cow.jpeg
which is: JPEG image (81.7 KB)
from: http://192.168.253.128:40000

What should Firefox do with this file?

☒ Open with Image Viewer (default)

☐ Save File

☐ Do this automatically for files like this from now on.



640 x 427

1 request 81.70 KB / 103 B transferred Finish: 1 ms

192.168.253.128:40000/ x 127.0.0.1:40000/ x +


192.168.253.128:40000

System Programming

Directory path : /home/kw2019202021/CS_2_2
total 568

NAME	Permissions	Link
2019202021_web_server.c	-rw-rw-r--	1
cat.png	-rw-rw-r--	1
catlink	lrwxrwxrwx	1
cow.jpeg	-rw-rw-r--	1
dog.jpg	-rw-rw-r--	1
folder1	drwxrwxr-x	3
Makefile	-rw-rw-r--	1
response_header	-rw-rw-r--	1
symlink_Makefile	lrwxrwxrwx	1
test	-rw-rw-r--	1
web_server	-rw-rw-r--	1

cow-1.jpeg



80%

192.168.253.128:40000/ x 127.0.0.1:40000/ x +

192.168.253.128:40000

GET

Schema: http
Host: 192.168.253.128:40000
Filename: /cow.jpeg
Address: 192.168.253.128:40000

Status: 200 OK
Version: HTTP/1.0
Transferred: 103 B (81.70 KB size)
Referrer Policy: strict-origin-when-cross-origin

Response Headers (103 B)

- Content-length: 83664
- Content-type: image/*
- Server: 2019202021 simple web server

Request Headers (388 B)

png와 마찬가지로 jpeg를 클릭했을 때 정상적으로 다운로드가 가능했으며 status는 200 OK, Content-type은 image/*로 정상적으로 header가 전달되었다.

M. 사진 jpg 클릭

Web Browser

192.168.253.128:40000/ x 127.0.0.1:40000/ x +

192.168.253.128:40000

System Programming Http

Directory path : /home/kw2019202021/CS_2_2
total 568

NAME	Permissions
2019202021_web_server.c	-rw-rw-r--
cat.png	-rw-rw-r--
catlink	lrwxrwxrwx
cow.jpeg	-rw-rw-r--
dog.jpg	-rw-rw-r--
folder1	drwxrwxr-x
Makefile	-rw-rw-r--
response_header	-rw-rw-r--
symlink_Makefile	lrwxrwxrwx
test	-rw-rw-r--
web_server	-rw-rw-r--

Opening dog.jpg

You have chosen to open:

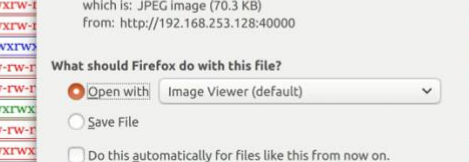
dog.jpg
which is: JPEG image (70.3 KB)
from: http://192.168.253.128:40000

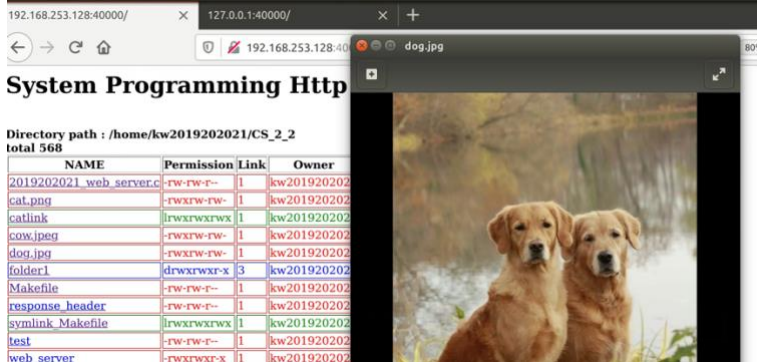
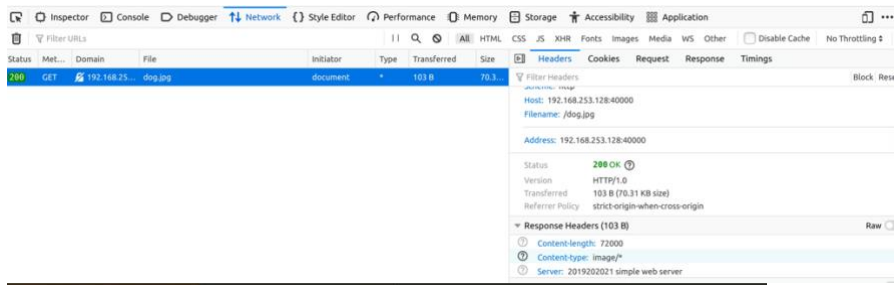
What should Firefox do with this file?

☒ Open with Image Viewer (default)

☐ Save File

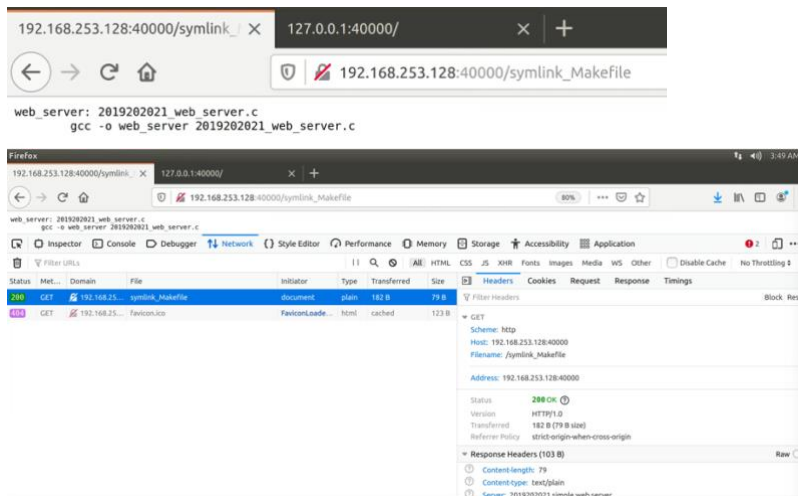
☐ Do this automatically for files like this from now on.





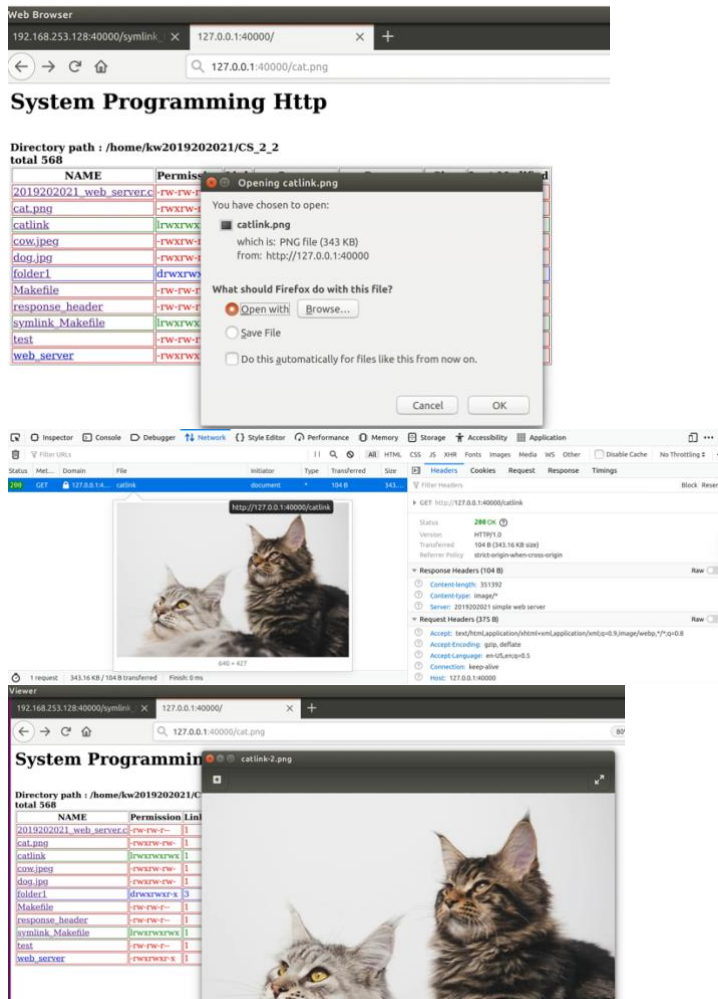
jpg를 클릭했을 때도 200 OK, Content-type image/* 로 정상적으로 헤더 전송과 파일 다운로드가 진행되었으며 특이점은 jpg 파일이지만 다운로드할 때 구분은 jpeg로 한다는 점이다. 이는 고찰에서 다시 다뤄본다.

N. source code에 대한 링크 파일 클릭



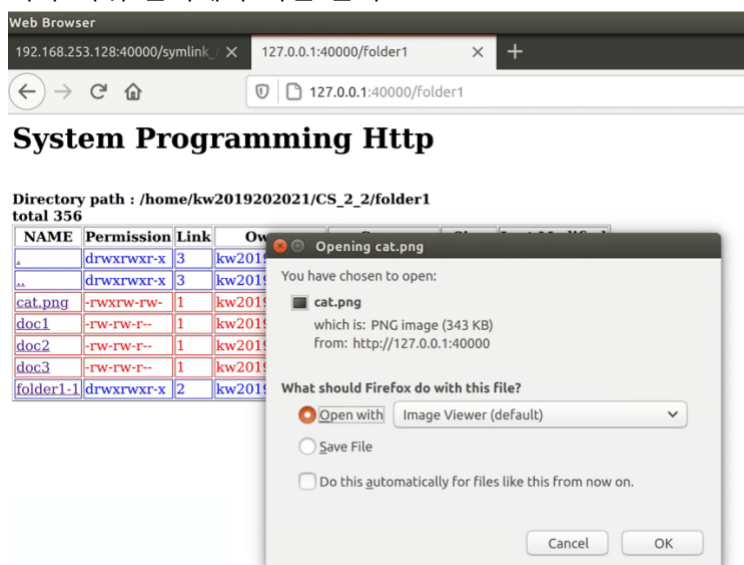
Makefile에 대한 symbolic link를 클릭했을 때 결과이며 200 OK, Content_type text/plain으로 원본을 가져와 전송했기 때문에 Makefile의 내용이 브라우저에 display 되었다.

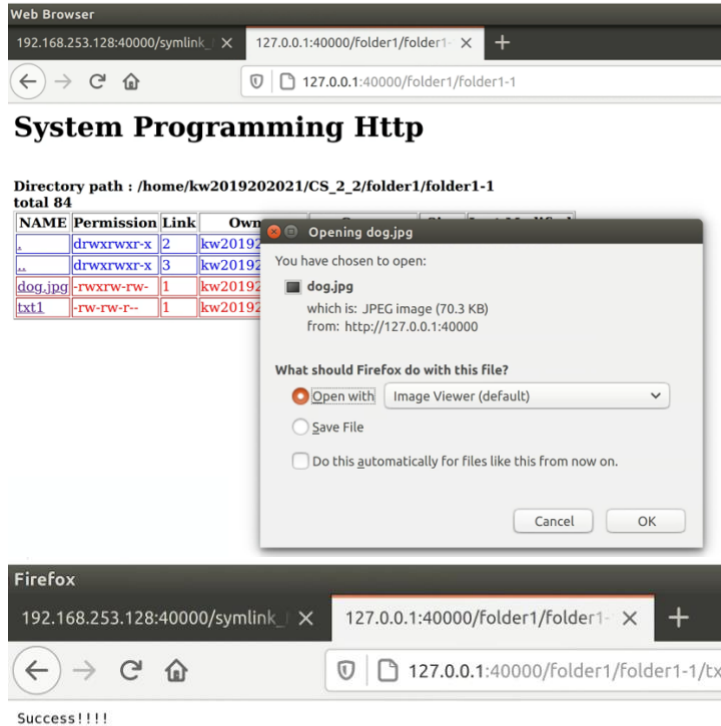
O. image에 대한 링크 파일 클릭



이미지에 대한 symbolic link를 클릭하였을 때 결과이다. symbolic link를 단순 text/plain으로 하는 것이 아니라 원본에 접근하였기 때문에 원본 이름인 cat.png 이름으로 구별하여 image/*로 헤더를 보냈으며 대신 이름은 catlink로 다운받도록 하였다.

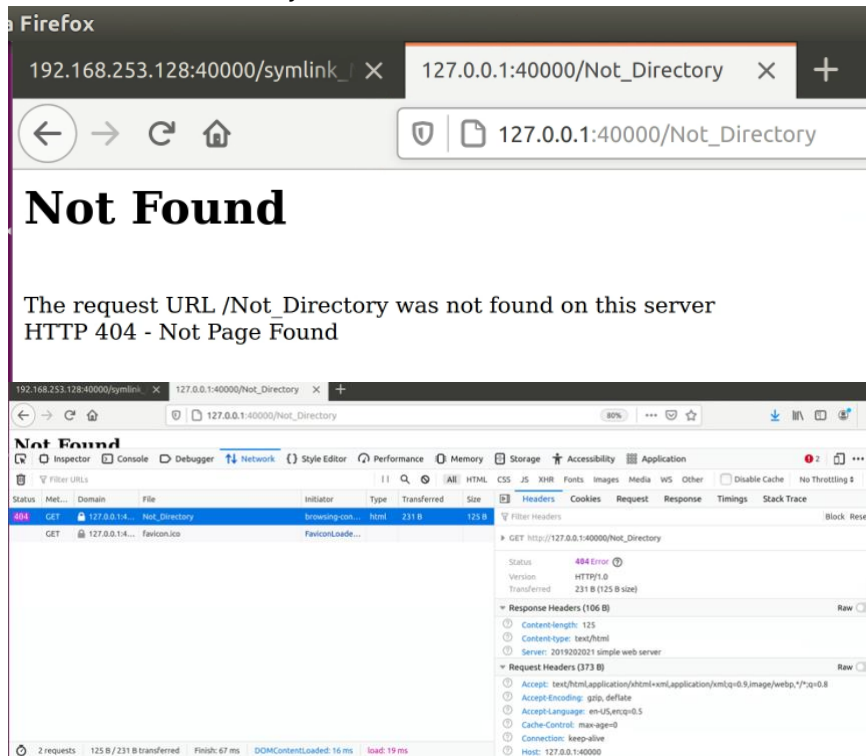
P. 기타 하위 폴더에서 파일 클릭





기타 하위 폴더에서도 파일들을 접근하면 문제없이 다운로드하거나 브라우저에 display하는 것을 확인하였다.

Q. 존재하지 않는 directory 접근 시



디렉토리가 아닌 곳으로 접근하였을 때 status는 404 Error가 되었으며 html 파일을 미리 작성하였기 때문에 content-type을 text/html로 지정하여 헤더로 전달하였다. 정해진 양식대로 오류가 출력됨을 확인할 수 있다.

5. 고찰

Assignment 2-2에서는 수업시간에 배운 socket을 바탕으로 리눅스 환경에 서버를 두어 브라우저에서 접근하여 ls의 결과를 html로 브라우저에 출력하거나 파일 다운로드 또는 출력을 하도록 설계하였다. 이 때, 수업시간이나 실습 수업에서는 response_header나 response_message는 write 함수로만 보냈지만, 파일 용량이 큰 경우 write만으로 전송하기에는 BUFSIZE가 작을 때는 직접 이미지를 바이너리 모드로 읽어 BUFSIZE만큼 패킷처럼 보내야하는 불편함이 있었고, 라이브러리에 대한 제한이 제안서에 없었기 때문에 한번에 처리할 수 있는 sendfile() 함수를 사용하여 설계하였다. 이를 통해 사이즈가 큰 이미지를 다른 처리를 하지 않고 보낼 수 있었다.

이미지를 보내는 중 한 가지 문제가 아닌 문제점이 있었다면 jpg 파일을 전송하고 다운로드 받을 때 firefox 브라우저에서 jpg 파일을 jpeg 파일처럼 읽고 표기한다는 점이였다. 이에 처음에는 asterisk를 사용하는 것이 아닌 각 확장자에 대해 표기를 해 줘야 하는 것인지, 따로 구별할 수 있는 점이 있는지 판별하려고 했으나 제안서에서 bonobono.jpg 또한 확장자가 jpg이지만 브라우저에서 jpeg로 읽는 것을 보고 문제가 없음을 확인하여 그대로 진행하였다. 이에 대해 왜 jpg를 jpeg로 읽는 지에 대해 찾아본 결과 jpg와 jpeg의 바이너리 헤더가 동일하기 때문에 이에 대한 구분을 두지 않고 한번에 처리함을 확인하였다. 이를 magic number라고도 한다.

마지막으로 매번 과제가 ls를 구현했던 것을 바탕으로 과제를 진행하는 데 이미 구현한 것으로 만들기 때문에 잘 활용하는 것이 중요하게 여겨지고 있다. 다행히 일부 출력에 대해 함수로 작게 소분하여 제작하였기 때문에 그대로 가져오는 경우가 있으나 후에 이를 발전시킬 때 또 다시 필요 함수를 만들 필요가 있기 때문에 이번 시스템프로그래밍 수업 내에 코드를 정돈할 수는 없겠지만 후에 다시 대규모 코딩을 진행한다면 미리 구조를 정해서 후에 수정할 때도 편리하게 할 수 있도록 해야 함을 배웠다.

6. Reference

- 1) 시스템프로그래밍실습 Assignment 2-2/광운대학교/컴퓨터정보공학부/김태석교수님/2023
- 2) 시스템프로그래밍 강의자료/광운대학교/컴퓨터정보공학부/김태석교수님/2023
- 3) jpeg signature format/ <https://www.ntfs.com/jpeg-signature-format.htm>