시스템프로그래밍

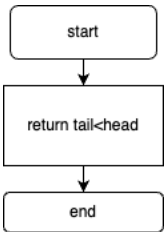
과제이름: Assignment 2-3

- 담당교수:김 태 석 교수님
- 학 과: 컴퓨터정보공학부
- 학 번: 2019202021
- 이 름: 정 성 엽
- 제출일: 2023/5/10

1. Introduction
   A. 과제 소개

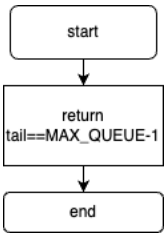
      이번 Assignment 2-3 과제는 이전 Assignment 2-2에서 설계하였던 리눅스 환경에서의 C언어 기반 socket server를 다중 접속과 접근 제어를 지원하는 웹 서버 프로그램을 작성하는 것이다. 이 때 클라이언트와 접속 연결 및 해제 시마다 클라이언트들의 정보를 출력한다. 또한 10초마다 연결되었던 클라이언트 최근 10개에 대한 정보를 출력한다. 마지막으로 접근을 제어하여 허용한 IP를 가진 사용자만 서버에 접속하도록 한다.

2. Flow Chart
   A. is_empty



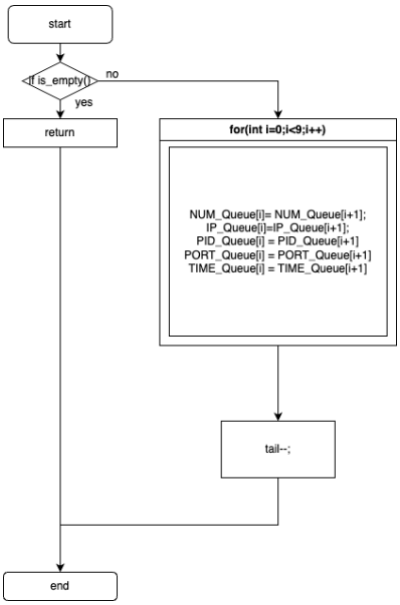
      Queue가 비어 있는지 확인하는 함수이다.

   B. is_full



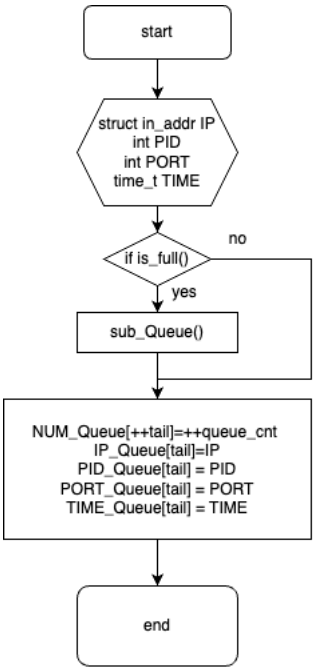
      Queue가 꽉 찼는지 확인하는 함수이다.

   C. sub_Queue



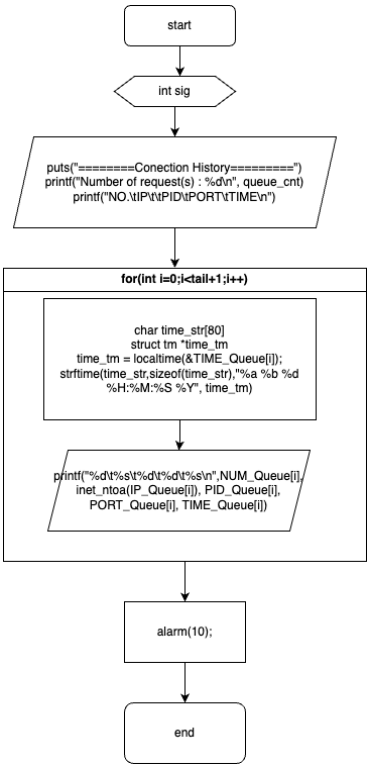
      Queue의 첫 번째 값을 없애는 함수이다.

D. add_Queue



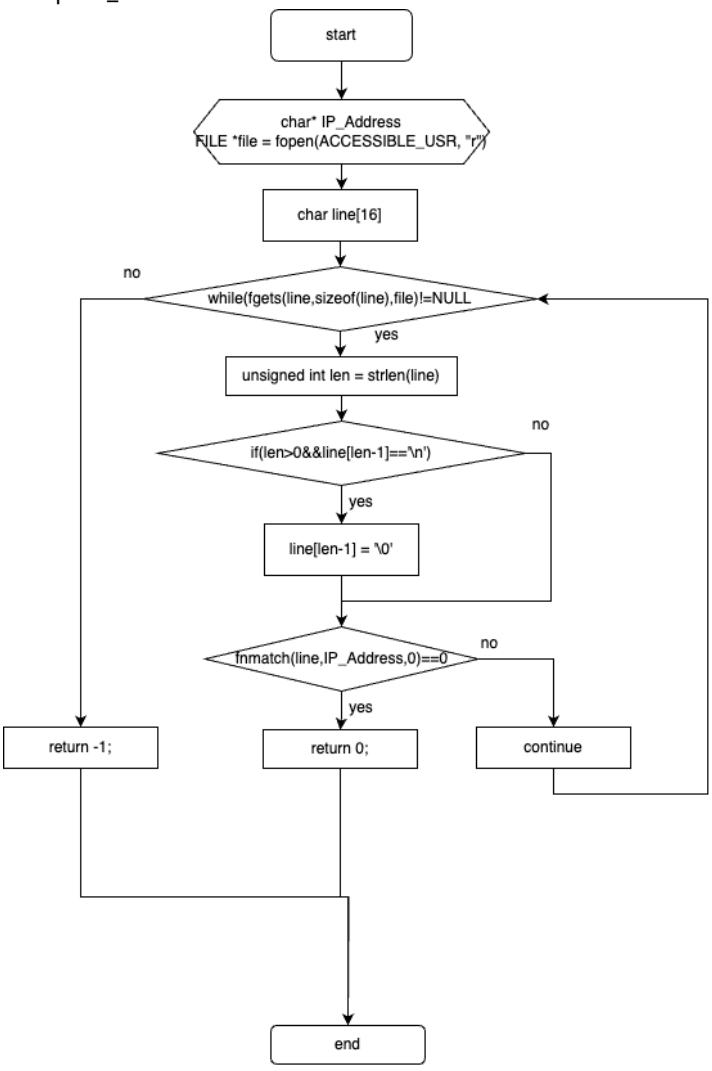
새로운 클라이언트가 연결되었을 때 queue에 연결정보를 저장하는 함수이다.

E. alarm_handler



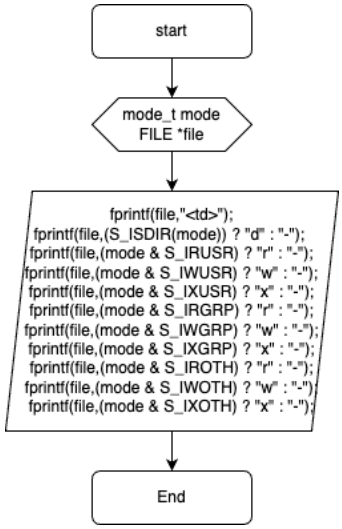
alarm이 10초가 되었을 때 작동하는 함수로 지금까지의 클라이언트 내역 중 최근 10개의 내역을 출력하며 출력 이후에는 새로운 알람을 호출한다.

F. compare_WhiteList



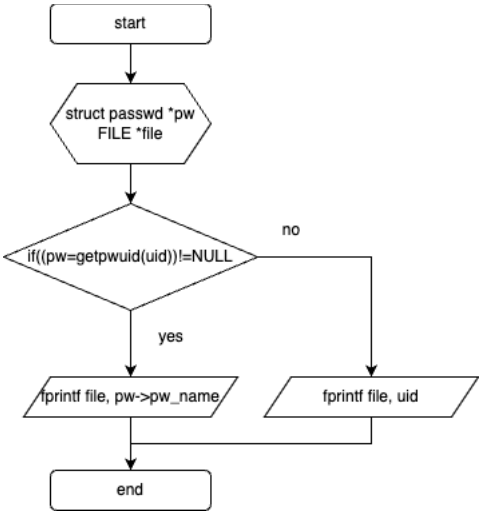
accessible.usr에 저장된 whitelist에 대한 정보를 읽고 패턴을 비교하여 일치하는 경우 0을 반환하고 일치하지 않는 경우 -1을 반환한다.

G. print_permission



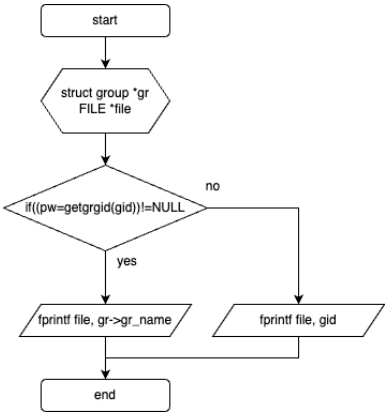
파일의 허용정보를 출력하는 함수이다.

H. print UID



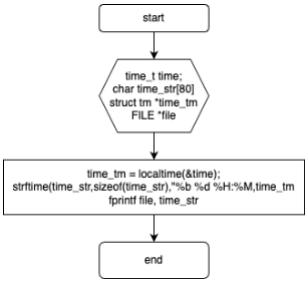
파일의 UID 정보를 출력하는 함수이다.

I. print GID



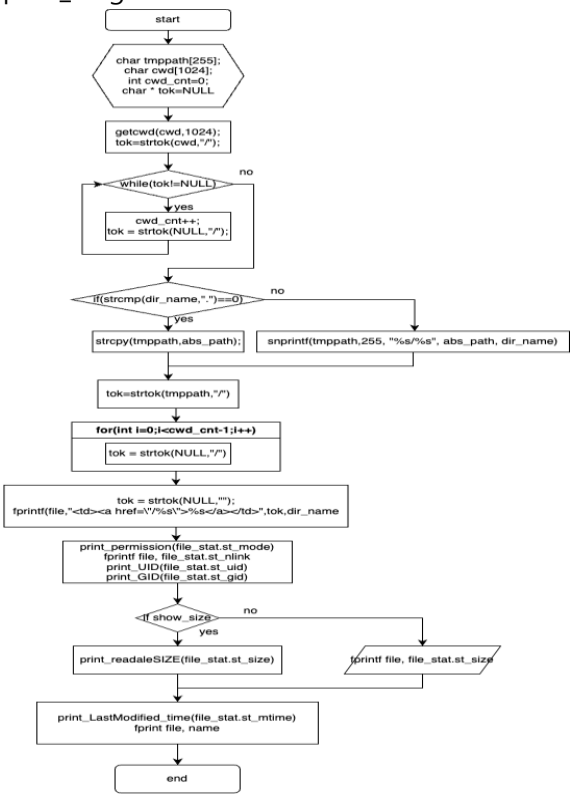
파일의 GID 정보를 출력하는 함수이다.

J. print_LastModified_time



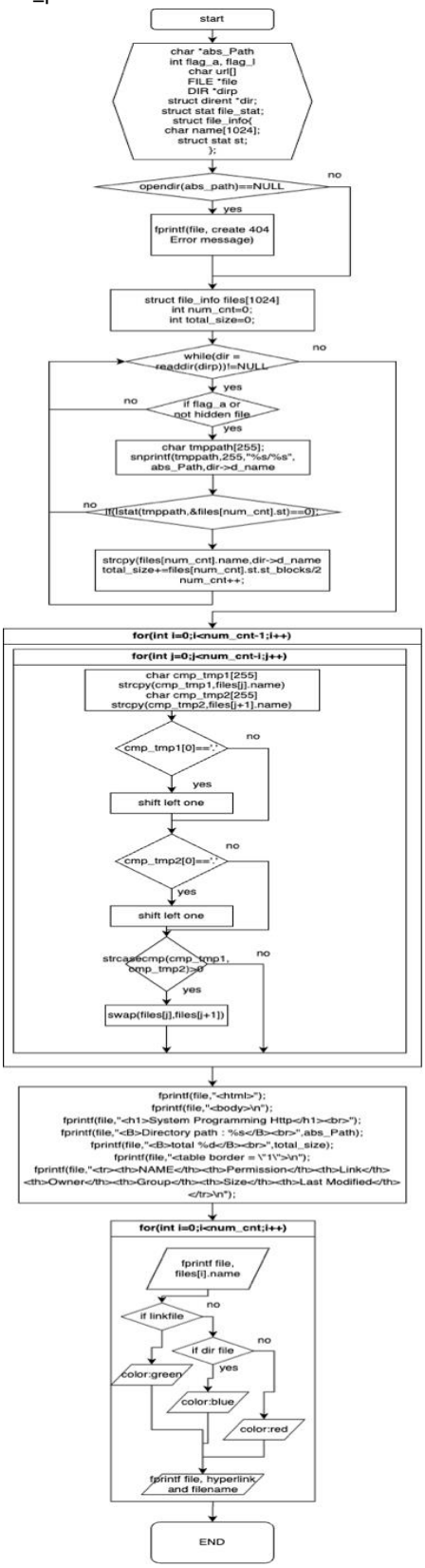
파일의 마지막 수정 일자와 시간을 출력하는 함수이다.

K. print_long



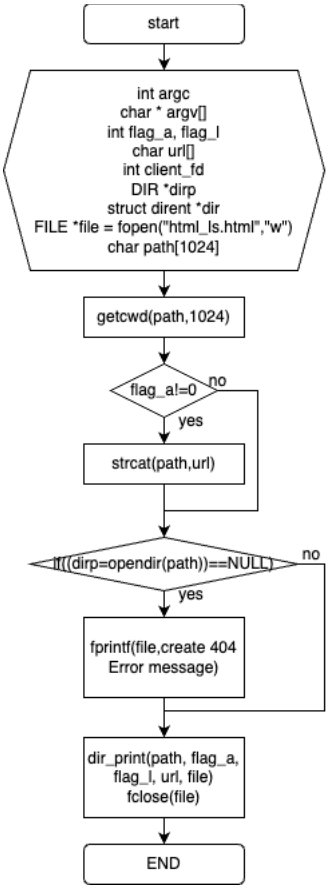
파일의 정보를 long format에 맞춰서 출력하는 함수이다.현재 working directory와 url 정보를 합쳐서 파일에 접근한다.

L. dir_print



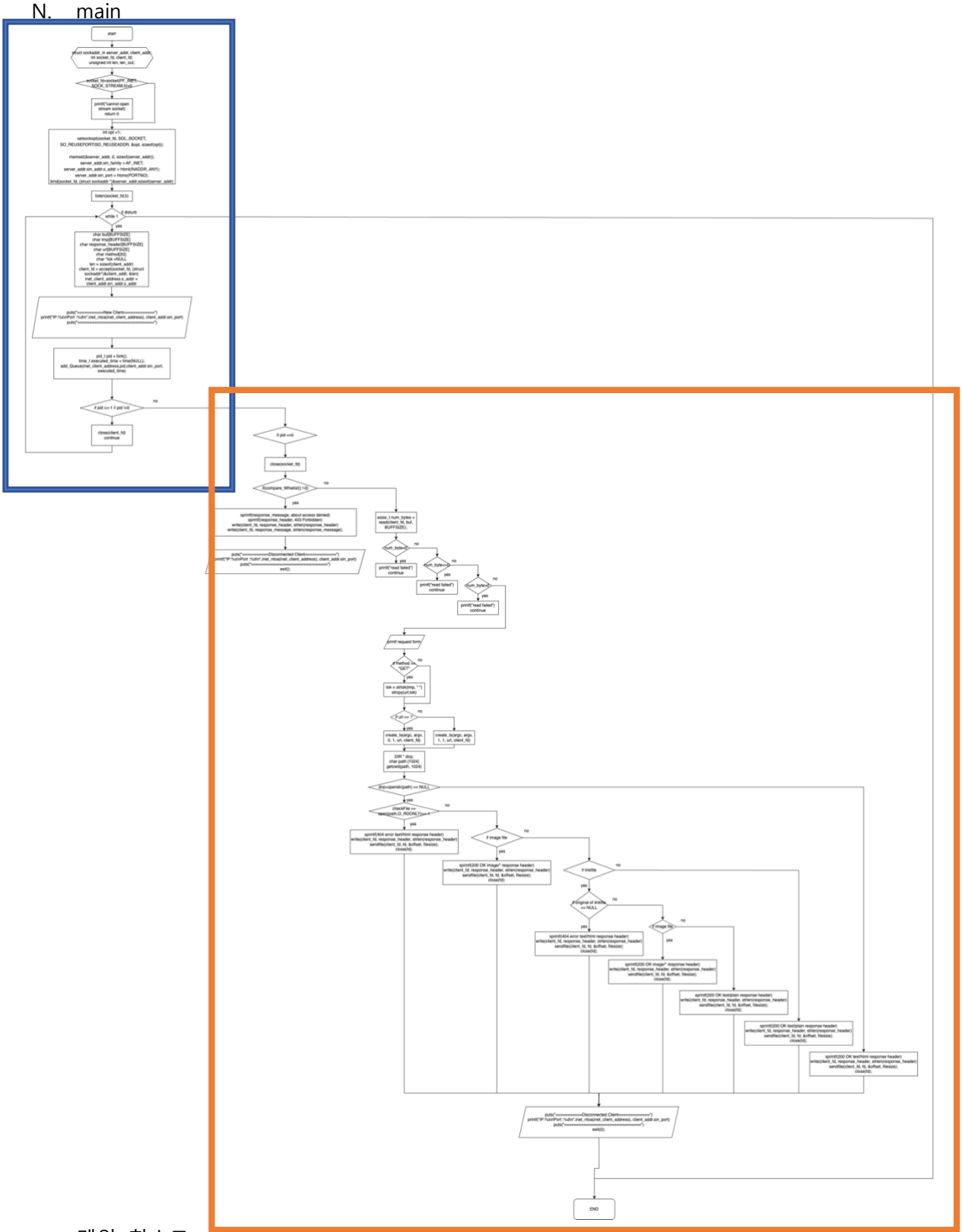
입력받은 인자가 경로(directory)인 경우 실행하는 함수이다. 없는 경로인 경우 404 error html을 작성하고 정상 경로인 경우 ls -l or -al의 결과를 html_ls.html에 작성한다.

M. 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.

N. main



메인 함수로 socket을 통해 client의 요청을 받고 요청에 대한 응답을 보내는 함수이다. IPv4 internet protocol을 받고 TCP 통신을 한다. while(1)문으로 요청을 무수히 받으며 잘못된 request가 넘어가서 buf에 대한 read가 0이하인 경우 다시 요청을 받도록 하고 BUFFSIZE를 넘는 경우에도 다시 받도록 한다. 그 이후 요청에 따른

response를 보내기 전에 fork 함수로 자식 프로세스와 나누고 response는 자식프로세스에서 처리해서 다중 클라이언트 접속이 가능하도록 한다. 또한 accessible.usr에 있는 접근 가능 IP 주소를 확인 후 패턴과 일치하면 접속하도록 하고 아니라면 403 forbidden을 보내도록 한다. response는 directory, 파일, 이미지파일, 링크파일, 등에 따라 나뉘어서 response_header를 작성하고 이를 write로 보내고 해당 파일을 client로 보낸다. 파란색 네모안의 flowchart는 부모 프로세스에서 동작 주황색 네모안의 flow chart는 자식 프로세스의

3. Pseudo Code

A. is_empty

```
return tail<head
```

B. is_full

```
return tail== MAX_QUEUE-1
```

C. void sub_Queue

```
if is_empty(){
        return;
}
for(int i=0;i<0;i++){
        NUM_Queue[i] = NUM_Queue[i+1];
        IP_Queue[i] = IP_Queue[i+1]
        PID_Queue[i] = PID_Queue[i+1]
        PORT_Queue[i] = PORT_Queue[i+1]
        TIME_Queue[i] = TIME_Queue[i+1]
}
tail--
```

D. void add_Queue

```
if is_full(){
        sub_Queue()
}
NUM_Queue[++tail] = ++queue_cnt;
IP_Queue[tail] = IP
PID_Queue[tail] = PID
PORT_Queue[tail] = PORT
TIME_Queue[tail] = TIME
}
```

E. alarm_handler

```
puts("============connection history============")
printf("Number of request(s) : %d₩n" queue_cnt);
printf("NO.₩tIP₩t₩tPID₩tPORT₩tTIME₩n")
for(int i=0; i<tail+1; i++)
```

```
        printf("%d₩t%s₩t%d₩t%d₩t%s₩n",NUM_Queue[i],
inet_ntoa(IP_Queue[i]), PID_Queue[i], PORT_Queue[i], time_str
```

F. compare_WhiteList

```
FILE *file = fopen(ACESSIBLE_USR,"r");
char line[16]
while(fgets(line, sizeof(line),file)!=NULL){
        unsigned int len =strlen(line)
        if(len>0&&line[len-1]=='₩n'
                line[len-1]='₩0'
        if(fnmatch(line,IP_Address,0)==0)
                return 0;
        else
                continue;
fclose(file)
return -1;
```

G. 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, fprintf file "r" else "-"
    if have write permission by other, fprintf file, "w" else "-"
    if have execute permission by other, fprint   file, "x" else "-"
```

H. print UID

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

I. print GID

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

J. 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
```

K. 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
```

L. 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"
```

M. 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{
                fprint file, 404 error message html
dir_print(path, flag_a, flag_l, url, file)
fclose(file)
```

N. 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
```

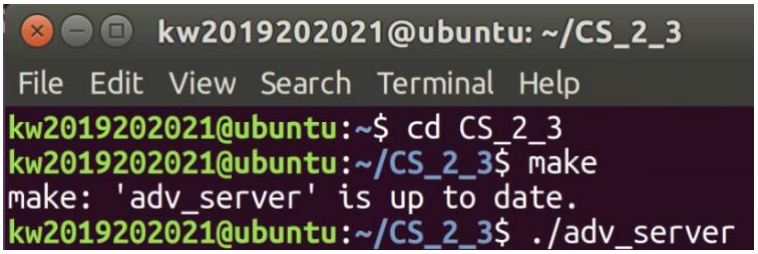
```
while(1)
        struct in_addr inet_client_address
        len = sizeof(client_addr)
        client_fd = accept socket_fd, (struct sockaddr*)&client_addr, &len
        inet_client_address.s_addr = client_addr.sin_addr.s_addr

        puts("============New Client==============")
        printf("IP : %s₩nPORT: %d₩n, inet_ntoa(inet_client_address),client_addr.sin_port)
        puts("====================================")

        pid_t pid=fork()
        time_t executed_time = time(NULL)
        add_Queue(inet_client_address,pid,client_addr.sin_port,executed_time)

if(pid != 0)
        close(client_fd)
        continue;
else if(pid ==0)
        close(socket_fd)
        if(compare_WhiteList(inet_ntoa(client_addr.sin_addr))!=0){
                sprintf(response_message, about access denied)
                sprintf(response_header, about 403 forbidden)
                write(client_fd, response_header, strlen(response_header)
                write(client_fd, response_message, strlen(response_message)
                exit()
        ssize_t num_bytes = read(client_fd, buf, BUFFSIZE)
        if num_bytes <=0
                exit()
        else if num_bytes > BUFFSIZE
                exit()

        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)
        else
                if image file
                        make 200 OK image/* response header
                        write(client_fd, response_header, strlen(response_header)
                        send(client_fd,fd,&offset, filesize)
                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)
                else
                if image file
                        make 200 OK image/* response header
                        write(client_fd, response_header, strlen(response_header)
                        send(client_fd,fd,&offset, filesize)
                else
                        make 200 OK text/plain response header
                        write(client_fd, response_header, strlen(response_header)
                        send(client_fd,fd,&offset, filesize)
                else
                        make 200 OK text/plain response header
                        write(client_fd, response_header, strlen(response_header)
                        send(client_fd,fd,&offset, filesize)
else
        make 200 OK text/html response header
        write(client_fd, response_header, strlen(response_header)
        send(client_fd,fd,&offset, filesize)
puts("============Disconnected Client==============")
printf("IP : %s₩nPORT: %d₩n, inet_ntoa(inet_client_address),client_addr.sin_port)
puts("=====================================")
remove(filename)
exit();
```

각 함수에 대한 설명은 위의 flow chart에서 설명한 것과 동일하다.

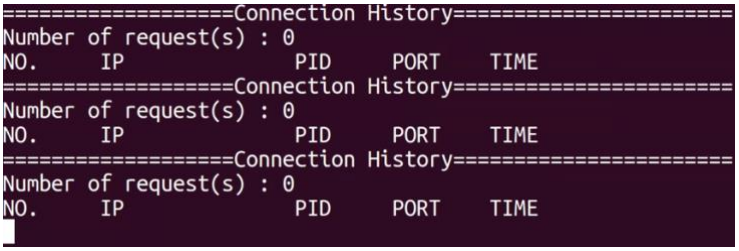
4. 결과화면

A. Web_server 실행



code를 gcc -o adv_server 2019202021_adv_server.c로 컴파일 후 ./adv_server를 실행하였다.

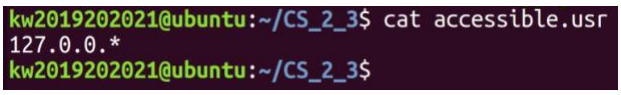
B. Alarm 확인



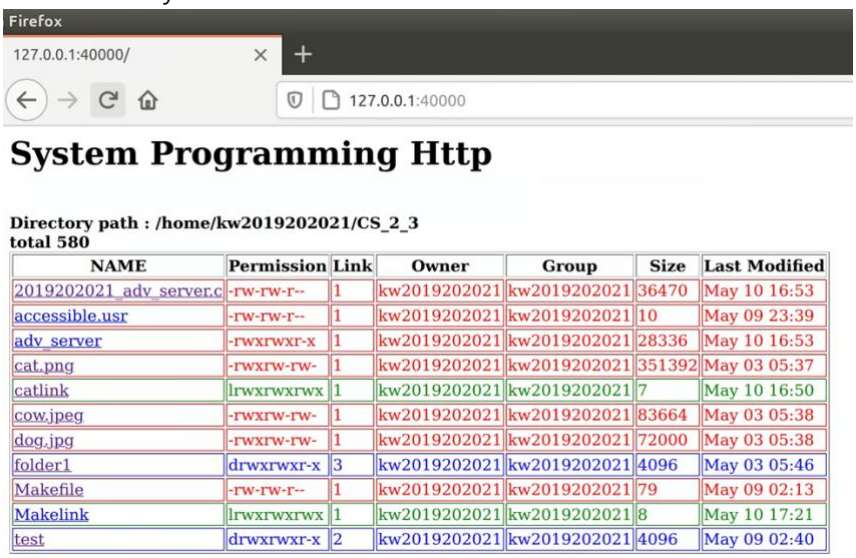
Alarm이 10초마다 connection history를 출력한다. 현재는 request 요청이 있던 적이 없으므로 0개이다.

C. IP 확인

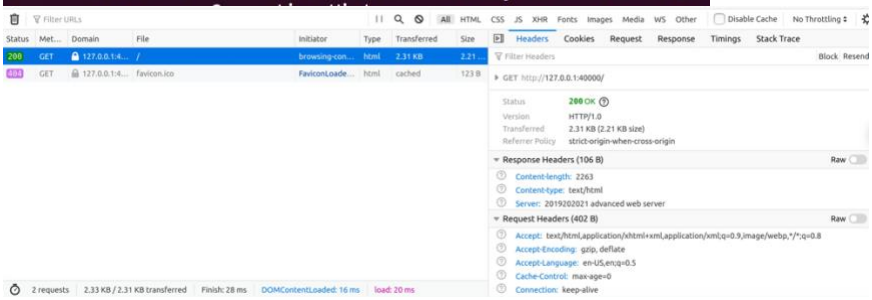
hostname(192.168.253.128)과 ip address(127.0.0.1)이다. 이 때 허용하는 IP address는 127.0.0.*로 한다.



D. Root directory

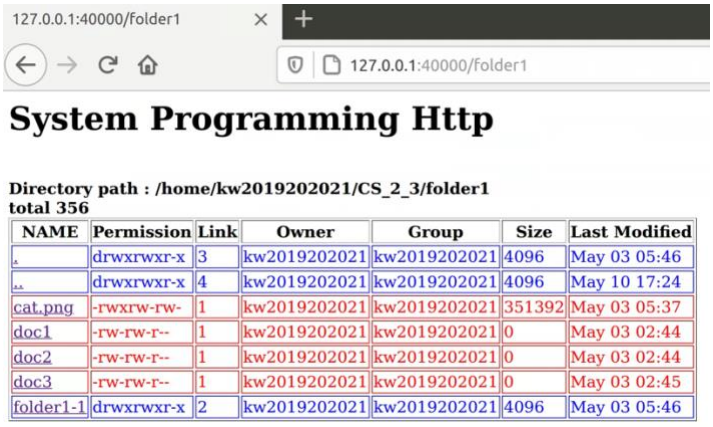


```
====================New Client====================
IP : 127.0.0.1
Port : 22202
==================================================

==============Disconnected Client================
IP : 127.0.0.1
Port : 22202
==================================================
==================Connection History==============
Number of request(s) : 1
NO.     IP              PID     PORT    TIME
1       127.0.0.1       14042   22202   Wed May 10 17:23:27 2023
```

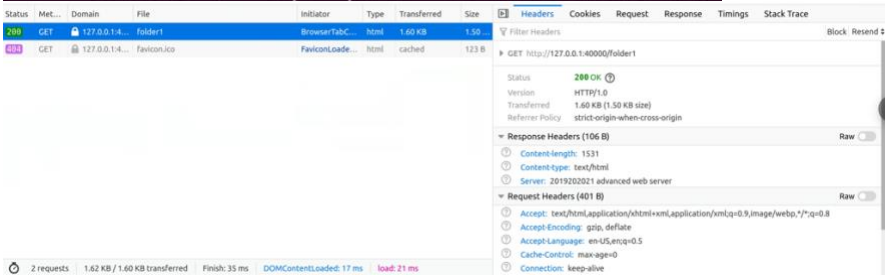


root directory에 접속했을 때 결과로 새로운 client의 정보와 끊겼을 때 정보, 그리고 history가 잘 출력되고 있다. 기존 html_ls처럼 directory는 파란색 link file은 초록색, 나머지는 빨간 색임을 확인할 수 있다. 이때 현재 페이지에 대한 html 파일은 출력하지 않도록 하였다.

E. 하위 directory




```
====================New Client====================
IP : 127.0.0.1
Port : 22714
==================================================

==============Disconnected Client================
IP : 127.0.0.1
Port : 22714
==================================================
==================Connection History==============
Number of request(s) : 2
NO.     IP              PID     PORT    TIME
2       127.0.0.1       14048   22714   Wed May 10 17:24:37 2023
1       127.0.0.1       14042   22202   Wed May 10 17:23:27 2023
```

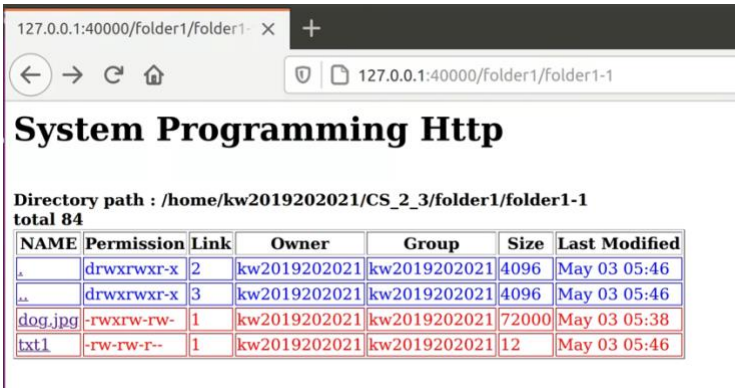


하위 디렉토리 중 하나인 folder1을 클릭했을 때 결과로 정상적으로 이동하고 출력하고 있다. response_header도 정상적으로 전달되었다.

F. 다음 하위 directory

다음 하위 폴더인 folder1-1에 클릭했을 때 결과로 정상적으로 이동하고 출력하고 있음을 확인할 수 있다. 200 OK로 Text/html type으로 display한다.

G. 현재 디렉토리(.) 클릭



```
====================New Client====================
IP : 127.0.0.1
Port : 23738
==================================================

===============Disconnected Client================
IP : 127.0.0.1
Port : 23738
==================================================
====================New Client====================
IP : 127.0.0.1
Port : 24250
==================================================

===============Disconnected Client================
IP : 127.0.0.1
Port : 24250
==================================================
==================Connection History==============
Number of request(s) : 5
NO.     IP              PID     PORT    TIME
5       127.0.0.1       14065   24250   Wed May 10 17:26:32 2023
4       127.0.0.1       14062   23738   Wed May 10 17:26:31 2023
3       127.0.0.1       14053   23226   Wed May 10 17:25:07 2023
2       127.0.0.1       14048   22714   Wed May 10 17:24:37 2023
1       127.0.0.1       14042   22202   Wed May 10 17:23:27 2023
```

현재 디렉토리를 클릭했을 때 결과로 현재 디렉토리 요청에 대한 client의 port는 23738이고 다음의 24250은 하이퍼링크에 커서를 올렸을 때 요청에 대한 결과이다.

H. 상위 디렉토리(..) 클릭

System Programming Http

Directory path : /home/kw2019202021/CS_2_3/folder1
total 356

NAME	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 03 05:46
..	drwxrwxr-x	4	kw2019202021	kw2019202021	4096	May 10 17:38
cat.png	-rwxrw-rw-	1	kw2019202021	kw2019202021	351392	May 03 05:37
doc1	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 03 02:44
doc2	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 03 02:44
doc3	-rw-rw-r--	1	kw2019202021	kw2019202021	0	May 03 02:45
folder1-1	drwxrwxr-x	2	kw2019202021	kw2019202021	4096	May 03 05:46

..의 상위 디렉토리로 클릭했을 때 40000/folder1에 접근하는 것을 확인할 수 있다.

I. 상위 디렉토리(..) 클릭

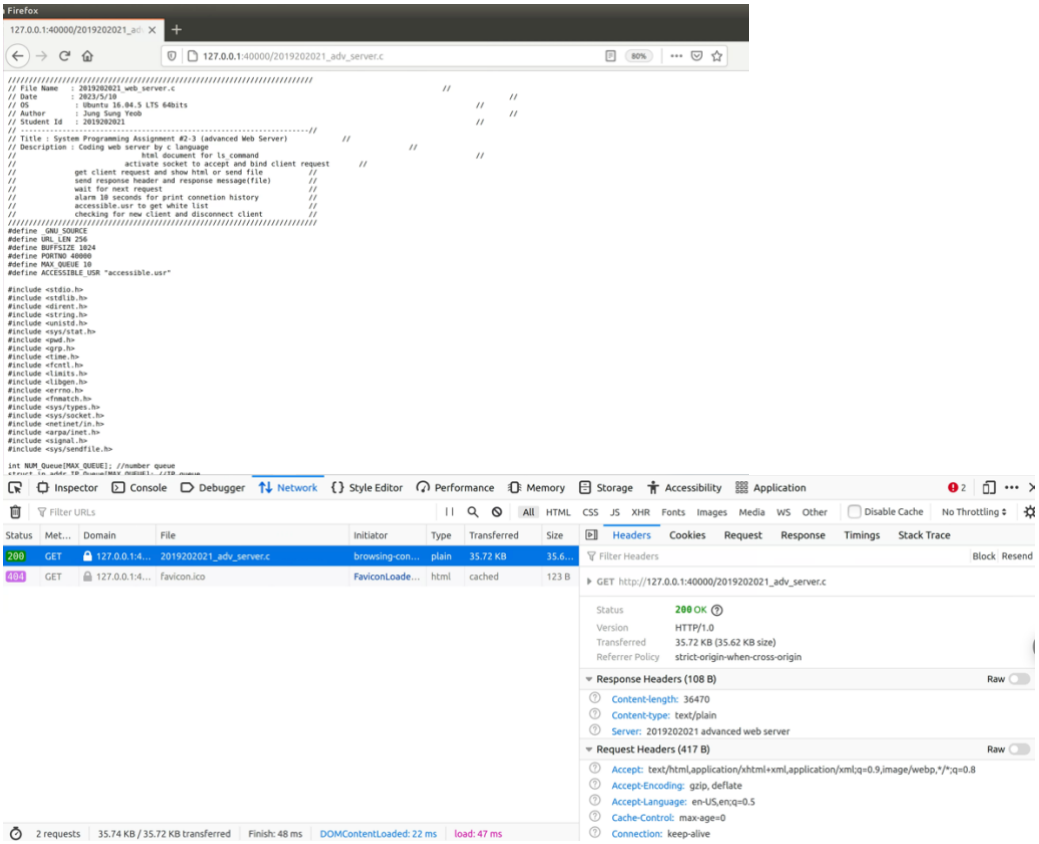
System Programming Http

Directory path : /home/kw2019202021/CS_2_3
total 580

NAME	Permission	Link	Owner	Group	Size	Last Modified
2019202021_adv_server.c	-rw-rw-r--	1	kw2019202021	kw2019202021	36470	May 10 16:53
accessible.usr	-rw-rw-r--	1	kw2019202021	kw2019202021	10	May 09 23:39
adv_server	-rwxrwxr-x	1	kw2019202021	kw2019202021	28336	May 10 16:53
cat.png	-rwxrw-rw-	1	kw2019202021	kw2019202021	351392	May 03 05:37
catlink	lrwxrwxrwx	1	kw2019202021	kw2019202021	7	May 10 16:50
cow.jpeg	-rwxrw-rw-	1	kw2019202021	kw2019202021	83664	May 03 05:38
dog.jpg	-rwxrw-rw-	1	kw2019202021	kw2019202021	72000	May 03 05:38
folder1	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 03 05:46
Makefile	-rw-rw-r--	1	kw2019202021	kw2019202021	79	May 09 02:13
Makelink	lrwxrwxrwx	1	kw2019202021	kw2019202021	8	May 10 17:21
test	drwxrwxr-x	2	kw2019202021	kw2019202021	4096	May 09 02:40

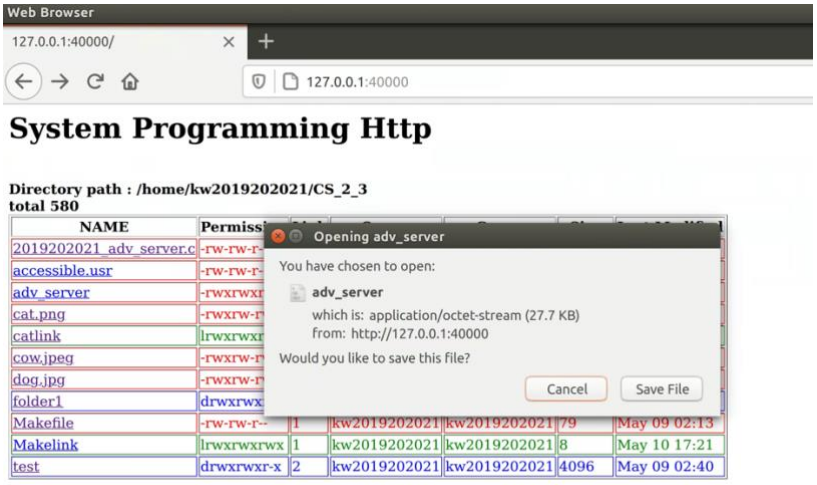
..의 상위 디렉토리를 클릭하였을 때 root 디렉토리에 접근하는 것을 확인할 수 있다.

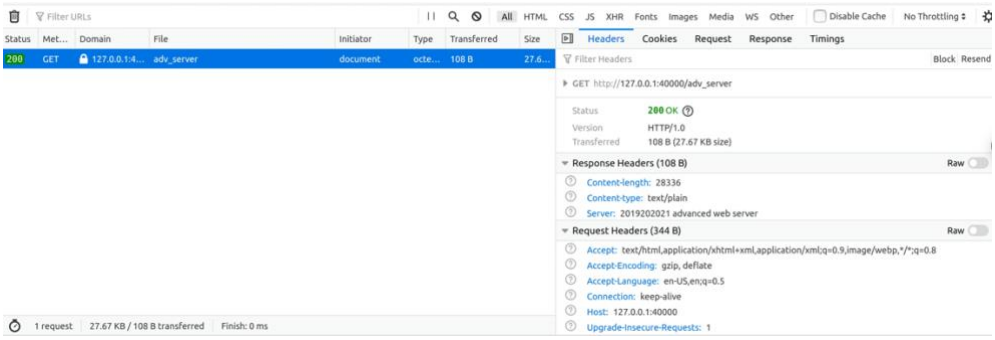
J. text or source code 파일 클릭



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

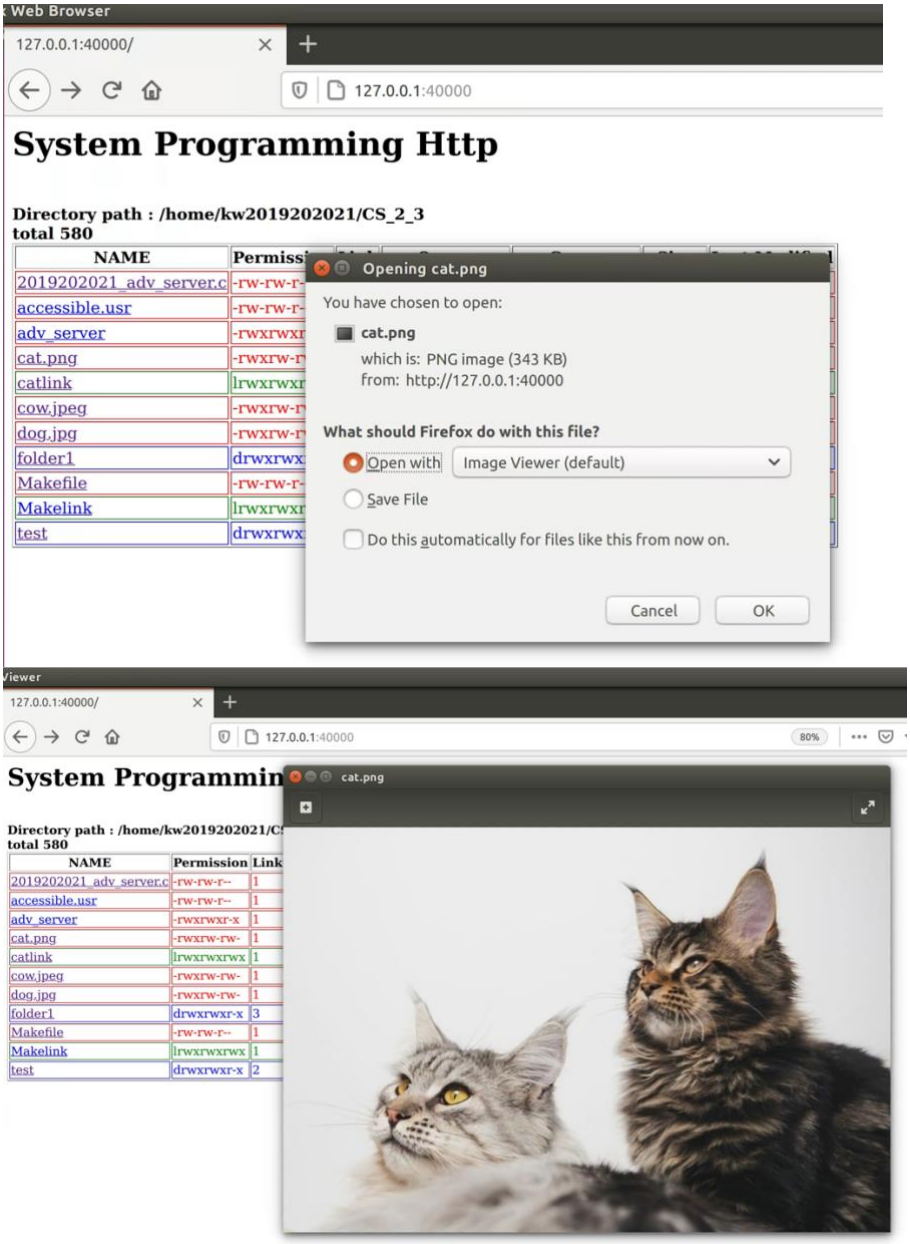
K. 실행 파일(바이너리 파일) 클릭

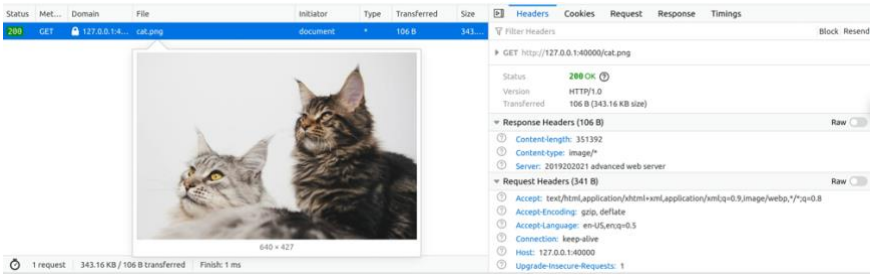




바이너리 파일을 클릭했을 때 Content-type이 text/plain이지만 실행 파일임으로 브라우저에 display 되는 것이 아닌 download를 진행하는 것을 확인할 수 있다.

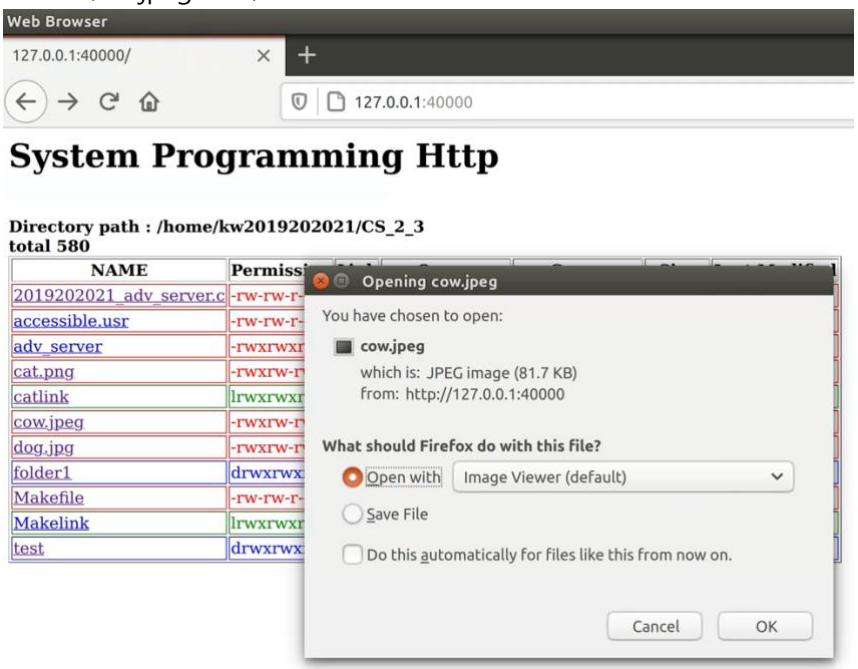
L. 사진 png 클릭

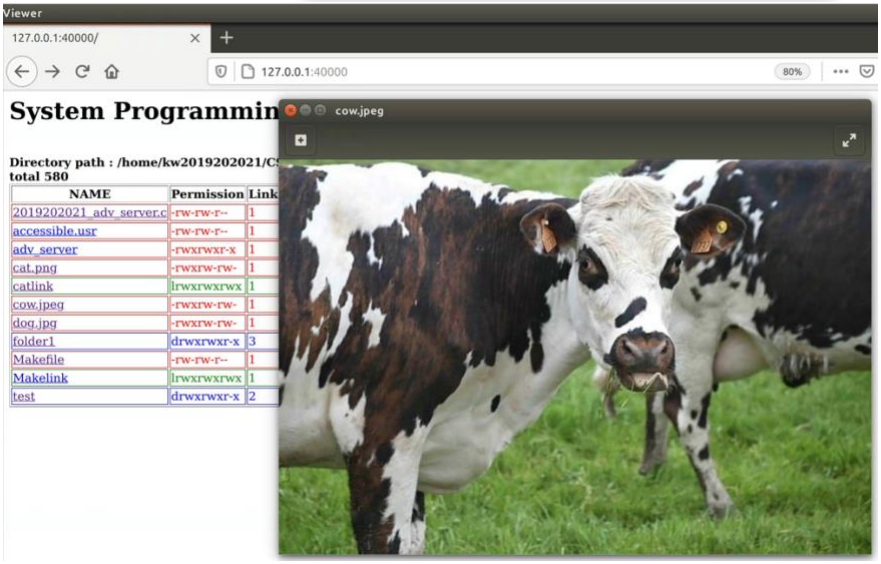


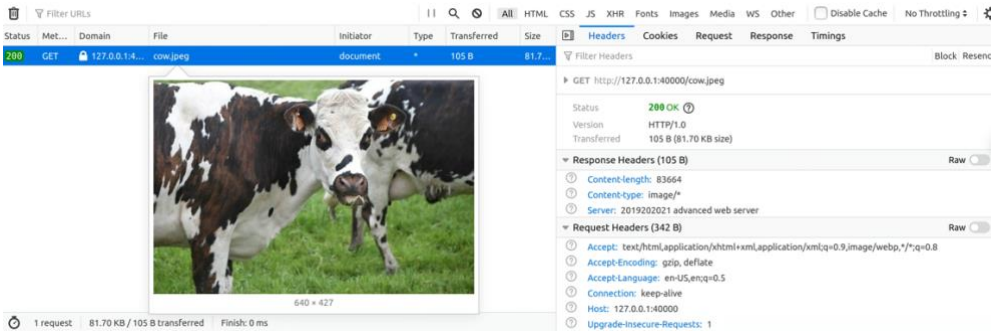


png 파일을 클릭했을 때 png image임을 구별하여 opening option을 주고 open with image viewer(default)로 진행하면 아래와 같이 사진이 출력됨을 확인할 수 있다. status는 200 OK이고 image/*로 content-type이 헤더로 전달됨을 확인할 수 있다.

M. 사진 jpeg 클릭

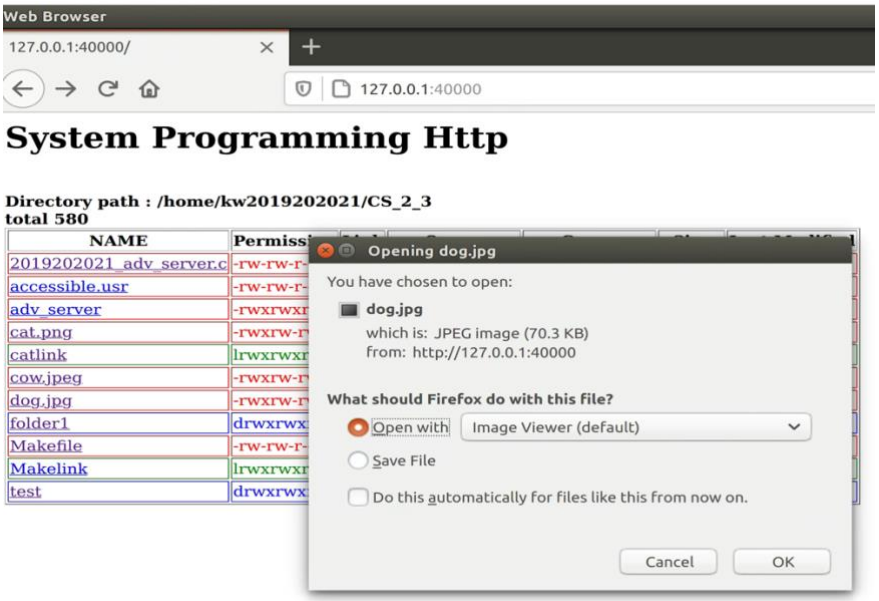


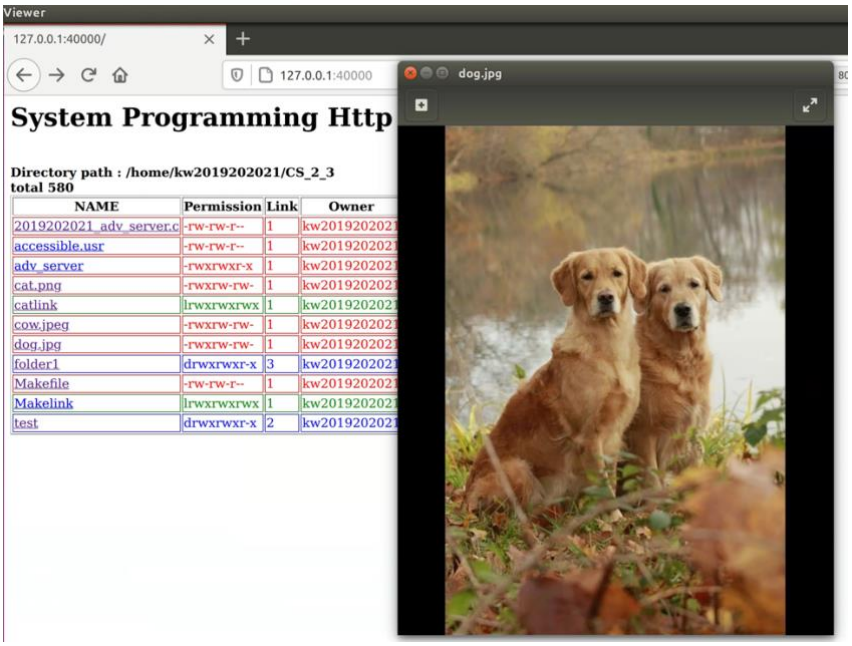


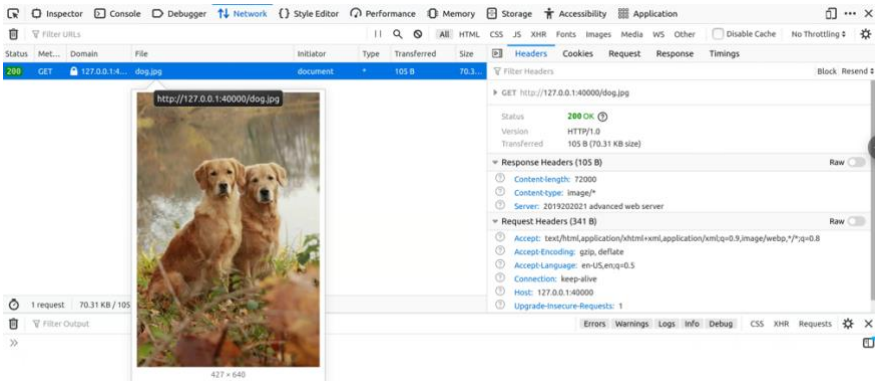


png와 마찬가지로 jpeg를 클릭했을 때 정상적으로 다운로드가 가능했으며 status는 200 OK, Content-type은 image/*로 정상적으로 header가 전달되었다.

N. 사진 jpg 클릭







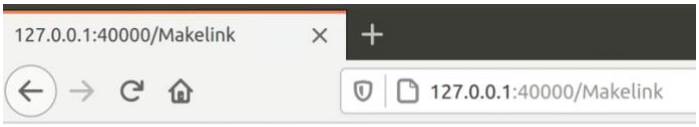
jpg를 클릭했을 때도 200 OK, Content-type image/* 로 정상적으로 헤더 전송과 파일 다운로드가 진행되었으며 특이점은 jpg 파일이지만 다운로드할 때 구분은 jpeg로 한다는 점이다.

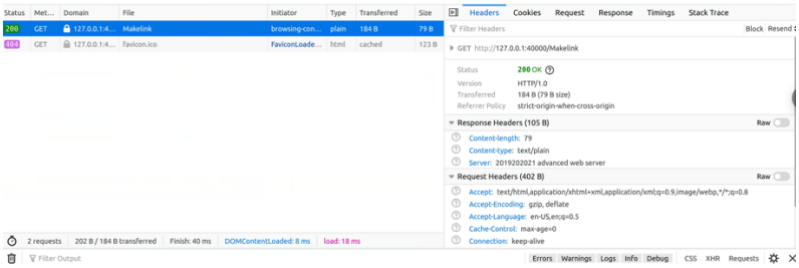
O. connection history 중간 점검

```
====================Connection History=======================
Number of request(s) : 32
NO.     IP              PID     PORT    TIME
32      127.0.0.1       14316   42682   Wed May 10 17:46:34 2023
31      127.0.0.1       14312   42170   Wed May 10 17:46:26 2023
30      127.0.0.1       14295   41658   Wed May 10 17:45:52 2023
29      127.0.0.1       14292   41146   Wed May 10 17:45:48 2023
28      127.0.0.1       14288   40634   Wed May 10 17:45:24 2023
27      127.0.0.1       14287   40122   Wed May 10 17:45:20 2023
26      127.0.0.1       14282   39610   Wed May 10 17:45:09 2023
25      127.0.0.1       14264   39098   Wed May 10 17:44:37 2023
24      127.0.0.1       14263   38586   Wed May 10 17:44:34 2023
23      127.0.0.1       14261   38074   Wed May 10 17:44:32 2023
```

검증은 15번 진행했지만 개발자모드에서 network를 확인할 때 reload하는 과정에서 새로운 connection이 존재했고 순서를 보면 최근 10개의 connection을 출력하는 것을 확인할 수 있다.

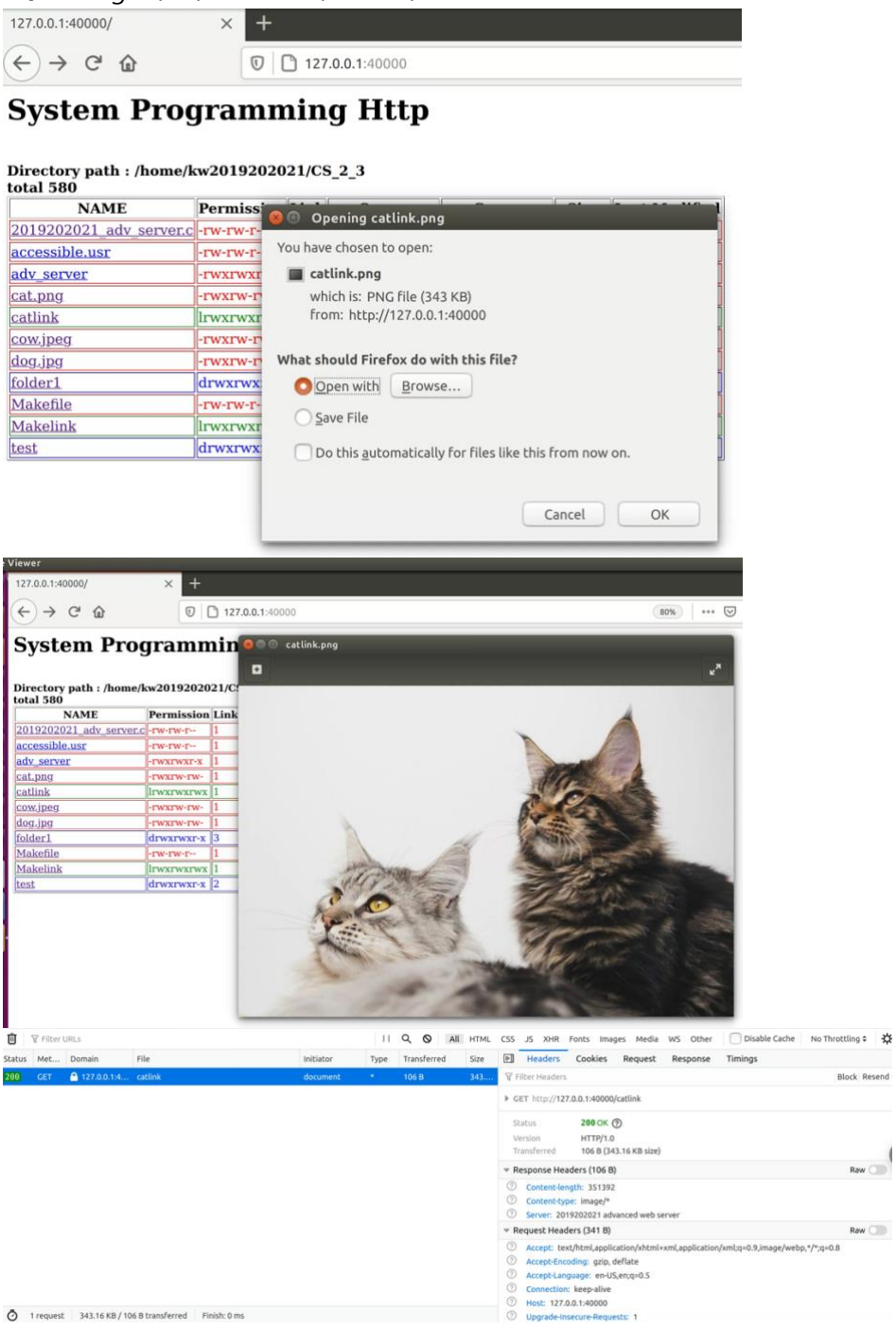
P. source code에 대한 링크 파일 클릭





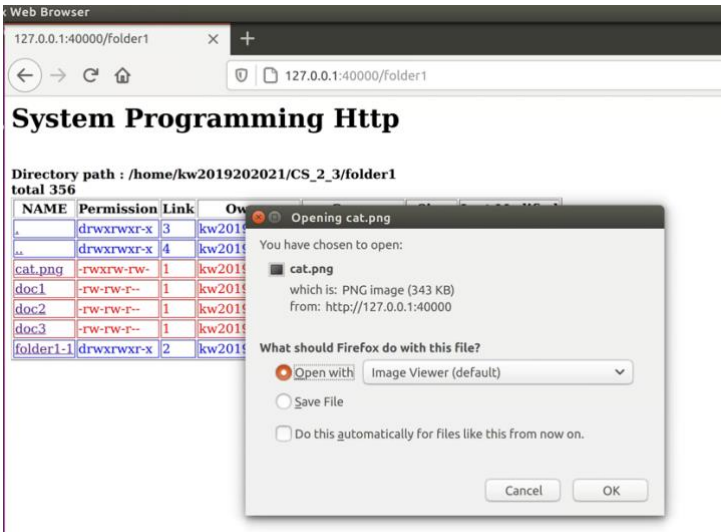
Makefile에 대한 symbolic link를 클릭했을 때 결과이며 200 OK, Content_type text/plain으로 원본을 가져와 전송했기 때문에 Makefile의 내용이 브라우저에 display 되었다.

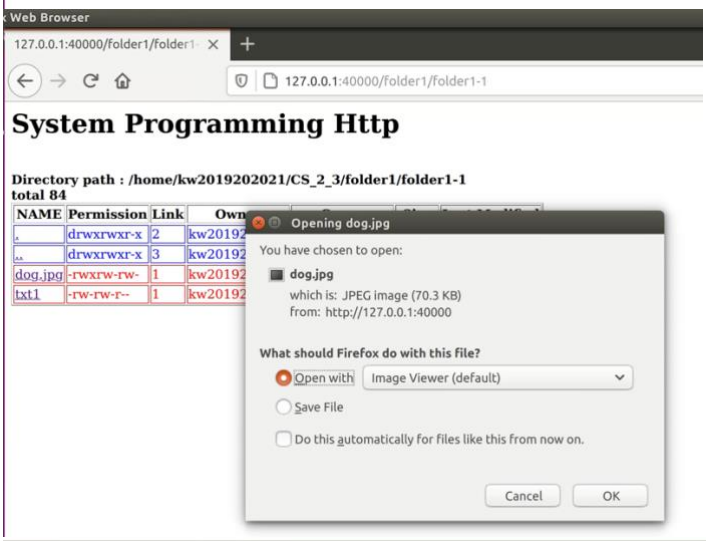
Q. image에 대한 링크 파일 클릭

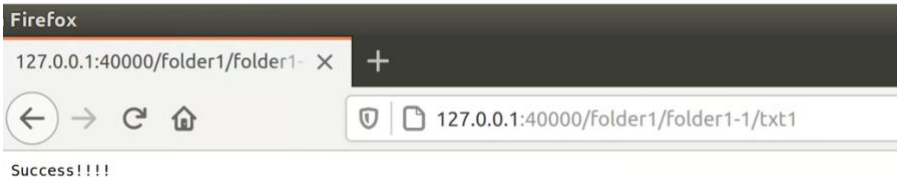


이미지에 대한 symbolic link를 클릭하였을 때 결과이다. symbolic link를 단순 text/plain으로 하는 것이 아니라 원본에 접근하였기 때문에 원본 이름인 cat.png 이름으로 구별하여 image/*로 헤더를 보냈으며 대신 이름은 catlink로 다운받도록 하였다.

R. 기타 하위 폴더에서 클릭

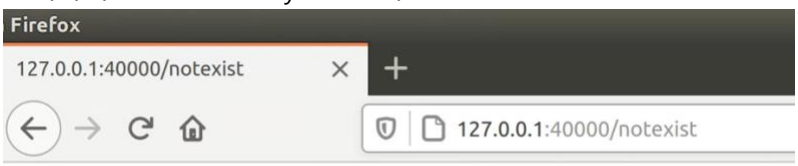






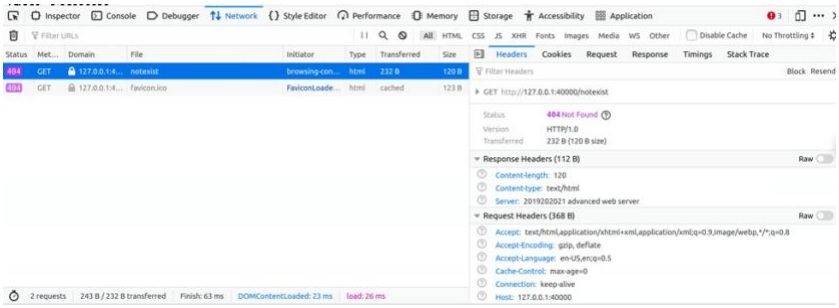
기타 하위 폴더에서도 파일들을 접근하면 문제없이 다운로드하거나 브라우저에 display 하는 것을 확인하였다.

S. 존재하지 않는 directory 접근 시



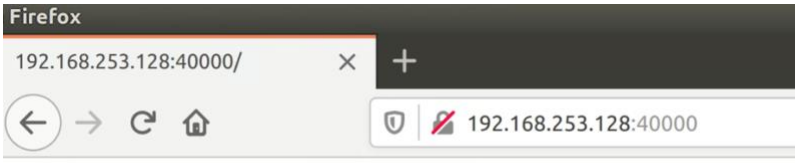
Not Found

The request URL /notexist was not found on this server
HTTP 404 - Not Page Found



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.

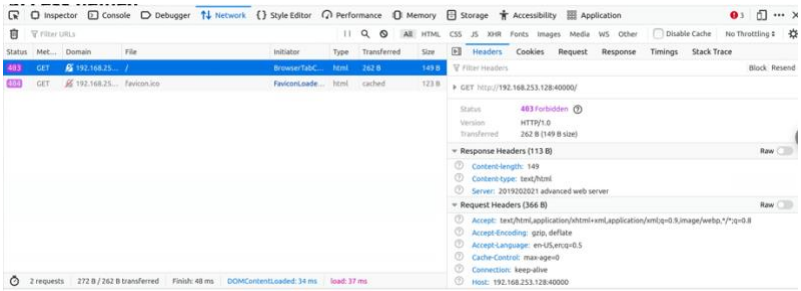
T. 허용한 ip가 아닌 경우



Access denied

Your IP: 192.168.253.128

You have no permission to access this web server.
HTTP 403.6 - Forbidden: IP address reject

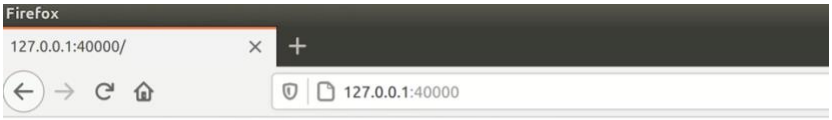


앞서 보았듯이 accessible.usr에 저장된 ip는 127.0.0.*이므로 192.168.253.128은 access denied되고 response header는 403 forbidden을 보낸다.

U. 새로고침을 빠르게 클릭했을 때 결과

```
====================Connection History=======================
Number of request(s) : 125
NO.      IP              PID     PORT    TIME
125      127.0.0.1       14671   50363   Wed May 10 18:14:17 2023
124      127.0.0.1       14670   49851   Wed May 10 18:14:16 2023
123      127.0.0.1       14669   49339   Wed May 10 18:14:16 2023
122      127.0.0.1       14668   48827   Wed May 10 18:14:16 2023
121      127.0.0.1       14667   48315   Wed May 10 18:14:16 2023
120      127.0.0.1       14666   47803   Wed May 10 18:14:16 2023
119      127.0.0.1       14665   47291   Wed May 10 18:14:15 2023
118      127.0.0.1       14664   46779   Wed May 10 18:14:15 2023
117      127.0.0.1       14663   46267   Wed May 10 18:14:15 2023
116      127.0.0.1       14662   45755   Wed May 10 18:14:15 2023
```

새로 고침을 빠르게 진행했을 때도 시간 순에 따라 정상적으로 출력되고 있음을 확인할 수 있다. 특히 초 단위로 출력하기 때문에 겹치는 시간이 있지만 PID나 PORT의 숫자를 보면 최신 순으로 출력되고 있음을 확인할 수 있다.



System Programming Http

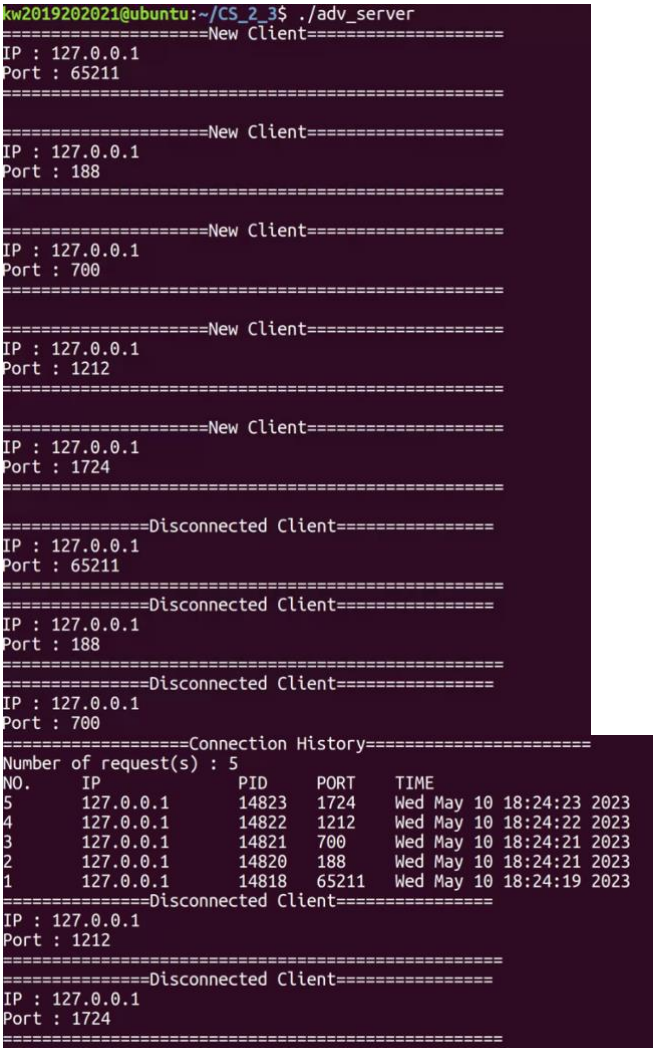
Directory path : /home/kw2019202021/CS_2_3
total 580

NAME	Permission	Link	Owner	Group	Size	Last Modified
2019202021_adv_server.c	-rw-rw-r--	1	kw2019202021	kw2019202021	36470	May 10 16:53
accessible.usr	-rw-rw-r--	1	kw2019202021	kw2019202021	10	May 09 23:39
adv_server	-rwxrwxr-x	1	kw2019202021	kw2019202021	28336	May 10 16:53
cat.png	-rwxrw-rw-	1	kw2019202021	kw2019202021	351392	May 03 05:37
catlink	lrwxrwxrwx	1	kw2019202021	kw2019202021	7	May 10 16:50
cow.jpeg	-rwxrw-rw-	1	kw2019202021	kw2019202021	83664	May 03 05:38
dog.jpg	-rwxrw-rw-	1	kw2019202021	kw2019202021	72000	May 03 05:38
folder1	drwxrwxr-x	3	kw2019202021	kw2019202021	4096	May 03 05:46
Makefile	-rw-rw-r--	1	kw2019202021	kw2019202021	79	May 09 02:13
Makelink	lrwxrwxrwx	1	kw2019202021	kw2019202021	8	May 10 17:21
test	drwxrwxr-x	2	kw2019202021	kw2019202021	4096	May 09 02:40

추가적으로 html파일을 생성하고 읽어서 보내는 동작을 하도록 했지만 child process 마지막에 html파일을 삭제하였고 이 때문에 연속된 새로고침에도 생성된 html 파일은 browser에 나타나지 않는다. 단, 다른 종류의 html 파일은 띄울 수 있다.

V. 다중 클라이언트 접속 확인

```
sleep(5);
puts("==============Disconnected Client==============");
printf("IP : %s\nPort : %d\n", inet_ntoa(inet_client_address), client_addr.sin_port); //if connected
puts("================================================");
```



disconnected 되기 전 sleep(5)를 두어 다중 클라이언트 접속이 가능한지 확인하였고 브라우저에 html을 정상적으로 전달함도 확인하였다.

5. 고찰

Assignment 2-3에서는 수업 시간에 배운 fork와 alarm과 signal을 이용하여 다중 클라이언트 접속 가능한 advanced web server를 구현하고 10초마다 connection history를 출력하는 프로그램을 설계하였다. 이 때 자식 프로세스에서 진행되는 코드와 부모 프로세스에서 진행되는 코드를 따로 두어 자식 프로세스에서 request에 대한 response를 보내도록 하였고, 부모 프로세스에서는 새로운 클라이언트를 accept 하도록 두었다. 특이점으로는 부모프로세스에서 자식프로세스의 종료를 받는 wait() 또는 waitpid()를 따로 두지 않아 좀비프로세스가 되는데 이는 제안서에서 제한하지 않아서 따로 설계에 염두하지 않았다.

Alarm을 통해 connection history 를 출력하는데 있어 connection history를 전역변수 queue에 저장하도록 설계하였다. 이때 C++에서는 vector가 있지만 C언어에서는 없기에 배열을 queue 처럼 만들어 10개 까지 최근 history를 저장하도록 했다. 10개가 초과되는 경우에는 가장 오래된 connection을 지우고 새로운 connection을 저장하도록 설계하였다.

또한 accessible.usr에 있는 허용 ip address에 대한 정보를 읽고 허용된 ip인 경우에만

접속이 가능하도록 설계해야 하는데 이 때는 전에 asterisk를 포함한 파일 경로를 읽을 때 사용한 fnmatch()함수를 사용하여 *로 이뤄져도 접속이 가능하도록 했다.

마지막으로 이번 과제는 기존 함수에 Alarm을 추가하고 fork로 기존 동작을 자식 프로세스에 넘기고 accessible.usr를 읽어 white list를 구분하는 코드를 추가하여 비교적 적은 시간으로 과제를 마무리 할 수 있었다.

6. Reference

1) 시스템프로그래밍실습 Assignment 2-3/광운대학교 / 컴퓨터정보공학부 / 김태석교수님/2023
2) 시스템프로그래밍 강의자료/광운대학교/컴퓨터정보공학부/김태석교수님/2023