

# 시스템프로그래밍실습

## assignment 2-3

학 과: 컴퓨터정보공학부

담당교수: 이기훈 교수님

학 번: 2019202050

성 명: 이강현

## 1. Introduction

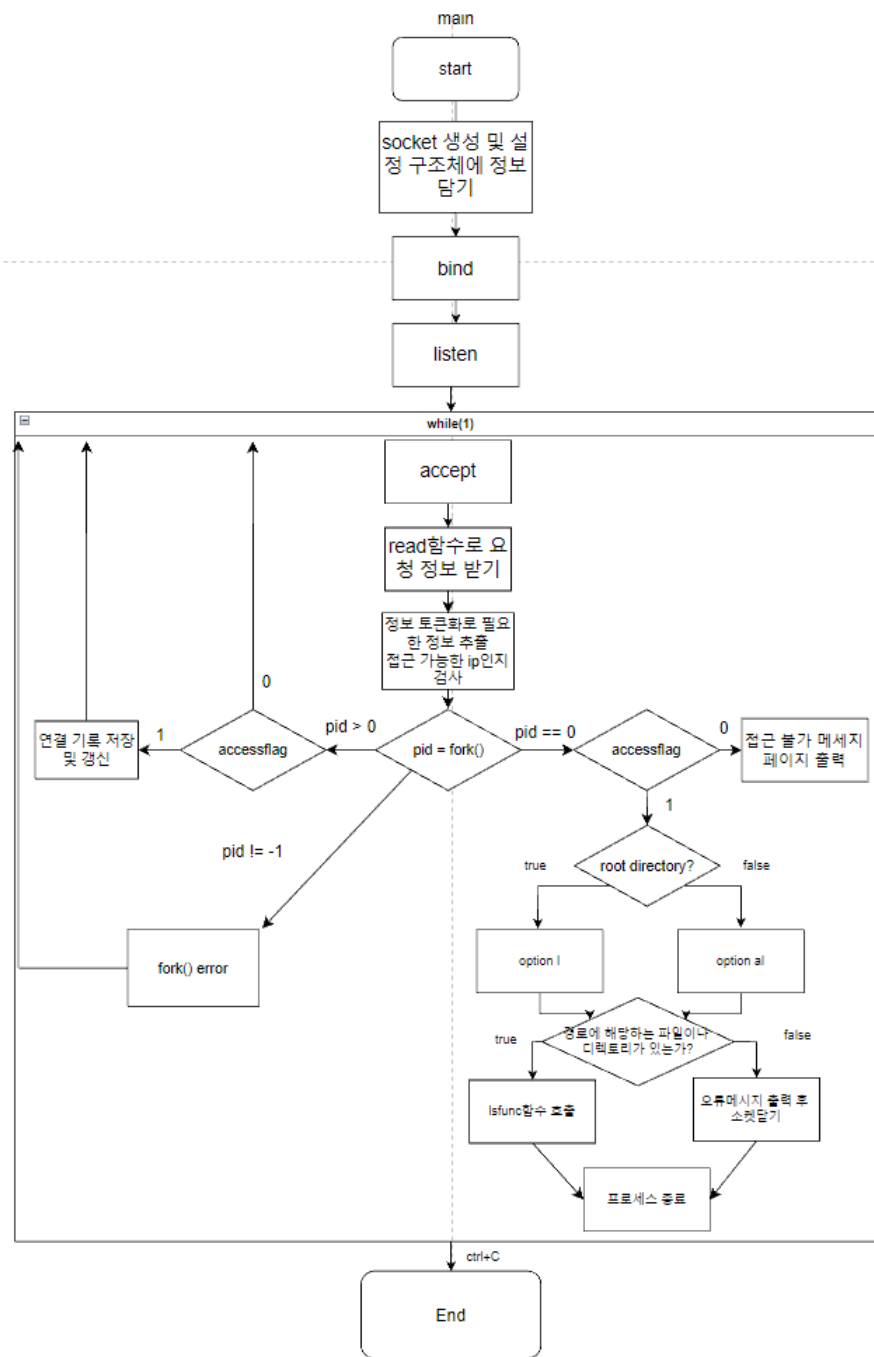
fork함수를 사용해보며 프로세스를 복제하는 법을 알고 해당 함수의 동작방식과 이를 통해 서로 다른 작업을 하도록 하는 방법을 안다.

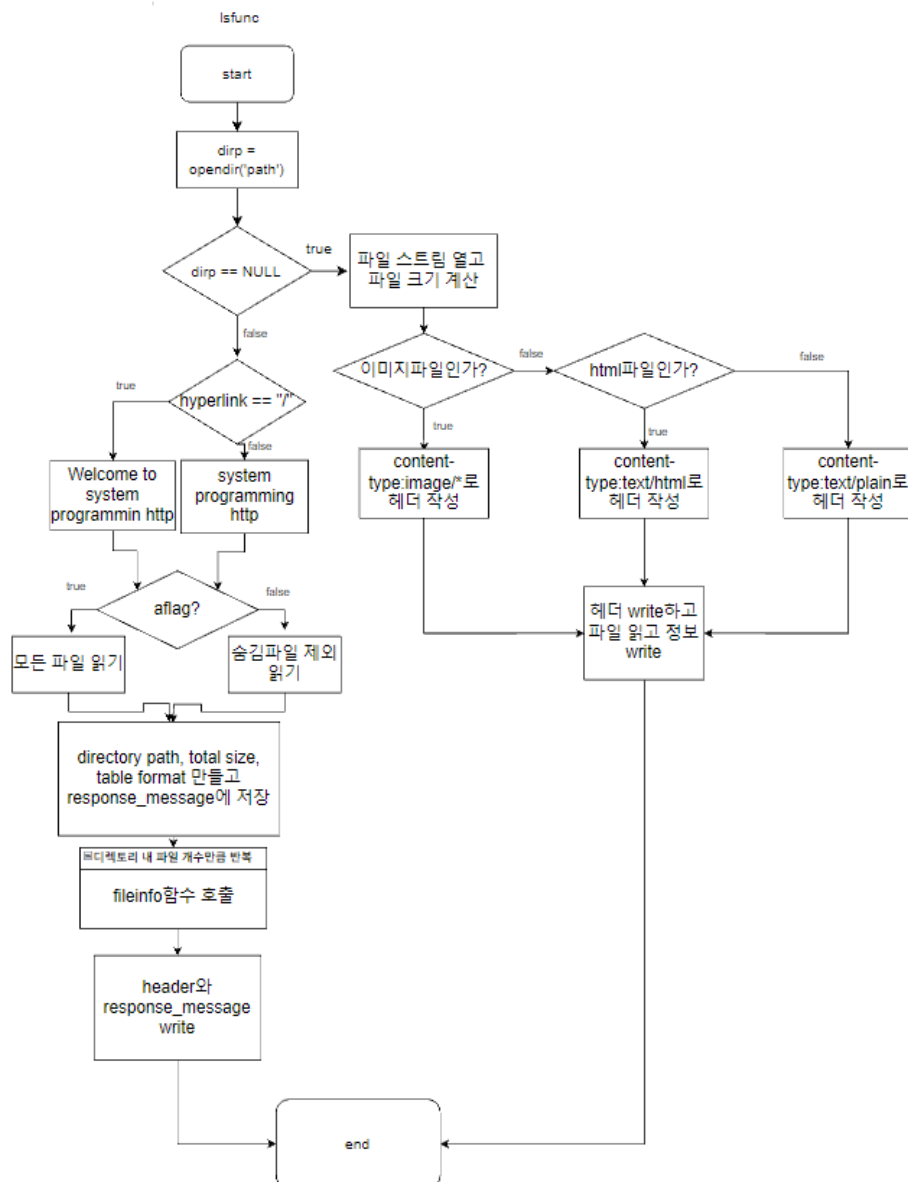
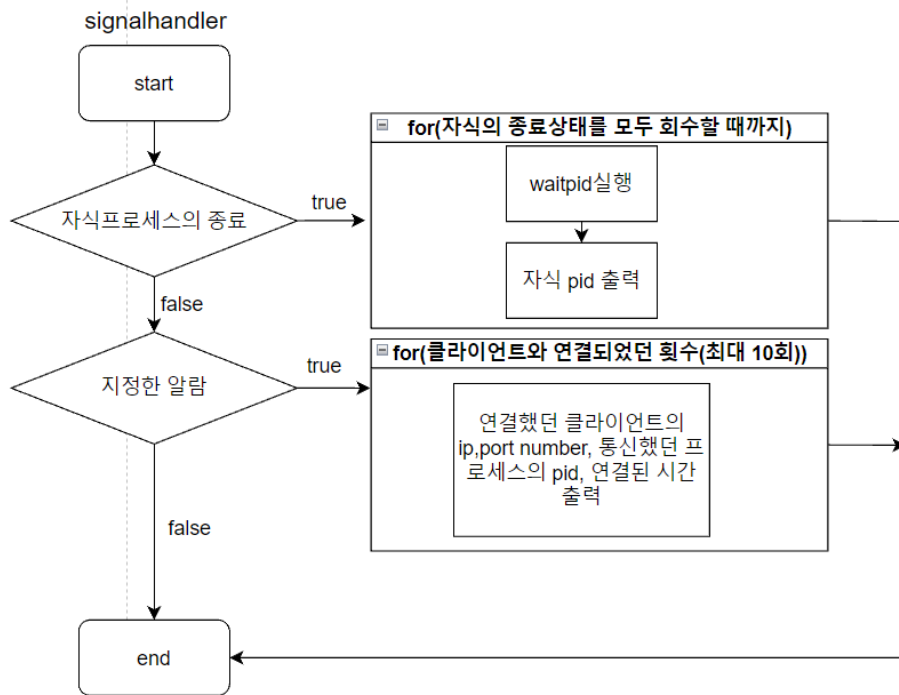
wait함수를 통해 부모 프로세스와 자식 프로세스의 관계를 알고 종료 상태의 개념을 안다.

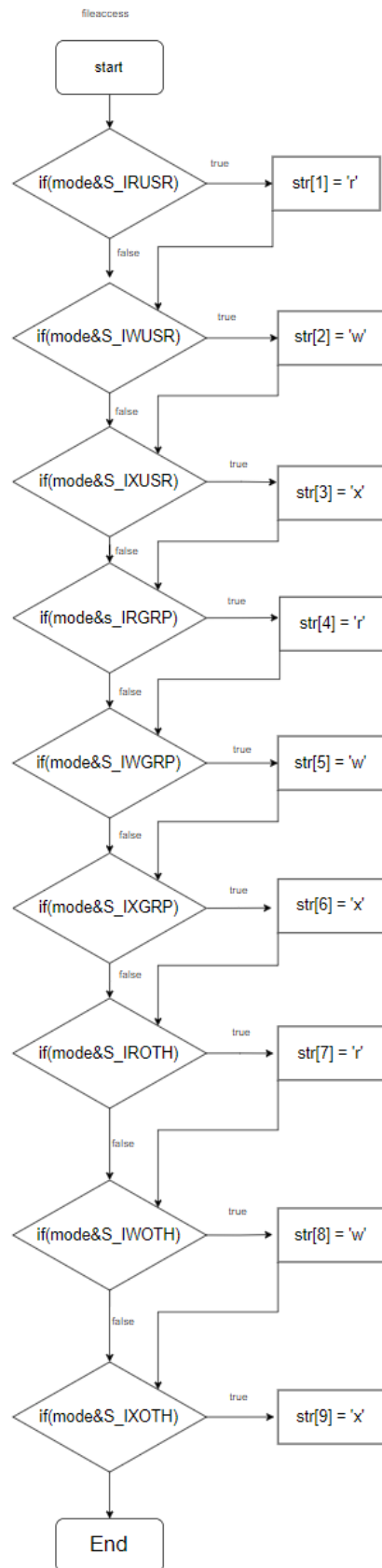
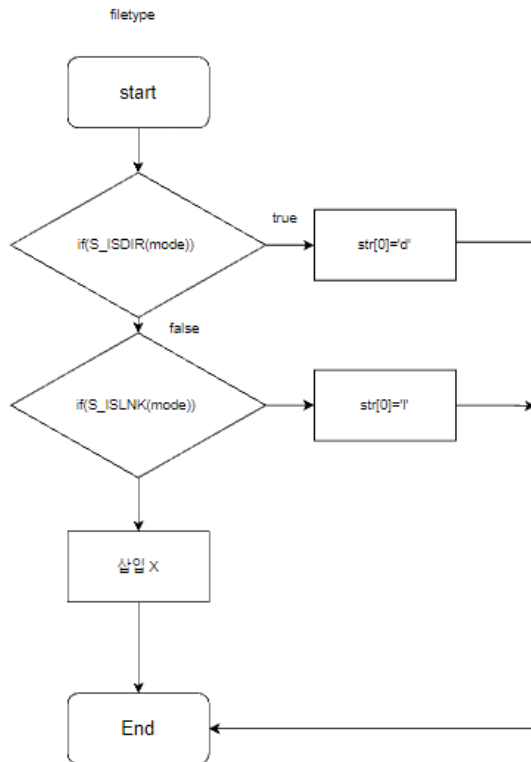
여러 상황에서 발생하는 signal의 종류를 알고 이를 handling 하는법을 익힌다.

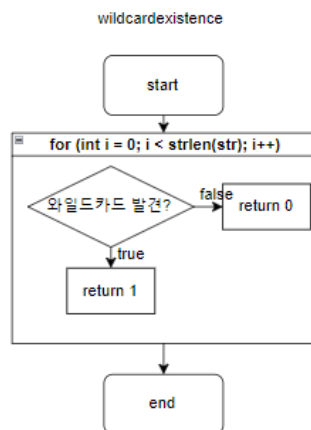
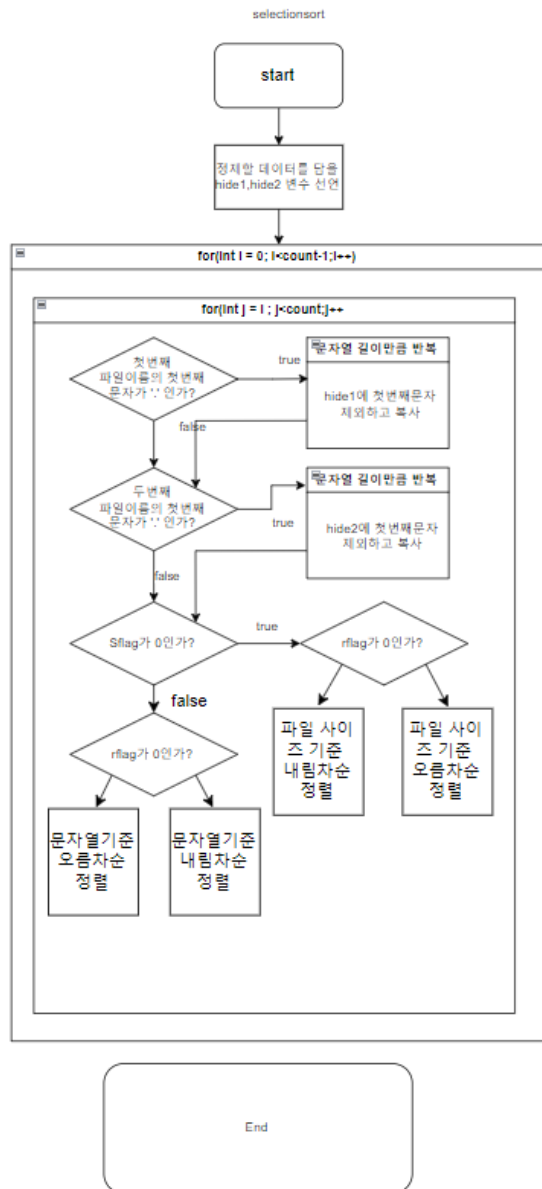
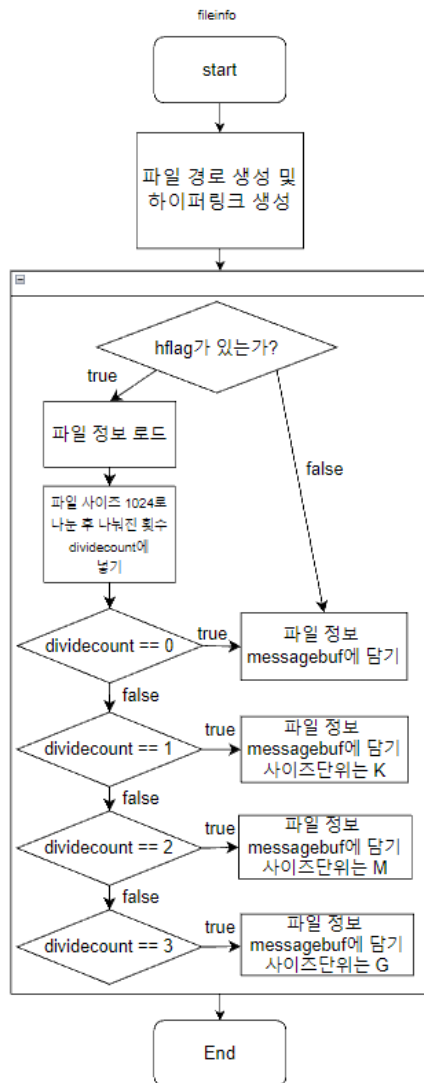
위의 개념을 이용하여 다중 접속 서버를 구현해보며 멀티 프로세스방식의 소켓 통신 개념을 익힌다.

## 2. Flow chart









모든 함수들의 흐름도이다. main 함수는 소켓을 생성하여 소켓을 설정을 마치고 bind,listen을 마무리한 후 while문에서 client의 요청을 기다린다. 그 후 요청이 들어오면 client가 보낸 요청을 해석하고 url을 얻어낸다. 이후 접근가능한 ip목록이 적혀있는 파일을 열어 클라이언트의 ip와 비교 후 flag를 설정한다. 이후 fork함수를 통해 자식 프로세스를 생성하고 자식프로세스는 flag가 0이라면 denied 페이지를 1이라면 요청한 페이지를 클라이언트에게 제공한다. 부모 프로세스는 자식프로세스가 연결한 클라이언트의 정보를 저장하고 새로운 클라이언트의 요청을 기다린다.

lsfunc 함수는 main 함수에서 넘겨받은 인자에 따라 파일인지 폴더인지를 체크한 후 분기를 나누고 각각 옵션에 따라 요구받은 행동을 취한다.

파일일 경우에는 파일포인터를 선언 후 파일 스트림을 통해 파일을 불러오고 이를 client에게 전송한다.

폴더일 경우에 root directory일 경우에는 l옵션만 적용하여 파일들의 정보를 html형식으로 만들어 보내준다.

그렇지 않을 경우에는 a 옵션은 적용하여 숨김파일에 대한 정보 또한 보낸다.

lsfunc 함수내에서 실행되는 fileaccess함수와 filetype함수는 l옵션에서 제일 먼저 출력되는 파일 타입과 접근권한을 숫자에서 문자데이터로 변환해주는 함수이다.

fileinfo 함수는 lsfunc에서 받은 경로를 통해 하이퍼링크를 생성하고 파일 정보와 함께 버퍼에 담는다.

selectionsort함수는 선택정렬을 구현한 함수로 숨김파일을 구분하지 않고 정렬하기 위해 임시변수를 활용하여 구현하였다.

signalhandler함수는 main에서 자식 프로세스의 종료 혹은 지정한 알람 signal이 발생할 때 작동하며 자식프로세스가 클라이언트에게 정보를 제공하고 종료되면 이를 부모 프로세스가 waitpid를 통해 자식 프로세스의 종료상태를 얻고 사용한 자원을 해제하는 동작을 진행하거나 혹은 연결되었던 클라이언트들의 정보와 이를 처리한 자식프로세스의 pid를 출력한다.

### 3. Pseudo Code

selectionsort  
파일 사이즈 변수선언  
파일 경로 만들기 위한 변수선언  
문자열 정제 후 저장할 변수 선언

만약 입력 경로가 NULL이 아니라면  
for(파일 개수만큼)  
    경로 생성하여 파일 사이즈를 얻어내고 변수에 저장  
for(파일개수 -1만큼)  
    for(파일개수 -1만큼)  
        만약 파일의 앞 문자가 '.' 이라면  
            앞문자 제외하여 변수에 저장  
        아니라면  
            그대로 저장

소문자 모두 대문자로 변경

if(!sflag)  
    if(!rflag)  
        문자열 기준 오름차순 정렬  
    else  
        문자열 기준 내림차순 정렬  
else  
    if(!rflag)  
        파일사이즈 기준 내림차순 정렬  
        사이즈가 같다면 문자열 기준 정렬  
    else  
        문자열 기준 오름차순 정렬  
        사이즈가 같다면 문자열 기준 정렬

filetype  
입력받은 파일의 mode가 디렉토리라면  
    입력받은 문자열의 맨 앞 글자를 d로 변경  
입력받은 파일의 mode가 심볼릭 링크라면  
    입력받은 문자열의 맨 앞 글자를 l로 변경  
|

fileaccess  
입력받은 파일의 mode의 user access가 read가 가능하다면  
    입력받은 문자열의 2번째를 r로 변경  
입력받은 파일의 mode의 user access가 write가 가능하다면  
    입력받은 문자열의 3번째를 w로 변경  
입력받은 파일의 mode의 user access가 execution이 가능하다면  
    입력받은 문자열의 4번째를 x로 변경

입력받은 파일의 mode의 group access가 read가 가능하다면  
    입력받은 문자열의 5번째를 r로 변경  
입력받은 파일의 mode의 group access가 write가 가능하다면  
    입력받은 문자열의 6번째를 w로 변경  
입력받은 파일의 mode의 group access가 execution이 가능하다면  
    입력받은 문자열의 7번째를 x로 변경

입력받은 파일의 mode의 other access가 read가 가능하다면  
    입력받은 문자열의 8번째를 r로 변경  
입력받은 파일의 mode의 other access가 write가 가능하다면  
    입력받은 문자열의 9번째를 w로 변경  
입력받은 파일의 mode의 other access가 execution이 가능하다면  
    입력받은 문자열의 10번째를 x로 변경

fileinfo  
구조체 선언  
입력받은 경로로 파일 정보 얻어와 구조체에 담기  
하이퍼링크 생성을 위해 입력받은 경로 정제  
파일 정보를 문자열로 변환  
if(hflag)  
    파일 정보중 파일 사이즈 부분을 나누어 단위를 붙여줌  
파일 정보 messagebuf에 담기

wildcardexistence  
wildcard = "\*?[]"  
if(와일드 카드를 입력받은 문자열에서 찾으면)  
    return 1  
못찾으면  
    return 0

lsfunc  
입력받은 경로로 디렉토리 열기  
디렉토리가 null이라면  
    파일 스트림 열고 파일 크기 저장  
    if(이미지파일)  
        이미지파일 전용 헤더 생성  
    elseif(html파일)  
        html파일 전용 헤더 생성  
    else  
        일반파일 전용 헤더 생성  
헤더 파일 write  
파일 바이트단위로 읽고 client에게 보내기  
종료

if(!aflag)  
    숨김파일 제외 파일 저장

else  
    모두 읽기

if(!lflag)  
    정렬 하고 파일들 이름만 출력

else  
    정렬하고 파일들 정보를 담은 테이블 생성  
    fileinfo 함수를 파일 개수만큼 호출  
    헤더파일 write하고 정제된 파일 정보 html형식으로 client에게 보내기  
종료

signalhandler

signal이 발생시 실행

if(signal이 자식이 종료되었음을 알림)

    종료된 자식들을 모두 회수할때까지 waitpid실행

else if(signal이 지정한 알람시간이 다 되었음을 알림)

    서버가 열린 이래로 연결되었던 클라이언트에 대한 정보와

    이를 처리했던 자식 프로세스의 pid, 연결 시간을 출력(최근 10회의 기록까지)

```

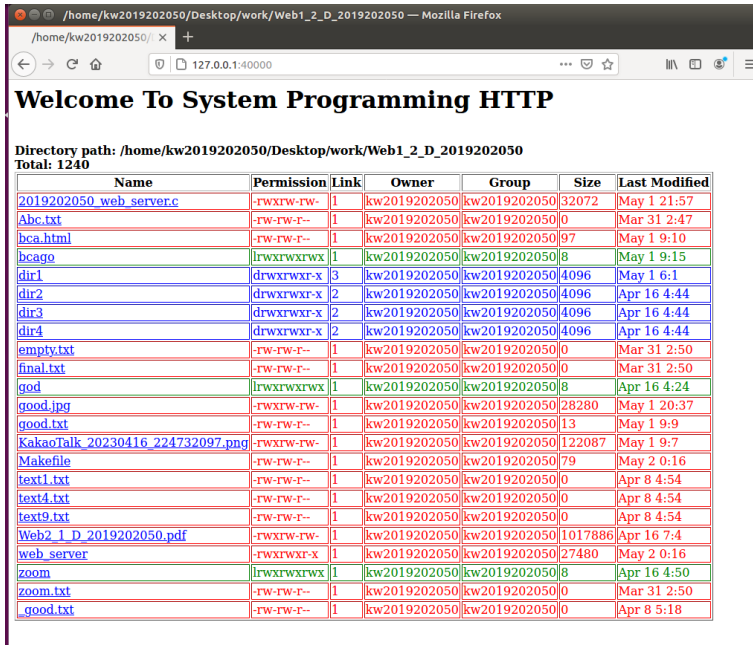
main
소켓 생성
소켓 설정
bind
listen
while(1)
    client accept
    method와 url 토큰으로 얻기
    접근가능한 ip주소가 저장된 파일 열고 현재 클라이언트 ip와 비교
    일치하면 accessflag = 1, 아니면 0
    fork()
    if(pid == 0)
        accessflag == 0일때
            접근 불가 메시지를 담은 페이지를 보내고 프로세스 종료
        accessflag == 1일때
            현재디렉토리를 루트 디렉토리로 간주하고 url이어붙여 경로 생성
            해당 경로에 파일이 존재하지 않는다면 에러전용 헤더와 메시지를 보내주기
            존재한다면 lsfunc함수 호출 후 client disconnect
            프로세스 종료
    else if(pid>0)
        클라이언트의 정보를 구조체에 저장
        자식프로세스의 pid를 저장
        클라이언트와 연결된 시간을 저장
    else
        fork error출력

소켓 닫기
종료

```



## 4. 결과화면



Directory path: /home/kw2019202050/Desktop/work/Web1\_2\_D\_2019202050  
Total: 1240

Name	Permission	Link	Owner	Group	Size	Last Modified
2019202050_web_server.c	-rwxrwxrwx	1	kw2019202050	kw2019202050	32072	May 1 21:57
Abc.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Mar 31 2:47
bca.html	-rw-rw-r--	1	kw2019202050	kw2019202050	97	May 1 9:10
bcago	lrwxrwxrwx	1	kw2019202050	kw2019202050	8	May 1 9:15
dir1	drwxrwxr-x	3	kw2019202050	kw2019202050	4096	May 1 6:1
dir2	drwxrwxr-x	2	kw2019202050	kw2019202050	4096	Apr 16 4:44
dir3	drwxrwxr-x	2	kw2019202050	kw2019202050	4096	Apr 16 4:44
dir4	drwxrwxr-x	2	kw2019202050	kw2019202050	4096	Apr 16 4:44
empty.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Mar 31 2:50
final.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Mar 31 2:50
god	lrwxrwxrwx	1	kw2019202050	kw2019202050	8	Apr 16 4:24
good.jpg	-rwxrwxrwx	1	kw2019202050	kw2019202050	28280	May 1 20:37
good.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	13	May 1 9:9
KakaoTalk_20230416_224732097.png	-rwxrwxrwx	1	kw2019202050	kw2019202050	122087	May 1 9:7
Makefile	-rw-rw-r--	1	kw2019202050	kw2019202050	79	May 2 0:16
text1.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Apr 8 4:54
text4.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Apr 8 4:54
text9.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Apr 8 4:54
Web2_1_D_2019202050.pdf	-rwxrwxrwx	1	kw2019202050	kw2019202050	1017886	Apr 16 7:4
web_server	-rwxrwxr-x	1	kw2019202050	kw2019202050	27480	May 2 0:16
zoom	lrwxrwxrwx	1	kw2019202050	kw2019202050	8	Apr 16 4:50
zoom.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Mar 31 2:50
good.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Apr 8 5:18

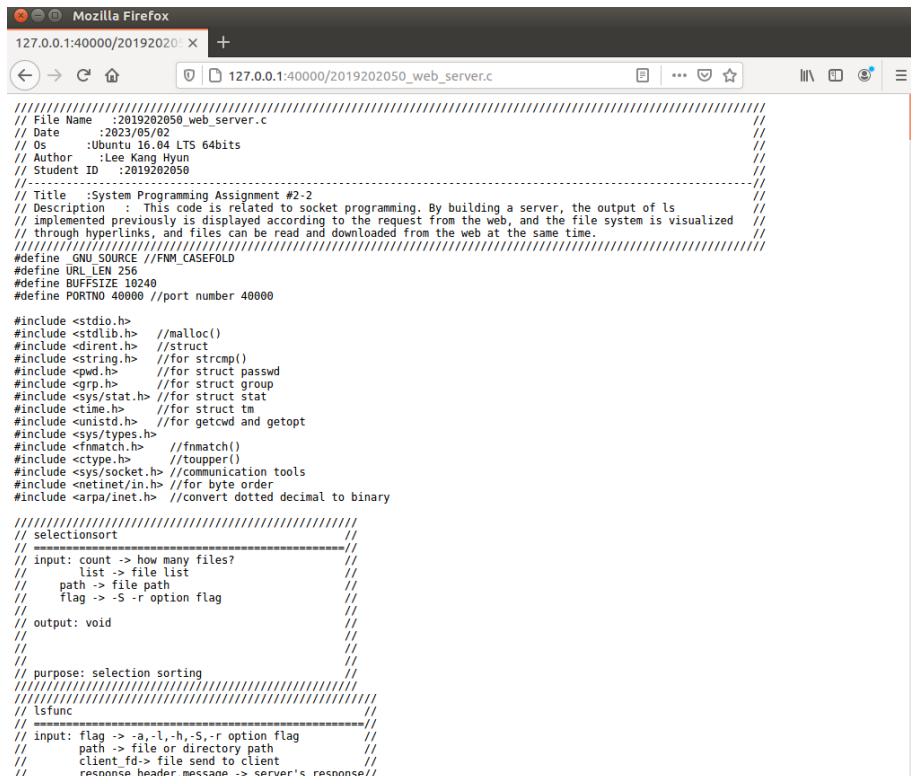
루트 디렉토리의 모습이다. 상위 디렉토리로 접근하지 못하도록 -l 옵션만 적용된 상태이다.



Directory path: /home/kw2019202050/Desktop/work/Web1\_2\_D\_2019202050/dir1  
Total: 12

Name	Permission	Link	Owner	Group	Size	Last Modified
.	drwxrwxr-x	3	kw2019202050	kw2019202050	4096	May 1 6:1
..	drwxrwxr-x	6	kw2019202050	kw2019202050	4096	May 2 0:16
hi	drwxrwxr-x	3	kw2019202050	kw2019202050	4096	May 1 6:1

하위 폴더로 하이퍼링크를 통해 들어간 모습이다. 하위 디렉토리이므로 상위 디렉토리로의 접근이 가능하게 -al 옵션이 적용되었다.

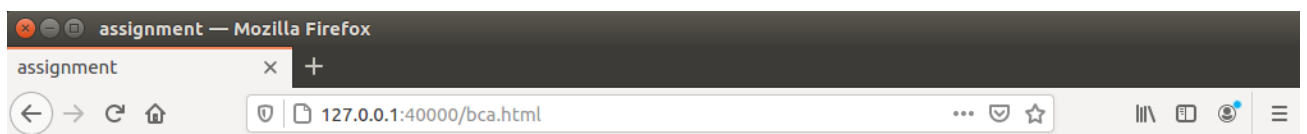


```
// File Name : 2019202050_web_server.c
// Date : 2023/05/02
// Os : Ubuntu 16.04 LTS 64bits
// Author : Lee Kang Hyun
// Student ID : 2019202050
// Title : System Programming Assignment #2-2
// Description : This code is related to socket programming. By building a server, the output of ls
// implemented previously is displayed according to the request from the web, and the file system is visualized
// through hyperlinks, and files can be read and downloaded from the web at the same time.
//
//=====
#define GNU_SOURCE //FNM_CASEFOLD
#define URL_LEN 256
#define BUFSIZE 10240
#define PORTNO 40000 //port number 40000

#include <stdio.h>
#include <stdlib.h> //malloc()
#include <dirent.h> //struct
#include <string.h> //for strcmp()
#include <pwd.h> //for struct passwd
#include <grp.h> //for struct group
#include <sys/stat.h> //for struct stat
#include <time.h> //for struct tm
#include <unistd.h> //for getcwd and getopt
#include <sys/types.h>
#include <fnmatch.h> //fnmatch()
#include <ctype.h> //toupper()
#include <sys/socket.h> //communication tools
#include <netinet/in.h> //for byte order
#include <arpa/inet.h> //convert dotted decimal to binary

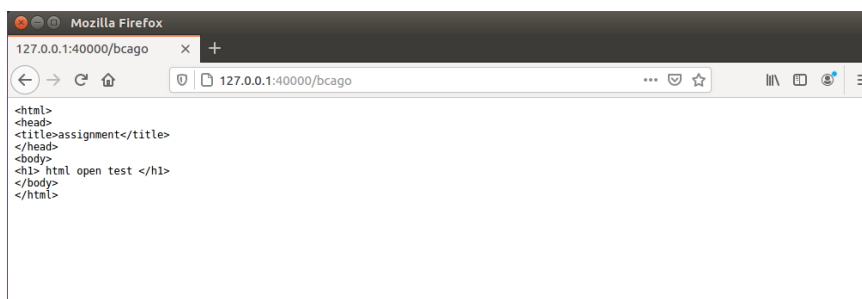
//=====
// selection sort
//
// input: count -> how many files?
// list -> file list
// path -> file path
// flag -> -S -r option flag
//
// output: void
//
// purpose: selection sorting
//=====
// lsfunc
//
// input: flag -> -a,-l,-h,-S,-r option flag
// path -> file or directory path
// client_fd -> file send to client
// response_header,message -> server's response//
```

source code를 하이퍼링크를 통해 열어본 모습이다. 서버에서 보낸 파일의 정보를 읽을 수 있다.

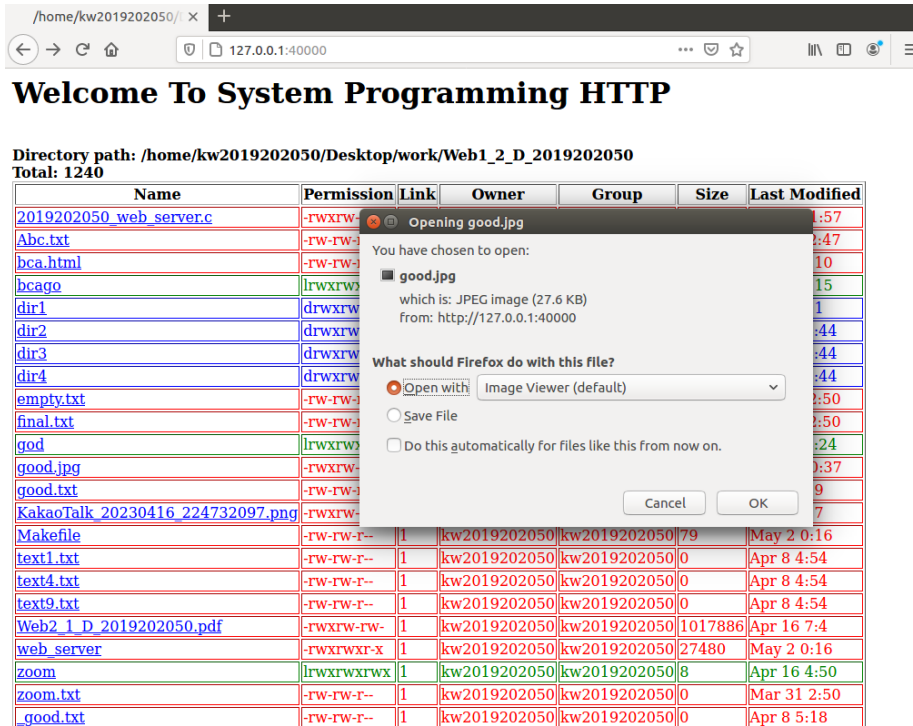


## html open test

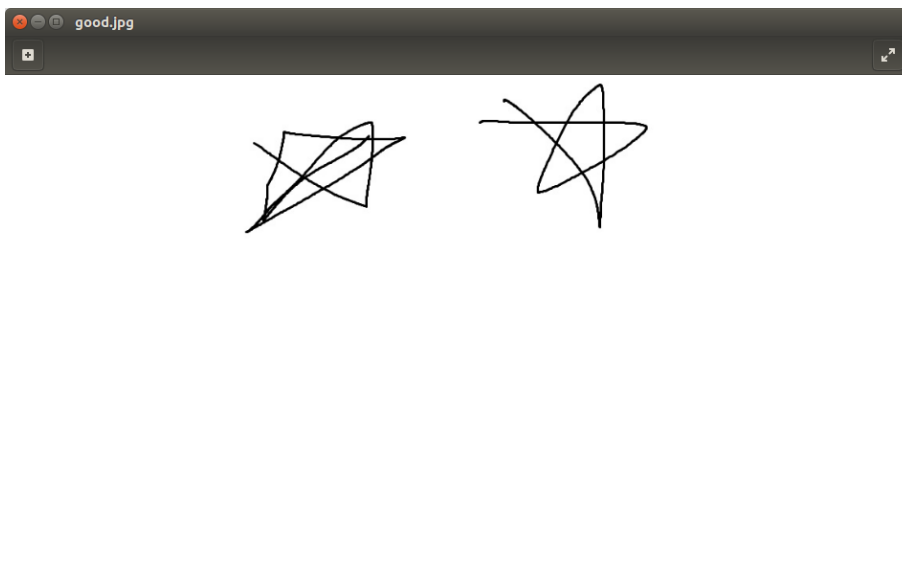
이번엔 html파일을 열어본 모습이다.



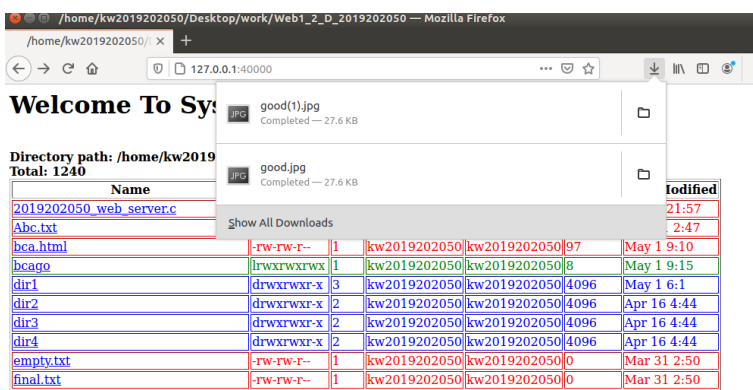
위의 html파일에 심볼릭 링크로 연결된 파일을 열었을 때 파일 자체에 적힌 코드를 확인할 수 있다.



이미지 파일을 클릭했을 때 확인할 수 있는 결과이다. open with를 클릭하면



위와 같이 사진을 볼 수 있고



save를 하게 되면 파일이 다운된 것을 알 수 있다.

## Welcome To System Programming HTTP

Directory path: /home/kw2019202050/Desktop/work/Web1\_2\_D\_2019202050  
Total: 1240

Name	Permission	Link	Owner	Group	Size	Last Modified
2019202050_web_server.c	-rwxrwx					:57
Abc.txt	-rw-rw					:47
bca.html	-rw-rw					10
bcago	lrwxrw					15
dir1	drwxrw					1
dir2	drwxrw					:44
dir3	drwxrw					:44
dir4	drwxrw					:44
empty.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Mar 31 2:50
final.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Mar 31 2:50
god	lrwxrwxrwx	1	kw2019202050	kw2019202050	8	Apr 16 4:24
good.jpg	-rwxrw-rw-	1	kw2019202050	kw2019202050	28280	May 1 20:37
good.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	13	May 1 9:9
KakaoTalk_20230416_224732097.png	-rwxrw-rw-	1	kw2019202050	kw2019202050	122087	May 1 9:7
Makefile	-rw-rw-r--	1	kw2019202050	kw2019202050	79	May 2 0:16
text1.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Apr 8 4:54
text4.txt	-rw-rw-r--	1	kw2019202050	kw2019202050	0	Apr 8 4:54

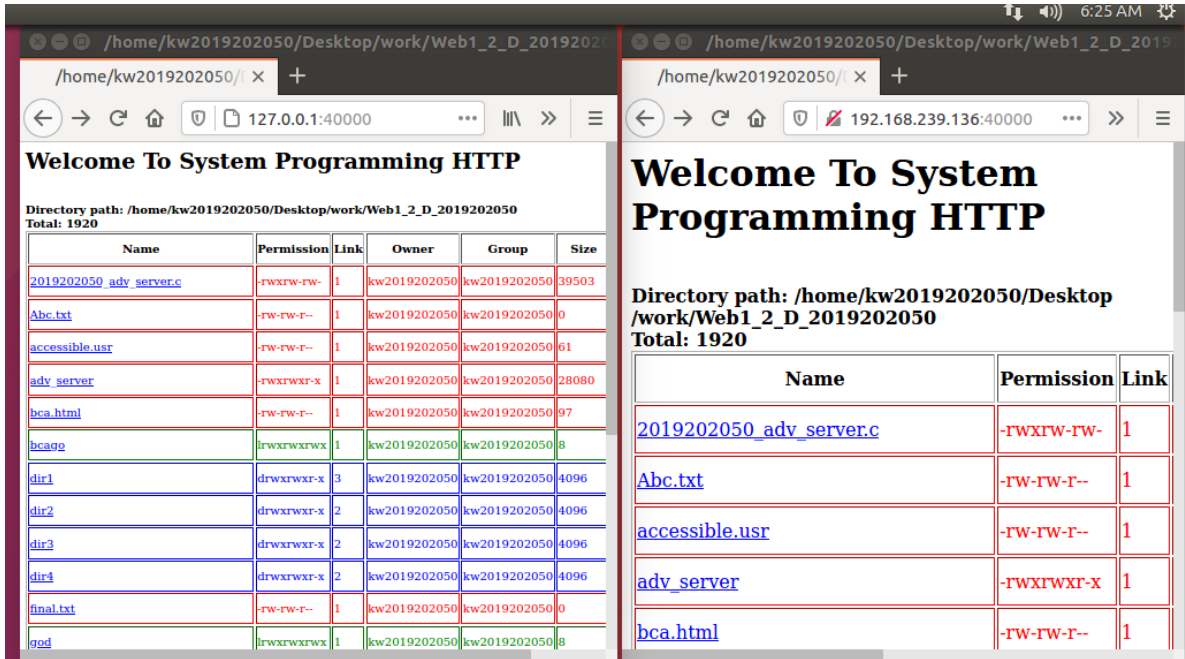
실행파일 또한 파일을 다운할 수 있다.

추가적으로 이번 과제에서 구현한 실행 결과를 검증한다.

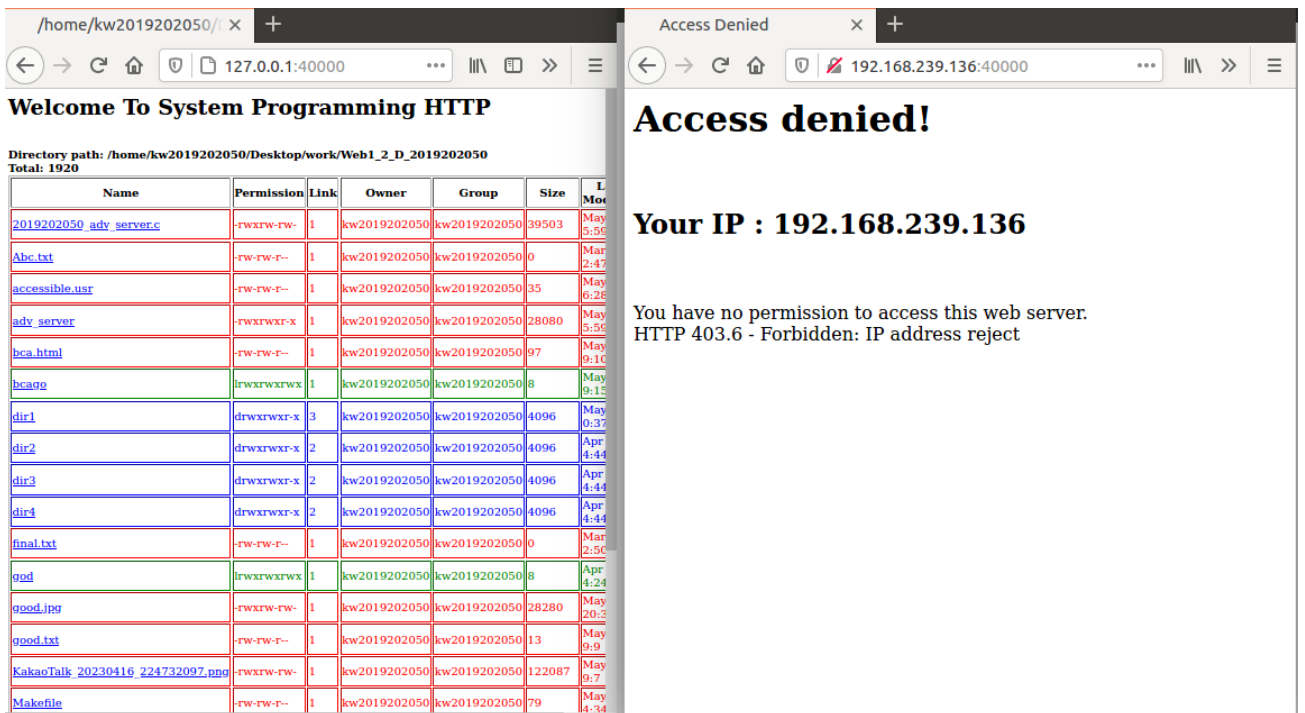
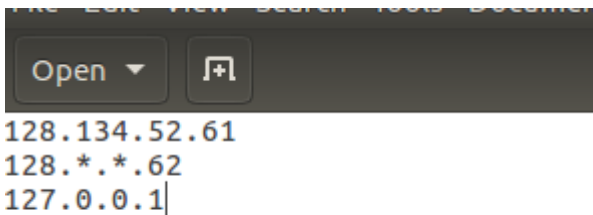
```
kw2019202050@ubuntu: ~  
kw2019202050@ubuntu:~$ ifconfig  
ens33      Link encap:Ethernet  HWaddr 00:0c:29:b4:82:98  
            inet addr:192.168.239.136  Bcast:192.168.239.255  Mask:255.255.255.0  
            inet6 addr: fe80::4438:d69f:cd3e:702a/64 Scope:Link  
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1  
            RX packets:9057 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:6524 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:1000  
            RX bytes:6508179 (6.5 MB)  TX bytes:1757660 (1.7 MB)  
  
lo         Link encap:Local Loopback  
            inet addr:127.0.0.1  Mask:255.0.0.0  
            inet6 addr: ::1/128 Scope:Host  
            UP LOOPBACK RUNNING  MTU:65536  Metric:1  
            RX packets:3208 errors:0 dropped:0 overruns:0 frame:0  
            TX packets:3208 errors:0 dropped:0 overruns:0 carrier:0  
            collisions:0 txqueuelen:1000  
            RX bytes:839415 (839.4 KB)  TX bytes:839415 (839.4 KB)
```

위 사진을 보면 local과 vmware에서 직접 이더넷과 연결된 두 가지의 ip를 확인할 수 있다.  
따라서 accessible usr를 통한 접근가능 ip에 대한 검증을 두 가지 ip를 통해 진행한다.

```
accessible usr (~/Desktop/work/Web1_2_D_2019202050) - gedit  
128.134.52.61  
192.168.*.*  
128.*.*.62  
127.0.0.1  
192.168.239.*
```



먼저 두 가지 ip를 모두 파일에 입력한 경우 두 경우 다 결과페이지를 보여준다.



192.168.239.136 ip는 접근가능한 ip목록에서 사라졌으므로 접근이 불가능하다는 것을 알린다.

또한 이번 과제는 여러 개의 클라이언트가 동시에 서버에 요청을 보낼 수 있게끔 하는 다중 접속 지원이 핵심이었기에 자식프로세스가 클라이언트의 요청을 처리하는 동안 부모 프로세스는 새로운 클라이언트와 자식프로세스를 연결해주는 것을 보여줄 필요가 있었다. 하지만 html파일을 보내주는 것만이 자식 프로세스의 역할이라 너무 빨리 종료되어 결과를 확인하면 다중 접속이 가능하다는 것을 입증하기가 어려웠다.

따라서 자식프로세스의 종료를 의도적으로 10초 늦추고 10초 안에 여러 개의 요청을 보냈을 때 웹 페이지를 보여주는지 확인하고 그 후에 자식프로세스가 종료되는 것을 signalhandler함수를 통해 보여주는 방법을 사용하였다.

따라서 아래와 같은 결과를 확인할 수 있다.

The screenshot shows a terminal window on the left and a web browser on the right. The terminal window displays the output of the `adv_server` program, which is a multi-threaded web server. It shows three clients connecting, each with a unique IP and port. The server processes each request and then waits for the child process to finish. The connection history shows three requests, each with a unique IP and port. The web browser on the right shows the directory listing for the path `/home/kw2019202050/Desktop/work/Web1_2_D_2019202050/`. The directory listing shows three files, each with a unique name and size.

```
kw2019202050@ubuntu: ~/Desktop/work/Web1_2_D_2019202050
kw2019202050@ubuntu:~/Desktop/work/Web1_2_D_2019202050$ ./adv_server
===== New Client =====
IP : 127.0.0.1
Port : 11949
=====
===== Disconnected Client =====
IP : 127.0.0.1
Port : 11949
=====
===== New Client =====
IP : 127.0.0.1
Port : 12461
=====
===== Disconnected Client =====
IP : 127.0.0.1
Port : 12461
=====
===== New Client =====
IP : 127.0.0.1
Port : 12973
=====
===== Disconnected Client =====
IP : 127.0.0.1
Port : 12973
=====
===== Connection History =====
Number of request(s) : 3
No.   IP       PID    PORT   TIME
1     127.0.0.1  5975   12973   Mon May 8 04:42:51 2023
2     127.0.0.1  5974   12461   Mon May 8 04:42:49 2023
3     127.0.0.1  5972   11949   Mon May 8 04:42:46 2023
wait for child process
child pid = 5972
wait for child process
child pid = 5974
wait for child process
child pid = 5975
```

System Programming HTTP

Directory path: /home/kw2019202050/Desktop/work/Web1\_2\_D\_2019202050/

Total: 12

Name	Permission	Link	Owner	Group	Size	Last Modified
1	drwxrwxr-x	3	kw2019202050	kw2019202050	4096	May 1 6:1
2	drwxrwxr-x	3	kw2019202050	kw2019202050	4096	May 2 0:37
bye	drwxrwxr-x	3	kw2019202050	kw2019202050	4096	May 1 6:1

3개의 클라이언트의 요청을 처리하는 것을 보면 순차적으로 처리하는 것으로 보이나 출력이 완료된 후 마지막에 signalhandler함수의 waitpid를 했을 때 나오는 출력문을 보고 3개의 클라이언트들은 각각 부모가 복제한 자식프로세스들과 각각 통신을 진행하고 있는 것을 확인할 수 있다.

마지막으로 가장 최근에 연결된 클라이언트들의 목록을 10개만 보여주는 결과이다.

```
===== Connection History =====
Number of request(s) : 10
No.  IP      PID    PORT    TIME
1    127.0.0.1 5988   17069   Mon May 8 04:44:01 2023
2    127.0.0.1 5987   16045   Mon May 8 04:43:48 2023
3    127.0.0.1 5986   15533   Mon May 8 04:43:47 2023
4    127.0.0.1 5985   15021   Mon May 8 04:43:45 2023
5    127.0.0.1 5984   14509   Mon May 8 04:43:45 2023
6    127.0.0.1 5983   13997   Mon May 8 04:43:44 2023
7    127.0.0.1 5981   13485   Mon May 8 04:43:44 2023
8    127.0.0.1 5975   12973   Mon May 8 04:42:51 2023
9    127.0.0.1 5974   12461   Mon May 8 04:42:49 2023
10   127.0.0.1 5972   11949   Mon May 8 04:42:46 2023

===== New Client =====
IP : 127.0.0.1
Port : 18605
=====
===== Disconnected Client =====
IP : 127.0.0.1
Port : 18605
=====
===== New Client =====
IP : 127.0.0.1
Port : 19117
=====
===== Disconnected Client =====
IP : 127.0.0.1
Port : 19117
=====

===== Connection History =====
Number of request(s) : 12
No.  IP      PID    PORT    TIME
1    127.0.0.1 6005   19117   Mon May 8 04:44:35 2023
2    127.0.0.1 6003   18605   Mon May 8 04:44:35 2023
3    127.0.0.1 5988   17069   Mon May 8 04:44:01 2023
4    127.0.0.1 5987   16045   Mon May 8 04:43:48 2023
5    127.0.0.1 5986   15533   Mon May 8 04:43:47 2023
6    127.0.0.1 5985   15021   Mon May 8 04:43:45 2023
7    127.0.0.1 5984   14509   Mon May 8 04:43:45 2023
8    127.0.0.1 5983   13997   Mon May 8 04:43:44 2023
9    127.0.0.1 5981   13485   Mon May 8 04:43:44 2023
10   127.0.0.1 5975   12973   Mon May 8 04:42:51 2023

wait for child process
child pid = 6003
wait for child process
child pid = 6005
```

10개로 가득 찬 history가 추가적으로 두 번 더 클라이언트의 요청을 받았을 때 최근 10번의 기록만을 보여주도록 갱신된 것을 확인할 수 있다.

## 5. 고찰

이번 과제는 fork함수를 통한 다중 접속 지원 서버를 구현하는 것이었다. fork함수의 반환값이 두 개라는 점이 내가 바라보는 코드는 하나인데 두 개가 다르게 실행된다는 것이 헛갈렸다. 자식은 부모가 연결해 준 클라이언트에게 보내주는 역할만 진행하고 부모는 그 사이에 또 다른 클라이언트의 요청을 받는다는 그 흐름을 직접 코드로 구현하기가 어색했고 또 자식프로세스가 클라이언트와 통신을 하기에 직접 클라이언트의 정보와 자신의 PID를 전역변수에 넣는다는 방법을 사용했으나 전역변수를 바꾸어도 부모의 전역변수는 바뀌지 않아서 어려움을 겪었다. 이를 fork가 아닌 vfork를 사용하여 전역변수를 공유하려고 했으나 이 또한 부모가 자식프로세스가 종료될때까지 block되어 있기 때문에 이는 다중접속지원이라는 취지에 맞지 않았다. 부모가 한번에 하나의 자식만 가지기 때문이다. 고민 끝에 부모는 자식의 PID를 가진다는 점과 자식은 연결된 이후의 일만 하면 되기에 부모가 전역변수를 갱신하도록 코드를 구현하여 해결하였다. 단순히 복제를 한다는 것보다 어떤 것을 공유하고 어떤 것은 공유하지 않는지 각자의 역할은 해당과제에서 무엇이어야 하는지를 곰곰히 고민해보아야 문제를 해결할 수 있었던 것 같다.

## 6. Reference

시스템프로그래밍실습 강의자료 참조