

시스템프로그래밍실습

assignment 3-1

학 과: 컴퓨터정보공학부

담당교수: 이기훈 교수님

학 번: 2019202050

성 명: 이강현

1. Introduction

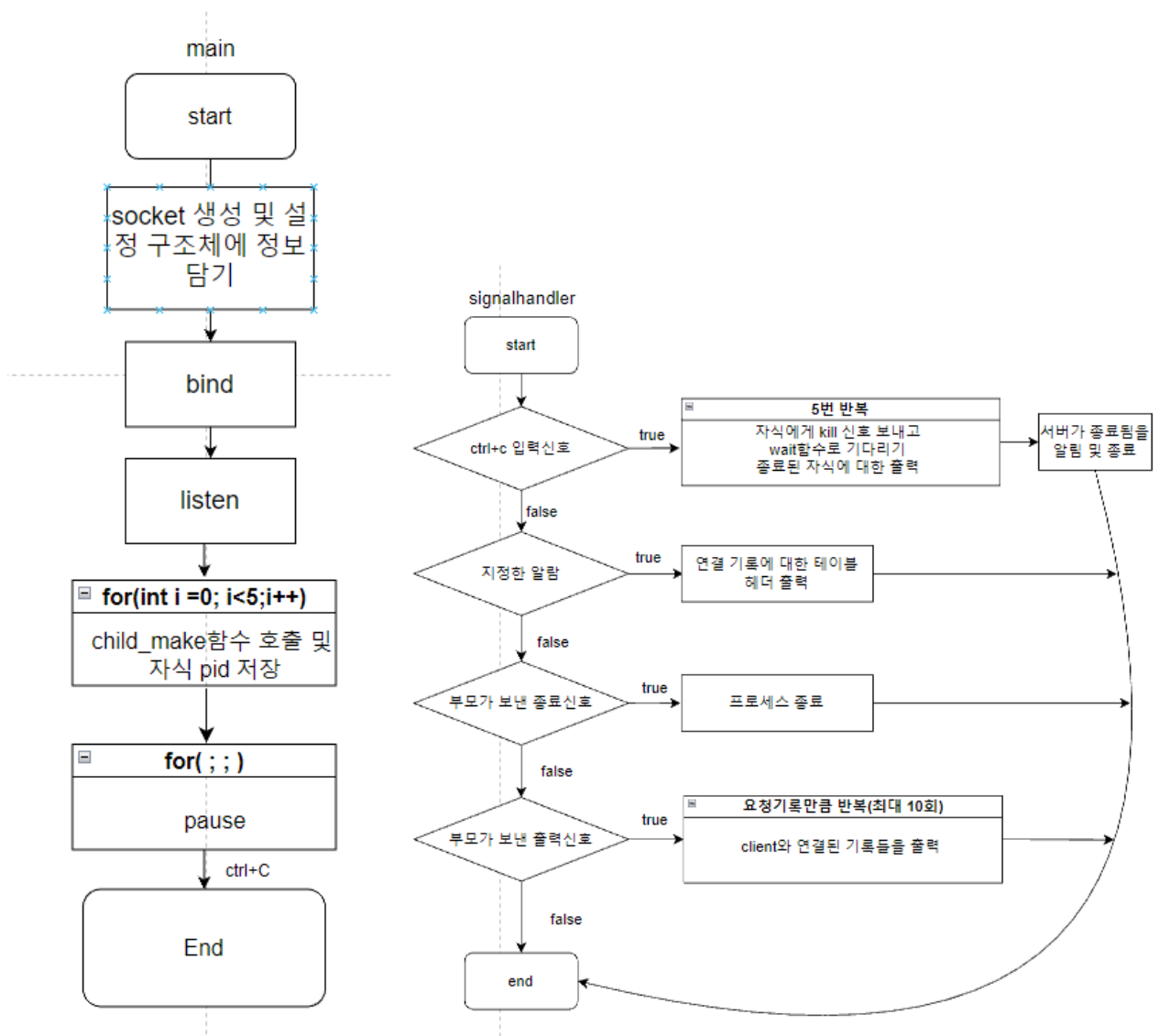
이번 과제는 pre-forked child process를 이용하여 server에서 client에게 서비스를 제공하는 방식으로 서버를 구현하여 본다.

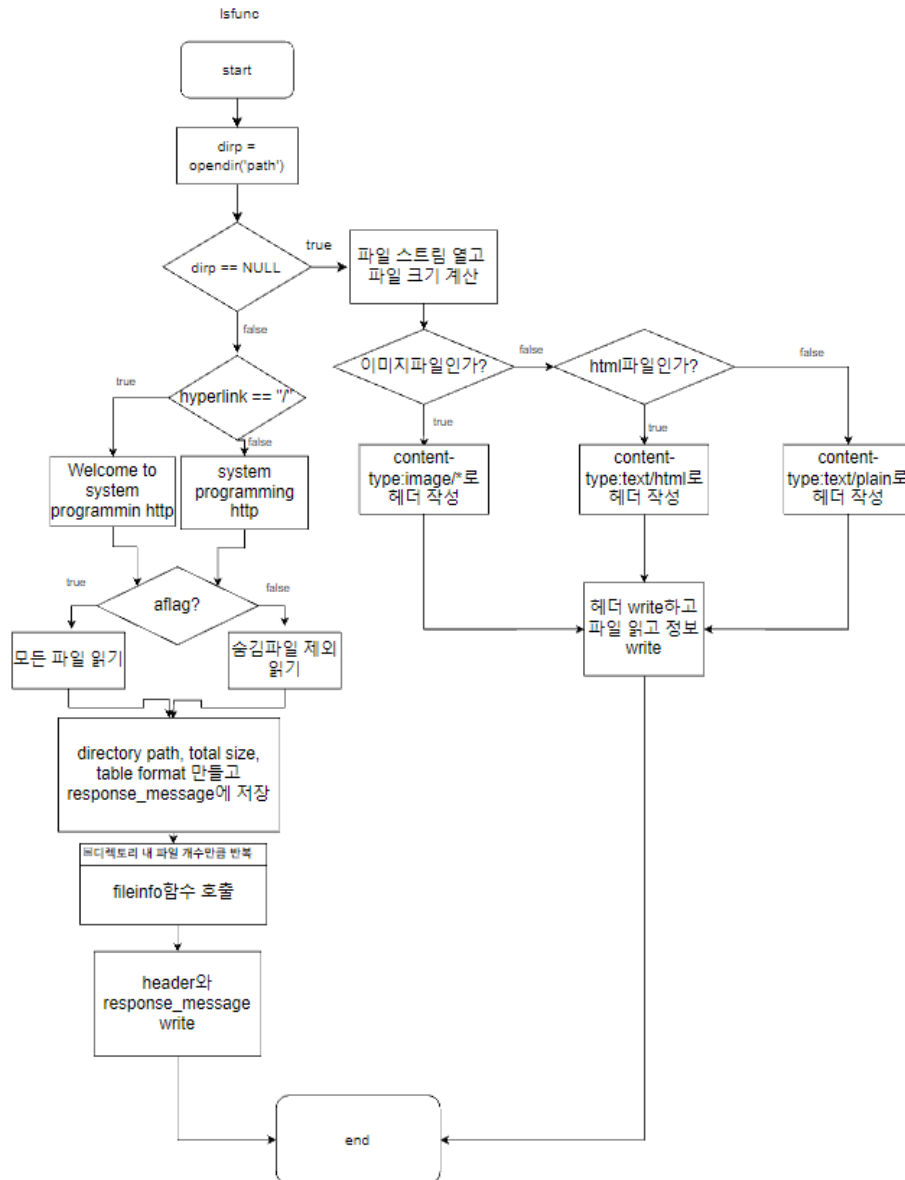
다양한 signal들을 추가적으로 학습하고 이를 활용하여 본다.

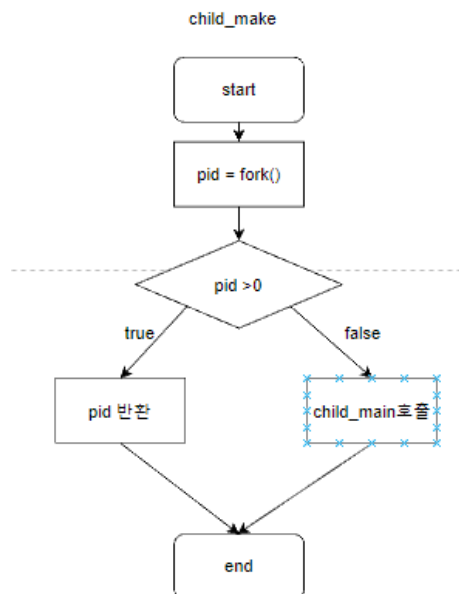
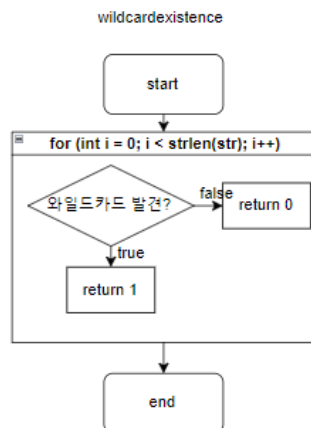
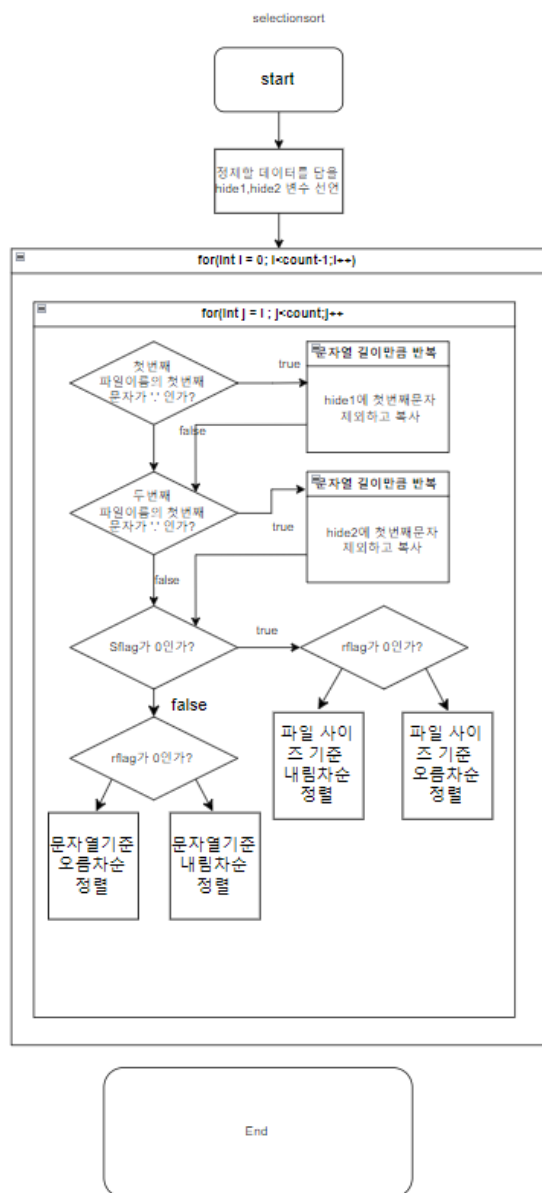
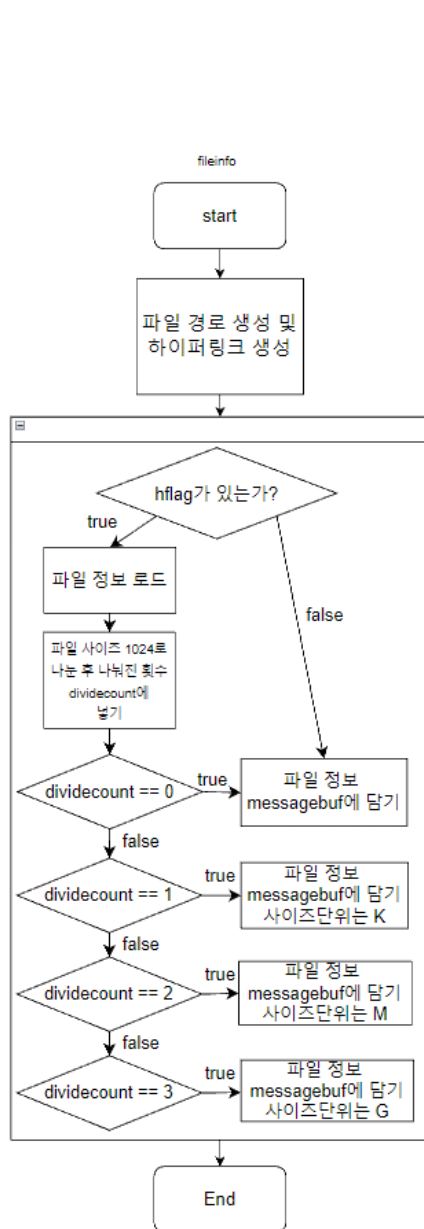
signal를 적절하게 활용하여 프로세스가 프로세스에게 신호를 보내는 법을 안다.

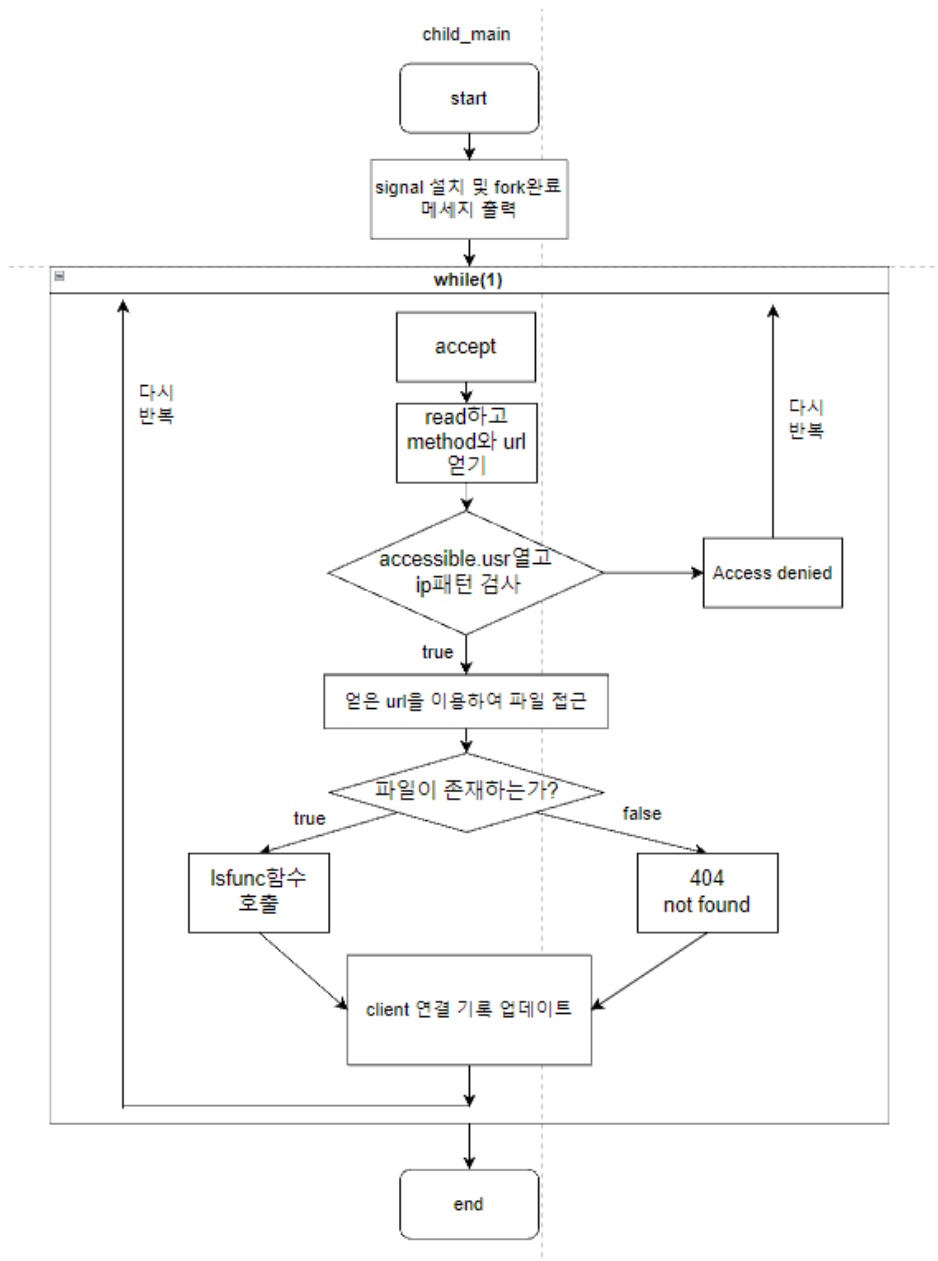
signal을 받았을 때 핸들링 하는 법을 알고 사용자정의 signal를 사용해 본다.

2. Flow chart









모든 함수들의 흐름도이다. main 함수는 소켓을 생성하여 소켓을 설정을 마치고 bind,listen을 마무리한 후 pre-forked방식에 맞게 5번 child_make함수를 호출한 후 pause함수로 무한루프에 빠진다. 따라서 signal만을 처리할 뿐 아무일도 하지 않는다.

child_make함수는 5번 fork함수를 호출 한 후 부모는 pid반환 자식은 child_main함수를 진행하도록 section을 나눈다.

child_main함수는 자식 프로세스의 일로 인자로 받은 서버 소켓을 이용하여 client와 accept를 진행하고 client의 요청을 수행하고 유지된다. 또한 부모 프로세스에게 받은 여러 signal들을 처리할 수 있게 되어있다.

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 al 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 SIGALRM, SIGINT, SIGTERM, SIGUSR1 4가지의 signal을 처리한다. SIGALRM, SIGINT는 부모가 처리할 신호들이고 SIGTERM, SIGUSR1은 부모가 자식에게 보내는 signal로서 자식이 처리해야 할 신호들이다. SIGALRM에서는 부모프로세스에서만 알람을 설치해 두었기에 부모가 알람신호를 받으면 연결기록 title을 출력한 후 kill함수를 통해 자식 프로세스에게 SIGUSR1신호를 보낸다. 따라서 SIGUSR1 신호를 받은 자식은 자신이 관리하는 client와의 연결기록을 출력한다. SIGINT신호는 터미널에서 ctrl+c를 입력하였을 때 생성되며 이를 부모가 처리하고 자식은 이에 대한 신호에 무시를 하도록 설정되어있다. 따라서 부모가 kill을 통해 자식에게 SIGTERM신호를 대신 보내고 자식은 SIGTERM신호를 받게되면 프로세스를 종료한다. 부모는 종료된 자식의 종료상태를 wait함수로 받게 되고 반환된 자식의 pid와 종료메세지를 출력한다. 이를 5번 반복하여 모든 자식프로세스를 종료시킨 후 부모는 스스로의 종료 메시지를 출력하며 종료된다.

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
```

```
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
```

```
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
```

```
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(ctrl+c 신호 발생시 SIGINT)
    for(5번 반복)
        모든 자식에게 kill함수를 통해 SIGTERM신호를 보냄
        wait로 자식의 종료상태를 받음
        자식의 종료 상태를 출력
        부모 스스로 종료메시지를 출력
        부모 종료
else if(signal이 지정한 알람시간이 다 되었음을 알림 SIGALRM)
    연결 기록 제목을 출력
    for(5번 반복)
        kill함수를 통해 자식들에게 SIGUSR1신호를 보냄
else if(SIGUSR1신호 일때)
    서버가 열린 이래로 연결되었던 클라이언트에 대한 정보와
    이를 처리했던 자식 프로세스의 pid, 연결 시간을 출력(최근 10회의 기록까지)
else if(SIGTERM신호 일때)
    프로세스 종료

```

```

main
소켓 생성
소켓 설정
bind
listen
child_make함수를 5번 반복
무한 루프에 빠져 pause함수호출

소켓 닫기
종료

```

```

child_make
fork함수로 프로세스 복제
if(pid>0일때)
    pid반환
else if(pid == 0 일때)
    child_main함수 호출

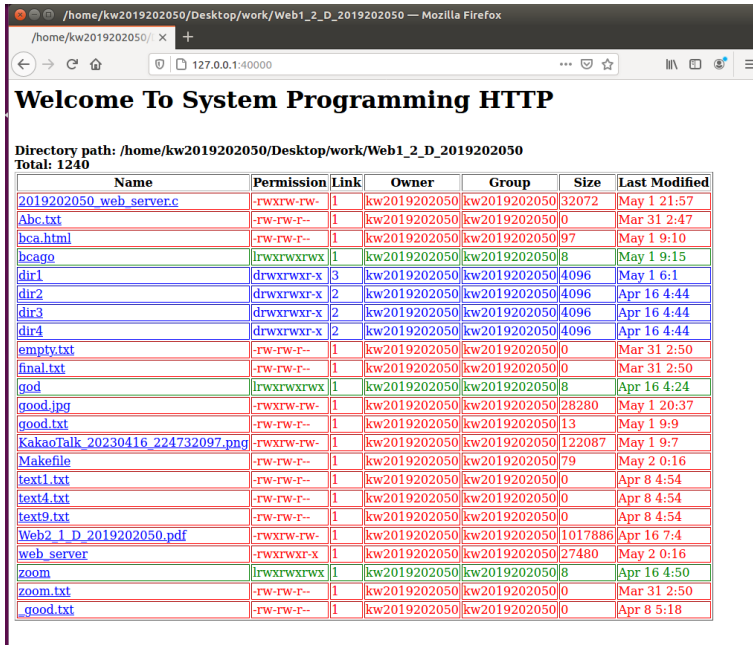
```

```

child_main
필요한 구조체, 변수 선언 및 초기화
while(1)
    client accept
    method와 url 토큰으로 얻기
    접근가능한 ip주소가 저장된 파일 열고 현재 클라이언트 ip와 비교
    일치하면 accessflag = 1, 아니면 0
    accessflag == 0일때
        접근 불가 메시지를 담은 페이지를 보내고 continue
    accessflag == 1일때
        현재디렉토리를 루트 디렉토리로 간주하고 url이어붙여 경로 생성
        해당 경로에 파일이 존재하지 않는다면 에러전용 헤더와 메시지를 보내주기
        존재한다면 lsfunc함수 호출 후 client disconnect
        continue
    클라이언트의 정보를 구조체에 저장
    클라이언트와 연결된 시간을 저장

```

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

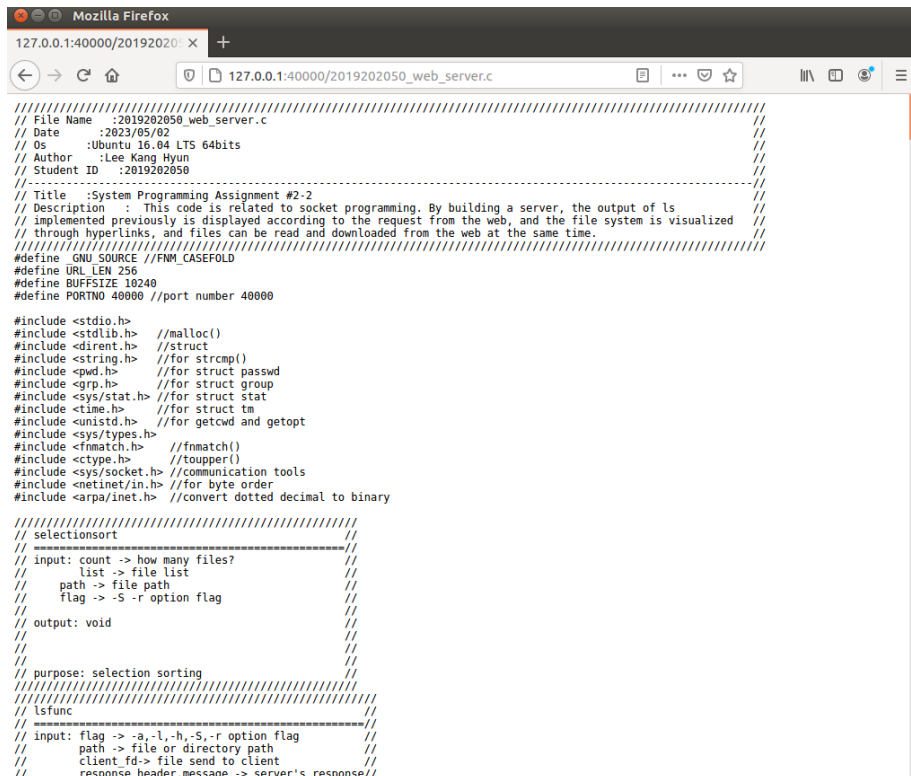
루트 디렉토리의 모습이다. 상위 디렉토리로 접근하지 못하도록 -l 옵션만 적용된 상태이다.



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.

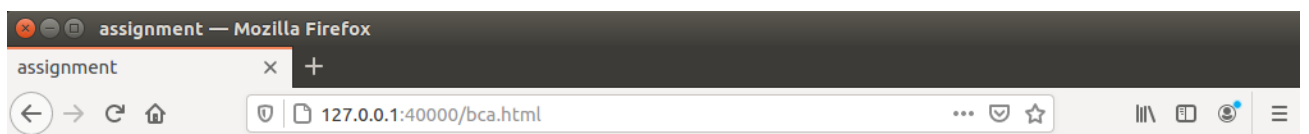


```
// File Name : 2019202050_web_server.c
// Date : 2023/05/02
// Os : Ubuntu 16.04 LTS 64bits
// Author : Lee Kang Hyun
// Student ID : 2019202050
// Title : System Programming Assignment #2-2
// Description : This code is related to socket programming. By building a server, the output of ls
// implemented 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

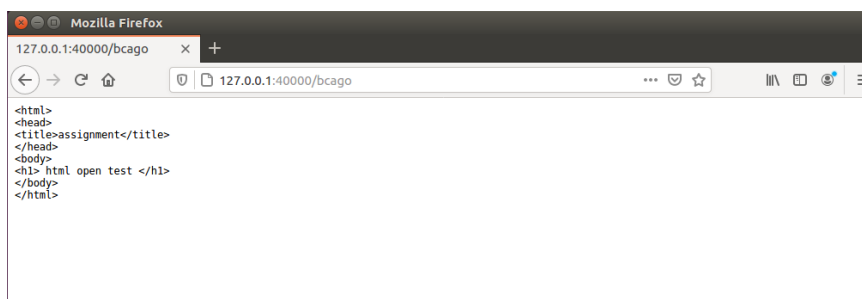
// =====
// selection sort
//
// input: count -> how many files?
// list -> file list
// path -> file path
// flag -> -S -r option flag
//
// output: void
//
// purpose: selection sorting
// =====
// lsfunc
//
// input: flag -> -a,-l,-h,-S,-r option flag
// path -> file or directory path
// client_fd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

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

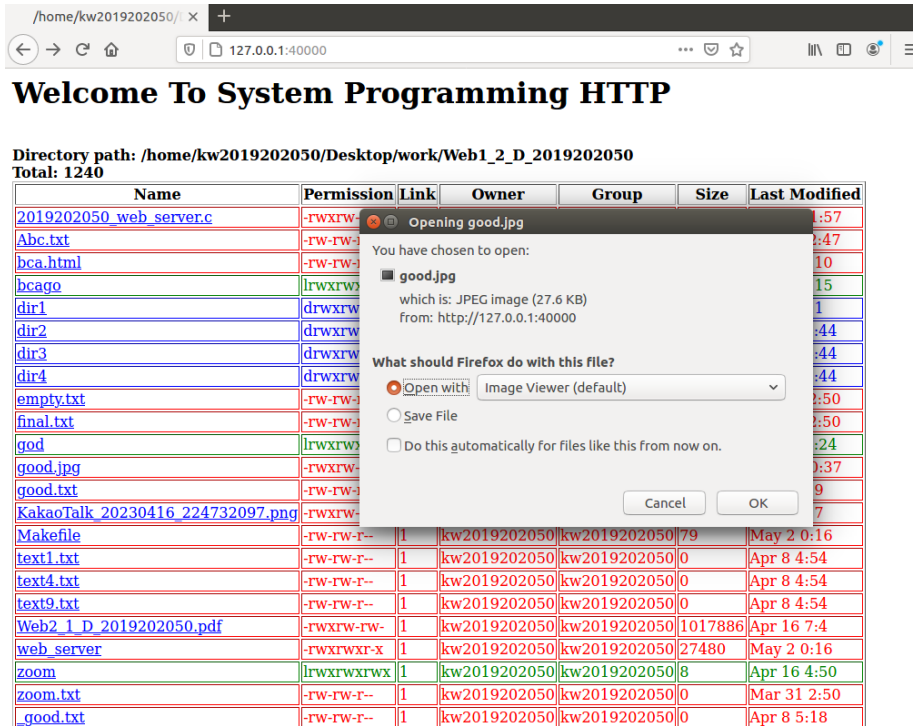


html open test

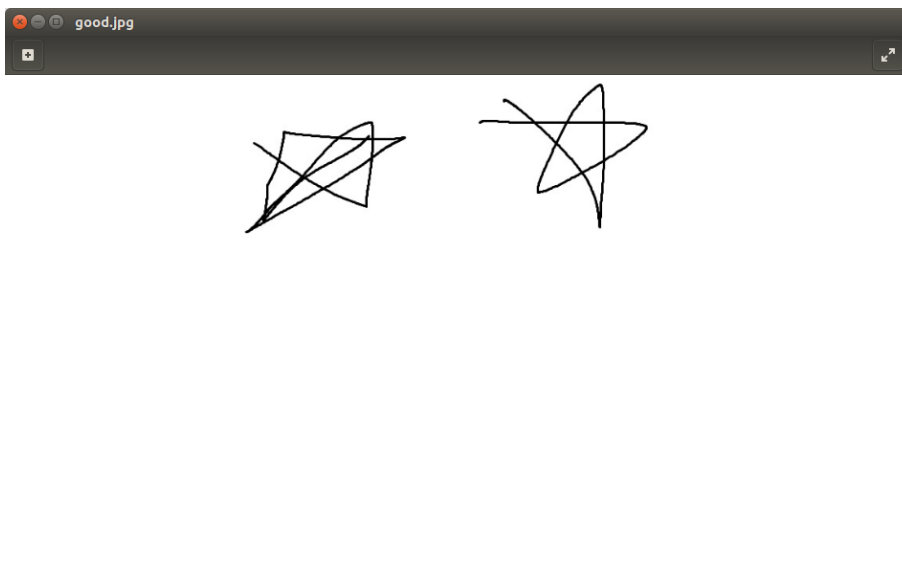
이번엔 html파일을 열어본 모습이다.



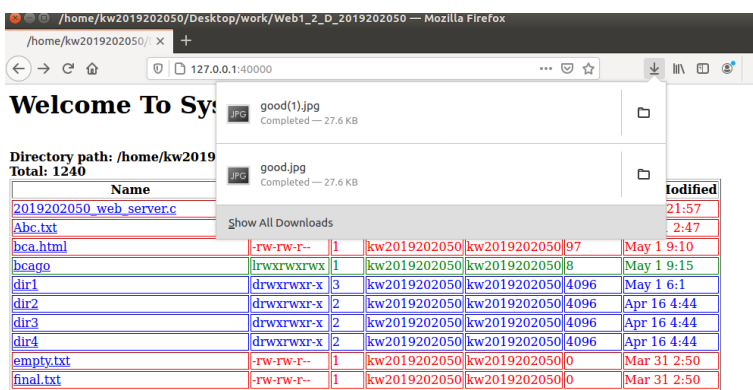
위의 html파일에 심볼릭 링크로 연결된 파일을 열었을 때 파일 자체에 적힌 코드를 확인할 수 있다.



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	-rwxrwxrwx					Mar 31 2:57
Abc.txt	-rwxrwxrwx					Mar 31 2:47
bca.html	-rwxrwxrwx					Mar 31 2:10
bcago	lrwxrwxrwx					Mar 31 2:15
dir1	drwxrwxrwx					Mar 31 2:1
dir2	drwxrwxrwx					Mar 31 2:44
dir3	drwxrwxrwx					Mar 31 2:44
dir4	drwxrwxrwx					Mar 31 2:44
empty.txt	-rwxrwxrwx					Mar 31 2:50
final.txt	-rwxrwxrwx					Mar 31 2:50
god	lrwxrwxrwx					Apr 16 4:24
good.jpg	-rwxrwxrwx				28280	May 1 20:37
good.txt	-rwxrwxrwx				13	May 1 9:9
KakaoTalk_20230416_224732097.png	-rwxrwxrwx				122087	May 1 9:7
Makefile	-rwxrwxrwx				79	May 2 0:16
text1.txt	-rwxrwxrwx				0	Apr 8 4:54
text4.txt	-rwxrwxrwx				0	Apr 8 4:54

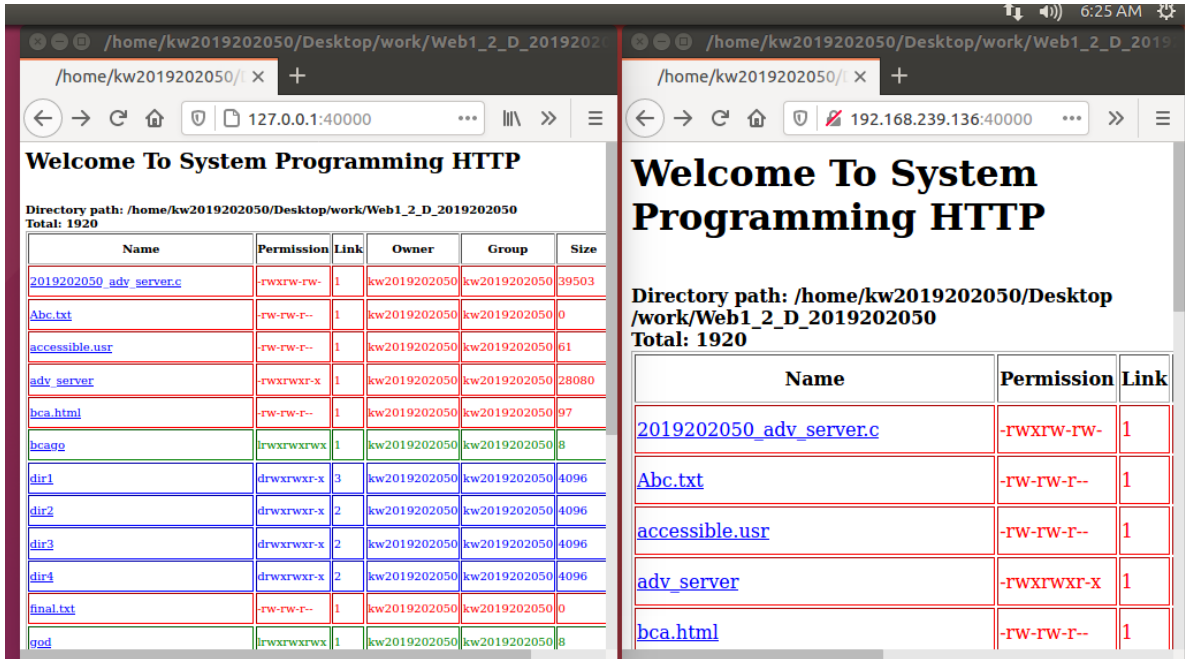
실행파일 또한 파일을 다운할 수 있다.

```
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)

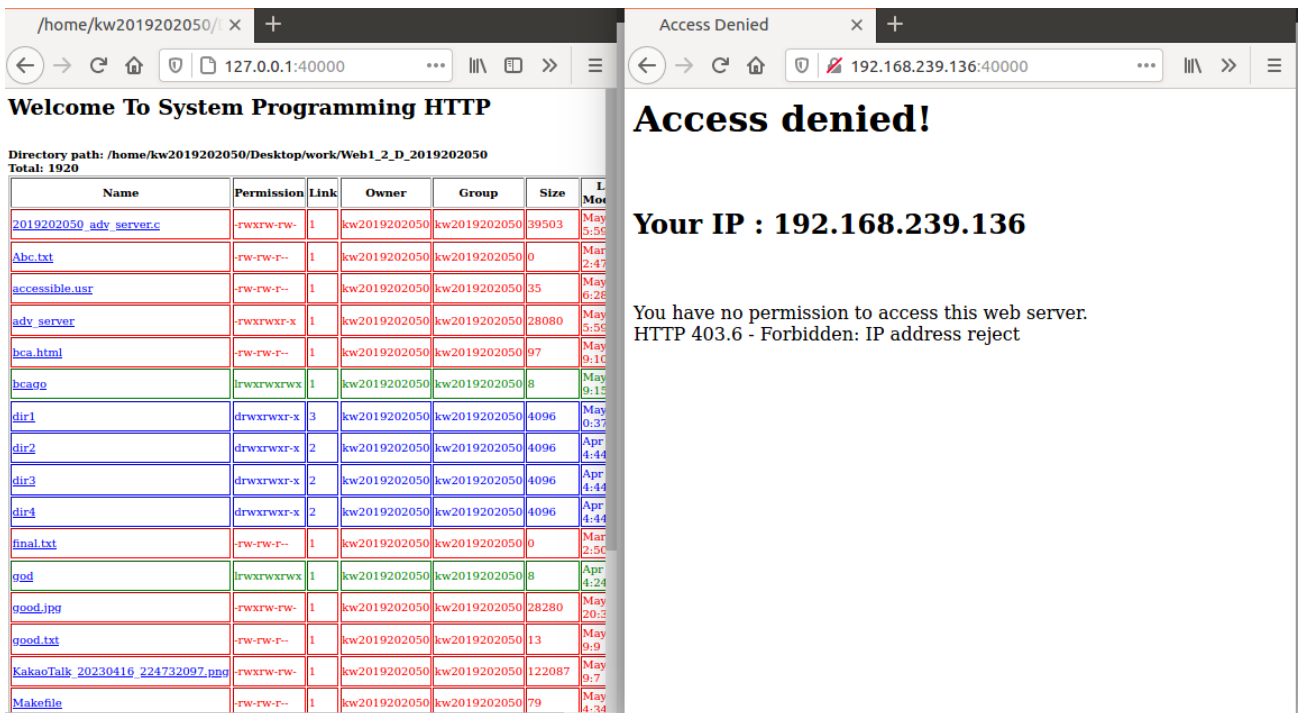
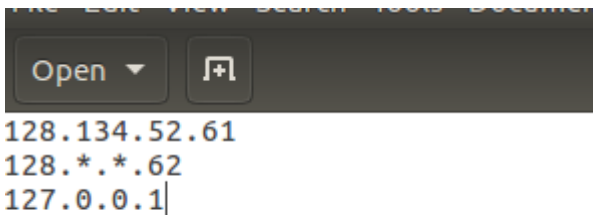
lo         Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:65536  Metric:1
            RX packets:3208 errors:0 dropped:0 overruns:0 frame:0
            TX packets:3208 errors:0 dropped:0 overruns:0 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.

```
kw2019202050@ubuntu:~/Desktop/work/Web1_2_D_2019202050$ ./preforked_server
[Sun May 14 03:31:43 2023] Server is started.
[Sun May 14 03:31:43 2023] 4222 process is forked.
[Sun May 14 03:31:43 2023] 4223 process is forked.
[Sun May 14 03:31:43 2023] 4225 process is forked.
[Sun May 14 03:31:43 2023] 4226 process is forked.
[Sun May 14 03:31:43 2023] 4224 process is forked.
```

먼저 실행파일을 실행한 결과 server가 먼저 시작되고 자식 프로세스들이 forked되는 것을 확인할 수 있다.

```
===== Connection History =====
No.      IP          PID      PORT      TIME
1        192.168.239.136 2969     696       Sun May 14 02:17:13 2023
2        127.0.0.1      2968     31919     Sun May 14 02:17:17 2023
1        192.168.239.136 2968     1208      Sun May 14 02:17:13 2023
1        127.0.0.1      2966     31407     Sun May 14 02:17:17 2023
2        192.168.239.136 2970     184       Sun May 14 02:17:09 2023
1        127.0.0.1      2970     16559     Sun May 14 02:16:26 2023
1        127.0.0.1      2967     30895     Sun May 14 02:17:15 2023
```

그 후 연결을 시도하여 요청을 완료한 결과를 보여주고 이는 각 자식 프로세스에 의해 출력되므로 pid에 따라 개별적으로 관리된다. 이전 과제의 연장으로 같은 pid에서 처리된 기록들은 최근 시간순으로 출력되는 것을 알 수 있다.

```
No.      IP          PID      PORT      TIME
12       192.168.239.136 2968     27320     Sun May 14 02:18:20 2023
11       127.0.0.1      2968     53935     Sun May 14 02:18:18 2023
10       192.168.239.136 2968     21688     Sun May 14 02:18:11 2023
9        192.168.239.136 2968     18616     Sun May 14 02:18:00 2023
8        127.0.0.1      2968     45231     Sun May 14 02:17:58 2023
7        192.168.239.136 2968     13496     Sun May 14 02:17:56 2023
6        192.168.239.136 2968     10936     Sun May 14 02:17:56 2023
5        127.0.0.1      2968     37551     Sun May 14 02:17:52 2023
4        127.0.0.1      2968     34991     Sun May 14 02:17:52 2023
3        127.0.0.1      2968     32431     Sun May 14 02:17:51 2023
11       192.168.239.136 2969     27832     Sun May 14 02:18:20 2023
10       127.0.0.1      2969     54447     Sun May 14 02:18:18 2023
9        192.168.239.136 2969     22712     Sun May 14 02:18:11 2023
8        192.168.239.136 2969     19128     Sun May 14 02:18:01 2023
7        127.0.0.1      2969     45743     Sun May 14 02:17:58 2023
6        192.168.239.136 2969     14008     Sun May 14 02:17:56 2023
5        192.168.239.136 2969     11448     Sun May 14 02:17:56 2023
4        192.168.239.136 2969     8888      Sun May 14 02:17:55 2023
3        127.0.0.1      2969     35503     Sun May 14 02:17:52 2023
2        127.0.0.1      2969     32943     Sun May 14 02:17:51 2023
12       192.168.239.136 2970     28856     Sun May 14 02:18:20 2023
11       192.168.239.136 2970     26296     Sun May 14 02:18:20 2023
10       127.0.0.1      2970     52399     Sun May 14 02:18:17 2023
9        192.168.239.136 2970     20152     Sun May 14 02:18:02 2023
8        127.0.0.1      2970     46767     Sun May 14 02:17:59 2023
7        127.0.0.1      2970     44207     Sun May 14 02:17:57 2023
6        192.168.239.136 2970     12472     Sun May 14 02:17:56 2023
5        192.168.239.136 2970     9912      Sun May 14 02:17:55 2023
4        127.0.0.1      2970     36015     Sun May 14 02:17:52 2023
3        127.0.0.1      2970     33455     Sun May 14 02:17:51 2023
12       192.168.239.136 2967     28344     Sun May 14 02:18:20 2023
11       192.168.239.136 2967     25784     Sun May 14 02:18:19 2023
10       127.0.0.1      2967     24788     Sun May 14 02:18:14 2023
9        192.168.239.136 2967     21176     Sun May 14 02:18:11 2023
8        192.168.239.136 2967     19640     Sun May 14 02:18:01 2023
7        127.0.0.1      2967     46255     Sun May 14 02:17:59 2023
6        192.168.239.136 2967     14520     Sun May 14 02:17:57 2023
5        192.168.239.136 2967     11960     Sun May 14 02:17:56 2023
4        192.168.239.136 2967     9400      Sun May 14 02:17:55 2023
3        127.0.0.1      2967     37039     Sun May 14 02:17:52 2023
11       192.168.239.136 2966     26808     Sun May 14 02:18:20 2023
10       127.0.0.1      2966     53423     Sun May 14 02:18:17 2023
9        192.168.239.136 2966     22200     Sun May 14 02:18:11 2023
8        192.168.239.136 2966     20664     Sun May 14 02:18:02 2023
7        127.0.0.1      2966     47279     Sun May 14 02:17:59 2023
6        127.0.0.1      2966     44719     Sun May 14 02:17:58 2023
5        192.168.239.136 2966     12984     Sun May 14 02:17:56 2023
4        192.168.239.136 2966     10424     Sun May 14 02:17:56 2023
3        127.0.0.1      2966     36527     Sun May 14 02:17:52 2023
2        127.0.0.1      2966     33967     Sun May 14 02:17:51 2023
```


여러 번 접속을 진행 하였을 때 각 프로세스마다 개별적으로 최대 10개씩의 가장 최근 연결 기록을 가지는 것을 확인할 수 있다.

```
^C[Sun May 14 02:19:08 2023] 2966 process is terminated.
[Sun May 14 02:19:08 2023] 2967 process is terminated.
[Sun May 14 02:19:08 2023] 2968 process is terminated.
[Sun May 14 02:19:08 2023] 2969 process is terminated.
[Sun May 14 02:19:08 2023] 2970 process is terminated.
[Sun May 14 02:19:08 2023] Server is terminated.
kw2019202050@ubuntu:~/Desktop/work/Web1_2_D_2019202050$
```

종료되었을 때 서버는 자식프로세스들을 순차적으로 종료시킨 후 스스로 종료된다.

```
^C[Sun May 14 02:19:08 2023] 2966 process is terminated.
[Sun May 14 02:19:08 2023] 2967 process is terminated.
[Sun May 14 02:19:08 2023] 2968 process is terminated.
[Sun May 14 02:19:08 2023] 2969 process is terminated.
[Sun May 14 02:19:08 2023] 2970 process is terminated.
[Sun May 14 02:19:08 2023] Server is terminated.
kw2019202050@ubuntu:~/Desktop/work/Web1_2_D_2019202050$ top

top - 02:20:14 up 37 min, 1 user, load average: 0.13, 0.14, 0.09
Tasks: 236 total, 1 running, 167 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.2 us, 0.3 sy, 0.0 ni, 99.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 4015896 total, 1971688 free, 909376 used, 1134832 buff/cache
KiB Swap: 998396 total, 998396 free, 0 used. 2763700 avail Mem
```

종료 후 좀비프로세스는 존재하지 않았다.

5. 고찰

이번 과제는 preforked방식으로 서버를 관리하는 것이었다. client의 요청을 받기 전에 미리 자식 프로세스들을 생성해 둬으로써 요청이 들어왔을 때 복제하는 시간과 비용을 아낄 수 있다는 장점이 있다. 또한 이번과제에서는 signal 개념을 학습하기에 좋았는데 SIGINT, SIGTERM, SIGUSR1신호를 추가적으로 사용해보면서 신호들을 어떻게 핸들링 할 것인지 과제에서 요구되는 부모가 처리해야 하는 signal과 자식이 처리해야 하는 signal이 어떻게 다른지 알고 신호처리가 어떻게 흘러가는지 배우기 좋은 과제였다. 자식들이 개별적으로 연결기록을 관리한다는 개념이 좀 어려웠는데 자식 프로세스에서 지역변수로 기록들을 저장하려 했으나 이를 signal로 핸들링 하게 되면 사용할 수 없었다. 이 같은 문제점을 fork를 사용하면 전역변수가 복사되고 이는 다른 프로세스와 공유하지 않는다는 점을 이용해서 해결할 수 있었다. 따라서 전역변수에 한번만 선언해두면 자식프로세스 모두가 fork되었을 때 동일한이름의 전역변수를 개별적인 저장공간으로 사용할 수 있었다.

6. Reference

시스템프로그래밍실습 강의자료 참조