시스템프로그래밍
assignment 2-3

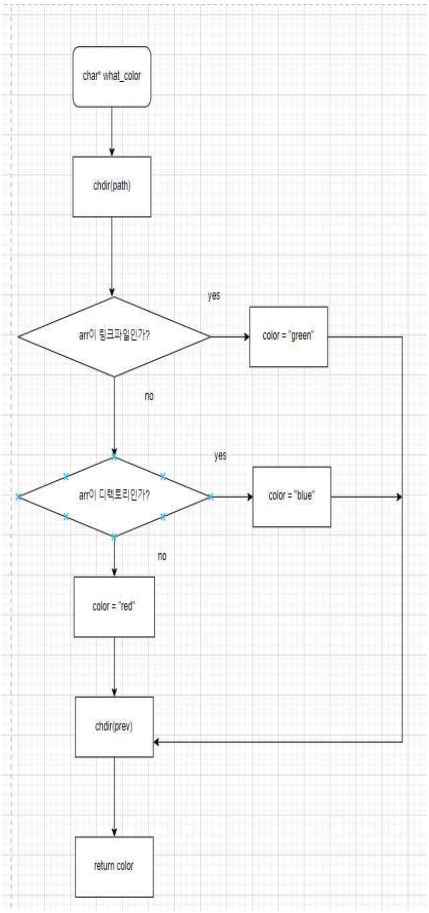
학번: 2021202045
이름: 김예은
담당 교수님: 최상호 교수님

Ⅰ. Introduction

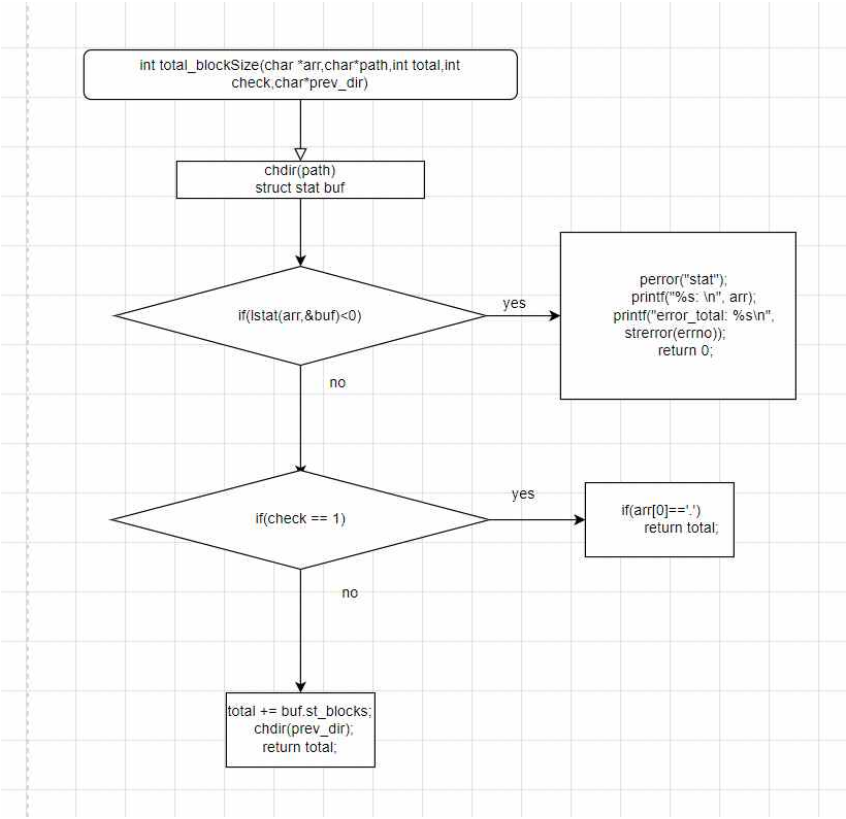
이번 과제는 저번 html 형식은 이어서 하되 클라이언트와 소켓 연결된 이후의 작업을 fork()를 통해 자식 프로세스에서 수행하게 하여 동시에 다수 클라이언트가 접속할 수 있도록 지원하는 발전된 server를 구현하는 것이다. 10초에 한 번씩 연결되었던 클라이언트의 정보를 format에 맞추어 terminal에 출력해준다. 또한, client의 ip 주소값과 port number를 client가 연결되었을 때와 연결이 끊겼을 때 각각 한번 출력해준다. 미리 허용된 ip 주소값에 한하여 client를 받고, match가 되지 않는 ip주소로부터 request가 올 시 오류 메시지를 출력한다.

Ⅱ.flowchart

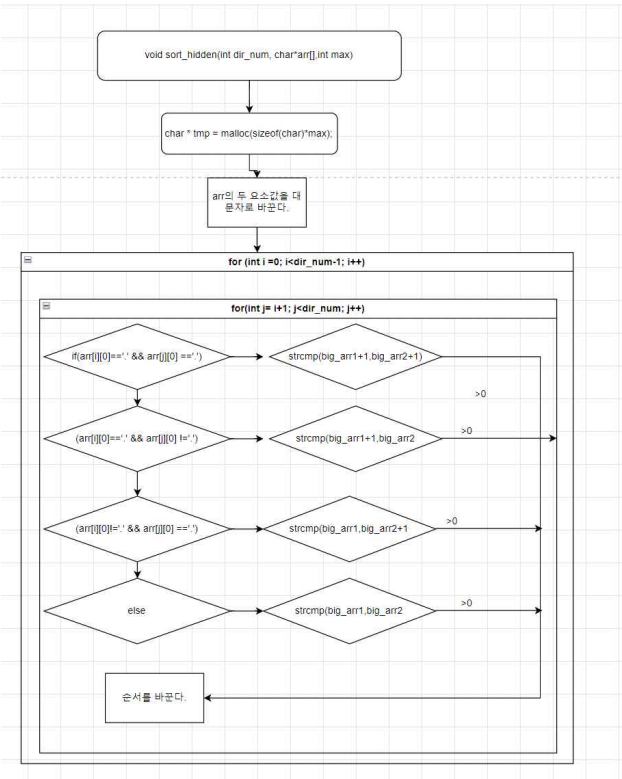
(1) what_color함수



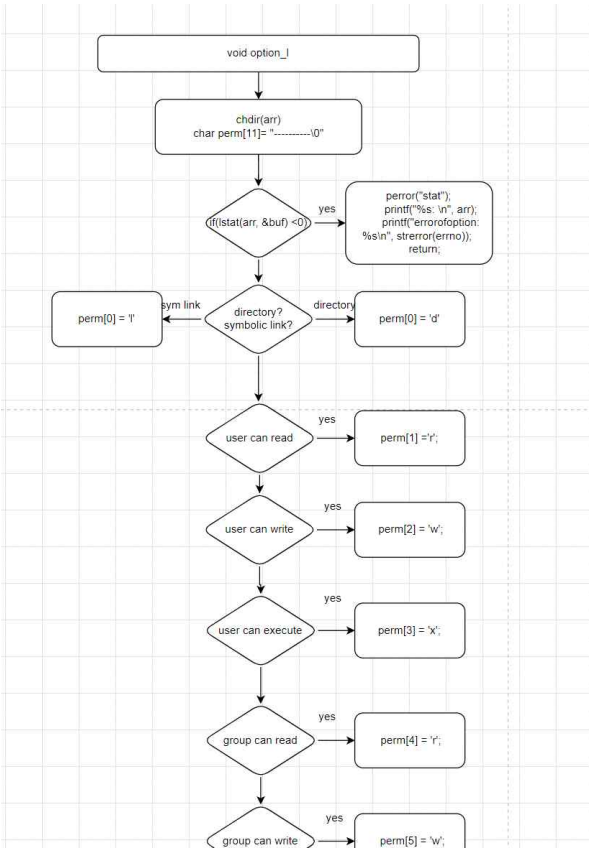
(2)total_blockSize함수

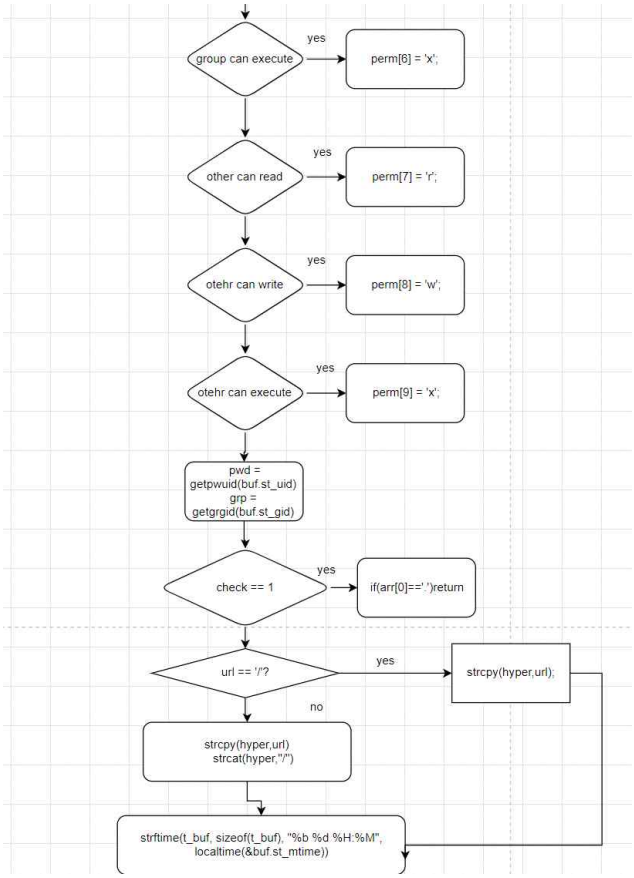


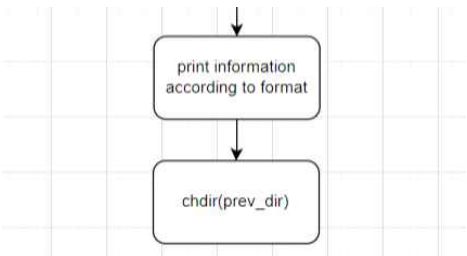
(3) sort_hidden함수



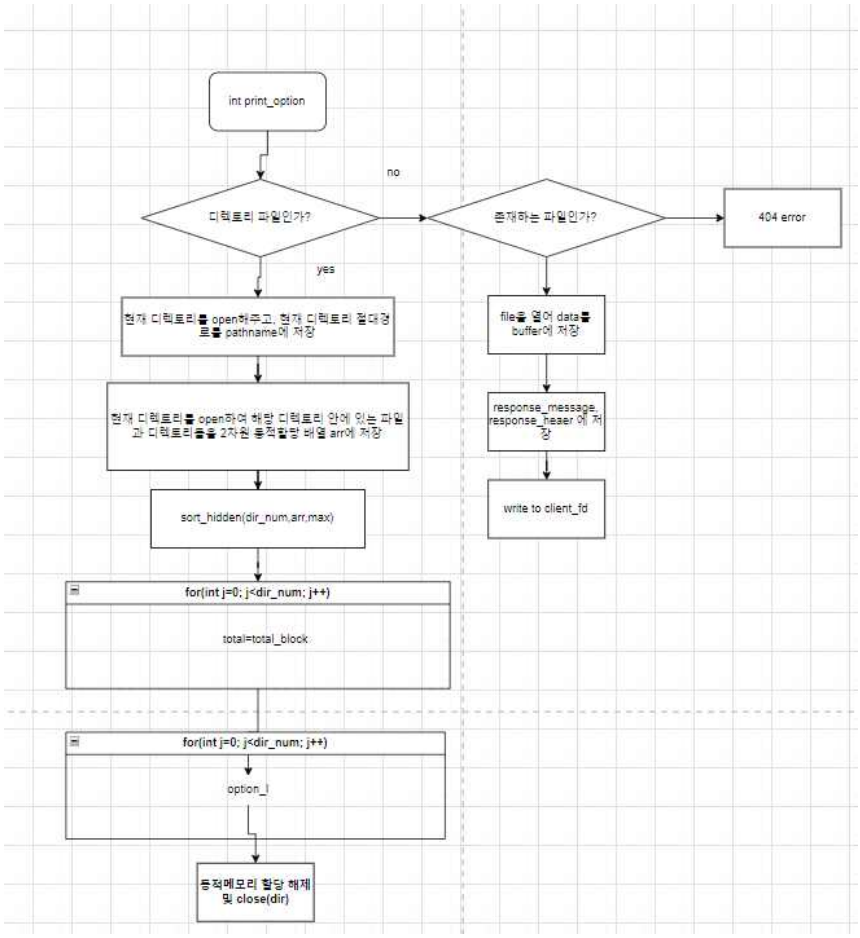
(4) option_l함수


void option_l
chdir(arr)
char perm[11]= "----------\0"
if(lstat(arr, &buf) <0)
yes
perror("stat");
printf("%s: \n", arr);
printf("errorofoption:
%s\n", strerror(errno));
return;
directory?
symbolic link?
sym link
perm[0] = 'l'
directory
perm[0] = 'd'
user can read
yes
perm[1] ='r';
user can write
yes
perm[2] = 'w';
user can execute
yes
perm[3] = 'x';
group can read
yes
perm[4] = 'r';
group can write
yes
perm[5] = 'w';

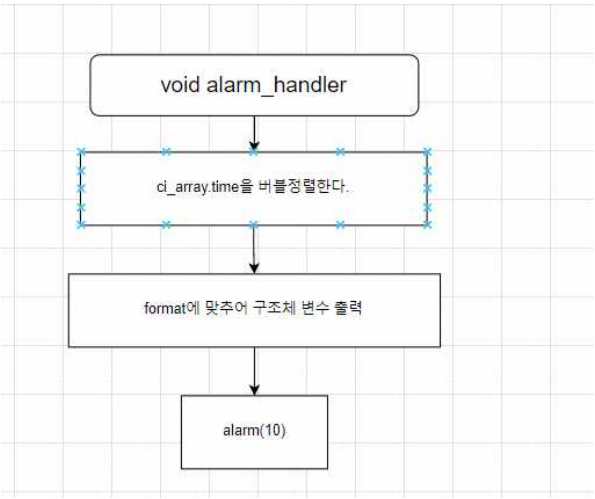




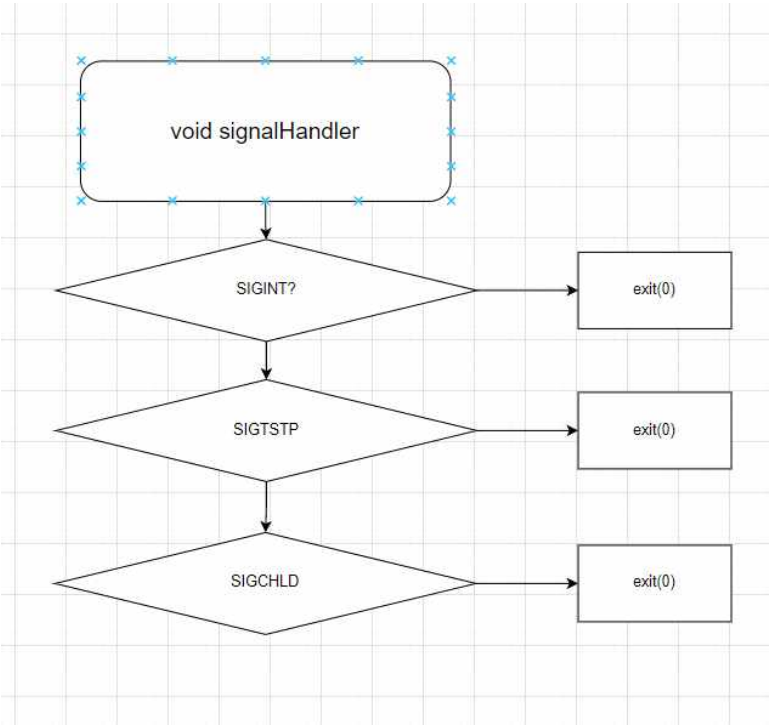
(5)print_option함수


int print_option
no
디렉토리 파일인가?
존재하는 파일인가?
404 error
yes
현재 디렉토리를 open해주고, 현재 디렉토리 절대경로를 pathname에 저장
file을 열어 data를 buffer에 저장
현재 디렉토리를 open하여 해당 디렉토리 안에 있는 파일과 디렉토리들을 2차원 동적할당 배열 arr에 저장
response_message, response_heaer 에 저장
sort_hidden(dir_num,arr,max)
write to client_fd
for(int j=0; j<dir_num; j++)
total=total_block
for(int j=0; j<dir_num; j++)
option_l
동적메모리 할당 해제 및 close(dir)

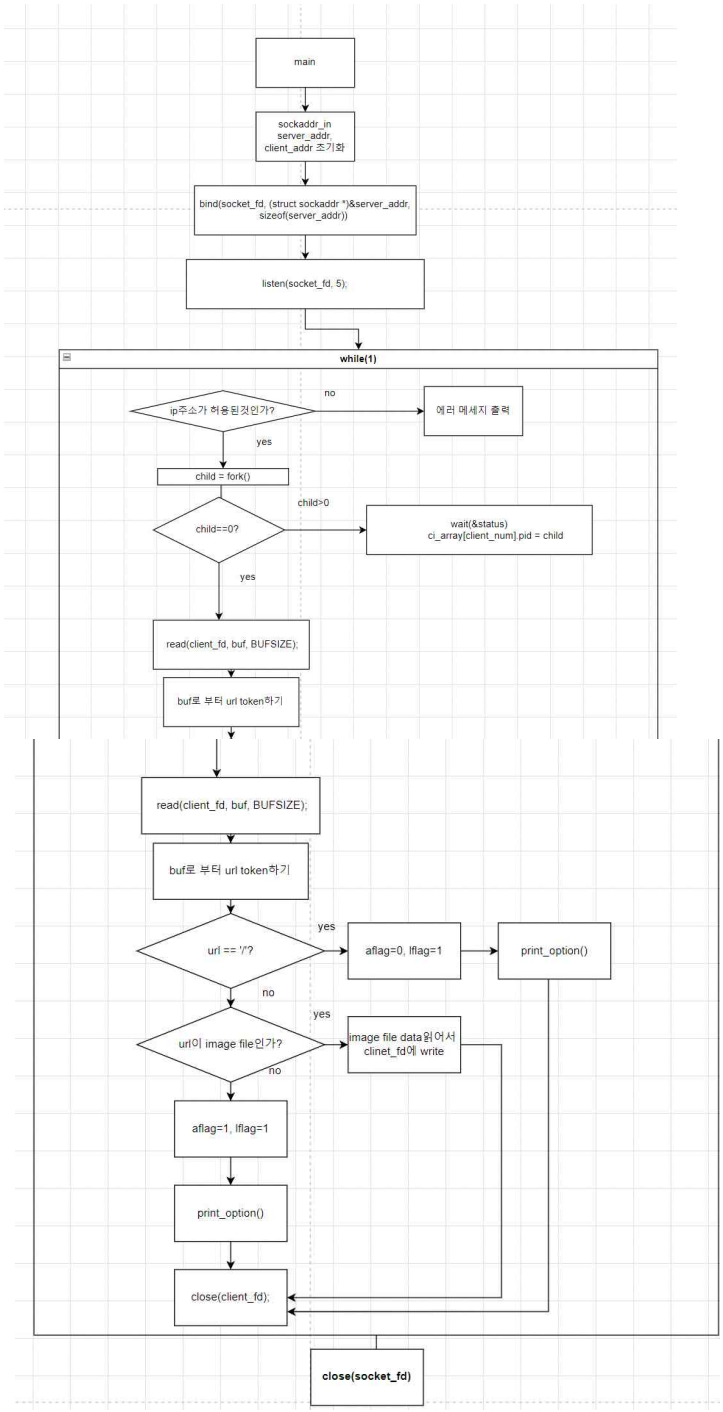
(6) alarm_handler함수



(7) signalHandler 함수



(8) main함수


main
sockaddr_in
server_addr,
client_addr 초기화
bind(socket_fd, (struct sockaddr *)&server_addr,
sizeof(server_addr))
listen(socket_fd, 5);
while(1)
ip주소가 허용된것인가?
no
에러 메세지 출력
yes
child = fork()
child>0
child==0?
wait(&status)
ci_array[client_num].pid = child
yes
read(client_fd, buf, BUFSIZE);
buf로 부터 url token하기
read(client_fd, buf, BUFSIZE);
buf로 부터 url token하기
yes
url == '/'?
aflag=0, lflag=1
print_option()
no
yes
url이 image file인가?
image file data읽어서
clinet_fd에 write
no
aflag=1, lflag=1
print_option()
close(client_fd);
close(socket_fd)

Ⅲ. Pseudo code

int total_blockSize는 디렉토리나 파일의 총 블록 사이즈를 구하는 함수이다.
인자로 받은 path로 우선 이동한다. 그 다음, stat구조체 변수 buf를 선언해준다.
stat 인자로는 인자로 받은 디렉토리명과 buf를 넣어준다.
만약 lstat에 문제가 생길 시, perror와 errno를 통해 오류 메세지를 출력해준다.
함수 인자로 받은 check는 숨김 파일을 고려해야할지 안해야할지를 알려주는 signal로
만약 check == 1이라면,
    숨김파일일 때, 블록 사이즈를 더하지 않고 함수를 끝낸다.
아니라면,
    함수인자로 받은 기존 total를 업데이트 해준다.
이전 경로를 함수인자로 받았으므로 이전경로(prev_dir)로 다시 나가주고, total을 반환하면서 함수를 끝낸다.

void option_l함수는 인자로 받은 파일이나 디렉토리의 정보를 구한 후, 출력해준다.
permission에 대한 정보를 저장하기 위해 perm[11]을 선언 및 맨 처음엔 모두 '-'로 초기화 해준다.
구조체 stat 변수 buf를 정의한다.
user name, group name을 출력하기 위한 passwd, group 구조체 변수 pwd, grp도 각각 선언한다.
인자로 받은 path로 이동한다.
만약 lstat에 대한 에러가 발생할 때, 에러 메세지를 출력해준다.
buf에 저장된 파일이나 디렉토리의 타입을 분석하여 perm[0]에 표시한다.
user, group, other에 대한 접근 권한 총 9가지에 대해 if문을 통해 분석하고,
해당 접근권한이 있을 시 각 자리에 맞게 r,w,x를 써준다.
pwd를 통해 user id를 가져오고, grp를 이용해 group id를 가져온다.
t_buf에 strftime을 통해 time을 localtime으로 바꿔준후, format형으로 저장해준다.
만약 함수인자로 받은 check가 1이라면 파일명이 '.'으로 시작하는 것은 출력을 무시한다.
hyper link를 만들어준다.
만약 인자로 받은 url 마지막이 '/'로 끝난다면, '\0'을 넣는다.
hyper에 '/'를 넣는다.
hyper에 url을 추가한다.
hyper에 arr을 추가한다.
하이퍼링크를 걸어 response_message를 갱신한다.
접근권한,link수, user name, group name,size,수정시간,파일명을 '\t'로 구분하여 출력한 후,
prev_dir로 되돌아간다.

what_color함수
인자로 받은 path로 이동
인자로 받은 arr에 대해 lstat구조체에 넣는다.
만약 arr이 링크파일이라면

color는 green
만약 arr이 directory라면
color는 blue
만약 그외라면
color는 red
인자로 받은 prev로 이동
color를 return

print_option함수
open한 경로가 디렉토리가 아니라면{
 {존재하고 있는 파일이 아니라면
  response_header를 404 error를 출력하게끔 갱신한다.
  response_message를 에러 내용으로 갱신한다.}
 {존재하는 파일이라면
  해당 file을 open한다.
  response_header를 갱신한다.
  해당 file을 읽어 nread에 읽은 byte수를 저장하고 해당 내용을 response_message에 저장한다.
  response_message를 response_header에 붙여 쓴다.
  client_fd에 response_header(+response_message)를 쓴다.
}
}
open한 디렉토리에서 해당 디렉토리 안에 있는 파일과 디렉토리들을 2차원 동적할당 배열 arr에 저장한다.
디렉토리내의 entry 수들을 dir_num으로 정의하고, entry name중 가장 긴 이름을 max로 정의한다.
sort_hidden함수를 call한다.
그 다음 정렬된 entry들을 각각 total_blockSize()에 넣는다.
갱신된 total 변수를 response_message에 넣어 갱신한다.
반복문을 통해 모든 entry들을 option_l함수에 넣는다.

void sort_hidden함수는 '.'으로 시작되는 파일도 포함하여 파일명을 정렬해주는 함수이다.
bubble sorting을 써줄 것이기 때문에 임시 저장해줄 array tmp를 동적할당 해준다.
해당 array는
array[i]와 array[j]를 모두 대문자로 바꿔준후 각각 big_arr1, big_arr2에 저장한다.
big_arr1[i]와 big_arr2[j]를 비교할 때
숨김 파일과 숨김 파일일 때,
숨김 파일과 숨김 파일이 아닐때,
숨김파일이 아니고 숨김파일일때,
둘다 숨김 파일이 아닐때
총 4개의 경우의 수를 통해 먼저 '.'을 빼고 비교해준 뒤, 다시 '.'을 포함하여 정렬해준다.

만약 숨김 파일이라면 arr[i][0] == '.'이므로 arr[i]+1을 해주면 '.'이후의 문자열을 가리키는 것이다.
만약 arr[0]와 arr[1]의 위치가 바뀌어야 한다면, tmp에 arrp[0]값을 저장
arr[0]에 arr[1]값을 복사하고
arr[1]에는 tmp값을 복사하여 저장한다.

main함수
sockaddr_in 구조체 변수인 server_addr, client_addr를 초기화 해준다.
PF_INET,SOCK_STREAM type을 가진 socket을 만들어 해당 descriptor을 socket_fd에 저장한다.
server_addr과 해당 socket을 bind한다.
queue사이즈를 5로 하고, listen을 보낸다.
10초에 한번씩 alarm signal을 보낸다.
while(1)
{
socket_fd로부터 accept한 것을 client_fd에 저장한다.
client_fd로 부터 읽은 내용을 buf에 저장한다.
client_num번째 c_info구조체 배열에 client port number을 저장한다.
client_num번째 c_info구조체 배열에 접속시간을 저장한다.
client_num번째 c_info구조체 배열에 client가 몇번째인지 저장한다.

"accessible.usr"파일을 읽기용으로 연다.
파일을 한 줄씩 읽으며 마지막까지 읽는동안{
is_match =1
만약, 파일에 저장된 ip주소값과 client ip 주소값이 일치하지않다면
is_match =0
}
만약 is_match=0이라면, client에 에러 메시지 출력
파일을 닫는다.
fork()를 통해 child process를 만든다.
만약 자식 프로세스라면{
buf에 저장된 내용을 method와 url부분으로 token한다.
aflag =0
lflag=0
으로 초기화한다.

만약, url이 '/'이라면,
lflag=1로 update한다.
현재 디렉토리를 link_path에 저장한다.
response_header(type : text/html)와 response_message를 update한다.
print_option함수를 call한다.

만약 url이 이미지파일(확장자가 .jpg, .png, .jpeg)라면,
response_header는 type: image/*로 update한 후, client_fd에 send한다.
url을 file open한다.
파일 끝까지 읽을 때까지,
사진 데이터를 읽어 buffer에 저장한다.
이 값을 client_fd에 send한다.
파일을 닫는다.

그외라면,
aflag=1, lflag=1로 update한다.
현재 디렉토리를 link_path, prev에 copy한다.
link_path에 url 부분을 추가한다.
response_header는 type : text/html로 갱신한다.
response_message를 갱신한다.
print_option함수를 call한다.
client_fd를 닫는다.
}
만약 부모프로세스라면{
자식 프로세스를 기다린다.
client_num번째 구조체 배열의 pid를 해당 return값으로 갱신한다.
만약 client_num이 10보다 크다면 읽을 index 시작점을 +1한다.}
socket_fd를 닫는다.

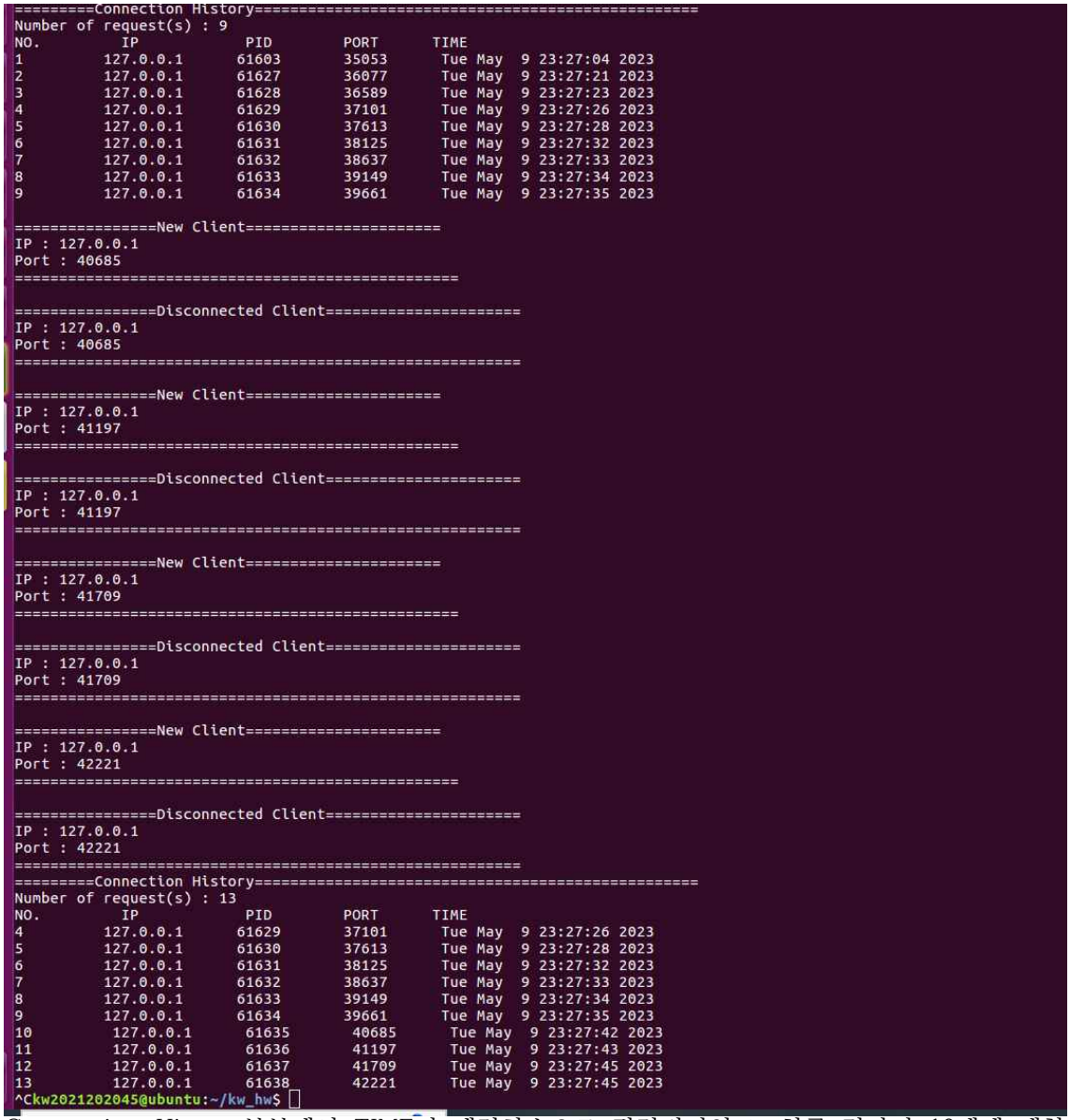
alarm_hander함수
구조체 배열 ci_array의 time부분을 오름차순으로 버블정렬한다.
start부터 client_num이하까지{
ci_array의 time을 ctime()을 이용하여 char형으로 바꿔준다.
구조체 정보들을 출력한다.
}
10초에 한번씩 알람을 준다.

signalHandler함수
인자로 받은 signal이 SIGINT라면, exit(0)
인자로 받은 signal이 SIGTSTP라면, exit(0)
인자로 받은 signal이 SIGCHLD라면, exit(0)

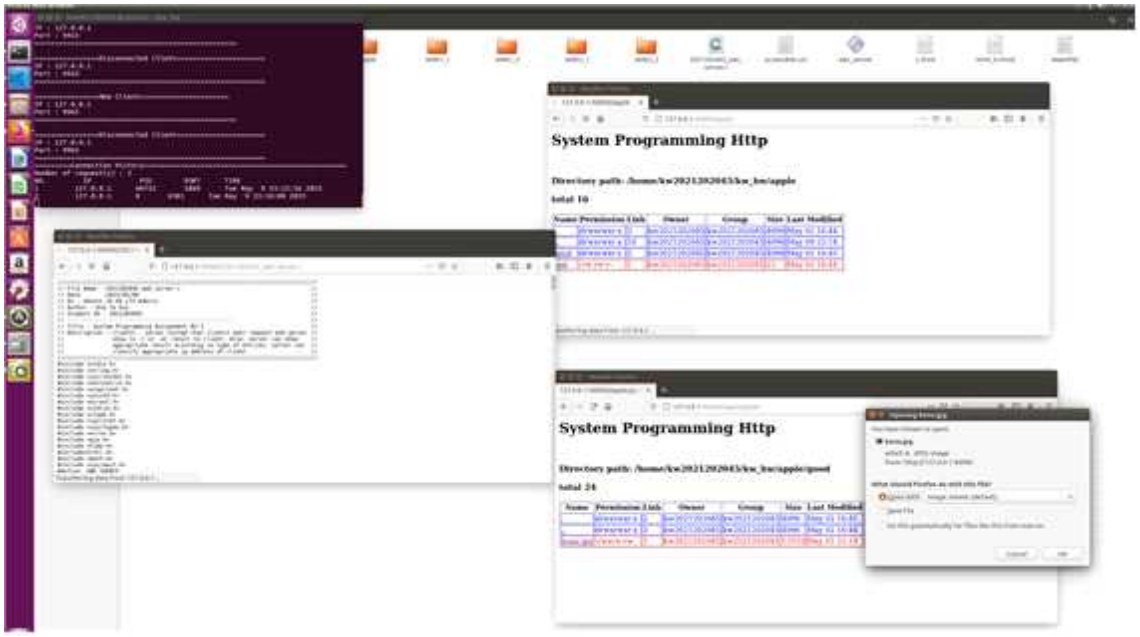
Ⅳ.결과화면

“accessbile.usr”파일에 안적힌 ip주소로 접근하면 위의 페이지가 나오면서 오류 메시지가 나온다.

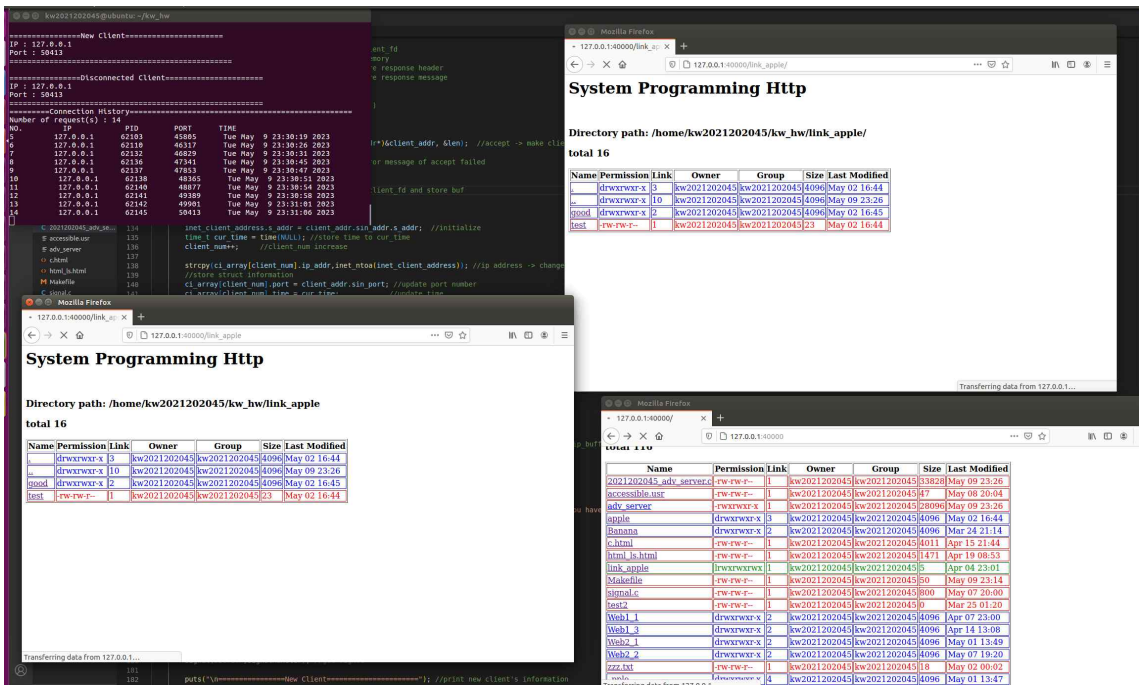
위의 파일에 127.*.*.1을 추가한 후 실행한 결과들은 아래와 같다.

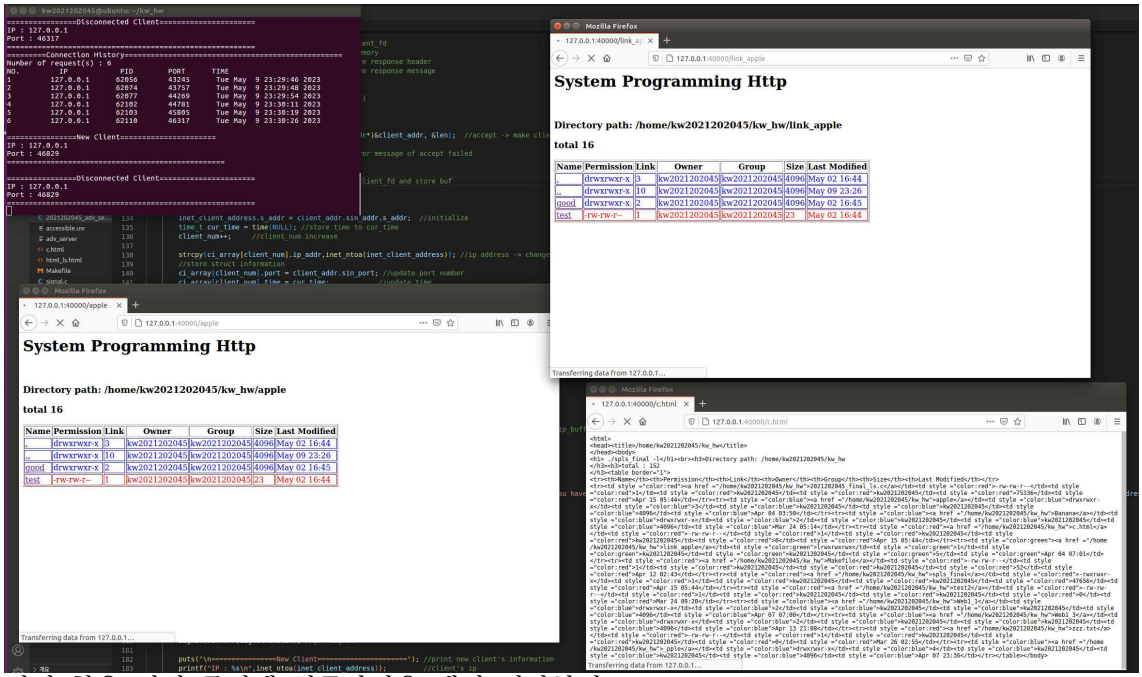


Connection History부분에서 TIME이 내림차순으로 정렬되어있고, 최근 것까지 10개에 대한 것만 출력해주고 있다. New Client에 대해 Disconnected메세지를 잘 보내준다.


System Programming Http
System Programming Http


bono.jpg
정말 대단해


System Programming Http
Directory path: /home/kw2021202045/kw_hw/link_apple/
total 16
System Programming Http
Directory path: /home/kw2021202045/kw_hw/link_apple
total 16



여러 창을 열어 동시에 접근하였을 때의 결과화면

V.고찰

alarm 및 signal 처리에 대한 함수는 signal이 있을 때 해당 정의된 함수로 가서 처리하는 것이다. 일반 함수와 달리 코드 순서에 제약을 받지 않는다는 점이 이해하는 데에 새로웠고, 어색했다. fork()를 통해 child process에서 여러 개의 client를 처리하는 부분은 크게 어렵지 않았으나 “accessbile.usr”파일에서 미리 허용한 ip주소값을 읽을 때 fgets()로 자동으로 한 줄씩 읽는데, 이 때 마지막 줄 제외 모든 줄의 마지막 요소인 ‘\n’이 같이 읽혀서 ip주소 fnmatch가 잘 되지 않아 이 부분을 ‘\n’을 만나면 ‘\0’으로 처리할 수 있도록 해주었다. 10초에 한 번씩 client의 information(ip,pid,port,access time 등)을 보여주기 위해 구조체를 도입하였고, 1번지부터 저장을 시작하였다. 구조체 배열에 몇 개가 저장되어 있든지 최신 10개까지 보여주기 위해 구조체 배열에 10개 넘게 저장되었다면 for문의 start를 하나씩 밀어주었다. 또한, hyper-link를 클릭하지 않고 마우스 커서를 위에 대기만 해도 new client를 보내주어서 client_fd로부터 내용을 read했는데, buf에 내용이 없다면 무시하도록 해주었다.