시스템프로그래밍
assignment 3-1

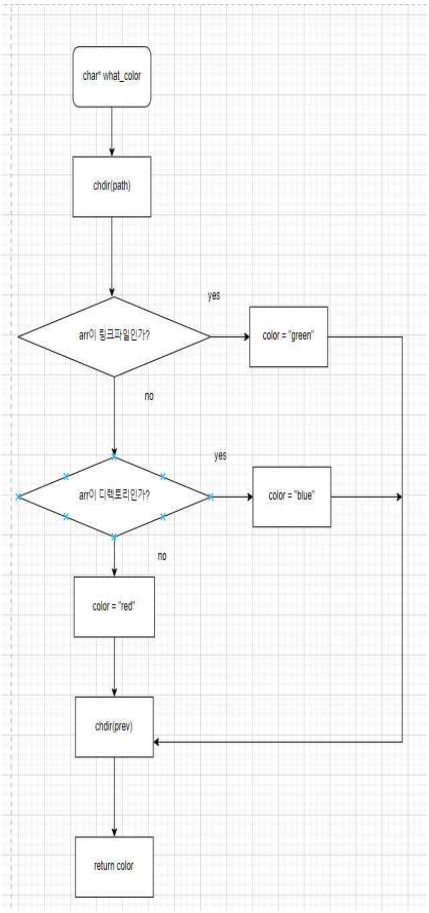
학번: 2021202045
이름: 김예은
담당 교수님: 최상호 교수님

Ⅰ. Introduction

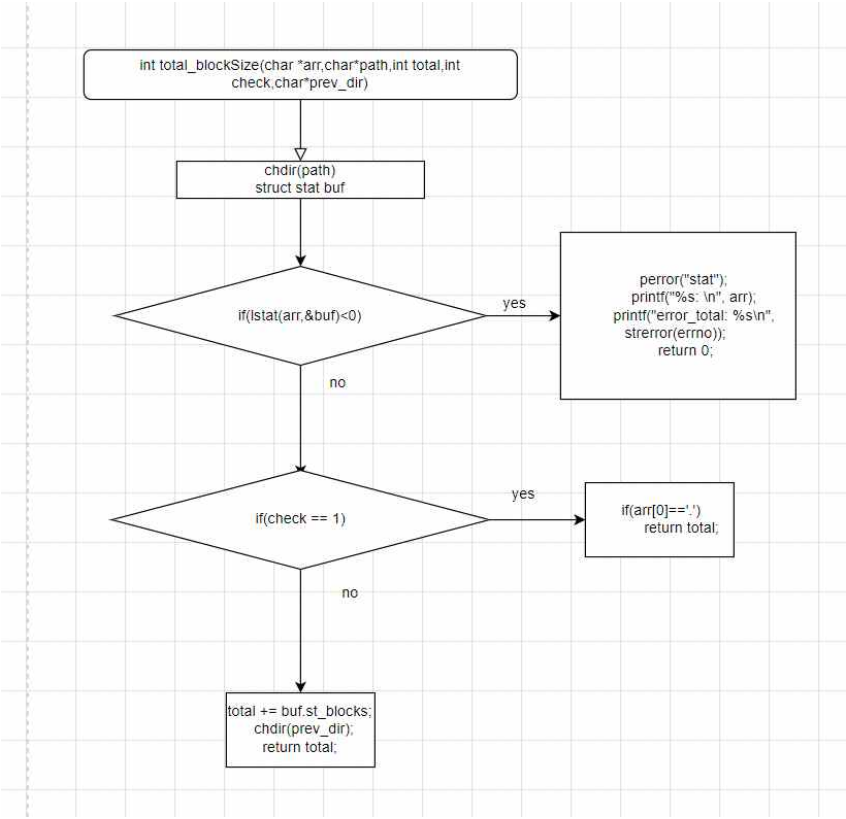
이번 과제는 저번 과제와 비슷하지만 저번에는 client를 accept한 뒤에 fork를 한 것과 달리 미리 5개의 child process를 fork한 뒤, 그 프로세스에서 client와 연결시켜준다. 10초마다 부모프로세서에서 history 목차를 출력해주고, 해당 목차에 맞게 구조체 배열을 출력해주는 일은 자식 프로세스에서 해준다. 각 자식 프로세스는 최대 10개의 history를 가지며, 각자 독립적인 client number를 가진다. 종료 시 종료시그널을 누르면 부모 프로세스가 자식 프로세스에 시그널을 보내 자식 프로세스들 먼저 종료시킨 뒤, 부모 프로세스를 종료한다. 프로세스 생성 및 종료때마다 알맞은 문구를 터미널에 출력해준다. 여기서 history 내용 출력 관한 동기화 문제는 고려하지 않는다.

Ⅱ.flowchart

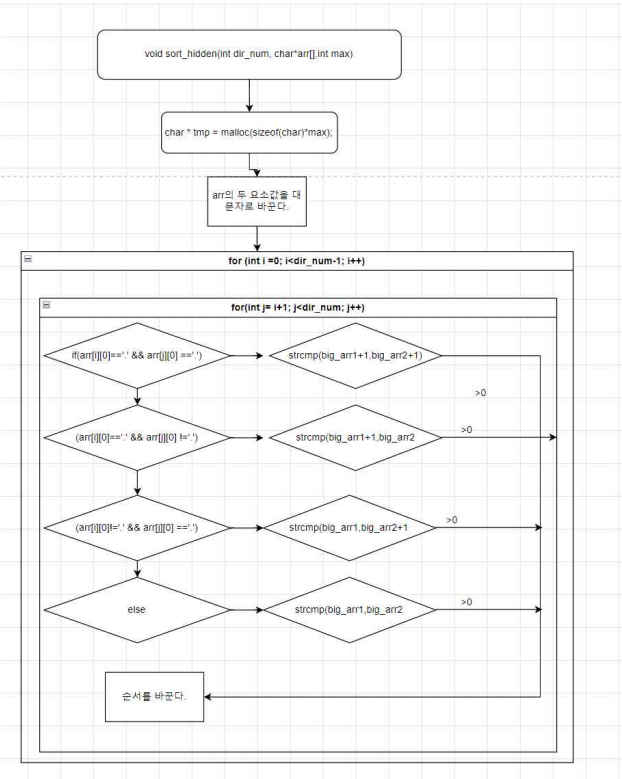
(1) what_color함수



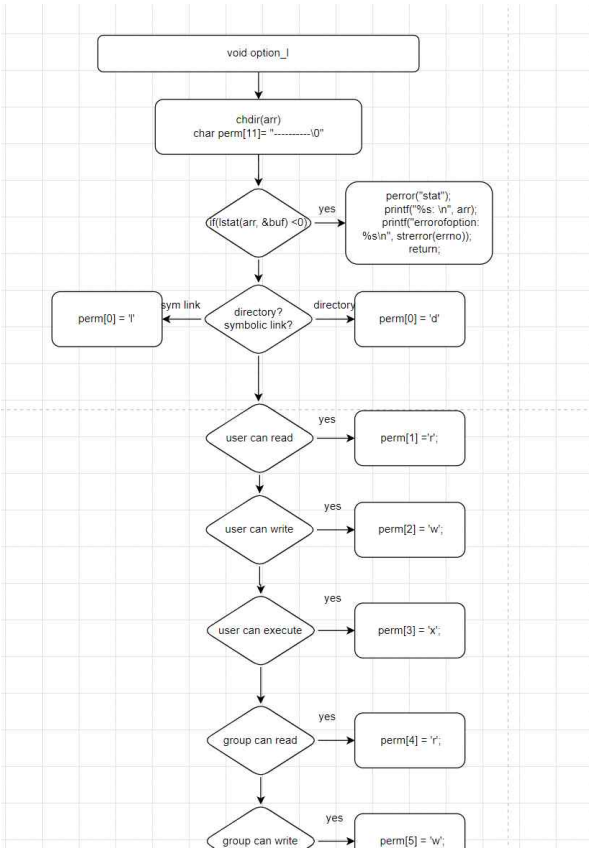
(2) total_blockSize함수

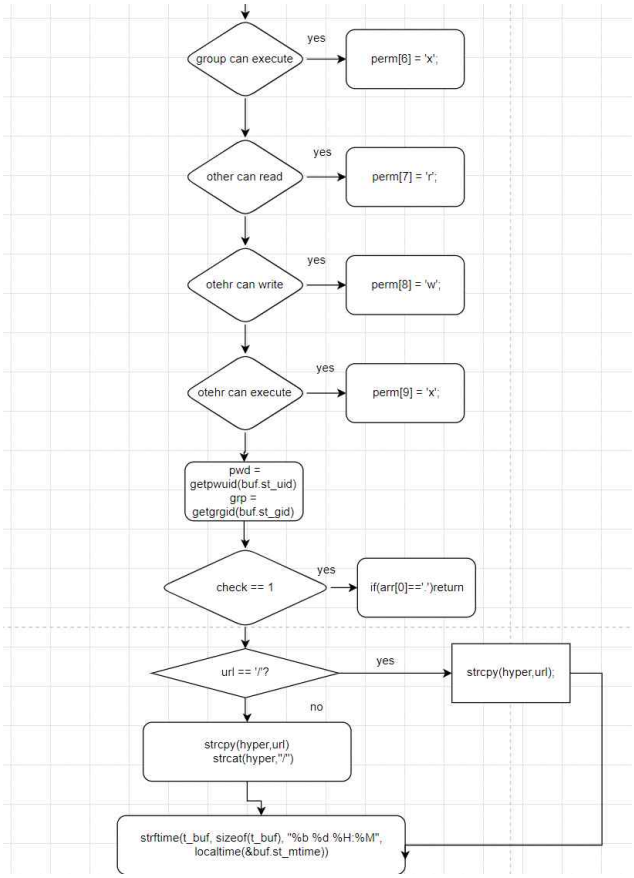


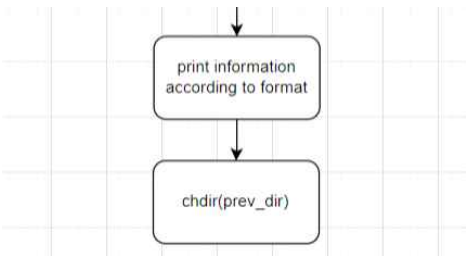
(3) sort_hidden함수



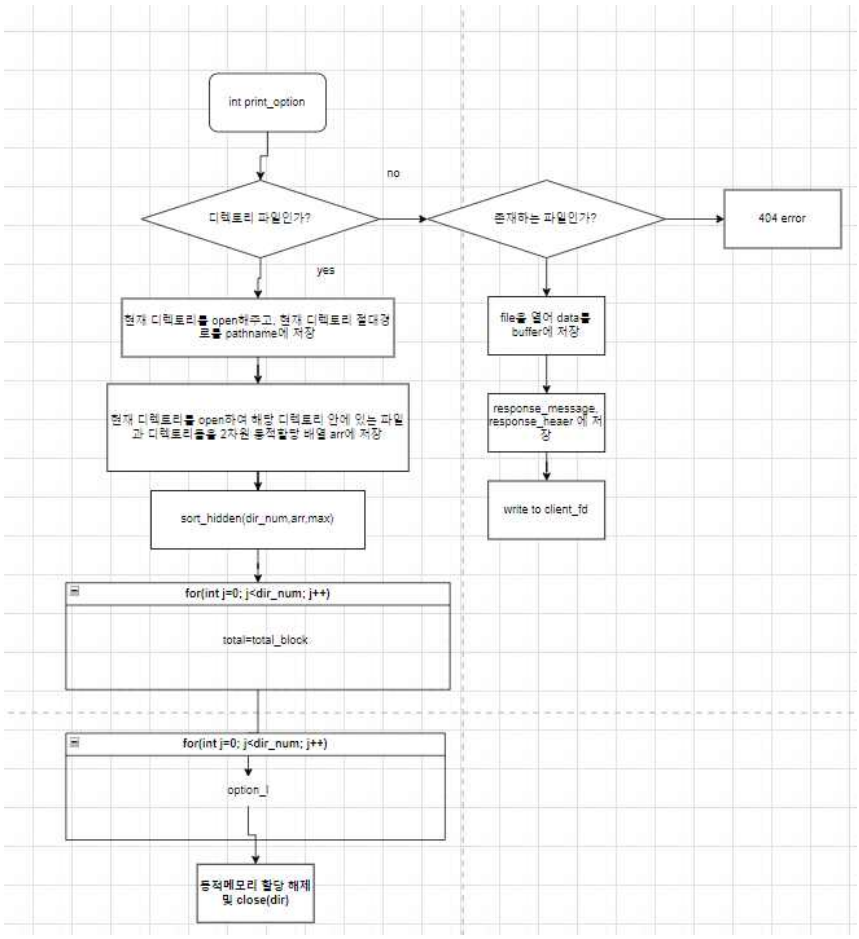
(4) option_l함수


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';

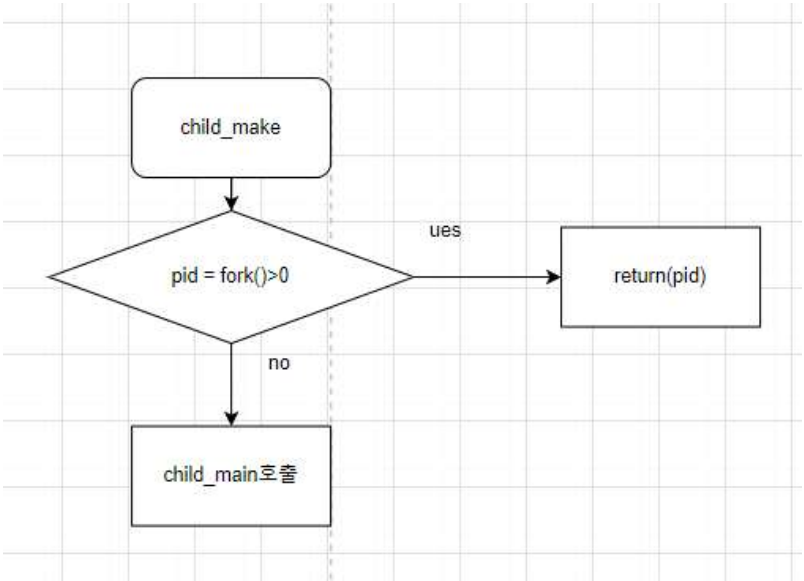

group can execute
yes
perm[6] = 'x';
other can read
yes
perm[7] = 'r';
otehr can write
yes
perm[8] = 'w';
otehr can execute
yes
perm[9] = 'x';
pwd = getpwuid(buf.st_uid)
grp = getgrgid(buf.st_gid)
check == 1
yes
if(arr[0]==' ')return
url == '/'?
yes
strcpy(hyper,url);
no
strcpy(hyper,url)
strcat(hyper,"/")
strftime(t_buf, sizeof(t_buf), "%b %d %H:%M", localtime(&buf.st_mtime))


print information according to format
chdir(prev_dir)

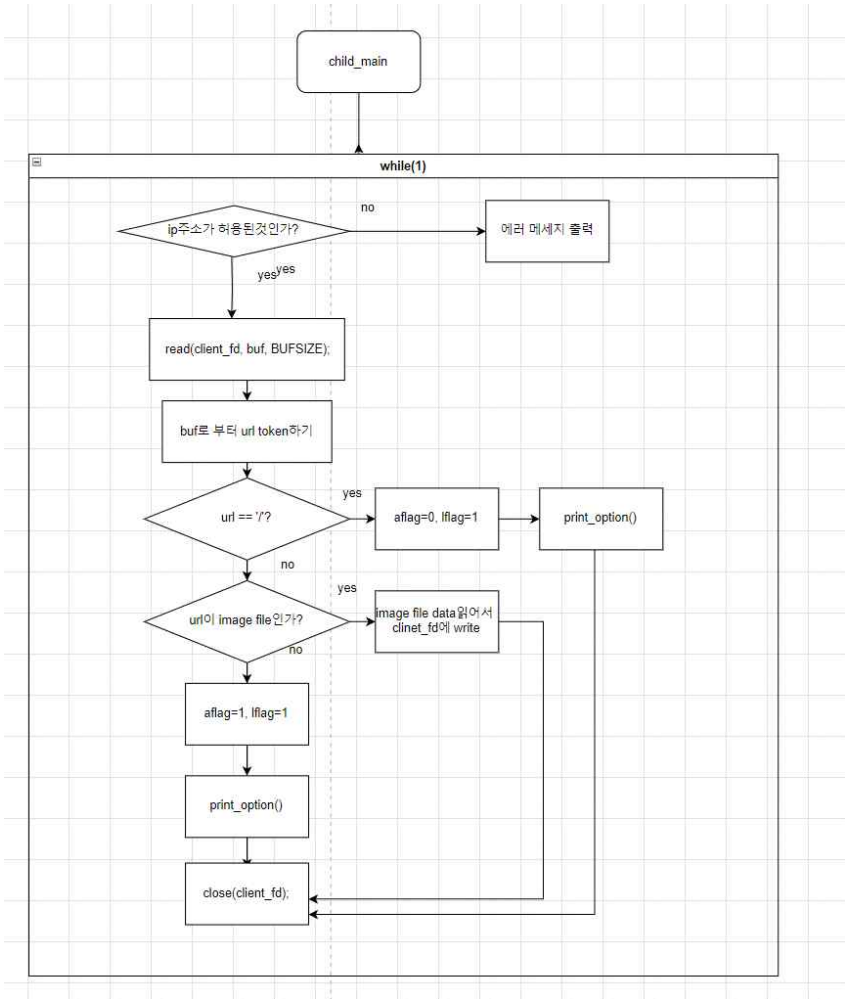
(5)print_option함수



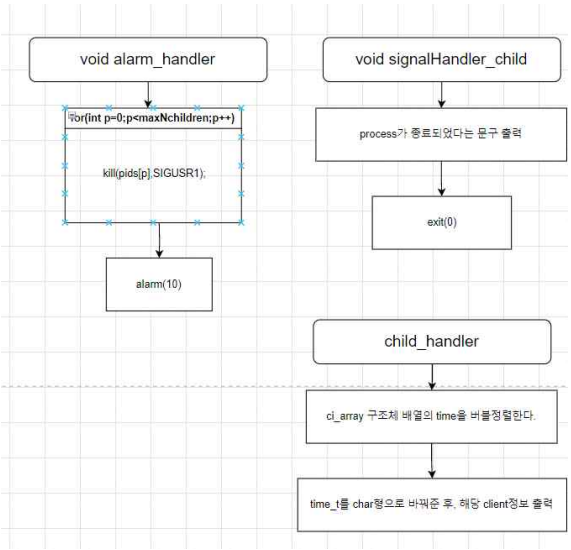
(6)child_make함수



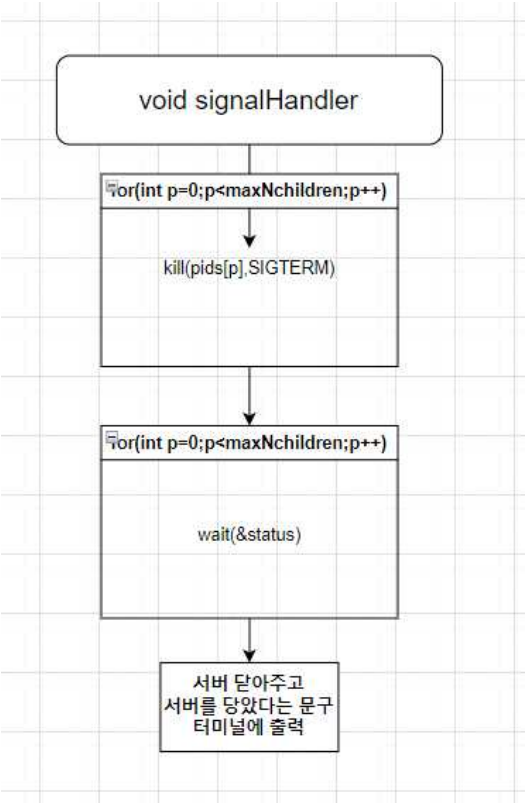
(7)child_main함수



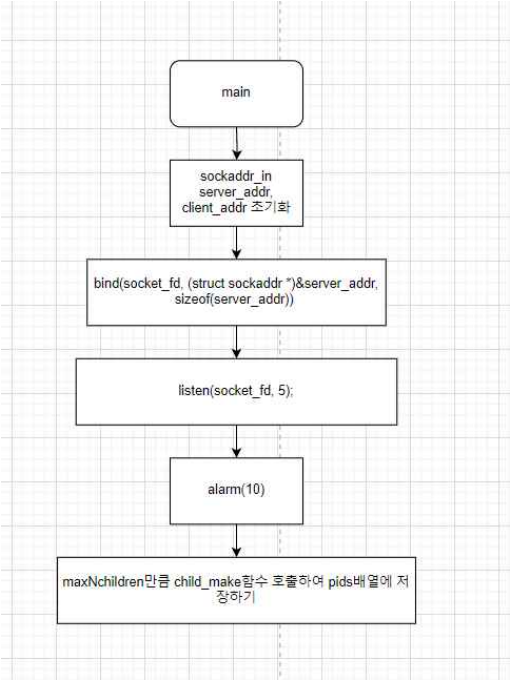
(8) alarm_handler, signalHandler_child, child_handler 함수



(9) signalHandler함수



(10) main함수



Ⅲ. Pseudo code

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.

signalHandler함수
만약 인자로 받은 sig가 SIGINT라면
p는 0부터 maxNchildren까지
pids[p]에 SIGTERM 신호를 준다.
p는 0부터 maxNchildren까지
부모프로세스는 자식프로세스를 wait한다.
server_end_time에 현재 시간을 time_t형으로 저장한다.
ctime을 이용해 char형으로 변환한다.
Server is terminated라는 문구를 출력한다.
부모프로세스를 종료한다.

signalHandler_child함수
만약 인자로 받은 sig가 SIGTERM이라면
child_end_time에 현재 시간을 time_t형으로 저장한다.
ctime을 이용해 char형으로 변환한다.
종료하는 자식 프로세스의 pid와 함께 종료 문구를 출력한다.
자식프로세스를 종료한다.

main함수
sockaddr_in 구조체 변수인 server_addr, client_addr를 초기화 해준다.
PF_INET,SOCK_STREAM type을 가진 socket을 만들어 해당 descriptor을 전역변수인 socket_fd에 저장한다.
server_addr과 해당 socket을 bind한다.
queue사이즈를 5로 하고, listen을 보낸다.
10초에 한번씩 alarm signal을 보낸다.
p는 0부터 maxNchildren까지
child_make함수를 호출한다.
함수의 return값을 pids[p]에 저장한다.
signal을 받을 때까지 무한루프를 돈다.

child_make함수
fork를 통해 자식프로세스를 만든다.
만약 부모프로세스라면
자식프로세스의 pid를 return해준다.
자식 프로세스라면
child_main함수를 호출한다.

child_main함수
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.

buf에 저장된 내용을 method와 url부분으로 token한다.
aflag =0
lflag=0

으로 초기화한다.

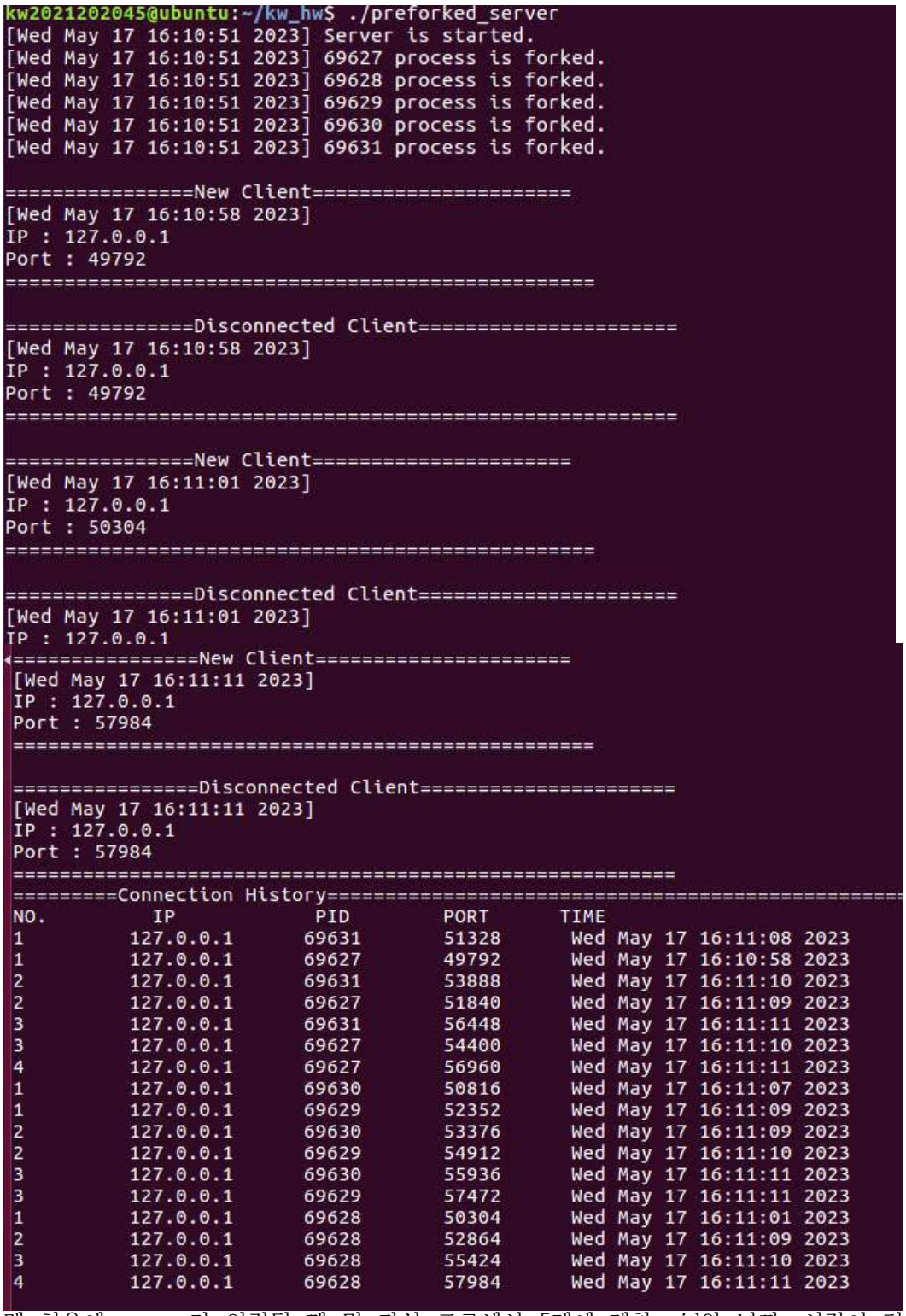
만약, url이 '/'이라면,
lflag=1로 update한다.
현재 디렉토리를 link_path에 저장한다.
response_header(type : text/html)와 response_message를 update한다.
print_option함수를 call한다.

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.

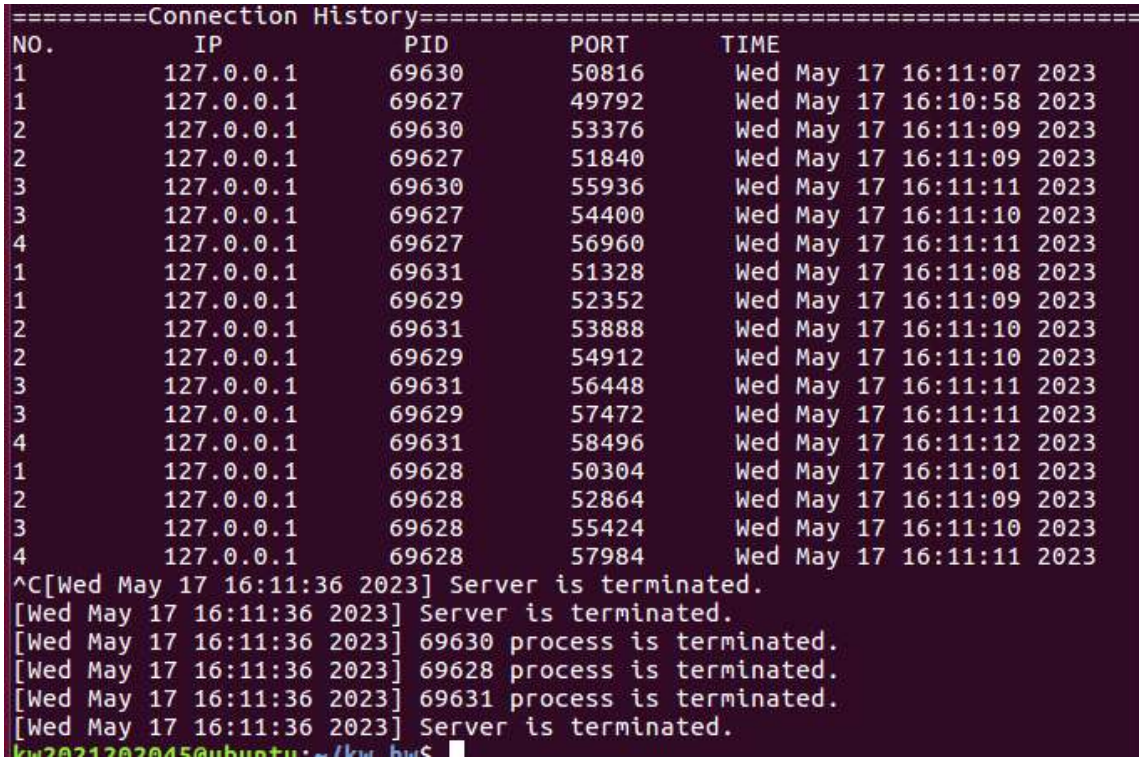
그외라면,
aflag=1, lflag=1로 update한다.
현재 디렉토리를 link_path, prev에 copy한다.
link_path에 url 부분을 추가한다.
response_header는 type : text/html로 갱신한다.
response_message를 갱신한다.
print_option함수를 call한다.
client_fd를 닫는다.

alarm_hander함수
만약 client_num이 10보다 크다면 읽을 index 시작점을 +1한다.}
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.

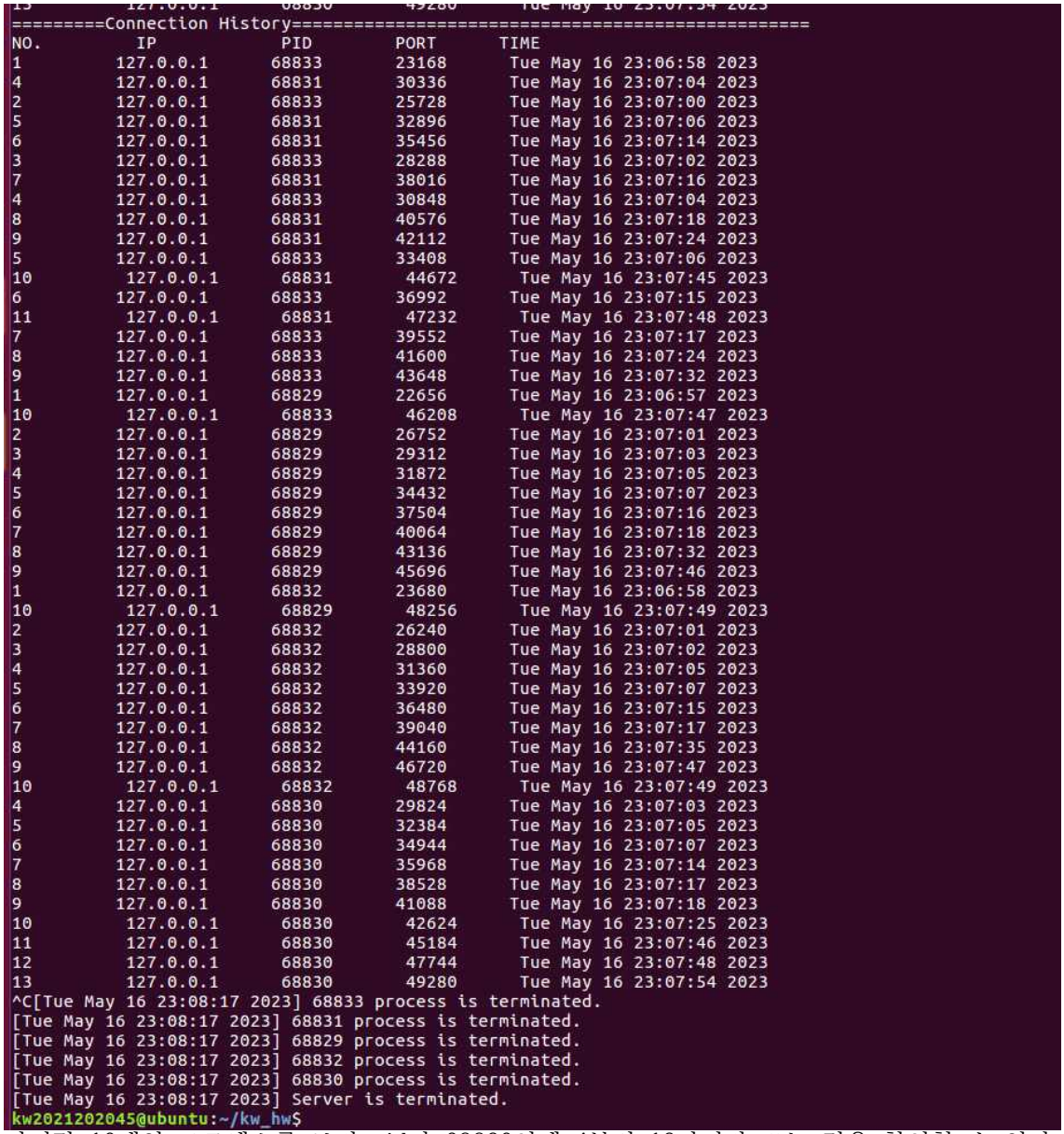
Ⅳ.결과화면



맨 처음에 server가 연결될 때 및 자식 프로세서 5개에 대한 pid와 날짜, 시간이 터미널에 출력된다. server가 열린상태로 접속을 하면 위와 같이 New Client와 Disconnected Client가 출력되고, 각 프로세스마다 독립적인 number가 history에 10초에 한번씩 출력된다.



ctrl+c를 통해 종료를 하면 SIGINT가 부모프로세스로 전달되면서 자식프로세스가 다섯 개 다 종료된 후, Server가 닫힌다는 문구(날짜및시간)이 터미널에 잘 출력되고 있다. pid를 보면 맨처음에 생성된 프로세스에 대응하는 pid의 process가 닫힌다.



마지막 10개의 프로세스를 보면 pid가 68830인데 4부터 13까지만 뜨는 것을 확인할 수 있다. 즉 가장 최근의 최대 10개만 출력해주는 것을 확인할 수 있다.

Ⅴ.고찰

signal에 대한 이해를 확실히 하지 못해 ctrl+c를 누르면 부모프로세스에서 먼저 자식 프로세스로 종료 시그널을 준 후, 그걸 기다렸다가 부모프로세스를 종료시켜야하는 코드가 아니라 ctrl+c를 누르면 바로 자식프로세스들이 종료되도록 구현했었다. 또한 kill함수 뒤에 바로 wait함수를 넣어 for문을 돌리니 process가 종료되는 것은 맞는데 문구가 계속 server가 terminated된다는 문구가 출력되어 wait함수를 for문 밖으로 빼서 다시 wait해주었더니 해결이 되었다. wait함수는 모든 자식 프로세스가 끝나기를 기다려서 blocking을 하기 때문에 다음 순서의 코드를 먼저 실행해서 그러는 것 같다. kill함수를 통해 시그널을 부모프로세스와 자식프로세스 간에 주고받는 것이 신기했다. 또한, 맨처음에는 5개의 자식프로세스이므로 구조체배열이나 client_num을 5개로 만들어줘야하나 생각했었는데 이론 시간에 각 자식 프로세

스는 copy on writing방식으로 메모리에 접근한다는 사실이 떠올라서 알아서 구조체배열을 독립적으로 가지지않을까하고 실험적으로 구조체 배열을 하나만 썼는데 정말 알아서 잘 독립적으로 접근하여서 신기하고, 코드의 복잡성이 줄어서 편리했다.