시스템 프로그래밍

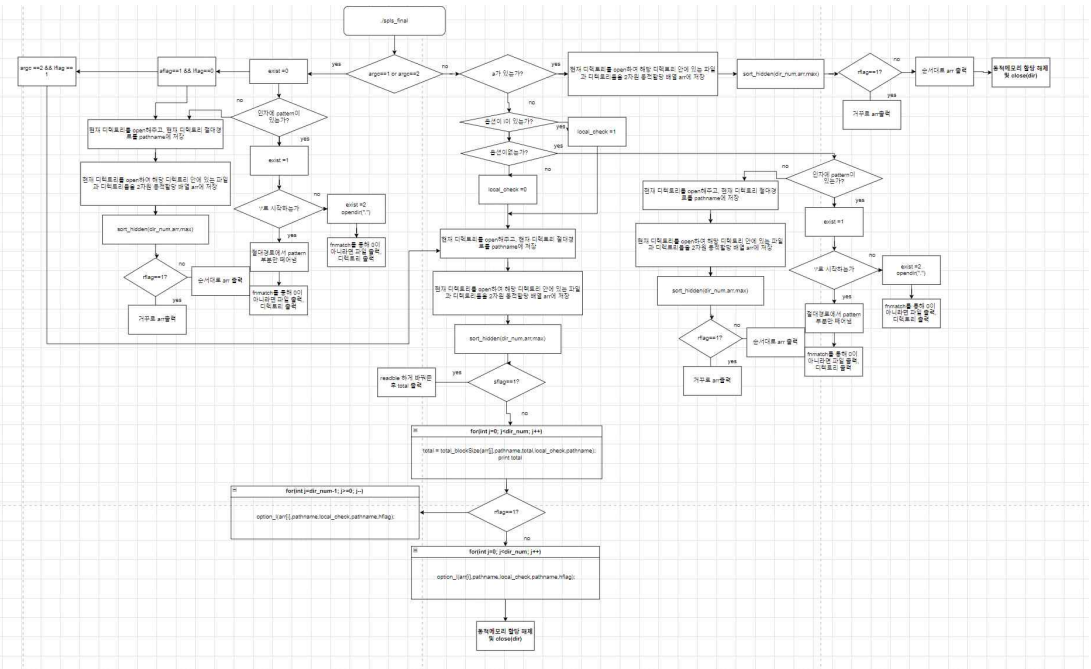
이름: 김예은
학번: 2021202045
담당 교수님: 최상호교수님

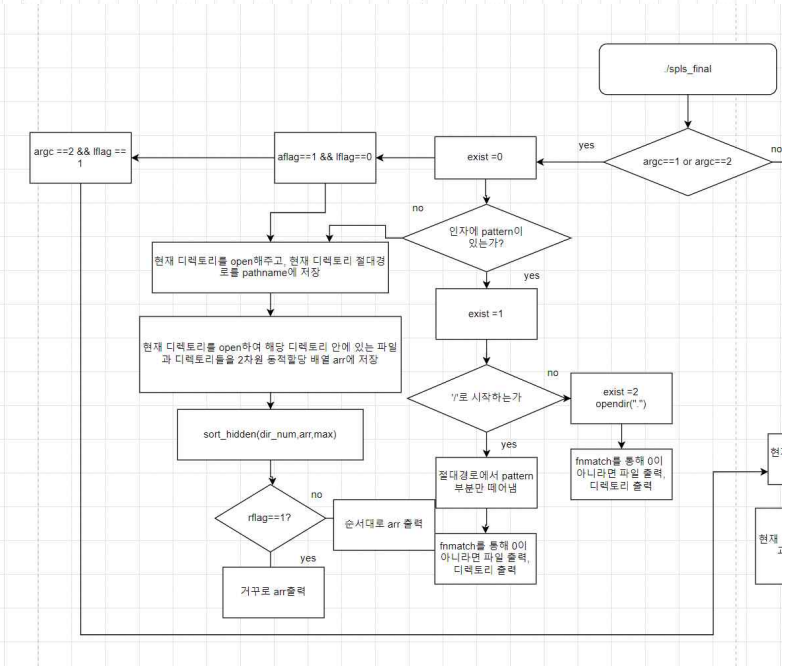
Ⅰ. Introduction

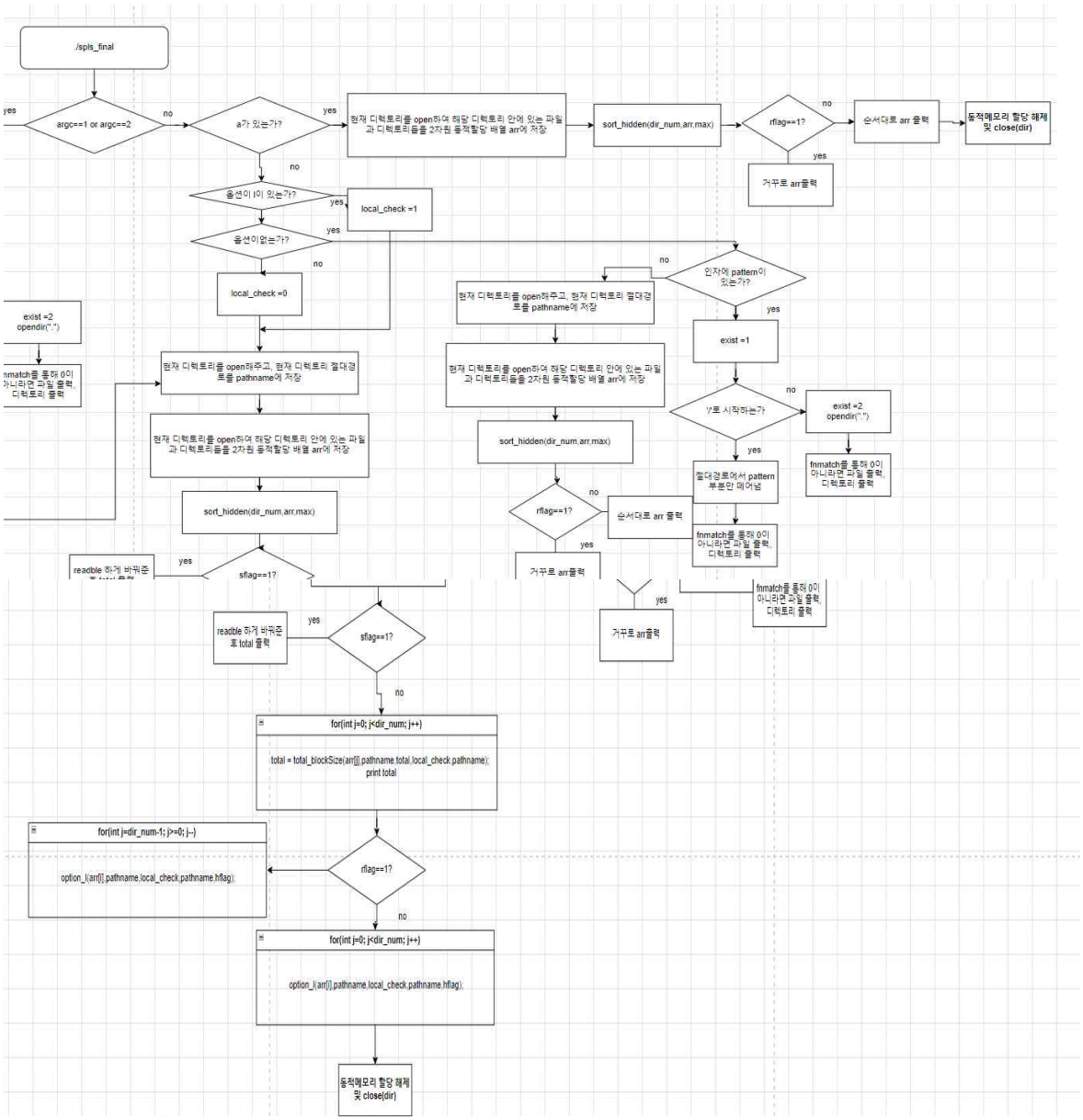
1-3과제는 1-2과제에서 구현한 ls(./spls_advanced)에서 옵션 h,r,S를 추가 구현하는 것이다. -h의 경우 파일 사이즈 및 total 블록 수를 readable하게 바꾸어 출력해주는 것이고, -r 옵션의 경우 문자열이 ascii code로 역순으로 출력되는 것이고, S의 경우 파일 사이즈 크기 순서대로 entry들이 출력되는 것이다. 이 외에, fnmatch()함수를 사용하여 wild card가 있을 때 matching이 되는 entry들을 출력하게끔 구현하는 것이 이번 1-3과제의 목적이다.

Ⅱ. Flowchart

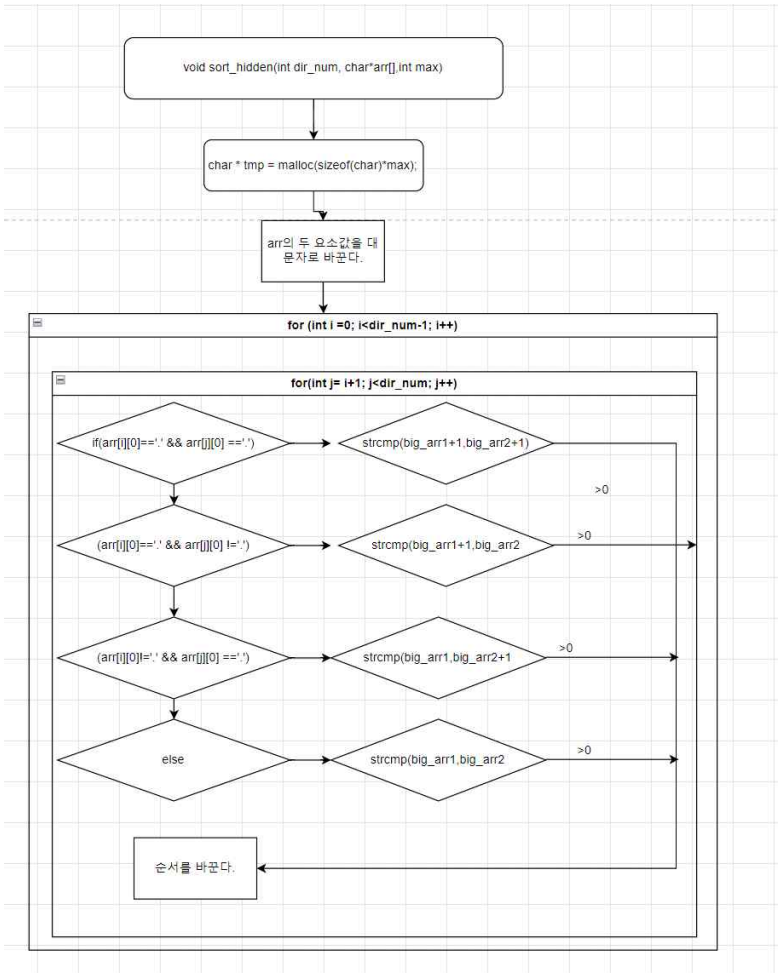
(1) main함수에 대한 flow chart이다. 잘 안보여서 전체사진, 3개로 분리한 사진을 첨부한다.



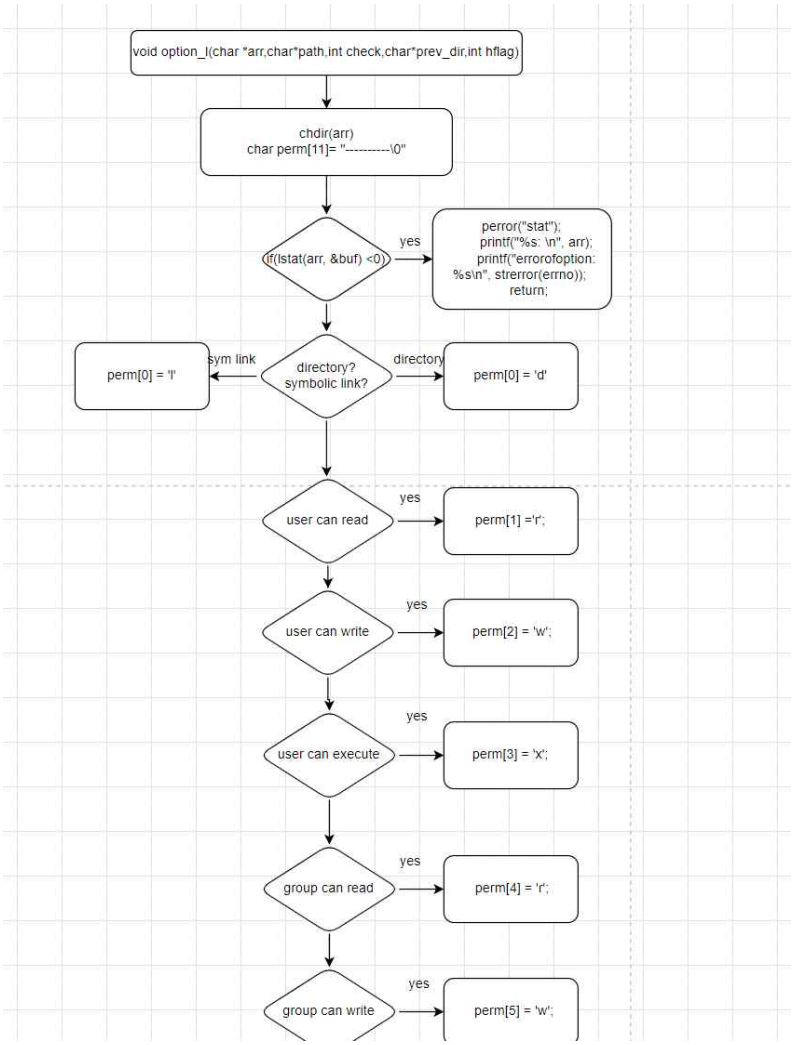


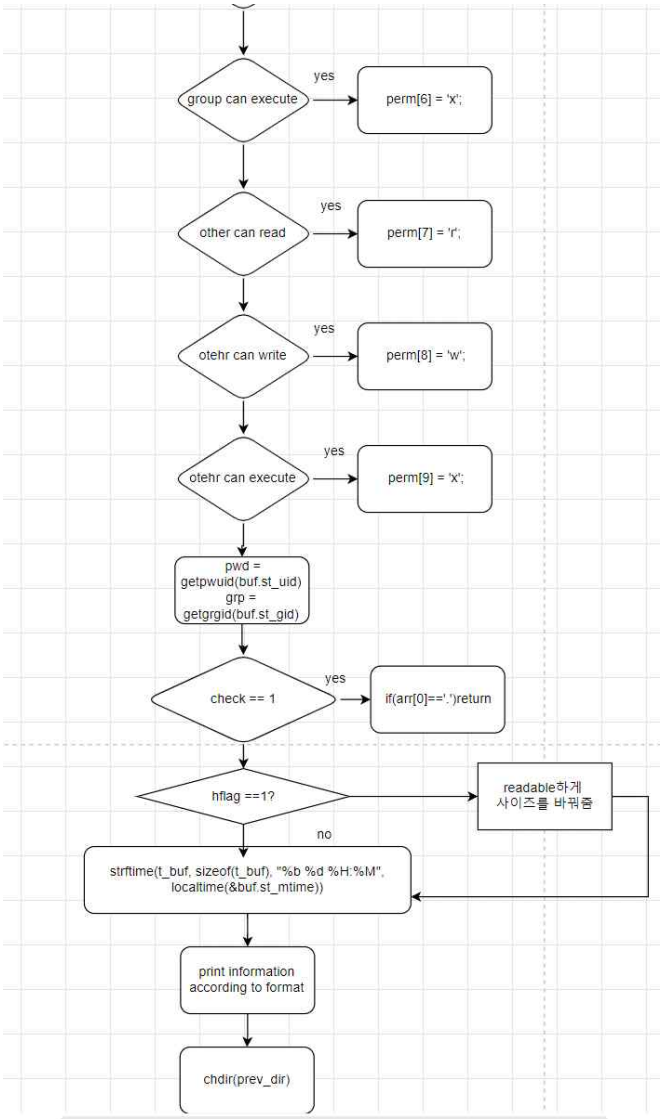


(2) sort_hidden함수

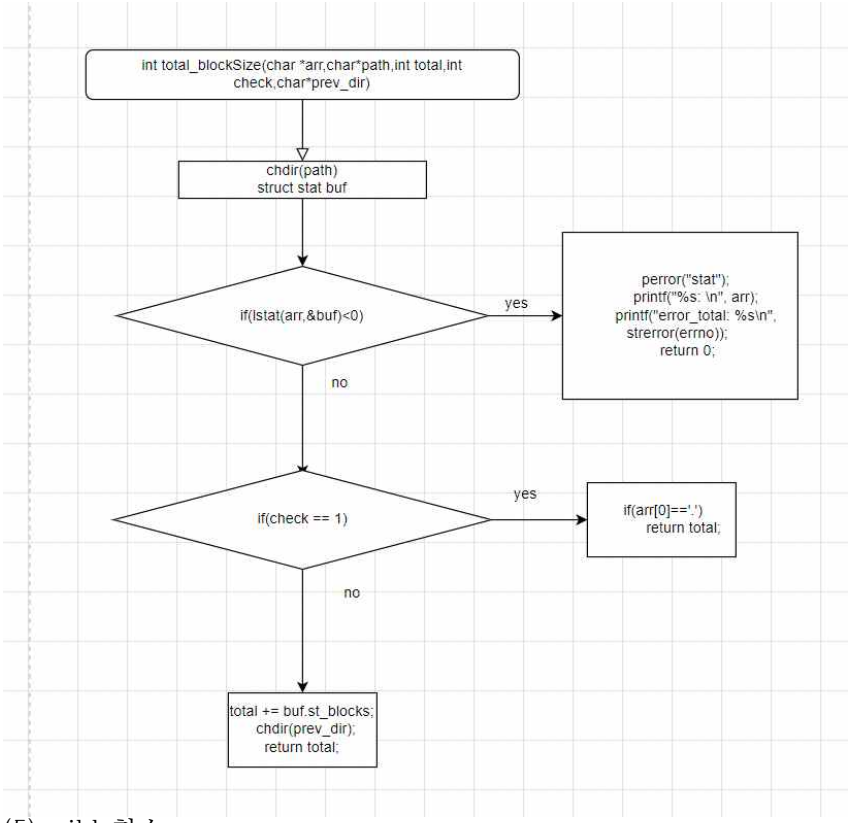

void sort_hidden(int dir_num, char*arr[],int max)
char * tmp = malloc(sizeof(char)*max);
arr의 두 요소값을 대문자로 바꾼다.
for (int i =0; i<dir_num-1; i++)
for(int j= i+1; j<dir_num; j++)
if(arr[i][0]=='.' && arr[j][0] =='.')
strcmp(big_arr1+1,big_arr2+1)
>0
(arr[i][0]=='.' && arr[j][0] !='.')
strcmp(big_arr1+1,big_arr2
>0
(arr[i][0]!='.' && arr[j][0] =='.')
strcmp(big_arr1,big_arr2+1
>0
else
strcmp(big_arr1,big_arr2
>0
순서를 바꾼다.

(3) option_l함수

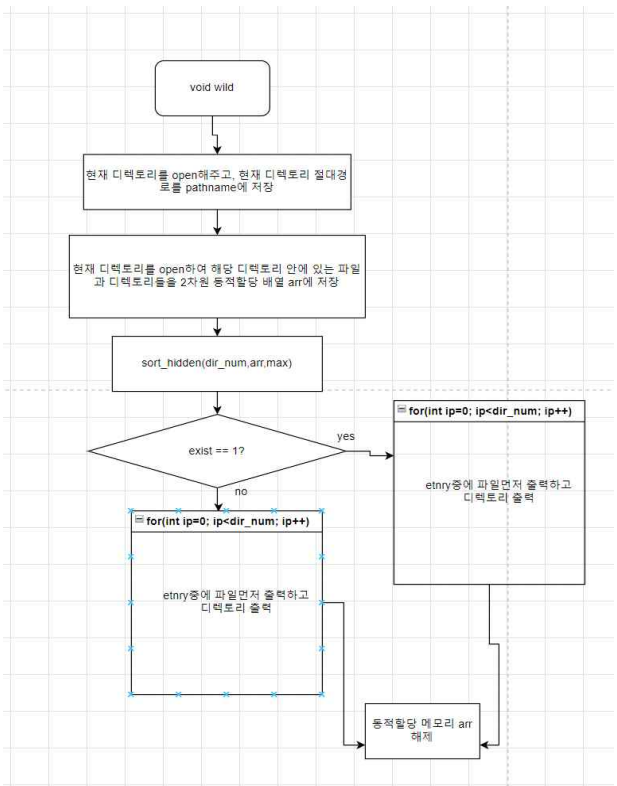

void option_l(char *arr,char*path,int check,char*prev_dir,int hflag)
chdir(arr)
char perm[11]= "----------\0"
if(lstat(arr, &buf) <0)
yes
perror("stat");
printf("%s: \n", arr);
printf("errorofoption: %s\n", strerror(errno));
return;
directory? symbolic link?
sym link
perm[0] = 'l'
directory
perm[0] = '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';



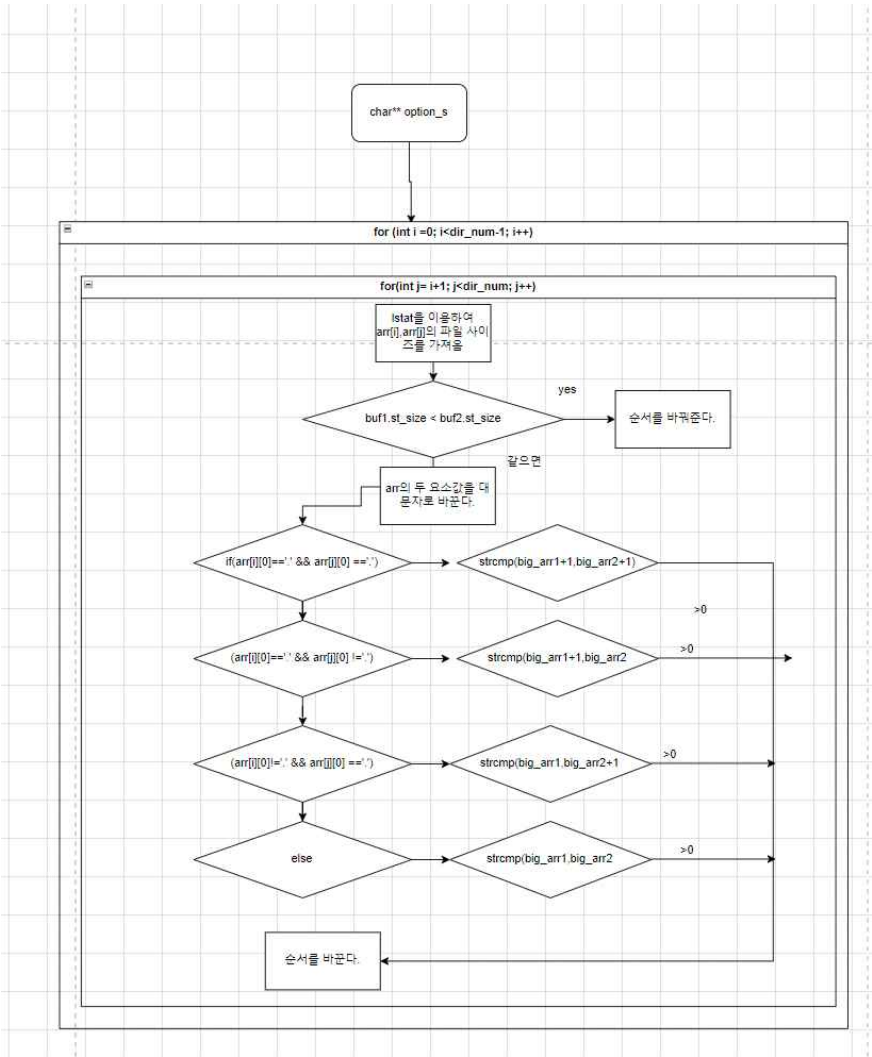
(4)total_blockSize함수


int total_blockSize(char *arr,char*path,int total,int check,char*prev_dir)
chdir(path)
struct stat buf
if(lstat(arr,&buf)<0)
yes
perror("stat");
printf("%s: \n", arr);
printf("error_total: %s\n", strerror(errno));
return 0;
no
if(check == 1)
yes
if(arr[0]=='.')
return total;
no
total += buf.st_blocks;
chdir(prev_dir);
return total;

(5) wild 함수


void wild
현재 디렉토리를 open해주고, 현재 디렉토리 절대경로를 pathname에 저장
현재 디렉토리를 open하여 해당 디렉토리 안에 있는 파일과 디렉토리들을 2차원 등적할당 배열 arr에 저장
sort_hidden(dir_num,arr,max)
exist == 1?
yes
no
for(int ip=0; ip<dir_num; ip++)
etnry중에 파일먼저 출력하고 디렉토리 출력
for(int ip=0; ip<dir_num; ip++)
etnry중에 파일먼저 출력하고 디렉토리 출력
동적할당 메모리 arr 해제

(6) option_s


char** option_s
for (int i =0; i<dir_num-1; i++)
for(int j= i+1; j<dir_num; j++)
lstat를 이용하여 arr[i],arr[j]의 파일 사이즈를 가져올
buf1.st_size < buf2.st_size
yes
순서를 바꿔준다.
같으면
arr의 두 요소값을 대문자로 바꾼다.
if(arr[i][0]=='.' && arr[j][0] ==',')
strcmp(big_arr1+1,big_arr2+1)
>0
(arr[i][0]=='.' && arr[j][0] !='.')
strcmp(big_arr1+1,big_arr2
>0
(arr[i][0]!='.' && arr[j][0] ==',')
strcmp(big_arr1,big_arr2+1
>0
else
strcmp(big_arr1,big_arr2
>0
순서를 바꾼다.

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.
hflag가 1이라면
파일 사이즈가 1024보다 작을 때,
파일 사이즈 출력
1024^2보다 작을 때
파일사이즈/1024.0을 올림하여 K붙여서 소수점 없이 출력
1024^3보다 작을 때
파일사이즈/1024.0^2을 올림하여 M붙여서 소수점 없이 출력
나머지
파일사이즈/1024.0^3을 올림하여 G붙여서 소수점 없이 출력
접근권한,link수, user name, group name,size,수정시간,파일명을 '\t'로 구분하여 출력한 후,
prev_dir로 되돌아간다.

option_s

인자로 받은 path로 이동한다.
i는 0부터 dir_num-1까지
j는 i+1부터 dir_num까지
arr[i]과 arr[j]의 파일 사이즈를 비교한다.
arr[i]의 사이즈 < arr[j] 사이즈
둘의 위치를 바꿔준다.
만약, 둘의 사이즈가 같다면
둘 모두 대문자로 바꿔준후,
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.

void wild
인자로 받은 path로 이동
해당 path를 opendir로 열어준 후, entry들을 arr에 저장한다.
check ==1이라면
모든 entry들을 반복문을 통해 돌며
만약 hidden file이라면 출력을 skip한다.
해당 entry가 opendir로 안열리면(directory가 아니라면)
출력한다.
해당 entry가 opendr로 열리면
출력한다.

check ==2이라면
모든 entry들을 반복문을 통해 돌며
만약 hidden file이라면 출력을 skip한다.
해당 entry가 opendir로 안열리면(directory가 아니라면)
출력한다.
해당 entry가 opendr로 열리면
출력한다.
arr 동적할당 해제
open한 directory닫기
이전 경로로 이동

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
getopt를 통해 옵션에 a가 있다면 aflag를 1로 set, l이 있다면 lflag를 1로 set한다. S가 있다면 sflag를, h가 있다면 hflag를, r이 있다면
rflag를 1로 바꾼다.
만약 argc ==1이거나, argc==2인데, lflag==0, aflag ==0일때,
argc가 1이 아니라면
인자에 wild card가 있는지 확인한다.
'/'로 시작하면 exist =1 , 현재디렉토리를 open한다.
아니라면 exist =2,
절대경로 중 wild card가 있는 부분 전까지 잘라서 path에 저장
나머지 뒷부분은 pattern array에 저장한다.
path를 open한다.
exist ==0이라면
인자값을 open한다. NULL값을 반환한다면,
현재 디렉토리를 연다.

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.
숨김파일이라면(파일이름이 .으로 시작하면) print를 하지 않고, entry들을 정렬된 순서대로 출력한다.
exist가 1이라면,
arr을 처음부터 끝까지 돌면서 pattern과 매칭이 되는 entry라면
entry가 디렉토리가 아니라면
경로 출력과 wild함수로 간다.
arr을 처음부터 끝까지 돌면서 pattern과 매칭이 되는 entry라면
entry가 디렉토리라면
경로출력과 wild함수로 간다.

exist가 2라면,
arr을 처음부터 끝까지 돌면서 pattern과 매칭이 되는 entry라면
entry가 디렉토리가 아니라면
wild함수로 간다.
arr을 처음부터 끝까지 돌면서 pattern과 매칭이 되는 entry라면
entry가 디렉토리라면
wild함수로 간다.
동적할당해준 것을 모두 해제해준 다음, main함수를 끝낸다.

argc==2 && aflag ==1 && lflag ==0이라면,
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.
만약 rflag ==1이라면,
sorting된 arr을 거꾸로 출력한다.
단, print를 할 때, '.'으로 시작하는 entry도 출력해준다.
아니라면,
처음부터 끝까지 순서대로 arr을 출력한다.
단, print를 할 때, '.'으로 시작하는 entry도 출력해준다.
동적할당해준 것을 모두 해제해준 다음, main함수를 끝낸다.

옵션이 l인데 argc==2일때 local_check =1, 옵션이 la이면 local_check =0으로 set한다.
자동으로 현재 디렉토리를 open해주고, 현재 디렉토리 절대경로를 pathname에 저장한다.
현재 디렉토리 안에 있는 entry들을 2d array에 저장하고 이를 차례로 total_blockSize()에 넣어 블록 사이즈의 총합을 알아낸다.
만약 hflag==0이라면,
total size를 출력한다.
만약 hflag ==1이라면,
total값이 1024보다 작으면
K와 함께 출력한다.
total값이 1024^2보다 작으면
total/1024.0을 소수점 없이 M과 출력
나머지라면
total/1024.0^2d을 소수점 없이 G와 출력
rflag가 1이라면
거꾸로 반복문을 돌며 모든 entry들을 option_l함수에 넣는다.
아니라면
반복문을 통해 모든 entry들을 option_l함수에 넣는다.
동적할당해준 것을 모두 해제해준 다음, main함수를 끝낸다.

옵션이 있고과 디렉토리, 파일 이름이 인자로 있으면
인자들을 파일인지 디렉토리인지 패턴인지 구분해서 opendir반환값에 따라(따로 정렬해주기 위해) 각각
for문을 통해 옵션다음부터 인자값 끝까지 돌면서 arr_file, arr_dir, arr_pat에 저장해준다.
연 directory는 다시 close해준다.
arr_file, arr_dir, arr_pat은 각각 요소들을 정렬해줘야하기 때문에 sort_hidden함수를 각각 호출하여 정렬해준다.

옵션이 a만 있다면,
file의 경우 arr_file 처음부터 끝까지 for문을 돌면서 F_OK를 통해 존재하는 파일인지 아닌지 판단한 다음 파일을 출력해준다.
directory의 경우 arr_dir 처음부터 끝까지 for문을 돌면서 해당 디렉토리를 open및 read하여 해당 디렉토리 내의 entry들을
arr이라는 2d array에 저장한다. arr의 요소들도 정렬해줘야하기 때문에, arr요소를 인자로 sort_hidden함수를 call 한다.
만약 rflag ==1이라면,
sorting된 arr을 거꾸로 출력한다.
단, print를 할 때, '.'으로 시작하는 entry도 출력해준다.
아니라면,
처음부터 끝까지 순서대로 arr을 출력한다.
단, print를 할 때, '.'으로 시작하는 entry도 출력해준다.
메모리 할당 해제 및 디렉토리를 닫아준다.

lfag==1 && rflag==0 이라면,
file의 경우 arr_file 처음부터 끝까지 반복하면서 파일이 없는 경우를 제외하고, 현재 디렉토리를 저장하여 출력한 다음, option_l함수로 넘어간다.
디렉토리의 경우, p_path, path를 동적할당 배열로 선언하여 절대경로를 만들어준다. 우선 현재 디렉토리를 prev_dir에 저장해놓고, p_path에 /을 저장한다.
p_path에 인자로 줄 디렉토리를 복사하여 최종적으로 /arr_dir[i]형식으로 만든다음 현재 디렉토리를 앞에 붙임으로써 절대경로를 완성시킨다.
arr_dir 처음부터 끝까지 for문을 돌면서 해당 디렉토리를 open및 read하여 해당 디렉토리 내의 entry들을
arr이라는 2d array에 저장한다. arr의 요소들도 정렬해줘야하기 때문에, arr요소를 인자로 sort_hidden함수를 call 한다.
그 다음 정렬된 entry들을 각각 total_blockSize()에 넣는다.
만약 hflag ==1이라면,
total값이 1024보다 작으면
K와 함께 출력한다.
total값이 1024^2보다 작으면
total/1024.0을 소수점 없이 M과 출력

나머지라면
total/1024.0^2d을 소수점 없이 G와 출력
만약, sflag==1이라면,
option_s함수로 들어간다.
반복문을 통해 모든 entry들을 option_l함수에 넣는다.
정렬된 entry들을 각각 option_l함수에 넣는다. 이때 옵션이 a가 있기 때문에 함수인자에 check부분에 0을 넣는다.
메모리 할당 해제 및 디렉토리를 닫아준다.

lfag==1 && rflag==1 이라면,
file의 경우 arr_file 처음부터 끝까지 반복하면서 파일이 없는 경우를 제외하고, 현재 디렉토리를 저장하여 출력한 다음, option_l함수로 넘어간다.
디렉토리의 경우, p_path, path를 동적할당 배열로 선언하여 절대경로를 만들어준다. 우선 현재 디렉토리를 prev_dir에 저장해놓고, p_path에 /을 저장한다.
p_path에 인자로 줄 디렉토리를 복사하여 최종적으로 /arr_dir[i]형식으로 만든다음 현재 디렉토리를 앞에 붙임으로써 절대경로를 완성시킨다.
arr_dir 처음부터 끝까지 for문을 돌면서 해당 디렉토리를 open및 read하여 해당 디렉토리 내의 entry들을
arr이라는 2d array에 저장한다. arr의 요소들도 정렬해줘야하기 때문에, arr요소를 인자로 sort_hidden함수를 call 한다.
그 다음 정렬된 entry들을 각각 total_blockSize()에 넣는다.
만약 hflag ==1이라면,
total값이 1024보다 작으면
K와 함께 출력한다.
total값이 1024^2보다 작으면
total/1024.0을 소수점 없이 M과 출력
나머지라면
total/1024.0^2d을 소수점 없이 G와 출력
만약, sflag==1이라면,
option_s함수로 들어간다.
거꾸로 반복문을 돌며 모든 entry들을 option_l함수에 넣는다.
정렬된 entry들을 각각 option_l함수에 넣는다. 이때 옵션이 a가 있기 때문에 함수인자에 check부분에 0을 넣는다.
메모리 할당 해제 및 디렉토리를 닫아준다.

옵션이 없다면,
pattern이 있을 경우,
해당 패턴이 '/'로 시작하면 exist =1, path는 해당 패턴으로 업데이트
해당 패턴이 '/'로 시작하지않다면, exist =2, 현재 디렉토리가 path, prev_dir이 된다. pattern을 저장한다.
마지막 '/'가 있을 때까지 num을 업데이트 해주다가, 마지막 '/'를 만나면 path[num]='\0'

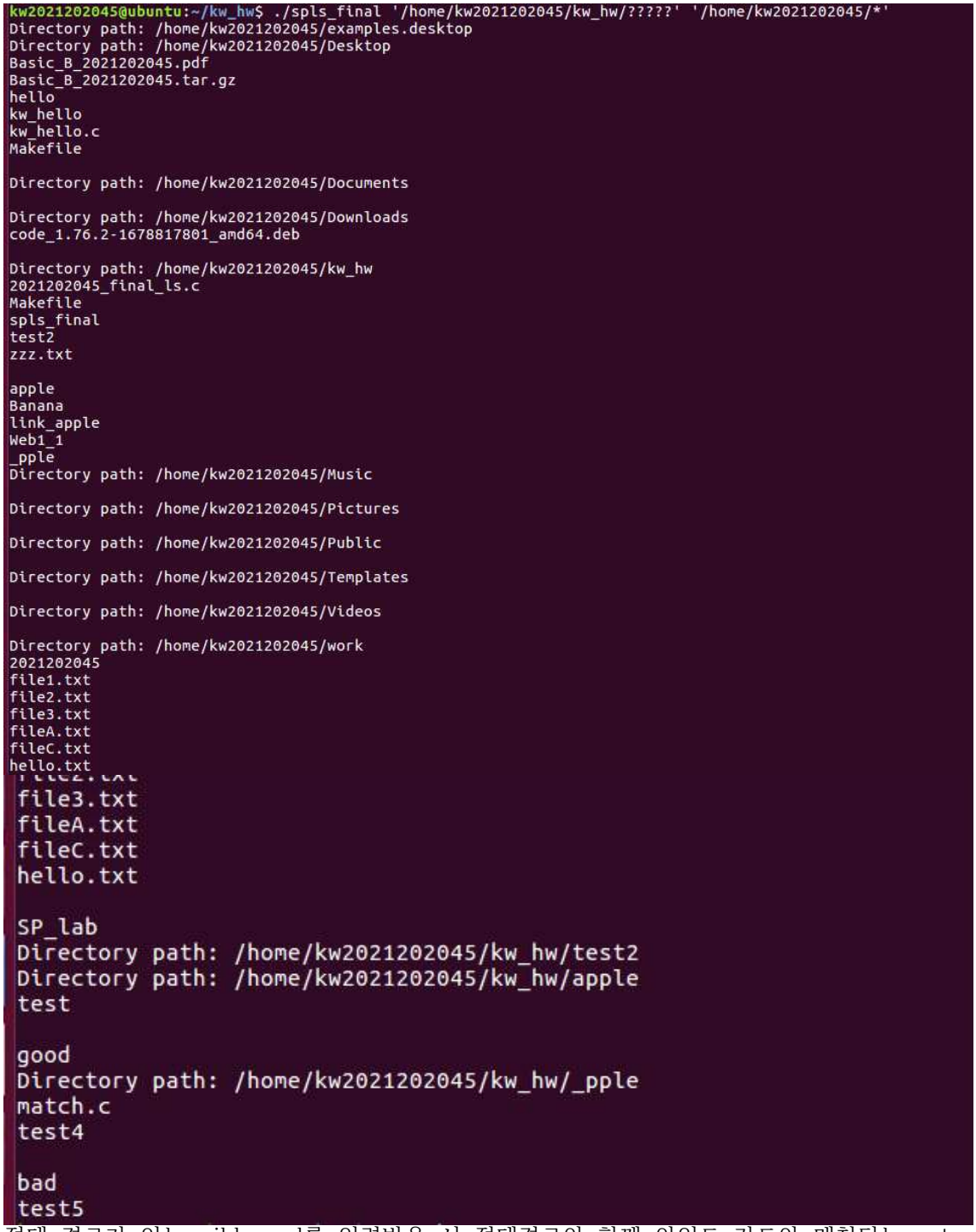
pattern이 있는 부분만 pattern에 update한다.
path를 opendir한다.
해당 directory를 읽으면서 entry들을 arr에 저장한다.
exist가 1이라면,
arr을 처음부터 끝까지 돌면서 pattern과 매칭이 되는 entry라면
entry가 디렉토리가 아니라면
경로 출력과 wild함수로 간다.
arr을 처음부터 끝까지 돌면서 pattern과 매칭이 되는 entry라면
entry가 디렉토리라면
경로출력과 wild함수로 간다.
exist가 2라면,
arr을 처음부터 끝까지 돌면서 pattern과 매칭이 되는 entry라면
entry가 디렉토리가 아니라면
wild함수로 간다.
arr을 처음부터 끝까지 돌면서 pattern과 매칭이 되는 entry라면
entry가 디렉토리라면
wild함수로 간다.

file의 경우 arr_file 처음부터 끝까지 for문을 돌면서 F_OK를 통해 존재하는 파일인지 아닌지 판단한 다음 파일을 출력해준다.
directory의 경우 arr_dir 처음부터 끝까지 for문을 돌면서 해당 디렉토리를 open및 read하여 해당 디렉토리 내의 entry들을
arr이라는 2d array에 저장한다.
arr요소를 인자로 sort_hidden함수를 call 한다.
arr의 처음부터 끝까지 '.'으로 시작되는 entry를 제외한 entry를 출력
메모리 할당 해제 및 open된 디렉토리를 닫아준다.

Ⅳ. 결과화면

```
kw2021202045@ubuntu:~/kw_hw$ ./spls_final -lSrh
Directory path: /home/kw2021202045/kw_hw
total :116K
-rw-rw-r--      1       kw2021202045    kw2021202045    0       Mar 26 02:55    zzz.txt
-rw-rw-r--      1       kw2021202045    kw2021202045    0       Mar 24 09:20    test2
lrwxrwxrwx      1       kw2021202045    kw2021202045    5       Apr 04 07:01    link_apple
-rw-rw-r--      1       kw2021202045    kw2021202045    52      Apr 08 00:36    Makefile
drwxrwxr-x      4       kw2021202045    kw2021202045    4K      Apr 07 23:36    _pple
drwxrwxr-x      2       kw2021202045    kw2021202045    4K      Apr 07 07:00    Web1_1
drwxrwxr-x      2       kw2021202045    kw2021202045    4K      Mar 24 05:14    Banana
drwxrwxr-x      3       kw2021202045    kw2021202045    4K      Apr 04 03:50    apple
-rwxrwxr-x      1       kw2021202045    kw2021202045    39K     Apr 12 00:50    spls_final
-rw-rw-r--      1       kw2021202045    kw2021202045    55K     Apr 12 00:41    2021202045_final_ls.c
```

인자가 없을 때 파일의 정보들을 크기 순서의 반대로(-Sr), 파일 사이즈가 같을 때는, ASCII 코드로 문자열이 더 뒤에 있는 순서가 먼저 출력한다. h옵션 때문에, 파일 사이즈와 total block수가 readable하게 바뀌어 출력된다.



절대 경로가 있는 wild card를 입력받을 시 절대경로와 함께 와일드 카드와 매칭되는 entry 들을 출력한다. 파일을 먼저 출력하고 directory를 출력하는데 그 사이에 '\0'으로 구분해주었다.

```
kw2021202045@ubuntu:~/kw_hw$ ./spls_final '/home/kw2021202045/kw_hw/?????'
Directory path: /home/kw2021202045/kw_hw/test2
Directory path: /home/kw2021202045/kw_hw/apple
test

good
Directory path: /home/kw2021202045/kw_hw/_pple
match.c
test4

bad
test5
```

kw_hw라는 directory내에 다섯글자로 된 파일을 먼저 절대경로와 함께 출력해준 뒤, 다섯글자로 된 디렉토리들을 절대경로와 함께 출력해준다. 디렉토리의 경우 그 디렉토리 내의 entry들까지 출력해주는데, 이때 파일이 먼저 출력되고 '\0'으로 구분한 다음, directory가 출력되는 것을 확인할 수 있다.

```
kw2021202045@ubuntu:~/kw_hw$ ./spls_final apple _pple Web1_1 -lSrh
Directory path: /home/kw2021202045/kw_hw/_pple
total :12K
-rw-rw-r--      1       kw2021202045    kw2021202045    0       Apr 07 08:16    test4
-rw-rw-r--      1       kw2021202045    kw2021202045    536     Apr 07 08:10    match.c
drwxrwxr-x      2       kw2021202045    kw2021202045    4K      Apr 07 08:16    test5
drwxrwxr-x      2       kw2021202045    kw2021202045    4K      Apr 07 07:42    bad
Directory path: /home/kw2021202045/kw_hw/Web1_1
total :184K
-rw-rw-r--      1       kw2021202045    kw2021202045    4K      Mar 26 15:20    2021202045_simple_ls.c
-rwxrw-rw-      1       kw2021202045    kw2021202045    179K    Mar 28 04:22    Web1_1_B_2021202045.pdf
Directory path: /home/kw2021202045/kw_hw/apple
total :4K
-rw-rw-r--      1       kw2021202045    kw2021202045    0       Mar 24 09:20    test
drwxrwxr-x      2       kw2021202045    kw2021202045    4K      Mar 24 09:20    good
```

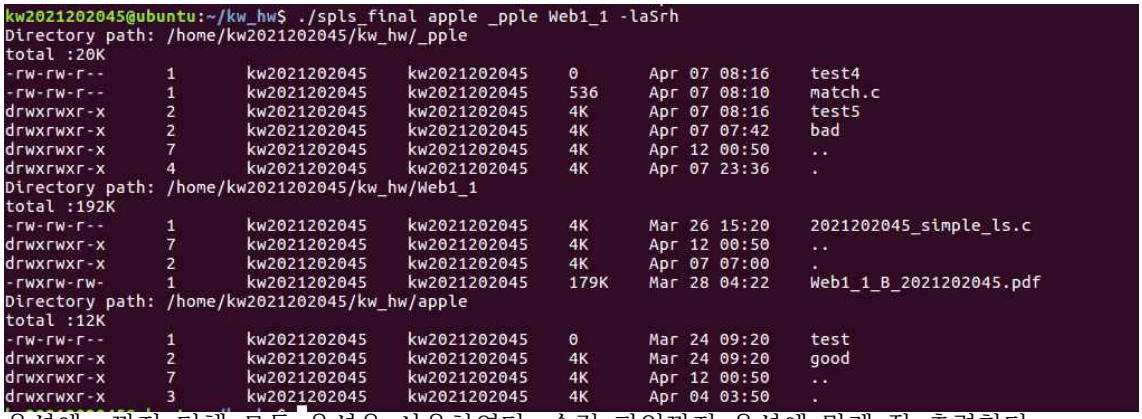
여러개의 파일 및 디렉토리 인자를 넣었을 때, 해당 인자들이 역순서로 열리면서 그 안의 entry들도 옵션에 맞게 출력되는 것을 확인할 수 있다.

```
kw2021202045@ubuntu:~/kw_hw$ ./spls_final '/home/kw2021202045/kw_hw/*.[a-z]'
Directory path: /home/kw2021202045/kw_hw/2021202045_final_ls.c
kw2021202045@ubuntu:~/kw_hw$ ls /home/kw2021202045/kw_hw/*.[a-z]
/home/kw2021202045/kw_hw/2021202045_final_ls.c
```

./spls_final과 ls의 출력을 비교한 것이다.

```
kw2021202045@ubuntu:~/kw_hw$ ./spls_final apple _pple Web1_1 -lSh
Directory path: /home/kw2021202045/kw_hw/apple
total :4K
drwxrwxr-x      2       kw2021202045    kw2021202045    4K      Mar 24 09:20    good
-rw-rw-r--      1       kw2021202045    kw2021202045    0       Mar 24 09:20    test
Directory path: /home/kw2021202045/kw_hw/Web1_1
total :184K
-rwxrw-rw-      1       kw2021202045    kw2021202045    179K    Mar 28 04:22    Web1_1_B_2021202045.pdf
-rw-rw-r--      1       kw2021202045    kw2021202045    4K      Mar 26 15:20    2021202045_simple_ls.c
Directory path: /home/kw2021202045/kw_hw/_pple
total :12K
drwxrwxr-x      2       kw2021202045    kw2021202045    4K      Apr 07 07:42    bad
drwxrwxr-x      2       kw2021202045    kw2021202045    4K      Apr 07 08:16    test5
-rw-rw-r--      1       kw2021202045    kw2021202045    536     Apr 07 08:10    match.c
-rw-rw-r--      1       kw2021202045    kw2021202045    0       Apr 07 08:16    test4
```

r이 없을 때의 결과화면이다.



옵션에 a까지 더해 모든 옵션을 사용하였다. 숨김 파일까지 옵션에 맞게 잘 출력한다.

Ⅴ.고찰

우선 특수문자가 있을 경우, 모두 대문자로 바꾸어 아스키코드에 맞추어 정렬을 해야했기 때문에 sort 함수를 다시 짰어야했다. toupper함수를 쓰는데 문자열 전체가 아니라 문자 하나하나에 접근하였기 때문에 for문으로 돌려야했다. strstr()함수의 경우 해당 문자열에 특정 문자가 있는지 없는지 검사해주는데, 이 함수를 처음 알게 되었다. fnmatch를 구현할 때, 잘못 이해해서 디렉토리 내부의 entry까지 매칭 시키는 경우를 만들어서 다시 짰는데 그 때문에 코드가 비효율적이게 되었다. h옵션을 쓸 때, block과 size에 대한 readable 단위로 고쳐주는 기준이 살짝 달라서 이 부분에서 고민했다. 또한, 인자가 하나 일 때, wild card를 구현하는 것은 괜찮았는데, 여러 인자를 받을 때, 이 인자가 디렉토리인지, 디렉토리가 아닌 entry인지, wild card인지 구분하여 2차원 배열에 넣어주는 것이 번거로웠다. 절대경로로 받을 때는 절대경로에서 wild card 앞부분은 path에 바로 뒤부터는 pattern에 저장하는 과정을 뒤늦게 더 효율적인 방법을 찾아서 아쉬웠다.