System Programming Report

Assignment 2-3 – Final ls

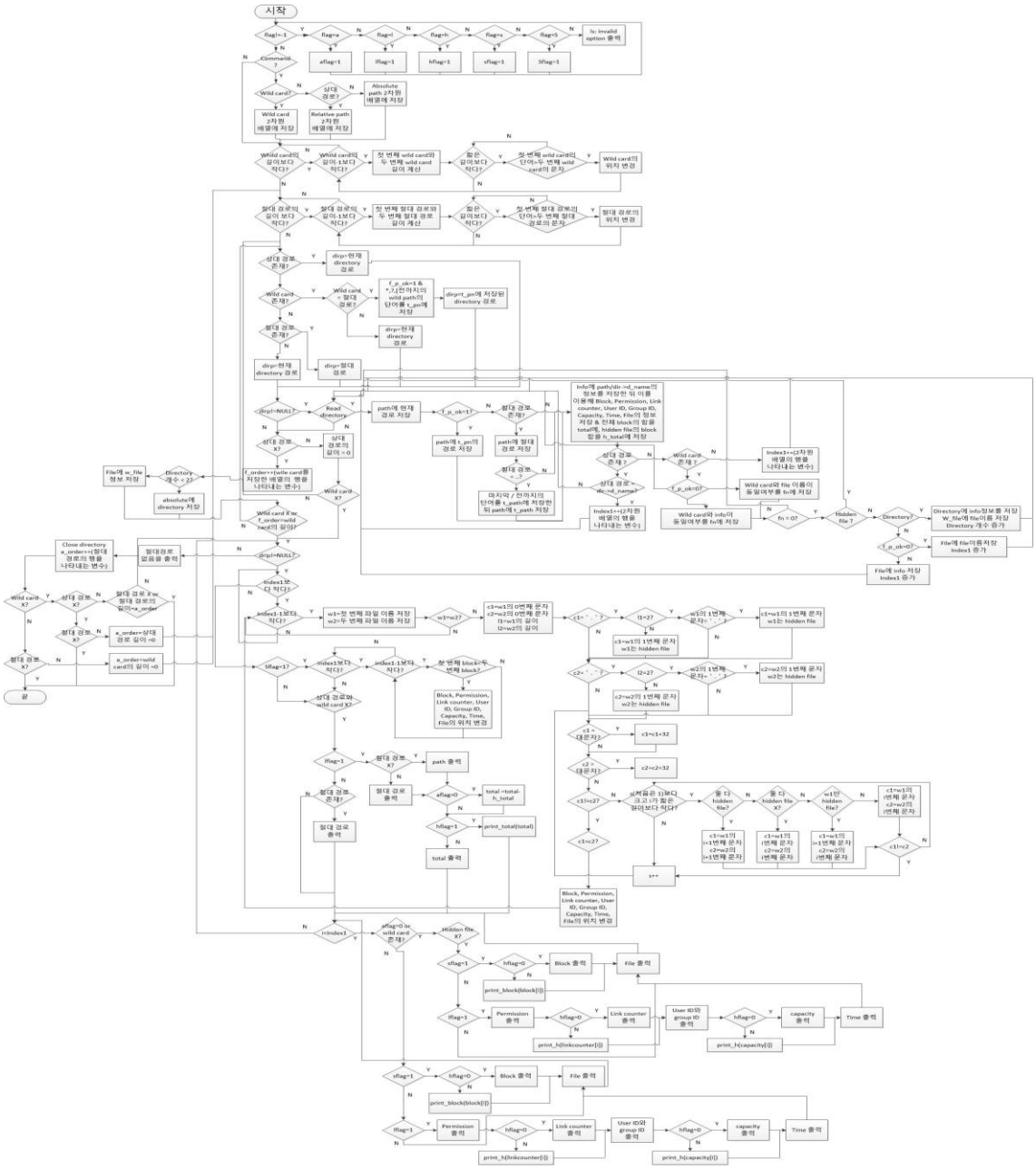
Professor	황호영 교수님
Department	Computer engineering
Student ID	2014722057
Name	김 진아
Class	설계 (화6 목4) / 실습 (금 56)
Date	2016. 4. 15

◆ **Introduction**

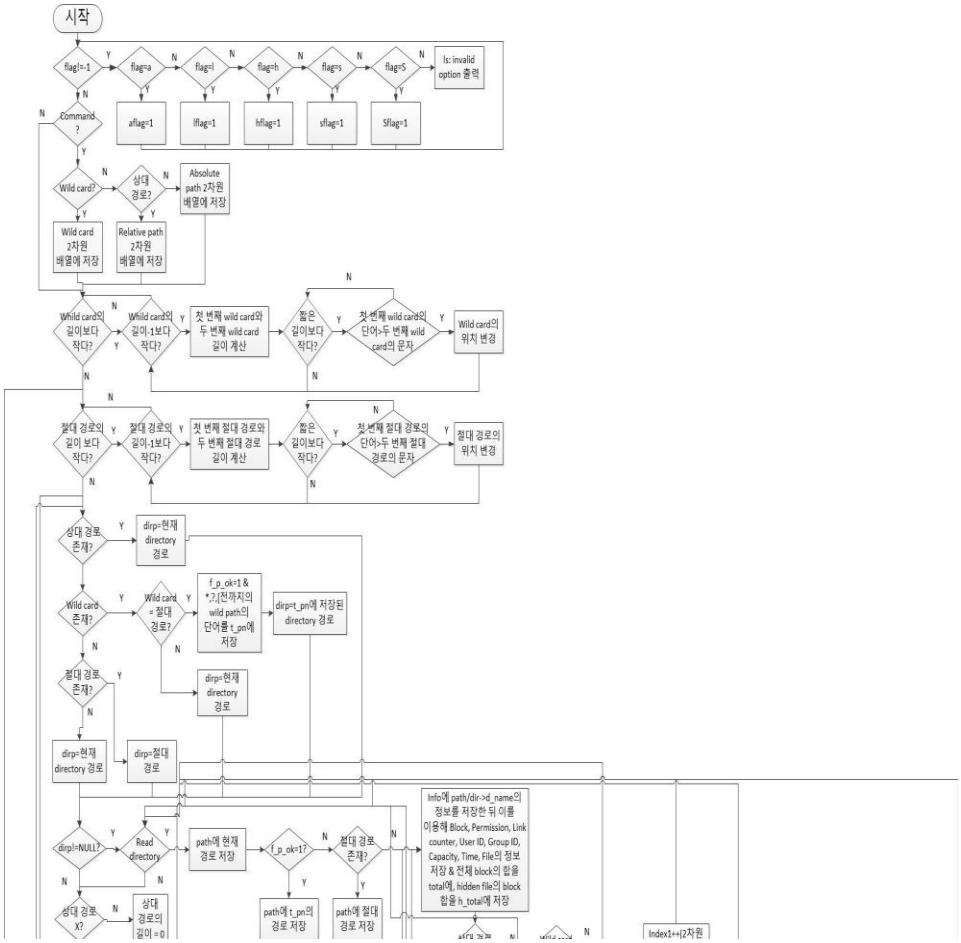
이번 과제는 final ls을 구현하는 것으로 지난 2-1와 2-2의 조건에 wild card matching('*', '?', '[seq]')을 하고 기존의 option에 –h, -s ,- S을 추가해 구현하는 것이다. wild card를 사용할 때 directory가 1개 나오면 이름만 출력하게 해준다. –h은 사람들이 보기 편한 형식으로 크기를 출력하고 –s는 각각의 파일에 할당된 block size를 출력한다. –S는 각각의 파일에 할당된 block size을 내림차순으로 출력한다. wild card를 할 때 '을 붙여서 사용한다.

◆ **Flowchart**

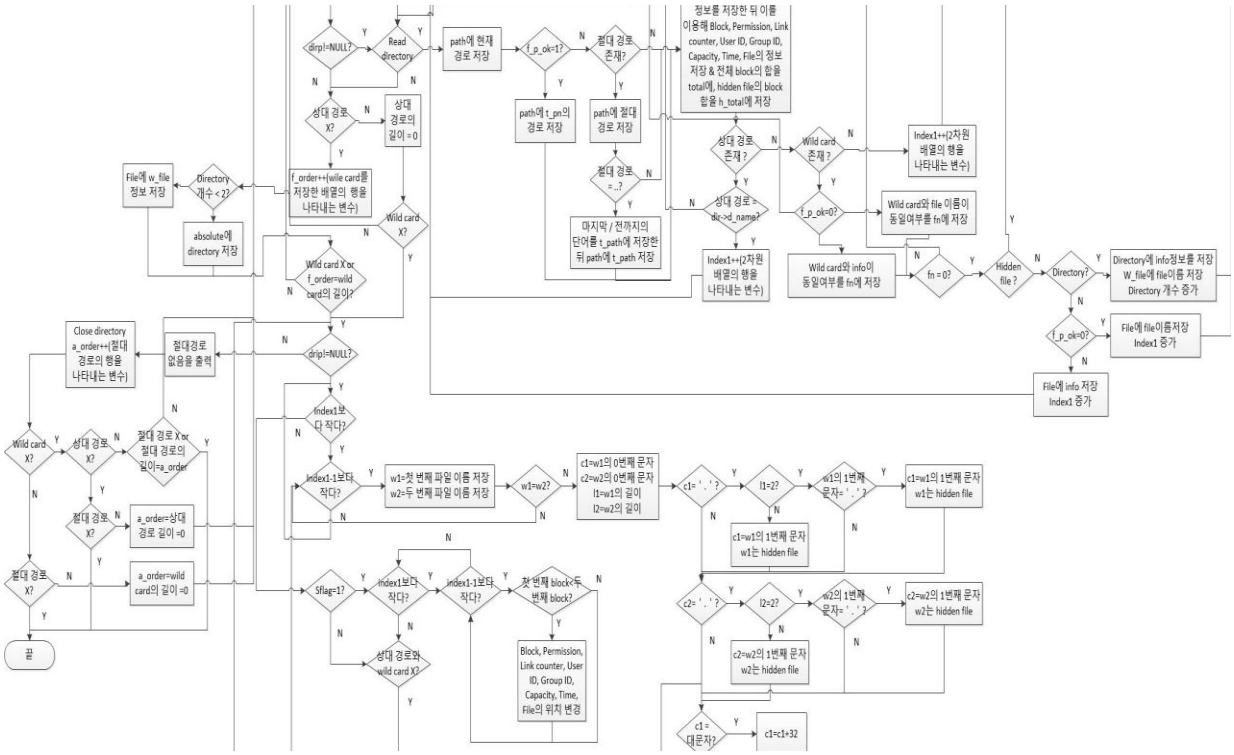
1) main 함수



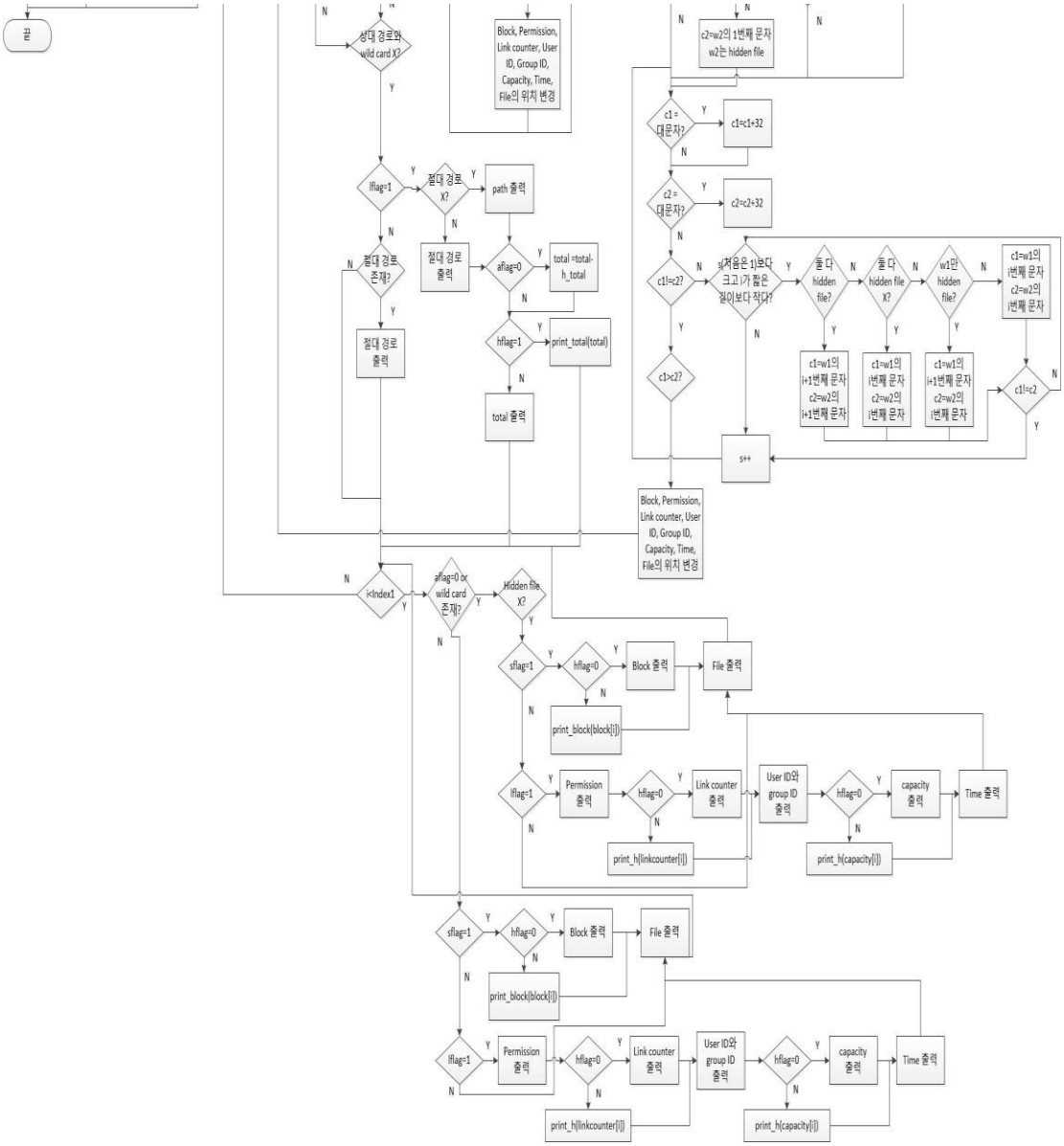
<1>


시작
flag!=1
flag=a
flag=l
flag=h
flag=s
flag=S
ls: invalid option 출력
Command ?
aflag=1
lflag=1
hflag=1
sflag=1
Sflag=1
Wild card?
상대 경로?
Absolute path 2차원 배열에 저장
Wild card 2차원 배열에 저장
Relative path 2차원 배열에 저장
Wild card의 길이보다 작다?
Wild card의 길이-1보다 작다?
첫 번째 wild card와 두 번째 wild card 길이 계산
짧은 길이보다 작다?
첫 번째 wild card의 단어>두 번째 wild card의 문자
Wild card의 위치 변경
절대 경로의 길이 보다 작다?
절대 경로의 길이-1보다 작다?
첫 번째 절대 경로와 두 번째 절대 경로 길이 계산
짧은 길이보다 작다?
첫 번째 절대 경로의 단어>두 번째 절대 경로의 문자
절대 경로의 위치 변경
상대 경로 존재?
dirp=현재 directory 경로
Wild card 존재?
Wild card = 절대 경로?
f_p_ok=1 & ',',[전까지의 wild path의 단어를 t_pn에 저장
dirp=t_pn에 저장된 directory 경로
절대 경로 존재?
dirp=현재 directory 경로
dirp=현재 directory 경로
dirp=절대 경로
dirp!=NULL?
Read directory
path에 현재 경로 저장
f_p_ok=1?
절대 경로 존재?
path에 t_pn의 경로 저장
path에 절대 경로 저장
Info에 path/dir->d_name의 정보를 저장한 뒤 이를 이용해 Block, Permission, Link counter, User ID, Group ID, Capacity, Time, File의 정보 저장 & 전체 block의 합을 total에, hidden file의 block 합을 h_total에 저장
상대 경로 X?
상대 경로의 길이 = 0
Index1++(2차원

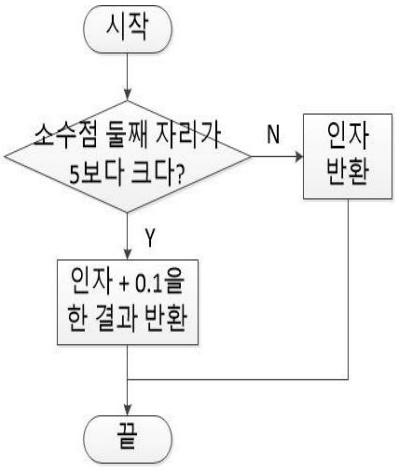
<2>


dirp!=NULL?
Read directory
path에 현재 경로 저장
f_p_ok=1?
절대 경로 존재?
path에 t_pn의 경로 저장
path에 절대 경로 저장
절대 경로 = ..?
마지막 / 전까지의 단어를 t_path에 저장한 뒤 path에 t_path 저장
정보를 저장한 뒤 이를 이용해 Block, Permission, Link counter, User ID, Group ID, Capacity, Time, File의 정보 저장 & 전체 block의 합을 total에, hidden file의 block 합을 h_total에 저장
상대 경로 X?
상대 경로의 길이 = 0
상대 경로 존재 ?
상대 경로 = dir->d_name?
Wild card 존재 ?
Index1++(2차원 배열의 행을 나타내는 변수)
Index1++(2차원 배열의 행을 나타내는 변수)
f_p_ok=0?
Wild card와 file 이름이 동일여부를 fn에 저장
Wild card와 info이 동일여부를 fn에 저장
fn = 0?
Hidden file ?
Directory?
Directory에 info정보를 저장 W_file에 file이름 저장 Directory 개수 증가
f_p_ok=0?
File에 file이름저장 Index1 증가
File에 info 저장 Index1 증가
File에 w_file 정보 저장
Directory 개수 < 2?
absolute에 directory 저장
f_order++(wile card를 저장한 배열의 행을 나타내는 변수)
Wild card X?
Wild card X or f_order=wild card의 길이?
Close directory a_order++(절대 경로의 행을 나타내는 변수)
절대경로 없음을 출력
dirp!=NULL?
Index1보다 작다?
Index1-1보다 작다?
w1=첫 번째 파일 이름 저장 w2=두 번째 파일 이름 저장
w1=w2?
c1=w1의 0번째 문자 c2=w2의 0번째 문자 l1=w1의 길이 l2=w2의 길이
c1= '.' ?
l1=2?
w1의 1번째 문자= '.' ?
c1=w1의 1번째 문자 w1는 hidden file
c1=w1의 1번째 문자 w1는 hidden file
c2= '.' ?
l2=2?
w2의 1번째 문자= '.' ?
c2=w2의 1번째 문자 w2는 hidden file
c2=w2의 1번째 문자 w2는 hidden file
c1 = 대문자?
c1=c1+32
Wild card X?
상대 경로 X?
절대 경로 X or 절대 경로의 길이=a_order
절대 경로 X?
a_order=상대 경로 길이 =0
절대 경로 X?
a_order=wild card의 길이 =0
끝
Sflag=1?
Index1보다 작다?
Index1-1보다 작다?
첫 번째 block<두 번째 block?
상대 경로와 wild card X?
Block, Permission, Link counter, User ID, Group ID, Capacity, Time, File의 위치 변경

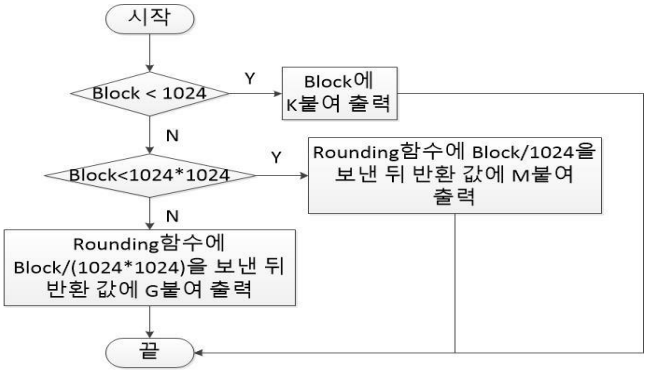
<3>



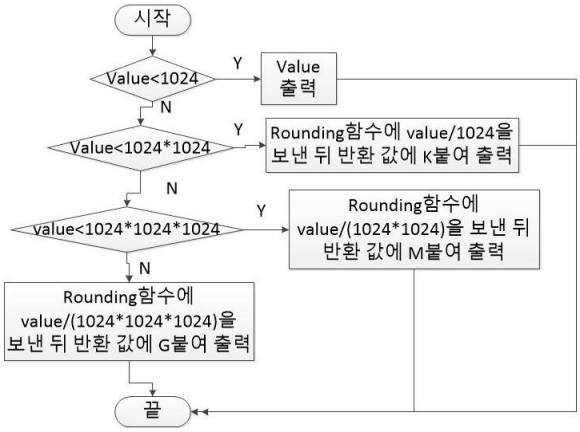
2) rounding 함수



3) print_block 함수



4) print_h 함수



◆ **Pseudo code**

1) main 함수

```
while(receive option){
        if flag is a:
                aflag is 1
        if flag is l:
                lflag is 1
        if flag is h:
                hflag is 1
        if flag is s:
                sflag is 1
        if flag is S:
```

```
                Sflag is 1
        if no option
                end program
}
for(i=0;i<1000;i++)
        initialize relative, absolute, fn_arr(wild card) 2D array
for(i=optind;i<argc;i++){
        f_ok is 0
        for(j=0;argv[i][j]!='₩0';j++){
                if wild card exist{
                        f_ok is 1;
                        out of loop
                }
        }
        if f_ok is 1 (exist wild card)
                fn_arr[f_len++] is argv[i]
        else{
                if relative path exist
                        relative[r_len++] is argv[i]
                if absolute path exist
                        absolute[a_len++] is argv[i]
        }
}
for(k1=0;k1<f_len;k1++){
        for(k2=0;k2<f_len-1;k2++){
                initialize length variables
```

```
                    calculate lengths of first and second wild card
                    compare wild cards' length and l is smaller length of wild cards
                    for(i=0;i<l;i++){
                        if first and second wild cards are different{
                            if fist wild card is more than second wild card{
                                first and second wild cards' position
                            }
                            out of loop
                        }
                    }
            }
    }
    while(infinite loop){
            initialize variables(l, total, h_total, index1, index2)
            initialize block, linkcounter, capacity, month, day, hour, minute 1D array
            for(i=0;i<1000;i++)
                    initialize permission, u_ID, g_ID, file 2D array
            for(k1=0;k1<a_len;k1++){
                    for(k2=0;k2<a_len-1;k2++){
                        initialize length variables
                        calculate lengths of absolute first path and second path
                        compare absolute paths' length and l is smaller length
                        for(i=0;i<l;i++){
                            if first absolute's character and second absolute's character
are different{
                                if first absolute's character is more than second
```

```
absolute's character{
                                                change first and second absolutes' position
                                        }
                                        out of loop
                                }
                        }
                }
        }
        while(1){
                initialize f_p_ok, d_count, d_index, i_index
                initialize information 1D array
                for(i=0;i<1000;i++)
                        initialize directory, w_file 2D array
                if relative path exist
                        dirp is current directory
                if wild card exist{
                        if wild card is absolute path{
                                initialize t_fn
                                for(i=0;fn_arr[f_order][i+1]!='*'&&        arr[f_order][i+1]!='?'&&
arr[f_order][i+1]!='[';i++)
                                        t_fn's ith character is fn_arr's ith character
                                dirp is t_fn's directory path
                                f_p_ok is 1
                        }
                        if wild card is relactive path
                                dirp is current directory
```

```
}
if absolute path exist
        dirp is absolute path's directory
if path doesn't exist
        dirp is current directory
if dirp isn't NULL{
        while(read directory){
                initialize variables(index2, w_end)
                path is current directory path
                if wild card is absolute path
                        path is t_fn's directory path
                if absolute path exist{
                        initialize t_path
                        path is absolute's directory path
                        iif abolute path is ".."{
                                for(i=0;path[i]!='₩0';i++){
                                        find position of last '/'
                                }
                                for(i=0;i<j;i++)
                                        t_path's ith character is path's ith
character
                                path is t_path's directory path
                        }
                }
                info is path/dir->d_name
                keep block size, permission, link counter, user ID, group ID,
```

capacity, month, day, hour, minute, file in arrays(block, permission, linkcounter, u_ID, g_ID, capacity, month, day, hour, minute)

```
total is total plus block size
if file is hidden file
        h_total is h_total plus block size
if relative path exist{
        for(i=0;i<r_len;i++){
                if relative path and file namd is same{
                        increase index1
                        out of loop
                }
        }
}
if wild card exist{
        if wild card is relative path
                if wild card and file name is same, fm is 0
        if wild card is absolute path
                if wild card and info is same, fm is 0
        if fm is 0{
        if file isn't hidden file{
                if file is directory{
                        keep info in directory
                        if wild card is relative path
                                keep file name in w_file
                        if wild card is absolute path
                                keep info in w_file
```

keep permission, u_ID, g_ID in w_file

keep block, linkcounter, capacity, month, day, hour, minute in information

directory count increase

}

if file is file{

if wild card is relative path

keep file name in file

if wild card is absolute path

keep info in file

increase index1

}

}

}

}

if wild card doesn't exist

increase index1

}

}

if relative path doesn’t exist{

increase wild card's order

if directory count is less than 2

save w_file’s information in each array(block, permission, linkcounter, u_ID, g_ID, capacity, month, day, hour, minute, file)

if directory count is more than 1{

```
                    for(i=0;i<d_index;i++)
                        keep directory[i] in absolute[a_len++]
                }
                if wild card doesn't exist or wild card's length and order are same
                    out of loop
            }
            if relative path exist {
                if wild card doesn't exist
                    out of loop
                initialize variable(relative paths' length)


            }
        }
        if directory exist{
            for(k1=0;k1<index1;k1++){
                for(k2=0;k2<index1-1;k2++){
                    initialize variables(l1, l2, s1, s2)
                    keep first file in w1 and keep second file in w2
                    if first word and second word are same
                        go to loop
                    c1 is w1's first character and c2 is w2's first character
                    calculate words' length
                    if c1 is '.'{
                        if w1's length is 2{
                            if w1 is parent directory{
                                c1 is w1's 1th character;
```

```
                    w1 is hidden file
            }
        }
        if w1's length isn't 2{
            c1 is w1's 1th character;
            w1 is hidden file
        }
    }
    if c2 is '.'{
        if w2's length is 2{
            if w2 is parent directory{
                c2 is w2's 1th character;
                w2 is hidden file
            }
        }
        if w2's length isn't 2{
            c2 is w2's 1th character;
            w2 is hidden file
        }
    }
    s is 1
    while(infinite loop){
        if c1 is capital letter, c1 is c1 plus 32
        if c2 is capital letter, c2 is c2 plus 32
        if c1 and c2 are different, out of loop
        if c1 and c2 are same{
```

```
compare words' length and l is smaller length
for(i=s;i<l;i++){
    if both aren't hidden file{
        c1 is ith w1's character
        c2 is ith w2's character
    }
    if both are hidden file{
        c1 is i+1th w1's character
        c2 is i+1th w2's character
    }
    if only first word is hidden file{
        c1 is i+1th w1's character
        c2 is ith w2's character
    }
    if only second word is hidden file
        c1 is ith w1's character
        c2 is 1+1th w2's character
    }
    if c1 and c2 different, out of loop
}
}
increase s(variable for loop)
}
if first word's character is more than second word's character{
    change position of file, block, permission, linkcounter,
```

u_ID, g_ID, capacity, month, day, hour, minute

}

initialize w1 and w2

}

}

if Sflag is 1(use option -S){

for(k1=0;k1<index1;k1++){

for(k2=0;k2<index1-1;k2++){

if first block is more than second block

change position of file, block, permission, linkcounter, u_ID, g_ID, capacity, month, day, hour, minute

}

}

}

}

if wild card and relative path don't exist{

if lflag is 1(use option -l){

if absolute path doesn't exist

print path

if absolute path exists

print absolute path

if aflag is 0 (not use option –a)

total is total minus h_total

if hflag is 1 (use option –h)

go to print_block function

if hflag is 0 (not use option –h)

```
                    print total
            }
            if lflag is 0 (not use option –l){
                if absolute path exist
                    print absolute path
            }
    }
    for(i=0;i<index1;i++){
            if aflag is 0 (not use option –a) and wild card exist{
                if file isn't hidden file{
                    if sflag is 1(use option -s){
                        if hflag is 0(not use option -h)
                            print block[i]
                        if hflag is 1(use option -h)
                            go to print_block function
                    }
                    if lflag is 1(use option -l){
                        print permission[i]
                        if hflag is 0(not use option -h)
                            print linkcounter[i]
                        if hflag is 1(use option -h)
                            go to print_h function
                        print u_ID[i], g_ID[i]
                        if hflag is 0(not use option -h)
                            print capacity[i]
                        if hflag is 1(use option -h)
```

```
                                        go to print_h function
                                print month[i], day[i], hour[i], minute[i]);
                        }
                        print file[i]
                }
        }
        if aflag is 1 (use option –a){
                if sflag is 1(use option -s){
                                if hflag is 0(not use option -h)
                                        print block[i]
                                if hflag is 1(use option -h)
                                        go to print_block function
                        }
                        if lflag is 1(use option -l){
                                print permission[i]
                                if hflag is 0(not use option -h)
                                        print linkcounter[i]
                                if hflag is 1(use option -h)
                                        go to print_h function
                                print u_ID[i], g_ID[i]
                                if hflag is 0(not use option -h)
                                        print capacity[i]
                                if hflag is 1(use option -h)
                                        go to print_h function
                                print month[i], day[i], hour[i], minute[i]);
                        }
```

```
                                        print file[i]
                                }
                        }
                }
        }
if directory path doesn’t exist
        print absolute[a_order])’s path
close directory
increase variable(absolute path's order)
if wild card doesn't exist{
        if relative path doesn't exist{
                if absolute path doesn't exist or done absolute path
                        out of loop
        }
        if relative path exists{
                if absolute path doesn't exist
                        out of loop
                if absolute path exist
                        initialize relative path's length & absolute path's order
        }
}
if wild card exist{
        if absolute path doesn't exist
                out of loop
        if absolute path exist
                initialize absolute path's length & wild card's length
```

```
        }
    }
```

2) rounding 함수

```
    h is num*100
    fl is h divided by 100
    if num*100-h*10's integer value is less than 5
        ro is fl
    if num*100-h*10's integer value is more than 5
        ro is fl plus 0.01;
    return ro
```

3) print_block 함수

```
    if block is less than 1024(K)
        print blockK
    if block is less than 1024*1024(M){
        h_block is block divided by 1024
        print value returned rounding function with M
    }
    if block is more than 1024*1024(G){
        h_block is block divided by 1024*1024
        print value returned rounding function with G
    }
```

4) print_h 함수

```
    if value is less than 1024
        print value
    if value is less than 1024*1024(K){
        h_value is value divided by 1024
```

```
        print value returned rounding function with K
    }
    if value is less than 1024*1024*1024(M){
        h_value is value divided by 1024*1024
        print value returned rounding function with M
    }
    if value is more than 1024*1024*1024(G){
        h_value is value divided by 1024*1024*1024
        print value returned rounding function with G
    }
```

◆ **Reference**

option –h를 하면 어떤 식으로 보여줘야 하는지 헷갈렸다. 그래서 강의자료와 sslab에 있는 조교님들의 답변들을 보며 어떤 식으로 구현해야 될지 알게 되었다. 소수점 둘째 자리에서 반올림해줘서 출력하게 구현했다. wild card에 대해 설명을 들었지만 어떤 식으로 되는지 알지 못해 강의자료를 참고하며 리눅스 상에서 하나씩 해보면서 정확하게 이해했다. wild card가 절대경로일 때 리눅스 상에서 나오는 것처럼 정렬해서 출력되도록 구현했다.