System Programming Report

Assignment 2-2 – Advanced ls

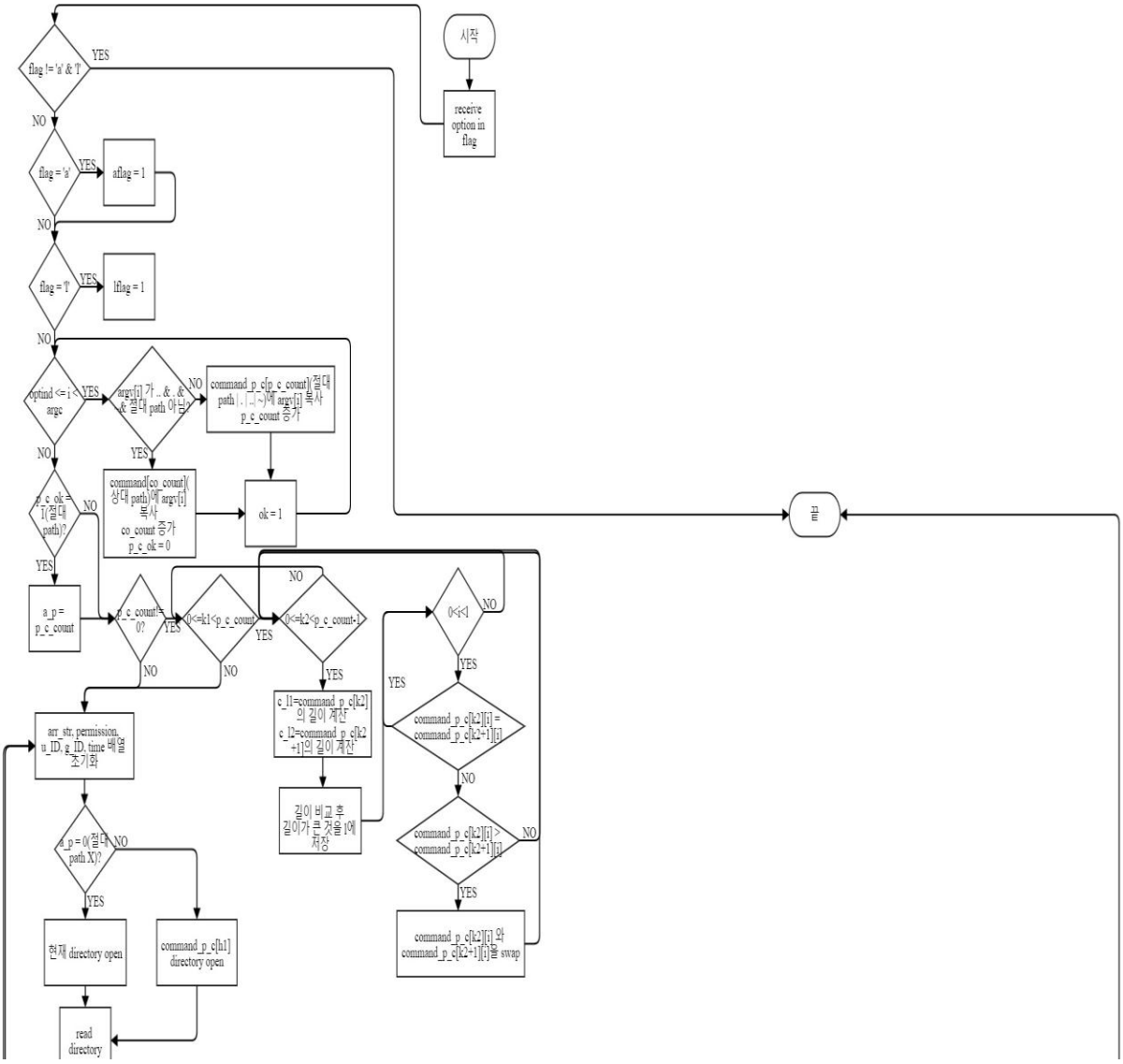
Professor	황호영 교수님
Department	Computer engineering
Student ID	2014722057
Name	김 진아
Class	설계 (화6 목4) / 실습 (금 56)
Date	2016. 4. 8

◆ **Introduction**

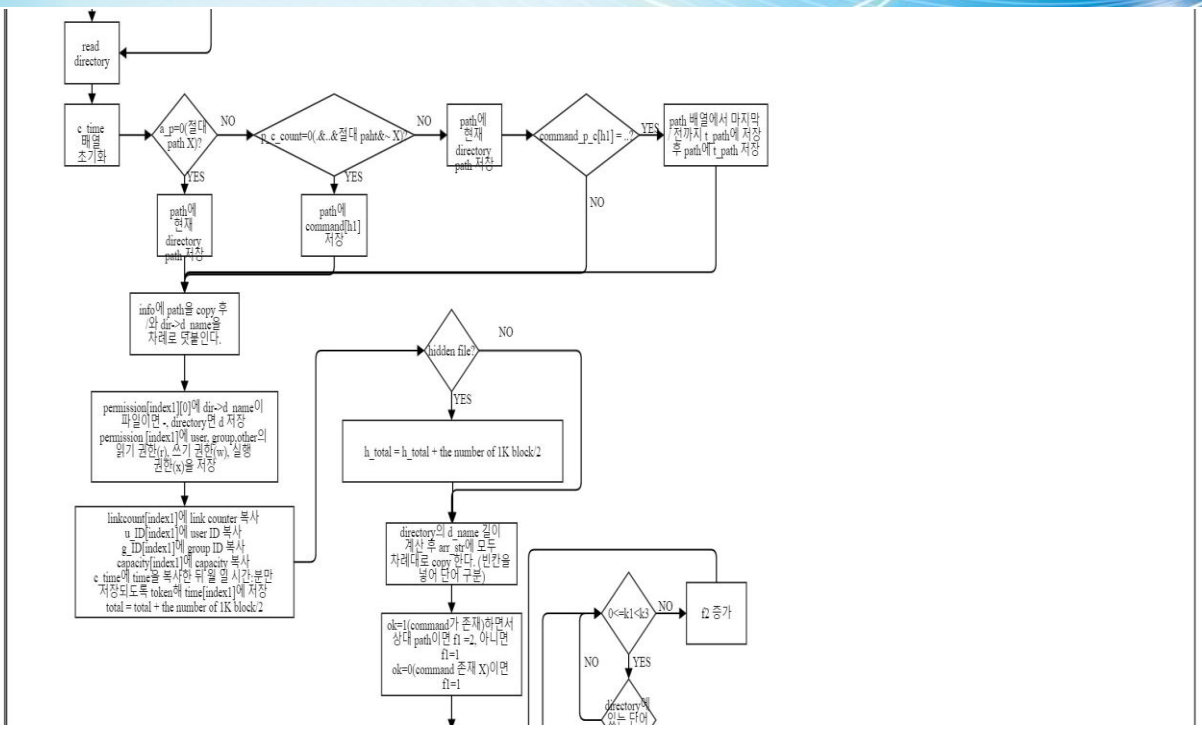
이번 과제는 저번 과제에서 구현한 simple ls에 option –a와 –l의 기능을 추가해주는 것이다. 또 directory의 위치와 1K blocks의 개수를 출력하도록 구현한다. 상대 path와 절대 path을 입력 받아 이에 따른 결과가 알맞게 출력되게 구현한다.

◆ **Flowchart**

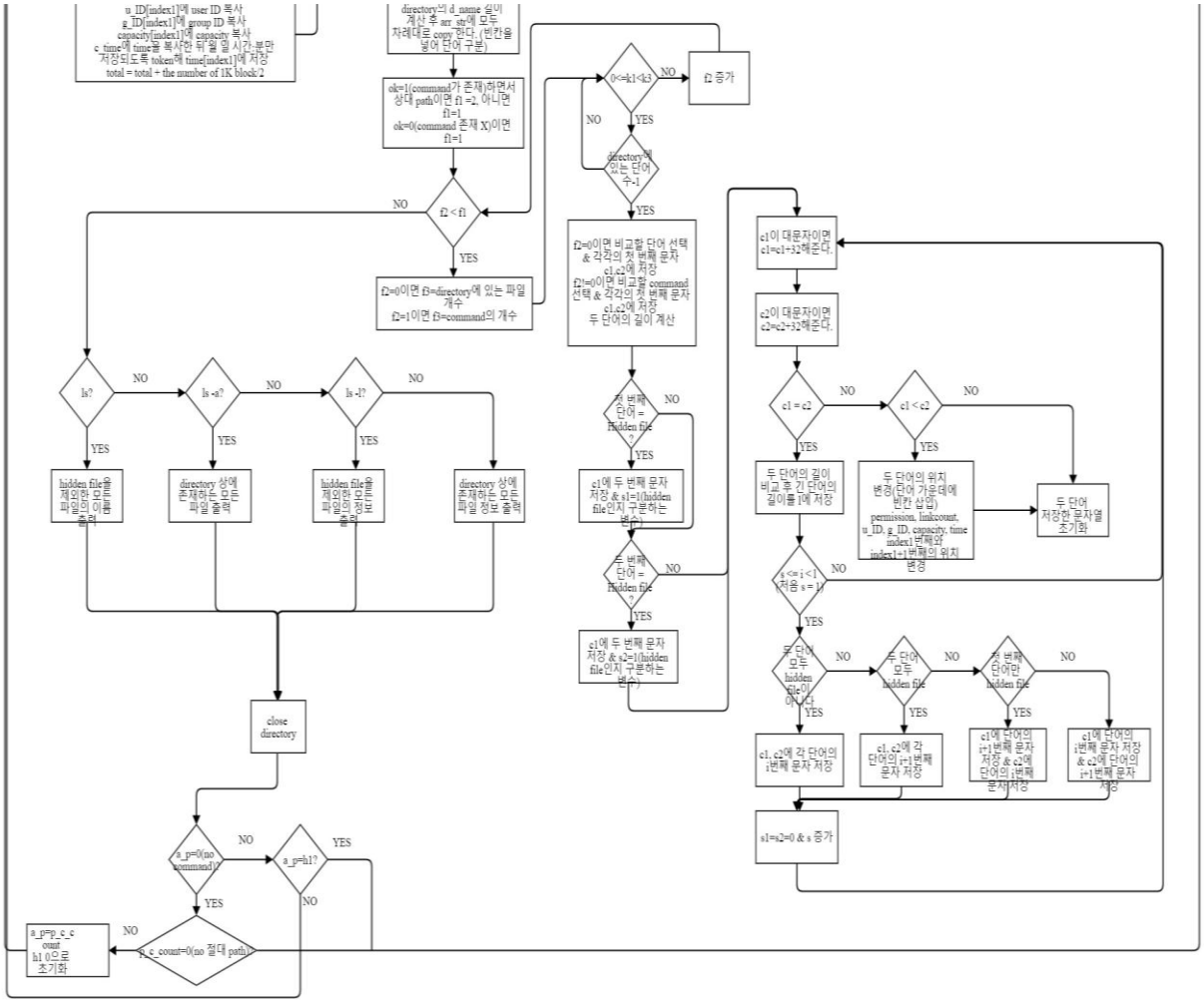
1)



2)


read directory
c_time 배열 초기화
a_p=0(절대 path X)?
NO
YES
path에 현재 directory path 저장
p_c_count=0(./&../&..&절대 pahat&~ X)?
NO
YES
path에 command[h1] 저장
path에 현재 directory path 저장
command_p_c[h1] = '/'
YES
NO
path 배열에서 마지막 전까지 t_path에 저장 후 path에 t_path 저장
info에 path을 copy 후 /와 dir->d_name을 차례로 덧붙인다.
permission[index1][0]에 dir->d_name이 파일이면 -, directory면 d 저장
permission [index1]에 user, group,other의 읽기 권한(r), 쓰기 권한(w), 실행 권한(x)을 저장
linkcount[index1]에 link counter 복사
u_ID[index1]에 user ID 복사
g_ID[index1]에 group ID 복사
capacity[index1]에 capacity 복사
c_time에 time을 복사한 뒤 월 일 시간-분만 저장되도록 token해 time[index1]에 저장
total = total + the number of 1K block/2
hidden file?
NO
YES
h_total = h_total + the number of 1K block/2
directory의 d_name 길이 계산 후 arr_str에 모두 차례대로 copy한다. (빈칸을 넣어 단어 구분)
ok=1(command가 존재)하면서 상대 path이면 f1 =2, 아니면 f1=1
ok=0(command 존재 X)이면 f1=1
0<=k1<k3
NO
f2 증가
NO
YES
directory에 있는 단어

3)


u_ID[index1]에 user ID 복사
g_ID[index1]에 group ID 복사
capacity[index1]에 capacity 복사
c_time에 time을 복사한 뒤 월 일 시간-분만 저장되도록 token해 time[index1]에 저장
total = total + the number of 1K block/2
directory의 d_name 길이 계산 후 arr_str에 모두 차례대로 copy한다. (빈칸을 넣어 단어 구분)
ok=1(command가 존재)하면서 상대 path이면 f1 =2, 아니면 f1=1
ok=0(command 존재 X)이면 f1=1
f2 < f1
NO
YES
f2=0이면 f3=directory에 있는 파일 개수
f2=1이면 f3=command의 개수
0<=k1<k3
NO
f2 증가
NO
YES
directory에 있는 단어 수-1
YES
f2=0이면 비교할 단어 선택 & 각각의 첫 번째 문자 c1,c2에 저장
f2!=0이면 비교할 command 선택 & 각각의 첫 번째 문자 c1,c2에 저장
두 단어의 길이 계산
첫 번째 단어 = Hidden file ?
NO
YES
c1에 두 번째 문자 저장 & s1=1(hidden file인지 구분하는 변수)
두 번째 단어 = Hidden file ?
NO
YES
c1에 두 번째 문자 저장 & s2=1(hidden file인지 구분하는 변수)
c1이 대문자이면 c1=c1+32해준다
c2이 대문자이면 c2=c2+32해준다.
c1 = c2
NO
YES
두 단어의 길이 비교 후 긴 단어의 길이를 l에 저장
c1 < c2
NO
YES
두 단어의 위치 변경(단어 가운데에 빈칸 삽입) permission, linkcount, u_ID, g_ID, capacity, time index1번째와 index1+1번째의 위치 변경
두 단어 저장한 문자열 초기화
s <= i < l (처음 s = 1)
NO
YES
두 단어 모두 hidden file인 아니다
NO
두 단어 모두 hidden file
NO
첫 번째 단어만 hidden file
NO
YES
c1, c2에 각 단어의 i번째 문자 저장
YES
c1, c2에 각 단어의 i+1번째 문자 저장
YES
c1에 단어의 i+1번째 문자 저장 & c2에 단어의 i번째 문자저장
YES
c1에 단어의 i번째 문자 저장 & c2에 단어의 i+1번째 문자 저장
s1=s2=0 & s 증가
ls?
NO
YES
hidden file을 제외한 모든 파일의 이름 출력
ls -a?
NO
YES
directory 상에 존재하는 모든 파일 출력
ls -l?
NO
YES
hidden file을 제외한 모든 파일의 정보 출력
directory 상에 존재하는 모든 파일 정보 출력
close directory
a_p=0(no command)?
NO
YES
a_p=h1?
YES
NO
p_c_count=0(no 절대 path)?
NO
YES
a_p=p_c_count
h1 0으로 초기화

◆ **Pseudo code**

```
while(receive option in flag){
        switch statement which is standard flag{
        if flag is a:
                aflag is 1;
                exit switch statement
        if flag is l:
                lflag is 1;
                exit switch statement
        if flag isn't a and l:
                end program
        }
    }
    for(i=optind;i<argc;i++){
        if argv[i] isn't ~ and dot and dot-dot and absolute path{
                command[co_count is command's order] is argv[i]
                increase co_count;
                relative path is exist
        }
        if argv[i] is ~ or dot or dot-dot or absolute path
                command_p_c[p_c_count is command_p_c's order] is argv[i]
                increase p_c_count;
        }
        argv[i] is exist
    }
    if ~ or absolute path or dot or dot-dot
```

```
        a_p is p_c_count
if p_c_count isn't 0 (if ~ or dot or dot-dot or absolute path exist){
        for(k1=0;k1<p_c_count;k1++){
        for(k2=0;k2<p_c_count-1;k2++){
                initialize first command's length and second command's length
                while(calculate first command's length until array is NULL)
                        increase variable indicate first command's length
                while(calculate second command's length until array is NULL)
                        increase variable indicate second command's length
                if first command's length is less than second command's length
                        l is second command's length
                if first command's length is more than second command's length
                        l is first command's length
                for(i=0;i<l;i++){ // compare commands' character
                        if first command and second command are same
                                continue for statement
                        if first command is more than second command
                                tmp is first command
                                first command is second command
                                second command is tmp
                                exit for statement
                        }
                        if first command is less than second command
                                exit for statement
                }
        }
```

```
                }
        }
while(infinite loop){
        initialize integer variables
        initialize arr_str
        for(i=0;i<1000;i++){
                initialize 2D arrays indicated permission, user ID, group ID, time
        }
        if a_p is 0 (if relative path)
                open directory(.)
        if a_p isn't 0 (if ~ or absolute path or dot or dot-dot)
                open directory(command_p_c's string)
        while(read directory){
                initialize c_time array
                if a_p is 0 (if relative path)
                        receive current directory path
                if a_p isn't 0 (if ~ or absolute path or dot or dot-dot)
                        if p_c_count is 0  (if relative path)
                                path is command[h1]
                        if p_c_count isn't 0 (if ~ or absolute path or dot or dot-dot)
                                initialize t_path array
                                path is command_p_c[h1]
                                if command_p_c[h1] is ..
                                        while(count path's length until path's character is NULL)
                                                increase integer variable indicated path's length
                                        for(k1=0;k1<s_l;k1++){ // find position of last /
```

```
                                        if path's kth character is '/' & path's (k+1)th isn't
NULL
                                                k2 is k1;
                                }
                                for(k1=0;k1<k2;k1++)
                                        t_path's k1th character is path's k1th character
                                path is t_path
                        }
                }
        }
        info is path
        add '/' behind info
        add dir->d_name behind info
        return information about info
        switch statement which is standard sta.st_mode and S_IFMT{
                if regular file:
                        permission[index1][index2++] is '-'
                        exit switch statement
                if directory:
                        permission[index1][index2++] is 'd';
                        exit switch statement
                if no argument:
                        exit switch statement
        }
        for(i=0;i<3;i++){
                if there is read permission
```

```
                permission[index1][index2++] is 'r'
        if there isn't read permission
                permission[index1][index2++] is '-'
        if there is write permission
                permission[index1][index2++] is 'w'
        if there isn't write permission
                permission[index1][index2++] is '-'
        if there is execute permission
                permission[index1][index2++] is 'x'
        if there isn's executer permission
                permission[index1][index2++] is '-'
}
permission[index1][index2] is '₩0';
linkcounter[index1] has information about link counter of file
u_ID[index1] has information about user ID of file
g_ID[index1] has information about group ID of file
capacity[index1 has information about capacity of file
c_time has information about time of file
t_time has word of c_time before blank or ₩t or :
t_time has word of NULL before blank or ₩t or :
time[index1] is t_time
while(rotate until t_time isn't NULL){
        t_time has word of NULL before blank or ₩t or
        if number is less than 3{
                if number is 2
                        add ':' behind time[index1]
```

```
                    if number isn't 2
                        add blank behind time[index1]
                    add t_time behind time[index1]
            }
            increase number
        }
        total is total plus file's the number of 1K blocks divides by 2
        if dir->d_name is hidden file
            h_total is h_total plus file's the number of 1K blocks divides by 2
        while(calculate d_name's length until d_name's character isn't NULL)
            increase variable indicated d_name's length
        for(i =0; i < d_name's length; i++)
            store directory's name in array one by one
        store blank in array
        count word
        overall length is overall length plus d_name's length plus 1
        increase index1
    }
if there are commands{
        if there is relative path
            f1 is 2
        if there is an absolute path or dot or dot-dot
            f1 is 1
    }
    if there aren't commands
        f1 is 1
```

```
while(if f2 is less than f1, rotate while statement){
if f2 is 0
        f3 is files' counter
if f2 isn't 0
        f3 is commands' counter
for(k1=0;k1<f3;k1++){
initialize c and j and index1(c and j are starting points which is words to compare)
for(k2=0;k2<f3-1;k2++){
        initialize s1 and s2(variable to check whether hidden file or not)
        if f2 is 0{
        for(count=0;count<2;count++)      // find words to compare
        {
                for(i=j,s=0;i<overall length;i++,s++){
                        if array's character is blank
                                exit loop
                        if array's character is character{
                                if count is 0{
                                        c1 is first word's first character
                                        w1 is first word stored in array
                                }
                                if count is 1{
                                        c2 is second word's first character
                                        w2 is second word stored in array
                                }
                        }
                }
```

```
        j is i plus 1
}
}
if f2 isn't 0{
        w1 is first command
        c1 is w1's first character
        w2 is second command
        c2 is w2's first character
}
initialize l1 and l2 (l1 is first word's length & l2 is second word's length
while(calculate first word's length until array is NULL)
        increase variable indicated first word's length
while(calculate second word's length until array is NULL)
        increase variable indicated second word's length
if first word's first character is '.' {
        if first word's length is 2{
                if first word's second character is '.'{
                        c1 is first word's second character
                        s1 is 1
                }
        }
        if first word's length isn't 2{
                c1 is first word's second character
                s1 is 1
        }
}
```

```
if second word's first character is '.'{
        if second word's length is 2{
                if(second word's second character is '.'){
                        c2 is second word's second character
                        s2 is 1
                }
        }
        if second word's length isn't 2
                c2 is second word's second character
                s2 is 1
        }
}
initialize s(start point to compare words)
while(infinite loop){
        if c1 is capital letter
                change capital letter to small letter
        if c2 is capital letter
                change capital letter to small letter
        if c1 and c2 are different
                exit infinite loop
        if c1 and c2 are same{
                if first word's length is bigger than second word's length
                        l is first word's length
                if second word's length is bigger than first word's length
                        l is second word's length
                for(i=s;i<l;i++){
```

```
                    if first word and second word aren't hidden file{
                        c1 is first word's ith character
                        c2 is second word's ith character
                    }
                    if first word and second word are hidden file{
                        c1 is first word's (i+1)th character
                        c2 is second word's (i+1)th character
                    }
                    if first word is hidden file and second word isn't hidden file{
                        c1 is first word's (i+1)th character
                        c2 is second word's ith character
                    }
                    if first word is hidden file and second word isn't hidden file {
                        c1 is first word's ith character
                        c2 is second word's (i+1)th character
                    }
                    if c1 and c2 are different
                        exit loop
                }
                initialize s1 and s2(variables to check whether hidden file or not)
            }
            increase start point
        }
        if first word's character is bigger than second word's character{
            if f2 is 0{
            for(i=c+l2+1,s=0;i<=c+l1+l2;i++,s++)
```

```
                change first word's position at second word's position
        for(i=c,s=0;i<c+l2;i++,s++)
                change second word's position at first word's position
        store blank between words
        j is j minus first word's length plus 1
        switch index1th word of permission with (index1+1)th word of permission
        switch index1th link counter with (index1+1)th link counter
        switch index1th word of user ID with (index1+1)th word of user ID
        switch index1th word of group ID with (index1+1)th word of group ID
        switch index1th capacity with (index1+1)th capacity
        switch index1th word of time with (index1+1)th word of time
        }
        if f2 isn't 0
                // change command's position
                switch (number)th command with (number+1)th command
        }
    }
    if first word's character is less than second word's character
        j is j minus second word's length plus 1
    c is j
    for(i=0;i<l1;i++)
        initialize first word
    for(i=0;i<l2;i++)
        initialize second word
    increase index1
}
```

```
}
        increase f2
}
if lflag is 1(use option –l)
        if there aren't commands
                print Directory path
                if aflag is 0(not use option –a)
                        print the number of 1K blocks
                if aflag is 1(use option –a)
                        print the number of 1K blocks
        }
        if there are commands
                if a_p is more than 0(absolute path exist){
                        print command_p_c[h1]
                        if aflag is 0(not use option –a)
                                print the number of 1K blocks
                        if aflag isn't 0(use option –a)
                                print the number of 1K blocks
                }
        }
}
if lflag isn't 1(not use option –l)
        if a_p is more than 0(absolute path exist)
                print command_p_c[h1]
}
initialize integer variables j and index1(order of arrays)
```

```
for(i=0;i<a_len;i++){
        if ith character of arr_str is blank{
                if both aflag and lflag are 0(no option(ls)){
                        if 0th character of print_arrprint_arr isn't '.'{
                                if there are commands
                                if a_p is 0(relative path){
                                for(number=0;number<co_count;number++){
                                if command[number] and print_arr are same
                                                print print_arr
                                }
                                }
                                if a_p isn't 0(absolute path){
                                        print print_arr
                                }
                                if there aren't commands
                                        print print_arr
                        }
                }
                if aflag is 1 and lflag is 0 (exist option-a(ls -a)){
                                if there are commands{
                                if a_p is 0(relative path){
                                        for(number=0;number<co_count;number++){
                                                if command[number] and print_arr are same
                                                        print print_arr
                                        }
                                }
```

```
                    if a_p isn't 0(absolute path)
                        print print_arr
                }
                if there aren't commands
                    print print_arr
            }
            if aflag is 0 and lflag is 1 (exist option-l(ls -l)){
                if 0th character of print_arrprint_arr isn't '.'{
                    if there are commands
                    if a_p is 0(relative path){
                    for(number=0;number<co_count;number++){
                    if command[number] and print_arr are same
                        print permission[index1], linkcounter[index1],
u_ID[index1], g_ID[index1], capacity[index1], time[index1], print_arr
                    }
                    }
                    if a_p isn't 0(absolute path){
                        print permission[index1], linkcounter[index1],
u_ID[index1], g_ID[index1], capacity[index1], time[index1], print_arr
                    }
                    if there aren't commands
                        print permission[index1], linkcounter[index1],
u_ID[index1], g_ID[index1], capacity[index1], time[index1], print_arr
                }
            }
            if both aflag and lflag are 1(option –l and -a(ls -al)){
```

```
                    if there are commands{
                        if a_p is 0(relative path){
                            for(number=0;number<co_count;number++){
                                if command[number] and print_arr are same
                                    print permission[index1],
linkcounter[index1], u_ID[index1], g_ID[index1], capacity[index1], time[index1], print_arr
                            }
                        }
                        if a_p isn't 0(absolute path)
                            print permission[index1], linkcounter[index1],
u_ID[index1], g_ID[index1], capacity[index1], time[index1], print_arr
                    }
                    if there aren't commands
                        print permission[index1], linkcounter[index1], u_ID[index1],
g_ID[index1], capacity[index1], time[index1], print_arr
                }
                for(k=0;k<j;k++)
                    kth character of print_arr is NULL
                initialize j
                increase index1
            }
            if ith character of arr_str isn't blank{
                jth character of print_arr is ith character of arr_str
                increase j
            }
        }
```

```
printf enter
close directory
increase h1
if a_p is 0{
        if p_c_count is 0(no absolute command)
                exit while statement
        if p_c_count isn't 0(exist absolute command){
                a_p is p_c_count
                initialize h1
                continue while statement
        }
}
if a_p isn't 0{
        if a_p is h1
                exit while statement
}
}
```

◆ **Reference**

시스템 프로그래밍 강의자료: 4.+time+and+date

◆ **Conclusion**

이번 과제에서는 directory 상에 존재하는 directory와 file의 정보(permission, user ID, group ID, capacity, link count, time)을 받기 위해 여러 함수와 구조체 변수를 사용해야 됐다. 이때 함수는 시스템 프로그래밍 실습과 시스템 프로그래밍의 강의자료를 보면 알 수 있어 무슨 함수를 가지고 구현해야 되는지 알기 쉬웠다. 하지만 어떤 구조체 변수를 사용해야 되는지 헷갈렸다. 또한 어떨 때 예외처리를 해야 하는지 몰라 리눅스 상에서 ls를 해봐 하나씩 비교하면서 같은 결과가 나오도록 구현했다.