System Programming Report

Assignment 3-1 – HTML_ls

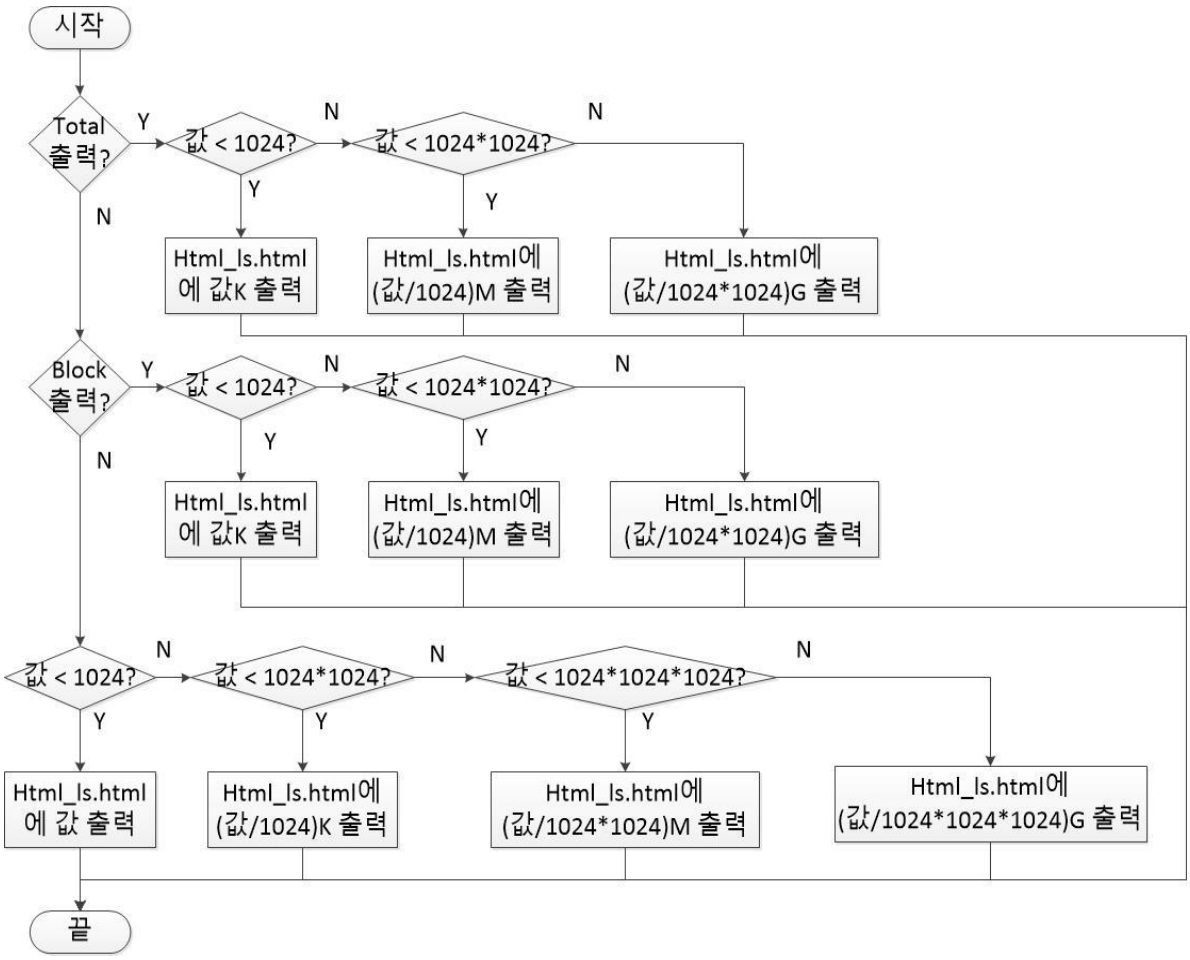
Professor	황호영 교수님
Department	Computer engineering
Student ID	2014722057
Name	김 진아
Class	설계 (화6 목4) / 실습 (금 56)
Date	2016. 4. 29

◆ **Introduction**

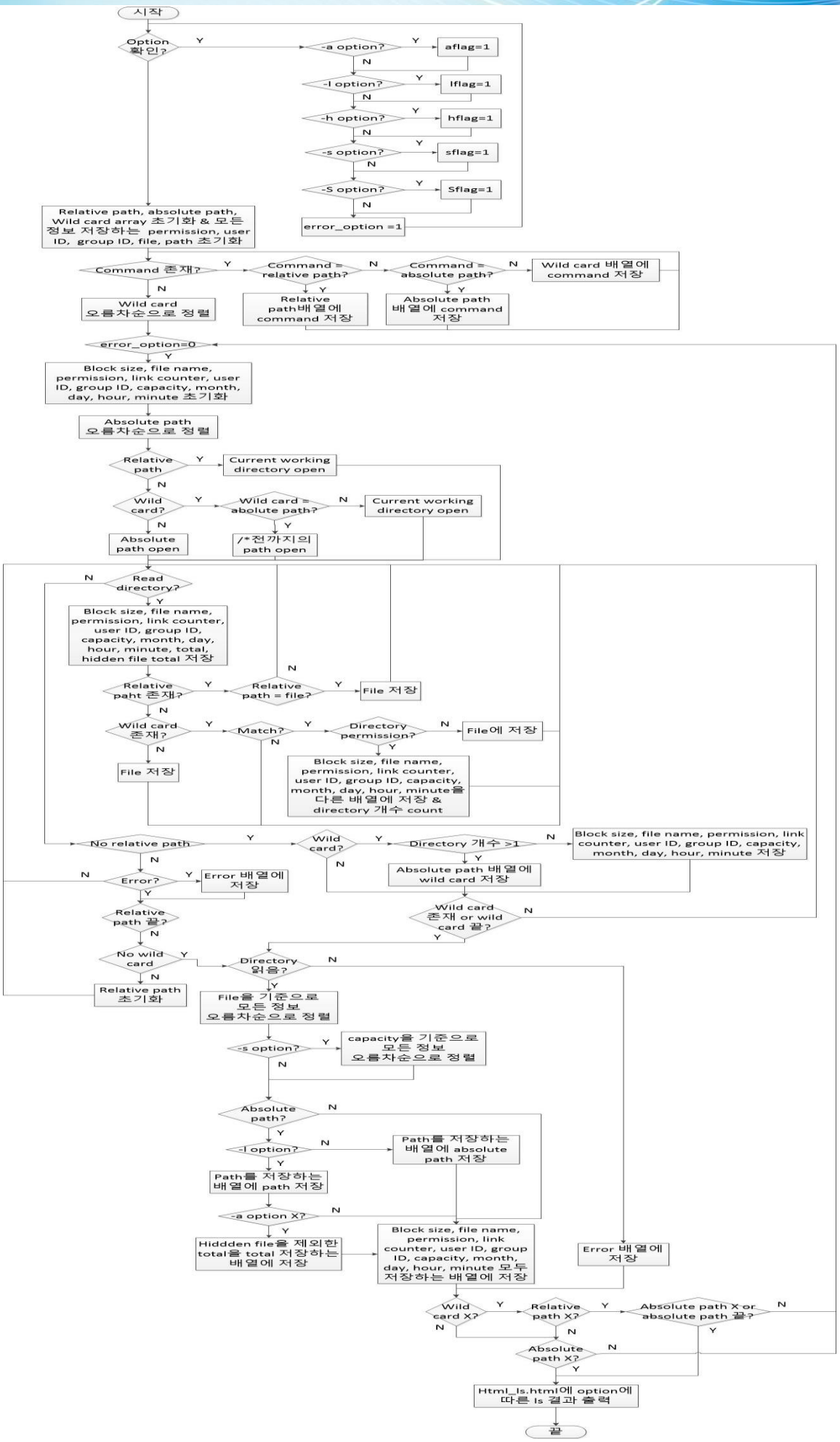
이번 과제는 저번 과제에서 만든 spls_final을 사용해 HTML document를 만드는 것이다. HTML document에 current directory가 title, command가 heading에 출력한다. ls 명령어의 결과를 table로 만들어 출력한다. 또한 file name에 hyperlink를 해 file로 이동하게 만든다. 마지막으로 output file name을 html_ls.html로 한다. 이때 처음 ls 명령어를 했을 때 html_ls.html가 출력되지 않도록 한다.

◆ **Flowchart**

1) print_h function



2) main function


시작
Option 확인?
-a option?
aflag=1
-l option?
lflag=1
-h option?
hflag=1
-s option?
sflag=1
-S option?
Sflag=1
error_option =1
Relative path, absolute path, Wild card array 초기화 & 모든 정보 저장하는 permission, user ID, group ID, file, path 초기화
Command 존재?
Command = relative path?
Command = absolute path?
Wild card 배열에 command 저장
Wild card 오름차순으로 정렬
Relative path배열에 command 저장
Absolute path 배열에 command 저장
error_option=0
Block size, file name, permission, link counter, user ID, group ID, capacity, month, day, hour, minute 초기화
Absolute path 오름차순으로 정렬
Relative path
Current working directory open
Wild card?
Wild card = abolute path?
Current working directory open
Absolute path open
/*전까지의 path open
Read directory?
Block size, file name, permission, link counter, user ID, group ID, capacity, month, day, hour, minute, total, hidden file total 저장
Relative paht 존재?
Relative path = file?
File 저장
Wild card 존재?
Match?
Directory permission?
File에 저장
File 저장
Block size, file name, permission, link counter, user ID, group ID, capacity, month, day, hour, minute을 다른 배열에 저장 & directory 개수 count
No relative path
Wild card?
Directory 개수 >1
Block size, file name, permission, link counter, user ID, group ID, capacity, month, day, hour, minute 저장
Absolute path 배열에 wild card 저장
Error?
Error 배열에 저장
Wild card 존재 or wild card 끝?
Relative path 끝?
No wild card
Directory 읽음?
Relative path 초기화
File을 기준으로 모든 정보 오름차순으로 정렬
-s option?
capacity을 기준으로 모든 정보 오름차순으로 정렬
Absolute path?
-l option?
Path를 저장하는 배열에 absolute path 저장
Path를 저장하는 배열에 path 저장
-a option X?
Hiddden file을 제외한 total을 total 저장하는 배열에 저장
Block size, file name, permission, link counter, user ID, group ID, capacity, month, day, hour, minute 모두 저장하는 배열에 저장
Error 배열에 저장
Wild card X?
Relative path X?
Absolute path X or absolute path 끝?
Absolute path X?
Html_ls.html에 option에 따른 ls 결과 출력
끝

◆ **Pseudo code**

1) print_h function

```
if print total
        if value is less than 1024
            print value attached K at html_ls.html
        if value is less than 1024*1024
            print value/1024 attached M at html_ls.html
        if total is more than 1024*1024
            print value/(1024*1024) attached G at html_ls.html
    }
    if print block size{
        if value is less than 1024
            print value attached K at html_ls.html
        if value is less than 1024*1024
            print value/1024 attached M at html_ls.html
        if total is more than 1024*1024
            print value/(1024*1024) attached G at html_ls.html
    }
    if print capacity{
        if value is less than 1024
            print value at html_ls.html
        if value is less than 1024*1024
            print value/1024 attached K at html_ls.html
        if value is less than 1024*1024*1024
            print value/(1024*1024) attached M at html_ls.html
        if value is more than 1024*1024*1024
```

```
                print value/(1024*1024*1024) attached G at html_ls.html
    }
```

2) main function

```
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:
                error_option is 1
    }
    for(i=0;i<1000;i++)
        initialize relative, absolute, fnmatch array, permission,user ID, group ID, file, path
    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 read_d, 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 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{
                        iif abolute path is ".."{
```

```
                    initialize t_path
                    path is absolute's directory path
                    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
                }
                if absolute path is "."
                    path is current working directory
                if absolute path isn't ".." and "."
                    path is absolute 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 name 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{
                        save s_index(array's index to save) into num_path array
                        if lflag is 1(use option -l){
                                save path into s_path array
                                if aflag is 0 (not use option –a)
                                        total is total minus h_total
                                save s_index(array's index to save) into num_total array
                                save total into s_total array
                        }
                        if lflag is 0 (not use option –l){
                                if absolute path exist
                                        save absolute path into s_path array
                        }
                }
                save s_index(array's index to save) into num_start array
                for(i=0;i<index1;i++)
                        save file, permission, user ID, group ID, block size, link counter,
capacity, month, day, hour, minute into other arrays (file, permission, user ID, group ID, block size, link
counter, capacity, month, day, hour, minute)
                save s_index(array's index to save) into num_end array
        }
```

```
        if directory path doesn’t exist
                store absolute[a_order])’s path into error array
        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
        }
}
html_ls.html file open
if html_ls.html file open{
        print error
```

```
        end program
}
get current working directory
print current path at html_ls.html file as title
for(i=0;argv[i]!='₩0';i++)
        print argv[i] at html_ls.html file
if(error_option==1){
        print error option at html_ls.html file
        end program
}
for(i=0;i<e_len;i++)
        print error file or directory at html_ls.html file
initialize variable (j, s, k1, k2)
for(i=0;i<s_index;i++){
        initialize variable (ok)
        if print directory path
                print directory path at html_ls.html file
        if print total{
                if exist option -h
                        go to print_h function
                if not exist option –h
                        print total at html_ls.html file
                if total is 0
                        ok is 1
        }
        if make table and total isn't 0{
```

```
                if -s option exist
                        print Block Size at html_ls.html file
                print File Name at html_ls.html file
                if -l option exist
                        print Permission, Link, Owner, Group, Size, Last Modified at
html_ls.html file
        }
        if file's first letter isn't '/'(file isn't directory path) {
                if file is relative path
                        info is current working directory
                if path exist
                        info is path
                info is info/dir->d_name
        }
        if file's first letter is '/'
                info is s_file
        if no option -a(except hidden file)
                if file isn't hidden fil{
                        if –s option exist{
                                if not use option -h
                                        print block size at html_ls.html file
                                if use option -h
                                        go to print_h function
                        }
                        print file name at html_ls.html file and create hyperlink
                        if –l option exist{
```

print permission, link counter, user ID, group ID at html_ls.html file

if not use option -h

print capacity at html_ls.html

if use option -h

go to print_h function

print month, day, hour, minute at html_ls.html

}

}

}

if exist option -a(include hidden file)

if –s option exist{

if not use option -h

print block size at html_ls.html file

if use option -h

go to print_h function

}

print file name at html_ls.html file and create hyperlink

if –l option exist{

print permission, link counter, user ID, group ID at html_ls.html file

if not use option -h

print capacity at html_ls.html

if use option -h

go to print_h function

print month, day, hour, minute at html_ls.html

}

```
}
close File
```

◆ **Conclusion**

저번 과제 때까지 ls명령어를 받으면 바로 출력하도록 설계했다. 하지만 이번 과제에서는 그렇게 하면 html_ls.html파일이 나오게 되므로 다 저장한 뒤 file을 open해서 출력해야 했다. 이때 메모리 문제가 발생할거 같아 최대한 크게 잡아 출력할 때 필요한 정보를 다 저장하도록 만들었다.

total이 0일 때 표가 생성되지 않아야 하는데 생성되도록 했다. 그래서 total이 0이 아닐 때만 표가 생성되도록 조건을 줬다.