시스템프로그래밍실습

Assignment2-1 과제

수업 명 : 시스템프로그래밍실습
과제 이름 : assignment2-1
담당 교수님 : 김태석 교수님
학 번 : 2019202005
이 름 : 남종식

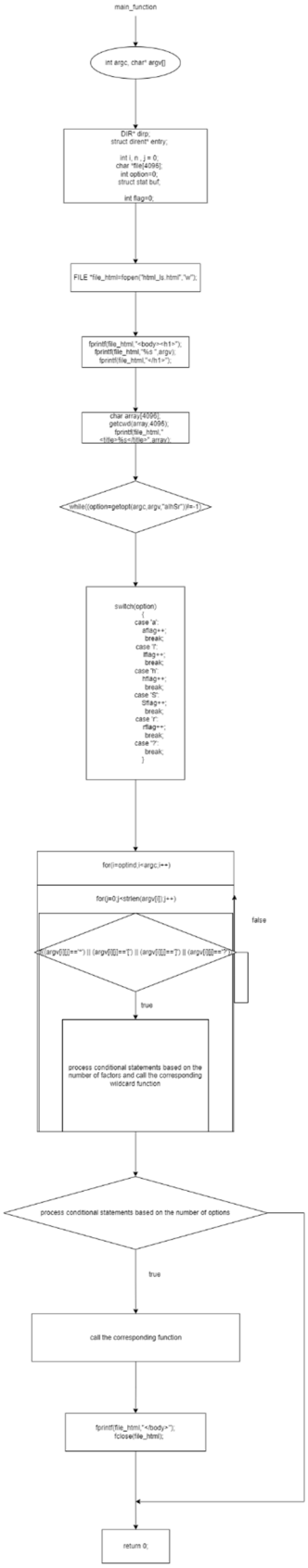
과제 소개

이번 과제는 저번과제에서 제출했던 final_ls.c 파일을 이용해서 html문서를 만드는 과제입니다. html문서를 만들기 위해서는 기존에 있던 코드에 파일을 생성하고 파일에 쓰기 위한 파일 포인터를 초기화하는 문장을 추가하면 됩니다. 이때 fopen함수를 사용하는데 함수의 첫 번째 인자는 파일의 경로와 이름을 포함한 문자열, 두 번째 인자는 파일 열기 모드입니다. `"w"`는 파일을 쓰기 모드로 바꿔줍니다. 저는 FILE *file_html=fopen("html_ls.html","w")를 추가하여 html_ls.htm파일을 생성하여 파일에 쓰기 위한 파일 포인터 file_html을 초기화 해주어 사용했습니다.

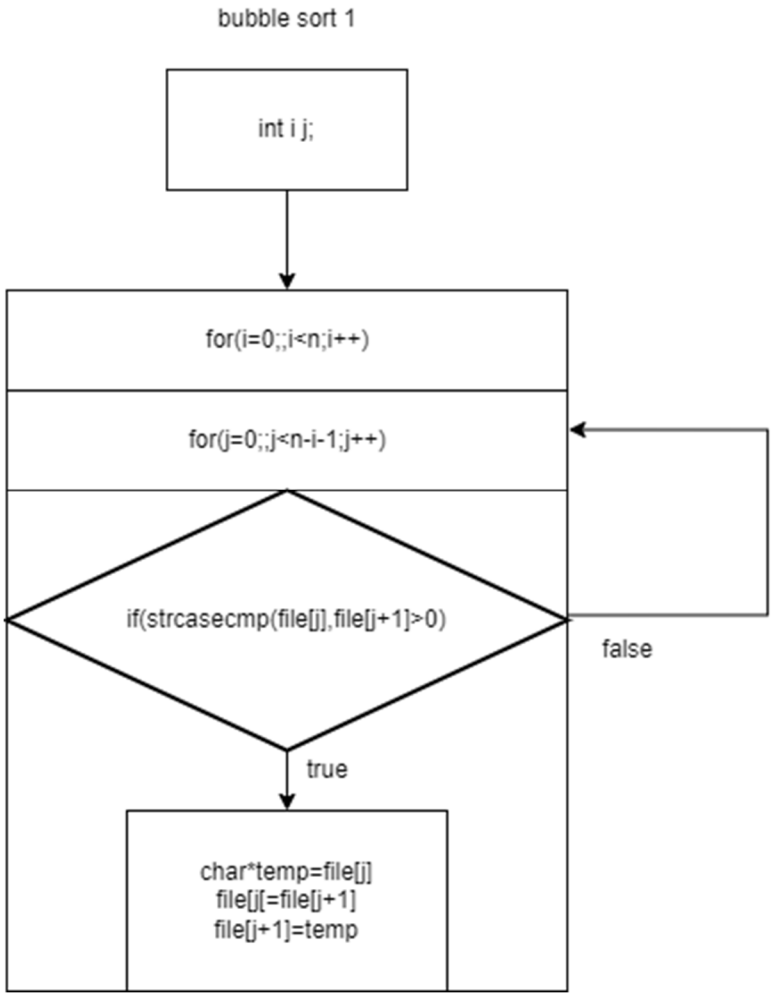
지금까지 진행한 ls명령어들을 html파일을 통해 직접 확인할 수 있으며 테이블로 파일들의 정보들을 묶어주었으며 디렉토리는 파란색, 링크 파일은 초록색이며 그 이 파일은 빨간색으로 나타내 주어야 합니다.

Flow Chart

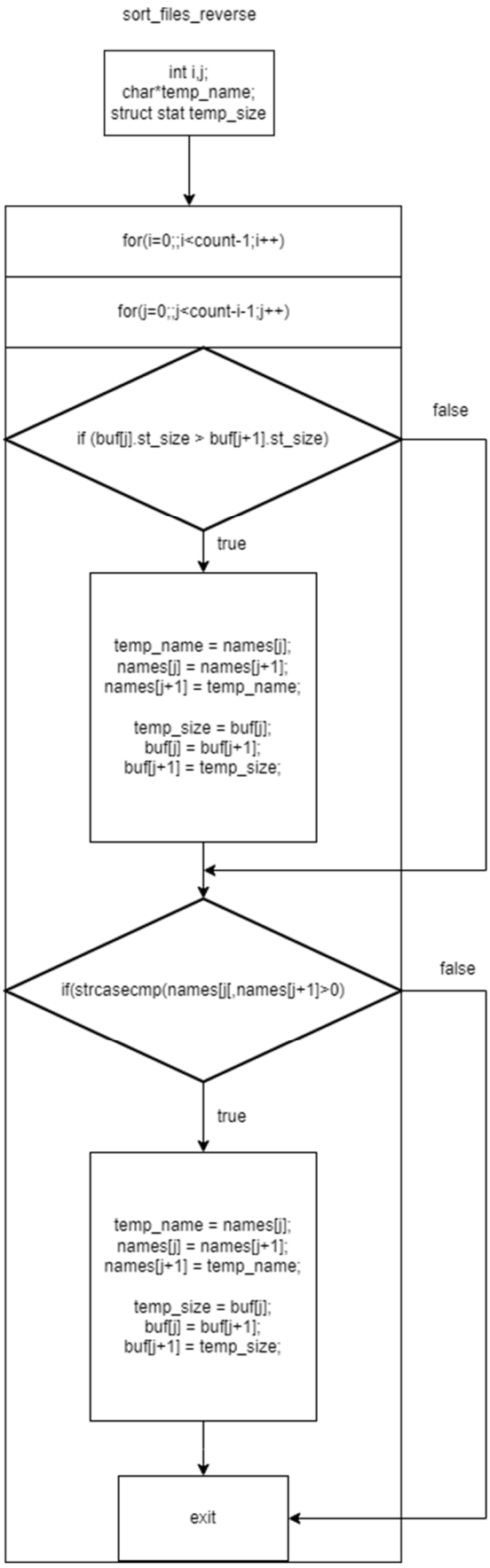
Main function



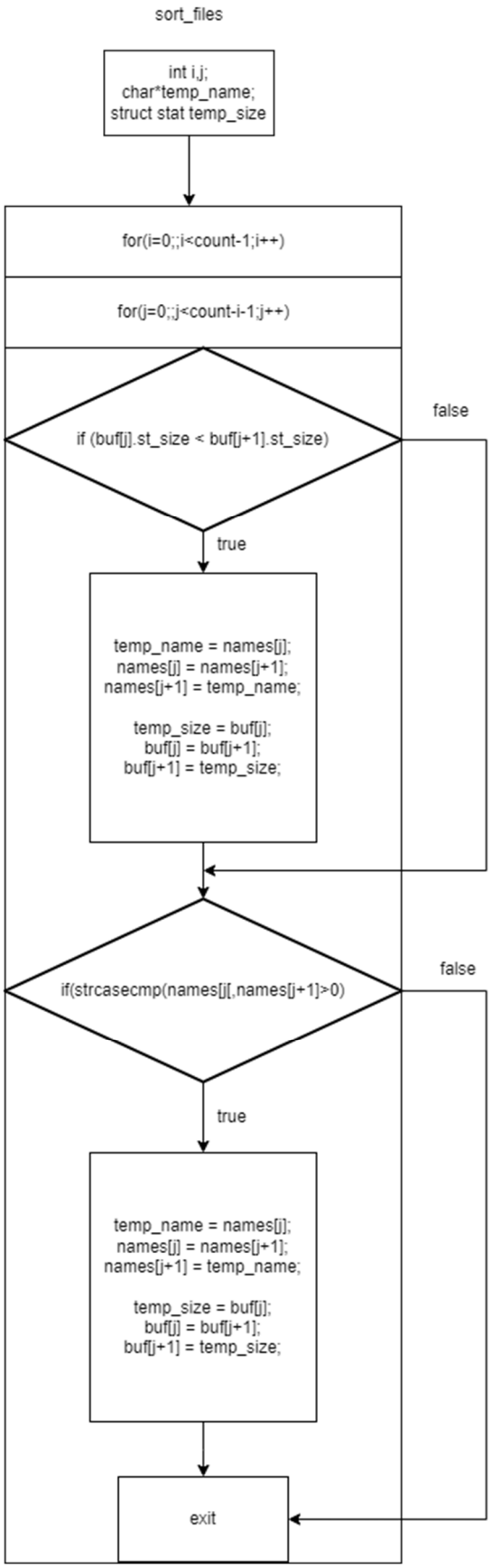
Bubble sort 1function



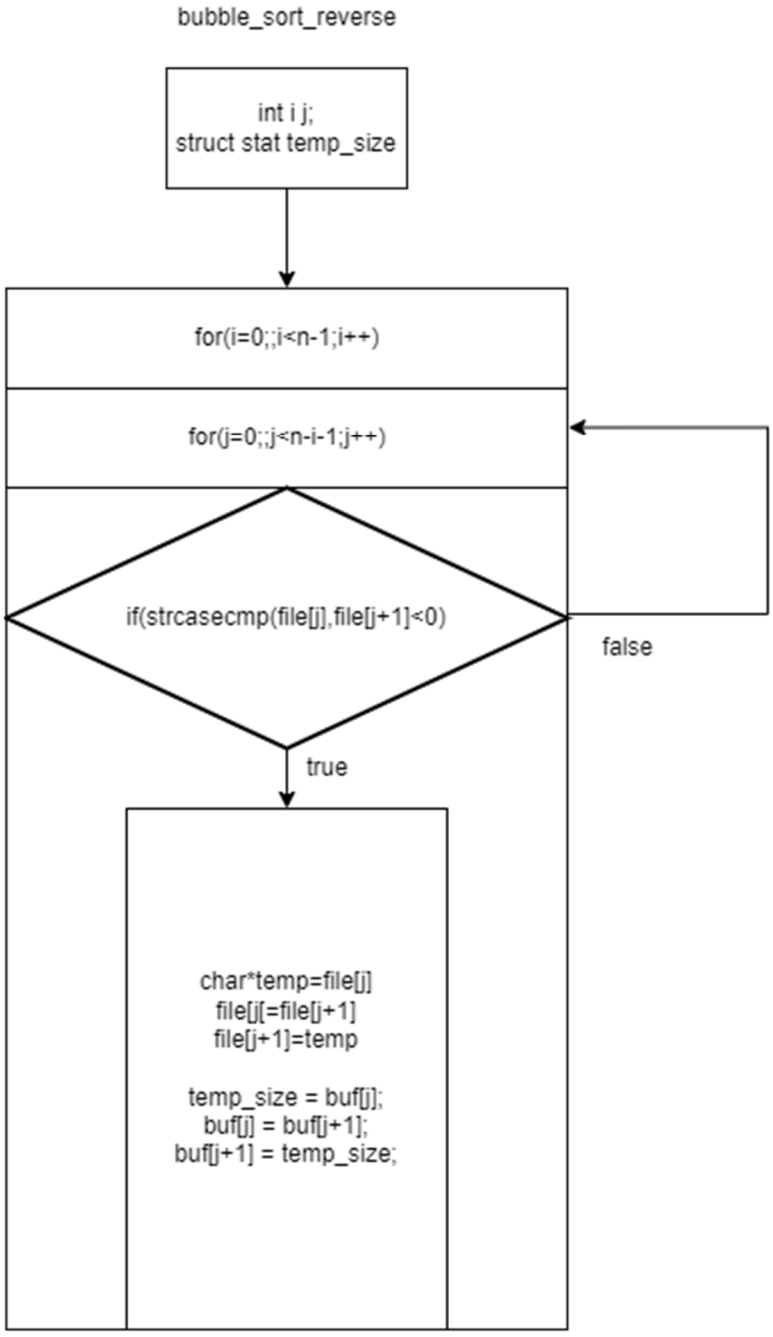
Sort_files_reverse



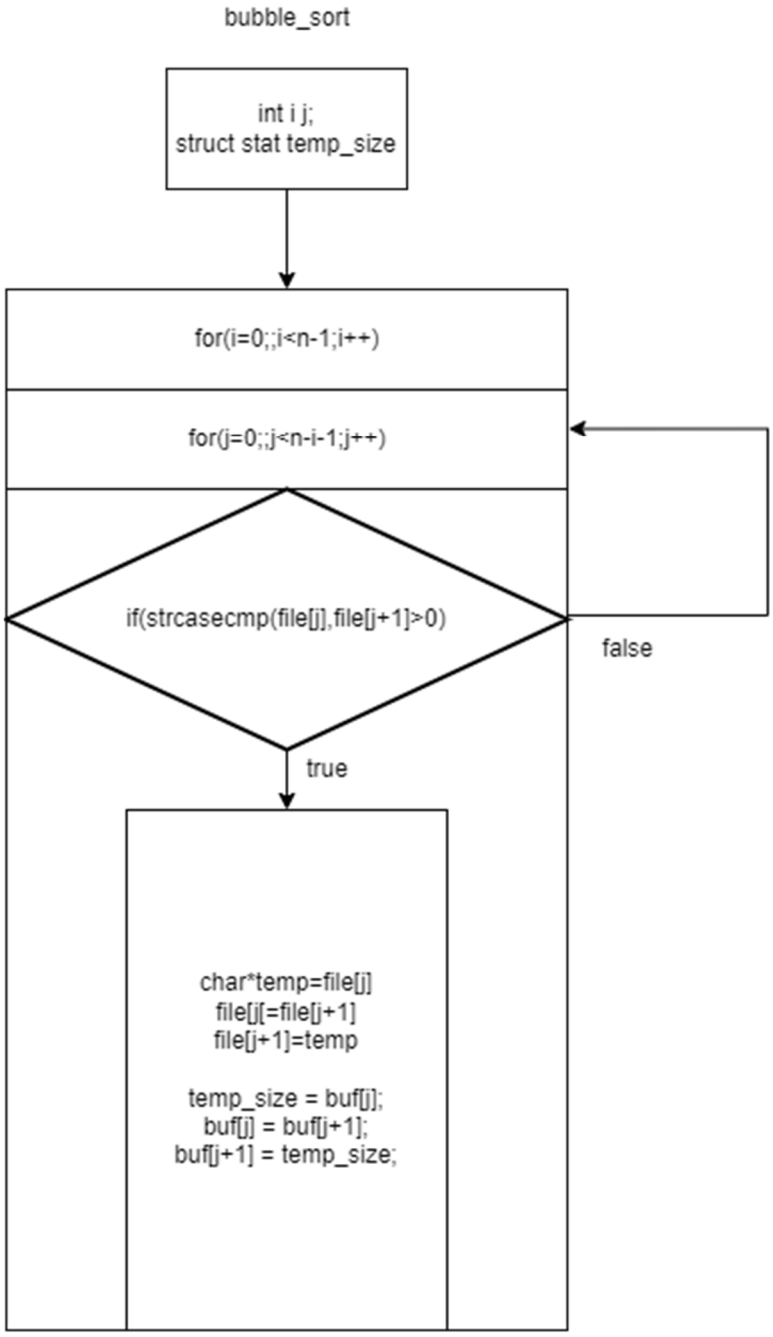
Sort_files


sort_files
int i,j;
char*temp_name;
struct stat temp_size
for(i=0;;i<count-1;i++)
for(j=0;;j<count-i-1;j++)
if (buf[j].st_size < buf[j+1].st_size)
false
true
temp_name = names[j];
names[j] = names[j+1];
names[j+1] = temp_name;
temp_size = buf[j];
buf[j] = buf[j+1];
buf[j+1] = temp_size;
if(strcasecmp(names[j[,names[j+1]>0)
false
true
temp_name = names[j];
names[j] = names[j+1];
names[j+1] = temp_name;
temp_size = buf[j];
buf[j] = buf[j+1];
buf[j+1] = temp_size;
exit

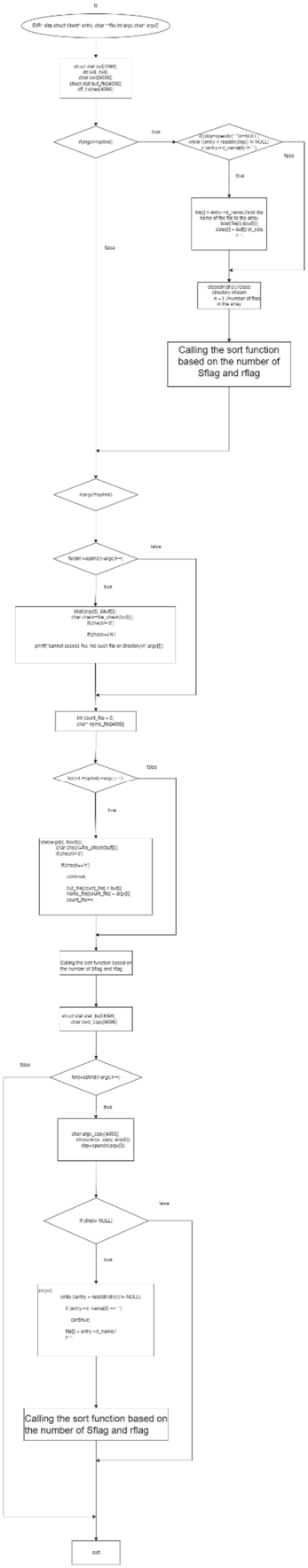
Bubble_sort_reverse



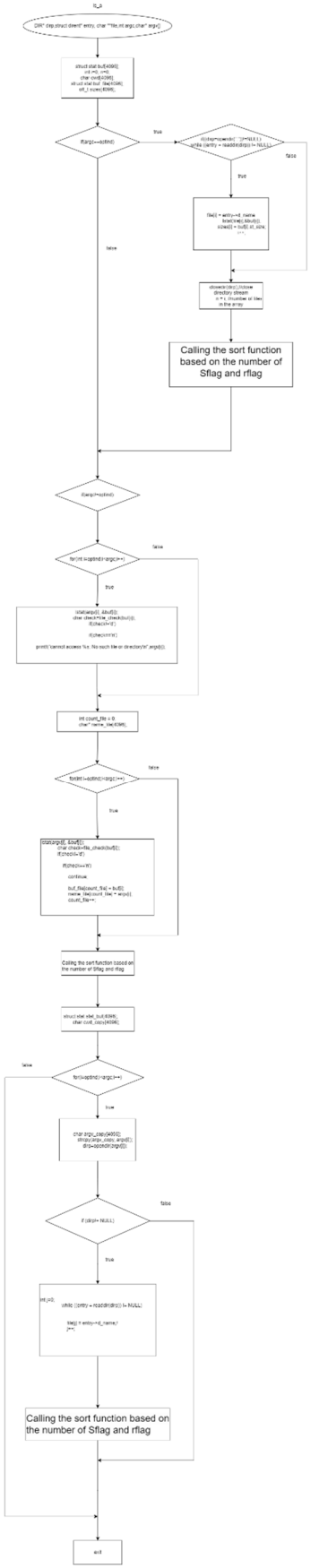
Bubble_sort


bubble_sort
int i j;
struct stat temp_size
for(i=0;;i<n-1;i++)
for(j=0;;j<n-i-1;j++)
if(strcasecmp(file[j],file[j+1]>0)
false
true
char*temp=file[j]
file[j[=file[j+1]
file[j+1]=temp
temp_size = buf[j];
buf[j] = buf[j+1];
buf[j+1] = temp_size;

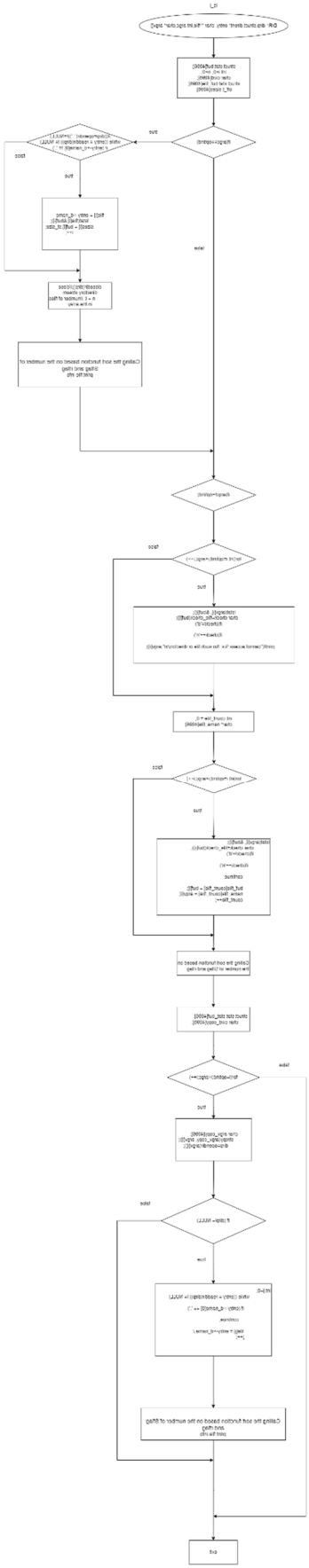
ls function


ls
Diff* dirp,struct dirent* entry, char **file,int argc,char* argv[]
struct stat buf[4096];
int i=0, n=0;
char cwd[4096];
struct stat buf_file[4096];
off_t size[4096];
if(argc==optind)
true
while ((entry = readdir(dirp)) != NULL)
if (entry->d_name[0] != '.')
false
true
file[i] = entry->d_name;//add the name of the file to the array
lstat(file[i],&buf[i]);
size[i] = buf[i].st_size;
i++;
false
closedir(dirp);//close directory stream
n = i; //number of files in the array
Calling the sort function based on the number of Sflag and rflag
if(argc!=optind)
for(int i=optind;i<argc;i++)
false
true
lstat(argv[i], &buf[i]);
char check=file_check(buf[i]);
if(check=='d')
if(check=='n')
printf("cannot access %s: No such file or directory\n",argv[i]);
int count_file = 0;
char* name_file[4096];
for(int i=optind;i<argc;i++)
false
true
lstat(argv[i], &buf[i]);
char check=file_check(buf[i]);
if(check=='d')
if(check=='n')
continue;
buf_file[count_file] = buf[i];
name_file[count_file] = argv[i];
count_file++;
Calling the sort function based on the number of Sflag and rflag
struct stat stat_buf[4096];
char cwd_copy[4096];
for(i=optind;i<argc;i++)
false
true
char argv_copy[4000];
strcpy(argv_copy, argv[i]);
dirp=opendir(argv[i]);
if (dirp!= NULL)
false
true
int j=0;
while ((entry = readdir(dirp)) != NULL)
if (entry->d_name[0] == '.')
continue;
file[j] = entry->d_name;
j++;
Calling the sort function based on the number of Sflag and rflag
exit

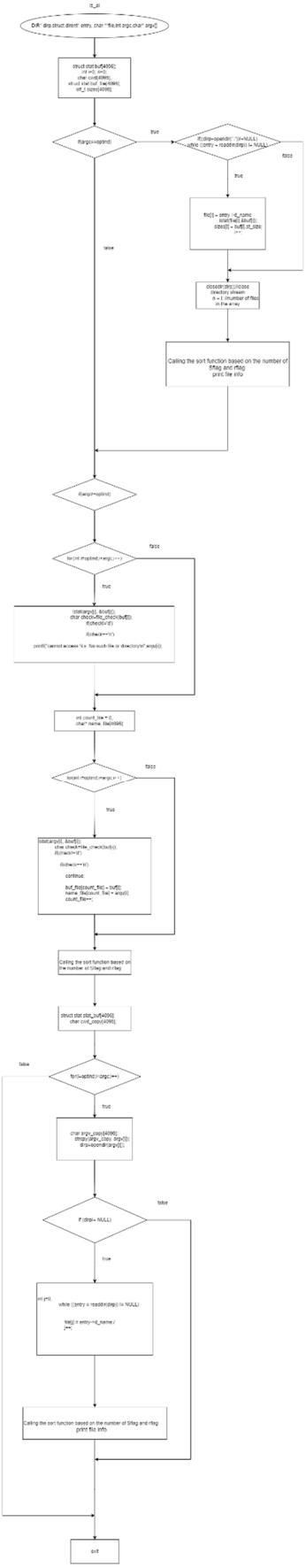
ls -a function



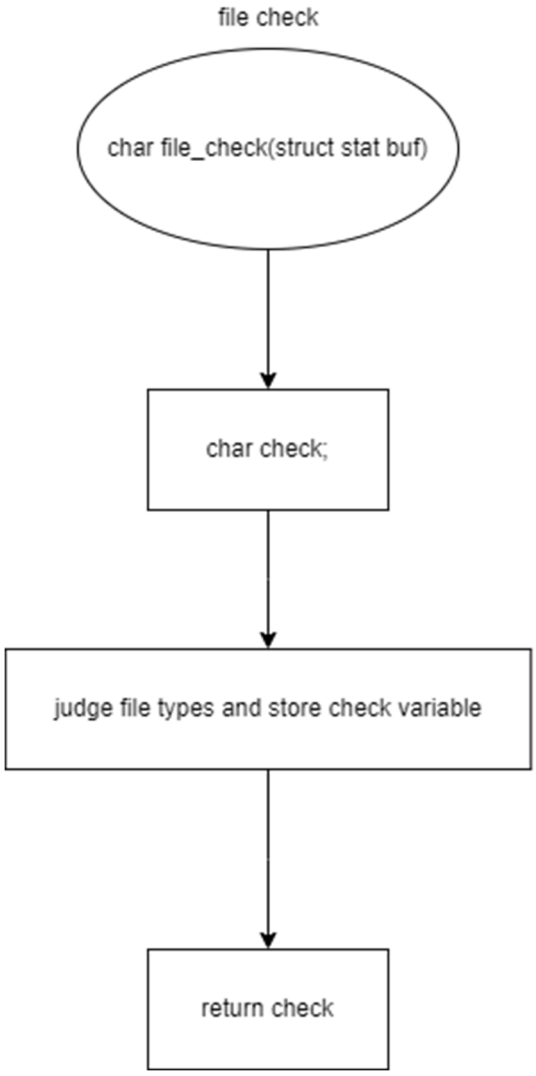
ls_l



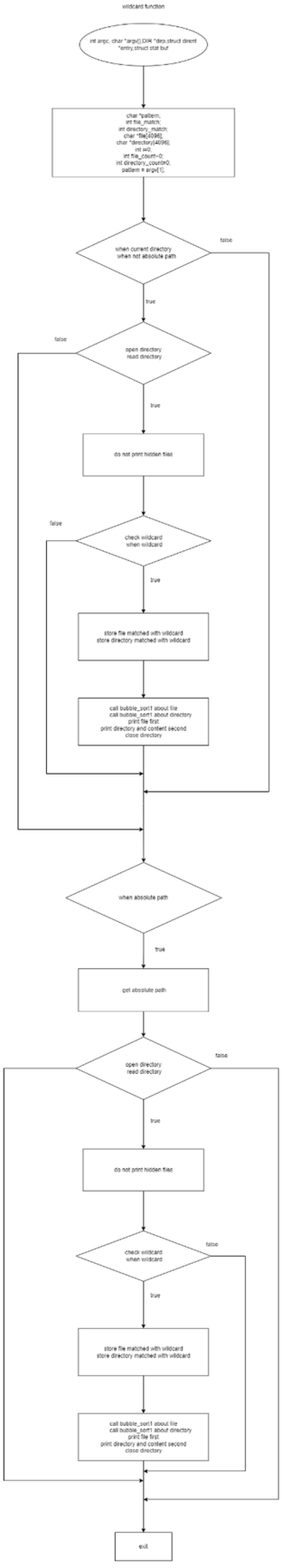
ls_al


ls_al
DIR* dirp,struct dirent* entry, char **file,int argc,char* argv[]
struct stat buff[4096];
int i=0, n=0;
char cwd[4096];
struct stat buf, fileb[4096];
off_t sizes[4096];
if(argc>optind)
true
if((dirp=opendir("."))!=NULL)
while ((entry = readdir(dirp)) != NULL)
false
true
false
file[i] = entry->d_name;
sizes[i] = buff[i].st_size;
i++;
closedir(dirp);//close directory stream
n = i; //number of files in the array
Calling the sort function based on the number of bflag and rflag
print file info
false
true
int count_file = 0;
char* name_file[4096];
false
true
struct stat stat_buff[4096];
char cwd_copy[4096];
false
true
char argv_copy[4096];
strcpy(argv_copy, argv[]);
dirp=opendir(argv[1]);
false
true
int j=0;
while ((entry = readdir(dirp)) != NULL)
file[j] = entry->d_name;
j++;
Calling the sort function based on the number of bflag and rflag
print file info
exit

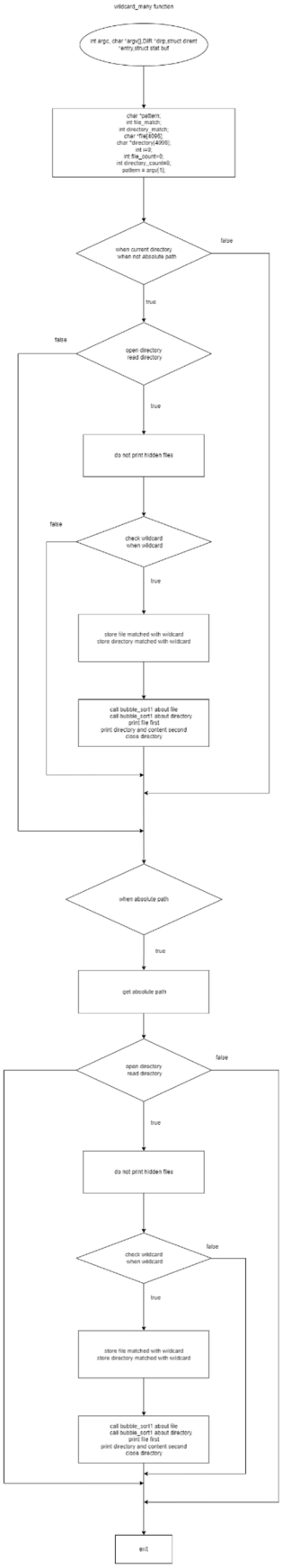
file check function



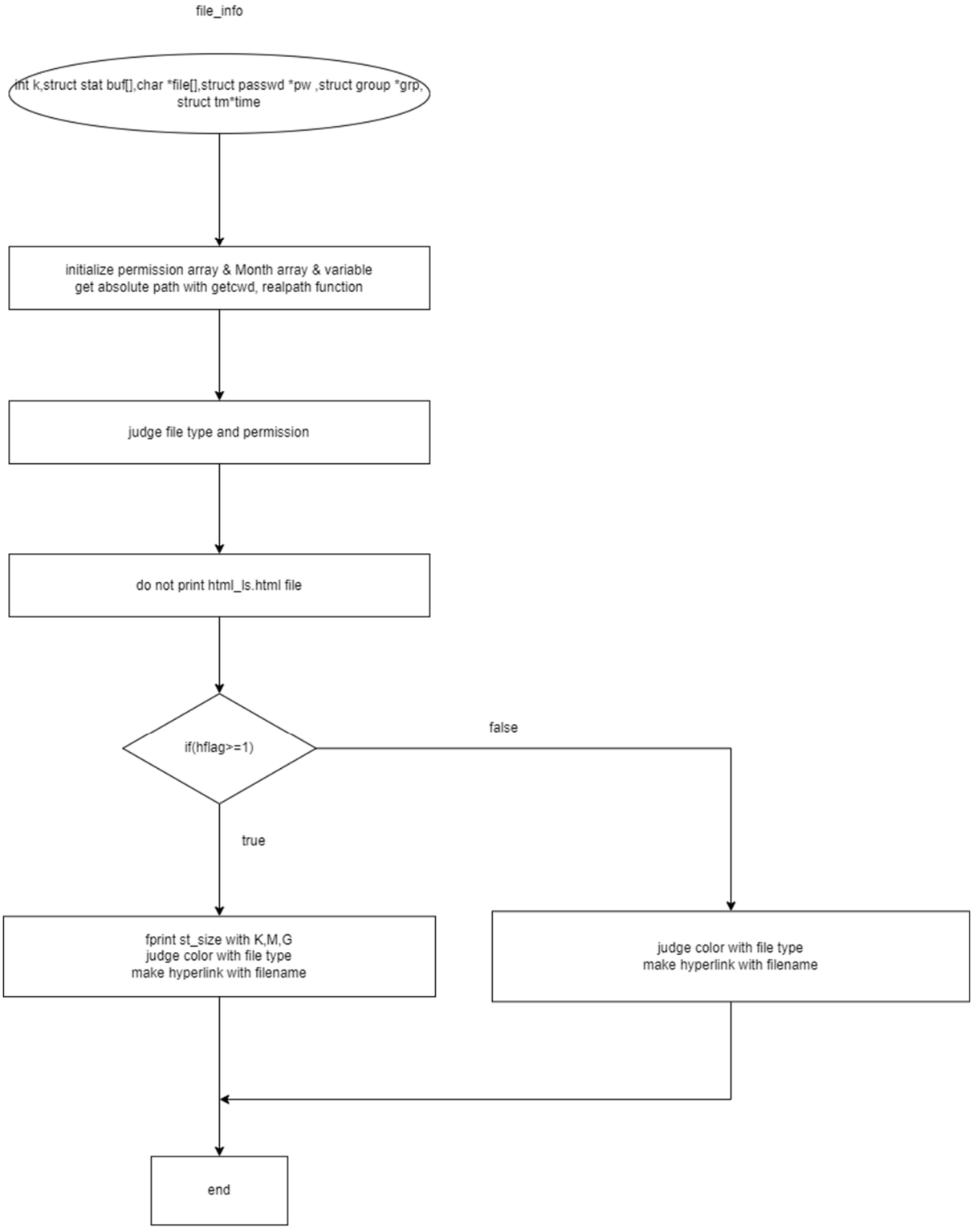
Wildcard function



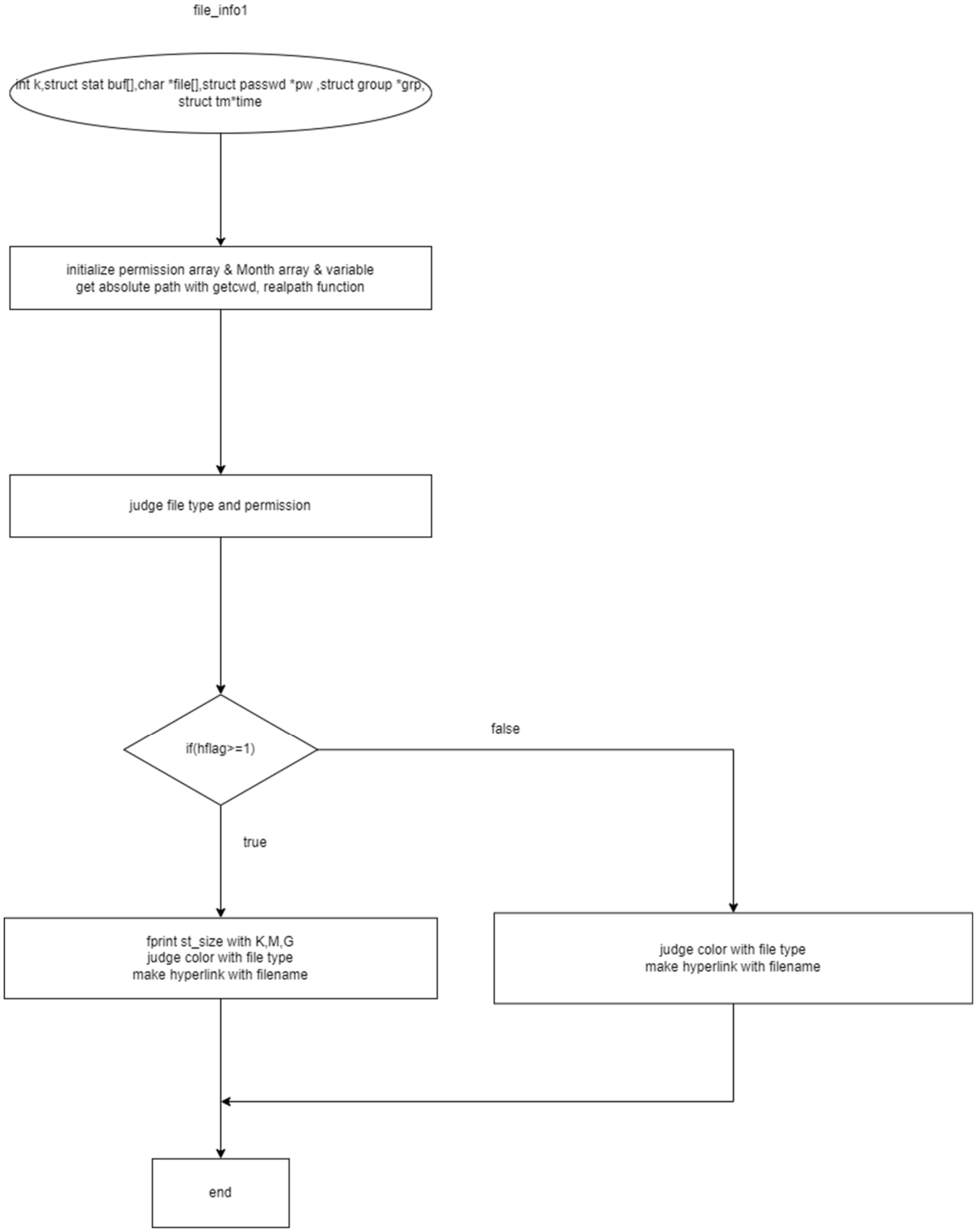
Wildcard_many function



file_info function



file_info1 function



Pseudo Code

```
Main function {
    pointer to directory stream
    Pointer to a dirent structure
    initialize variable
    declare struct stat buf

    file open(html_ls.html)
    print command in html
    print current directory as title in html

    while check option
    {
        Option a -> aflag++ &break
        Option l -> lflag++ & break
        Option S -> Sflag++ & break
        Option h -> hflag++ & break
        Option r -> rflag++ & break
        Other option -> break
    }

    Call the corresponding function based on the number of wildcards

    Call a function that corresponds to the condition based on the number of
flags
    quit main function
}

bubble_sort1 {
    initialize variable
    sort the file names alphabetically in wildcard function
}
```

```
sort_files_reverse {
	initialize variable

	sort the file names alphabetically
	sort the file sizes in reverse order
}

sort_files
{
	initialize variable

	sort the file names alphabetically
	sort the file sizes in order
}


bubble_sort(char**file, int n, struct stat buf[]) {
	initialize variable
	sort the file names alphabetically with ls option
}

bubble_sort_reverse {
	initialize variable
	Bubble sort the file names reverse alphabetically with ls option
}

wildcard
{
    initialize variable

	when current directory
```

```
when not absolute path
{
    open directory
    {
        read directory
        {
            do not print hidden files in html
            {
                get info
                when not directory
                {
                    check wildcard
                    when wildcard
                    {
                        store file matched with wildcard
                    }
                }
                check directory
                {
                    check wildcard
                    when wildcard
                    {
                        store directory matched with wildcard

                    }
                }
            }
        }
        call bubble_sort1 about file
        call bubble_sort1 about directory

        print file first in html with table
```

```
            initialize variable

                open directory
                print directory in html
                while read directory
                    except hidden file
                        store directory

             print content of directory with table in html
             close directory
        }
}

when absolute path
{
        initialize variable
        to get wildcard location
        {
            when absolute path
            {
                get path length except wildcard
            }
        }
        get absolute path

        opden directory
        {
            read directory
            {
                initialize variable
```

```
            absolute path + d_name
            get info
            except hidden file
            {
                when not directory
                {
                    check wildcard
                    when file matched with wildcard
                    {
                        store file matched with wildcard
                    }
                }

                when directory
                 {
                    check wildcard
                    when directory matched with wildcard
                    {
                        store directory matched with wildcard
                    }
                }
            }
        }

        bubble_sort1 about file
        bubble_sort1 abobut directory

        print directory and content of directory with table in html
        close directory
    }
}
```

```
wildcard_many
{
    initialize variable

    check wildcard

    check wildcard exsist
    {
        for(i=optind;i<argc;i++)
        {
            when wildcard exsist
            {
                when absolute path
                {
                    initialize variable
                    get absolute path

                    open directory
                    {
                        read directory
                        {
                            except hidden file
                            {
                                when not directory
                                {
                                    check wildcard
                                    when file matched with wildcard
                                    {
                                        store file matched with wildcard
                                    }
                                }
                                when directory
```

```
                                {
                                        check wildcard
                                        when directory matched with wildcard
                                        {
                                              store directory matched with
wildcard
                                        }
                                }
                            }
                        }
                    }
                when not absolute path
                {
                        open directory
                        {
                            read directory
                            {
                                except hidden files
                                {
                                    when not directory
                                    {
                                        check wildcard
                                        when file matched with wildcard
                                        {
                                              store file matched with wildcard
                                        }
                                    }
                                    when directory
                                    {
                                        check wildcard
                                        when directory matched with wildcard
```

{
store directory matched with wildcard

}
}
}
}
}
}
}
bubble_sort1 about file
bubble_sort1 abobut directory

print file with table in html
print directory and content of directory with table in html
close directory
}
}
}

ls function
{
initialize variable

When there is no file or directory as an argument
{
open current directory
{
read directory
{
do not print hidden files in html

```
                add the name of the file to the array
            }
            close directory stream

            Call the corresponding function based on the conditions of the
given option
        }
    }
    When there is a file or directory as an argument
    {
        for(int i=optind;i<argc;i++)
        {
            type check
            not directory
            {
                no exsist file
                {
                    Exception when file does not exist
                }
            }
        }

        for(int i=optind;i<argc;i++)
        {
            save file information into buf
            type check
            not directory
            {
                no exsist file
                {
                    count file
```

```
                }
            }
        }
```

Call the corresponding function based on the conditions of the given option

```
        for(i=optind;i<argc;i++)
        {
            open directory
            {
                initialize value
                read directory
                {
                    do not print hidden files in html
                    add the name of the file to the array
                }
                close directory stream

                get directory absolute path in html
                print directory path in html
                Call the corresponding function based on the conditions of the given option
            }
        }
    }
}

ls_a function
{
    initialize variable
```

```
When there is no file or directory as an argument
{
    open current directory
    {
        read directory
        {
            add the name of the file to the array
        }
        close directory stream

        Call the corresponding function based on the conditions of the given option
    }
}
When there is a file or directory as an argument
{
    for(int i=optind;i<argc;i++)
    {
        type check
        when not directory
        {
            no exsist file
            {
                Exception when file does not exist
            }
        }
    }

    for(int i=optind;i<argc;i++)
    {
        save file information into buf
        type check
```

```
        not directory
        {
            no exsist file
            {
                count file
            }
        }
    }
```

Call the corresponding function based on the conditions of the given option

```
    for(i=optind;i<argc;i++)
    {
        initialize variable
        open directory
        {
            initialize valiable
            read directory
            {
                do not print hidden files in html
                add the name of the file to the array
            }

            close directory stream

            get directory absolute path
            print directory path in html
            Call the corresponding function based on the conditions of the given option
        }
    }
```

```
    }
}

ls_l function
{
    initialize value
    get file information

    When there is no file or directory as an argument
    {
        open directory
        {
            read directory
            {
                do not print hidden files in html
                {
                    store file information
                }
            }

            for (int j = 0; j < i; j++)
            {
                get block size
            }

            get current directory
            print directory path in html
            print total block size in html

            Call the corresponding function based on the conditions of the
given option
            print file information with table in html
```

```
        }
        close directory
    }

    When there is a file or directory as an argument
    {
        for(int i=optind;i<argc;i++)
        {
            save file information into buf
            type check
            not directory
            {
                no exsist file
                {
                    Exception when file does not exist
                }
            }
        }
        initialize variable
        for(int i=optind;i<argc;i++)
        {
            save file information into buf
            type check
            not directory, exsist file
            {
                count file
            }
        }
```

Call the corresponding function based on the conditions of the given option

print file information with table in html

```
for(i=optind;i<argc;i++)
{
    open directory
    {
        for(i=optind;i<argc;i++)
        {
            {
                read directory
                {
                    do not print hidden files in html
                    {
                        store file name into file array
                    }
                }
                    for (int q = 0; q < k; q++)
                    {
                        get absolute path
                        save file information into buf
                        get block size
                    }

                    print directory path in html
                    print total size in html

                    Call the corresponding function based on the conditions of the given option
                    print file information with table in html

            }
            number of file initialize
        }
```

```
            }
        }
        close directory
}

ls_al function
{
    initialize value
    get file information

    When there is no file or directory as an argument
    {
        open directory
        {
            read directory
            {
                {
                    store file information
                }
            }

            for (int j = 0; j < i; j++)
            {
                get block size
            }

            get current directory
            print directory path in html
            print total block size in html

            Call the corresponding function based on the conditions of the
given option
```

```
            print file information with table in html
        }
        close directory
    }

    When there is a file or directory as an argument
    {
        for(int i=optind;i<argc;i++)
        {
            save file information into buf
            type check
            not directory
            {
                no exsist file
                {
                    Exception when file does not exist
                }
            }
        }
        initialize variable
        for(int i=optind;i<argc;i++)
        {
            save file information into buf
            type check
            not directory, exsist file
            {
                count file
            }
        }
```

Call the corresponding function based on the conditions of the given option

print file information with table in html

for(i=optind;i<argc;i++)
{
open directory
{
for(i=optind;i<argc;i++)
{
{
read directory
{
{
store file name into file array
}
}
for (int q = 0; q < k; q++)
{
get absolute path
save file information into buf
get block size
}

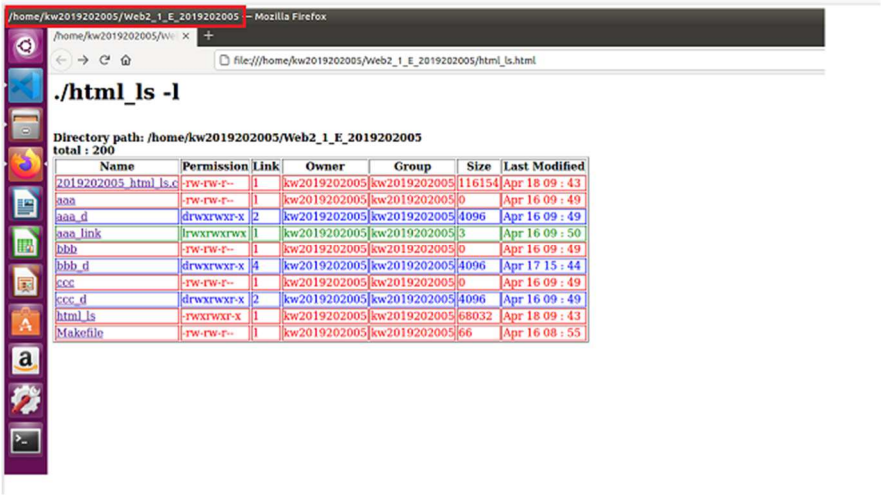
print directory path in html
print total size in html

Call the corresponding function based on the conditions of the given option
print file information with table in html

}
number of file initialize
}

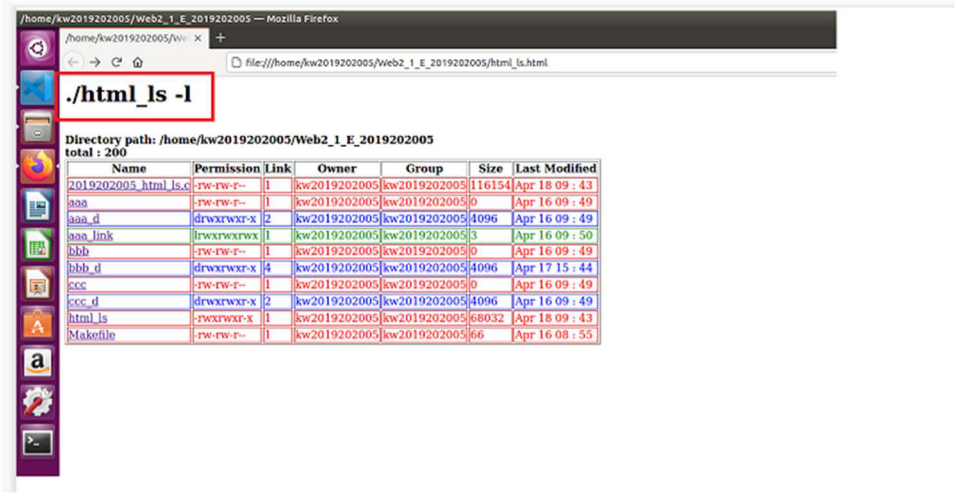
```
            }
        }
        close directory
}

char file_check
{
        file check
        judge file types and store check variable
}

void file_info about directory
{       judge file type and permission in html
        print permission with table in html
        print link size with table in html
        print user name with table in html
        print group info with table in html
        Process the flag & file size conditions and print file size with table in html
        print month day time with table in html
        print file name with color and hyperlink with table in html
}

void file_info about file
{       judge file type and permission
        print permission with table in html
        print link size with table in html
        print user name with table in html
        print group info with table in html
        Process the flag & file size conditions and print file size with table in html
        print month day time with table in html
        print file name with color and hyperlink with table in html
}
```

결과화면

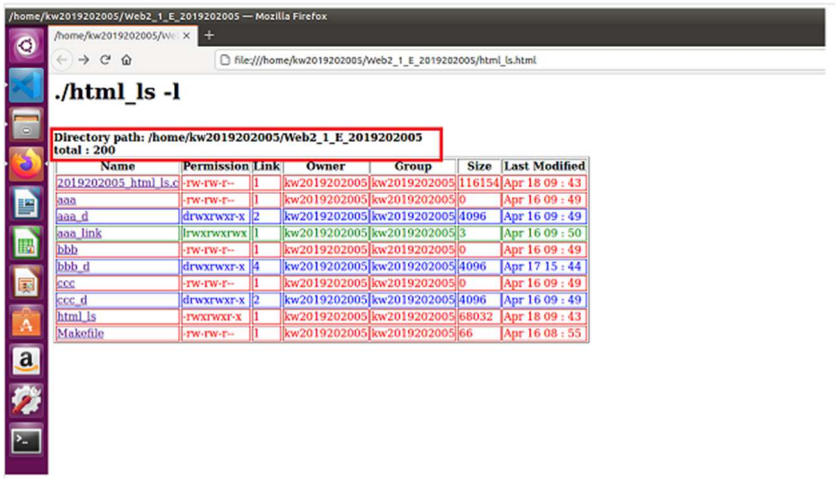


Tiltle 태그는 웹 페이지의 제목을 지정합니다. 브라우저에서 페이지 제목을 보여줍니다.

current directory를 title태그를 이용하여 위 사진처럼 출력해주었습니다.



다음으로 command부분을 헤더 태그를 이용해 위 사진처럼 출력해주었습니다.



명령어 ls -l 입력 시 출력되는 directory path와 total을 똑같이 출력해주었습니다.

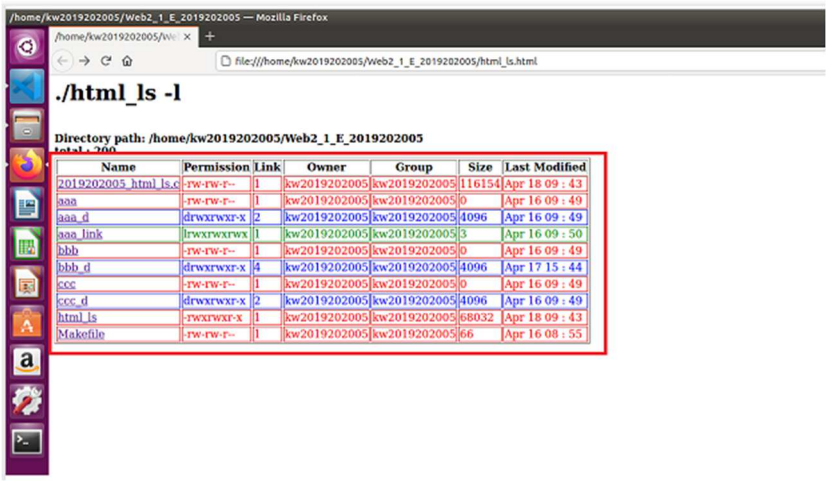


Table 태그는 데이터를 표 형태로 나타내기 위해 사용되는 태그입니다.

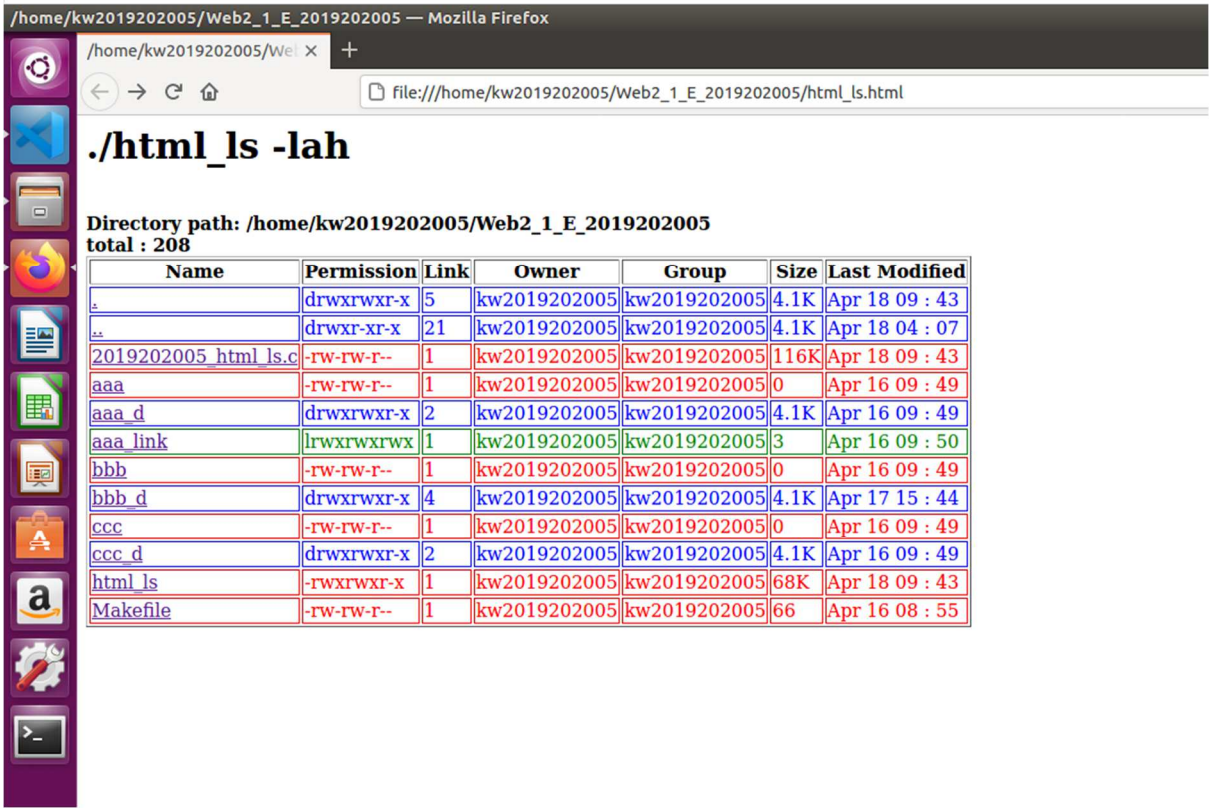
- <tr> : 테이블의 각 행을 나타냅니다.
- <th> : 테이블의 헤더 셀을 나타냅니다.
- <td> : 테이블의 일반 셀을 나타냅니다.

Table 태그를 이용해 ls -l 입력 시 출력되는 파일 및 디렉토리의 정보를 표 형태로 출력해주었습니다.

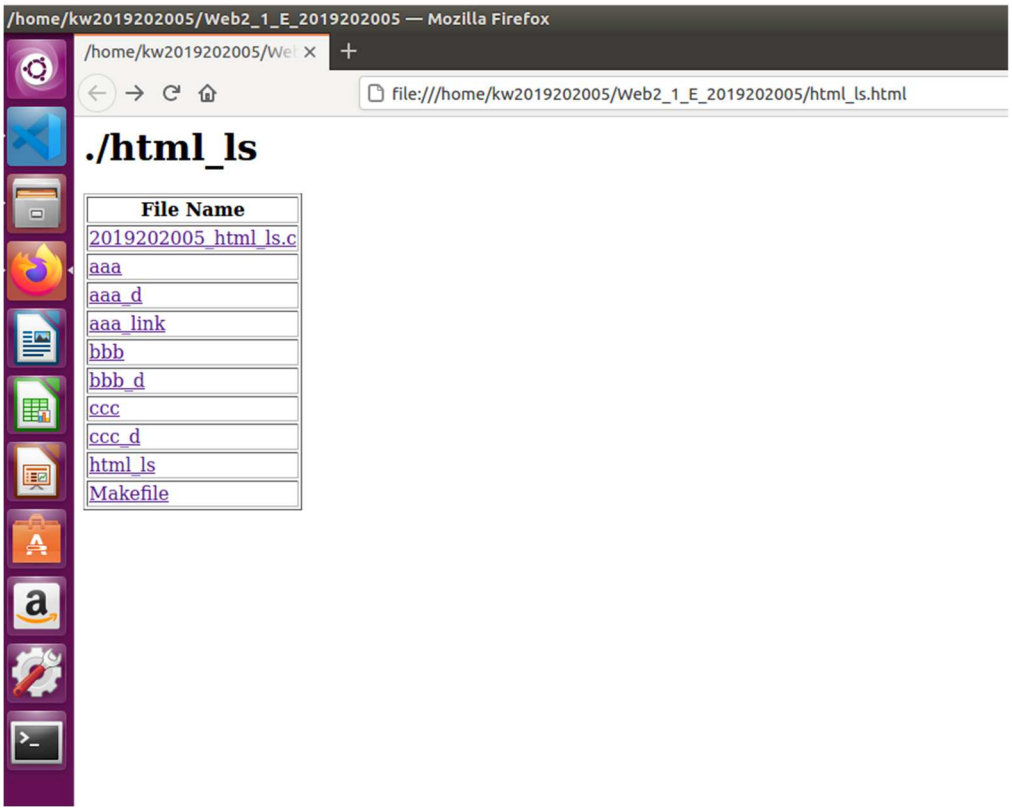
<A HREF="URL"> text </A>태그를 이용하여 각 파일 및 디렉토리의 이름에 하이퍼링크를 달아주었습니다.

Style="color:####"을 이용하여 디렉토리, 링크 파일, 파일의 색깔을 각각 파란색, 초록색, 빨간색으로 출력해주었습니다.

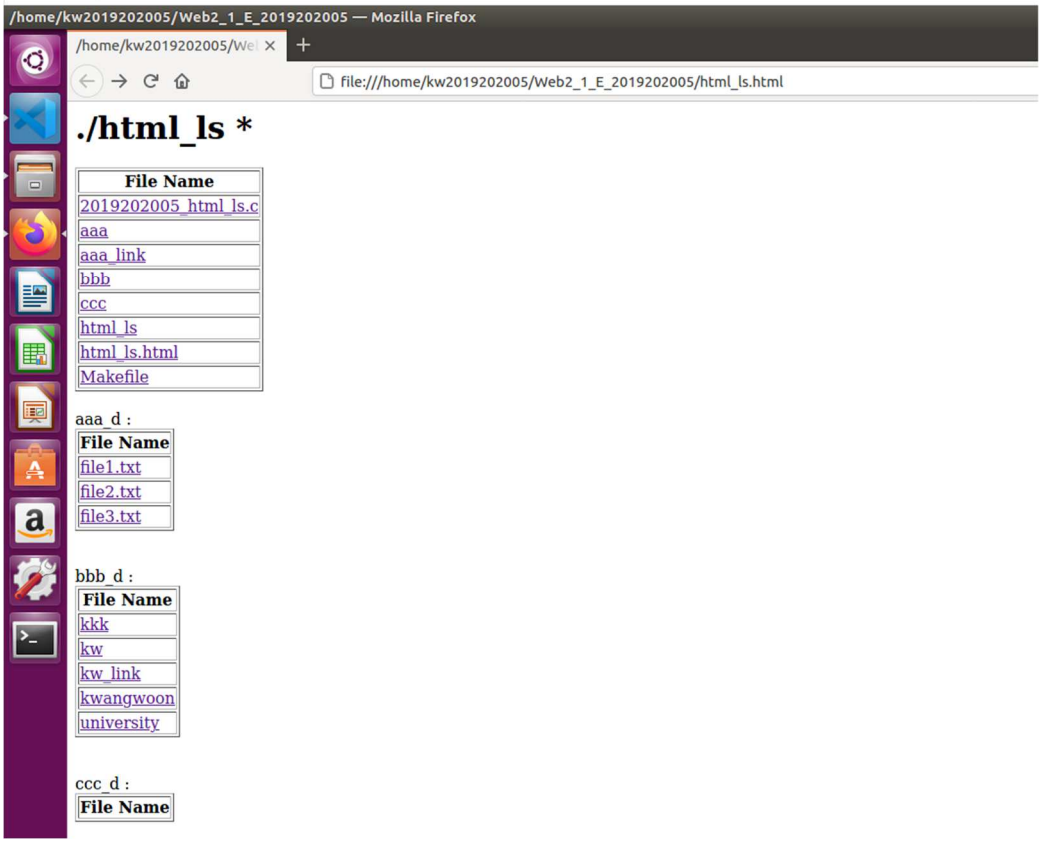
출력된 html파일의 이름은 html_ls.html이며 이 파일은 표에 출력되지 않게 해주었습니다.



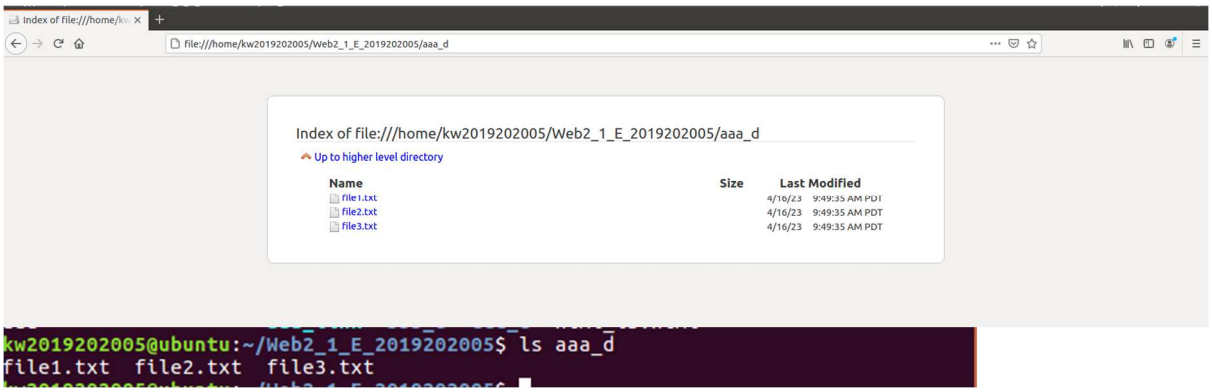
다음은 ./html_ls -lah를 입력했을 때 출력되는 화면입니다.



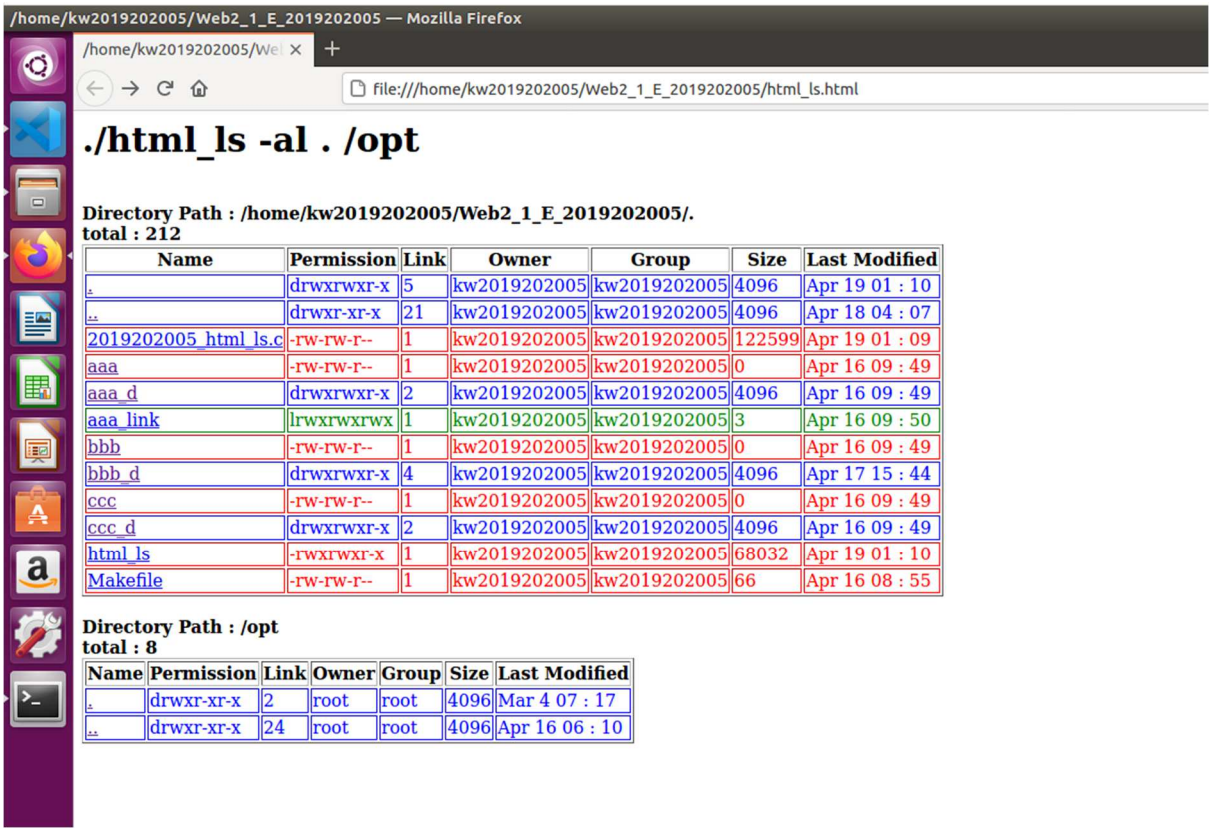
다음은 기본 ls명령어 입력 시 출력화면입니다.



다음은 와일드카드를 입력했을 때 출력되는 화면입니다.



다음은 aaa_d 디렉토리의 하이퍼링크를 타고 들어갔을 때 출력되는 화면입니다. ls 명령어를 통해 aaa_d 디렉토리의 파일들과 같게 출력된 것을 확인했습니다.



다음은 ./html_ls -al . /opt를 출력한 화면입니다.

고찰

이번 과제는 저번 과제에서 진행한 filnal_ls코드를 바탕으로 html문서를 생성해 웹에서 직접 확인하는 과제였습니다. HTML에 대해서는 많이 들어는 봤지만 예전에 살짝 유튜브를 통해 HTML에 대해서 접해본 경험 외에는 처음 접하는 시간이었기 때문에 걱정도 많이 되었습니다. 실습 시간에 HTML파일을 생성하고 웹에서 확인하는 수업을 듣고 난 후 생각보다 과제를 진행하면서 이전 과제들보다 수월할 수 있겠다고 생각이 들었습니다. 이전에 ls명령어를 직접 구현하는 과제에서 아무것도 없는 날 것 상태에서 모든 것을 다 구현하려다 보니 정말 시행착오도 많았고 시간이 많이 소요되었습니다. 하지만 이번 HTML과제에서는 이미 구현한 코드를 토대로 약간의 변형 및 살을 덧붙이는 과정이 대부분이었기 때문에 좀 더 수월하게 진행할 수 있었습니다.
일단 모든 printf문에는 printf문을 한 줄 더 만들어 fprintf문으로 바꿔주었으며 printf문을 통해 터미널에서 결과 화면을, fprintf문을 통해 브라우져에서 결과화면을 확인했습니다.
이전에 과제에서는 결과화면을 터미널을 통해서만 확인할 수 있어 약간 보기

불편한 감이 없지 않았었는데 이번 과제에서는 웹을 통해 직접 확인할 수 있어 진짜 무언가를 만든 느낌이 들었습니다. 그리고 웹에서 파일 및 디렉토리의 테이블을 만들어 하이퍼링크도 달아주고 색깔도 파일의 타입에 따라서 입혀주다 보니 ls의 출력문을 이전보다 더 보기 쉬웠고 변경한 부분을 즉각적으로 쉽게 확인할 수 있어 수정할 때에 있어서도 수월하게 진행할 수 있었습니다.
HTML에 대해서는 정말 기초적인 부분만 사용했지만 조금이라도 배울 수 있어서 좋은 시간이었고 흥미로웠습니다.

Reference

2023년 1학기 시스템프로그래밍 & 시스템 프로그래밍 실습 강의자료
Assignment 2-1
2023년 1학기 시스템 프로그래밍 1학기 강의자료 3. Files and directories
2023년 1학기 시스템 프로그래밍 실습 8주차 강의자료 HTML-ls