2019. 4. 12.

Assignment # 2-3

금요일 - 이성원 교수님

2015722087 컴퓨터정보공학부

김민철

Spls\_Final\_ls 구현

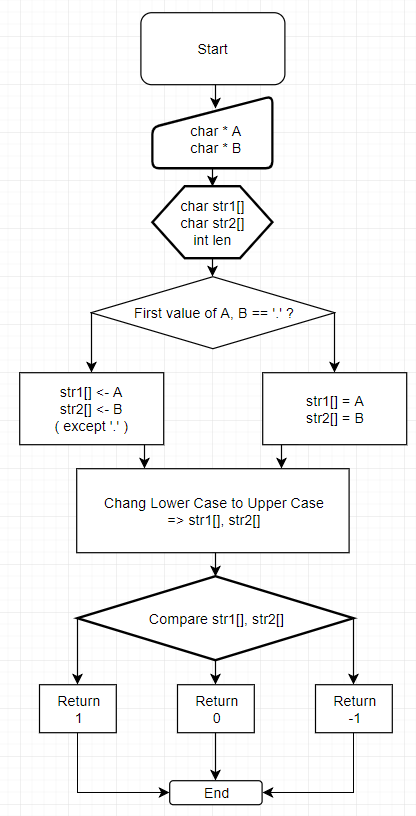
**Introduction**

이번 과제에서는 2-2 과제에서 구현하였던 advanced\_ls 에서 옵션 –h : 파일 크기를 1024로 나누어 단위 표시, -S : 파일 크기로 정렬, 같으면 이름순 정렬, -r : 정렬을 역순으로 변경

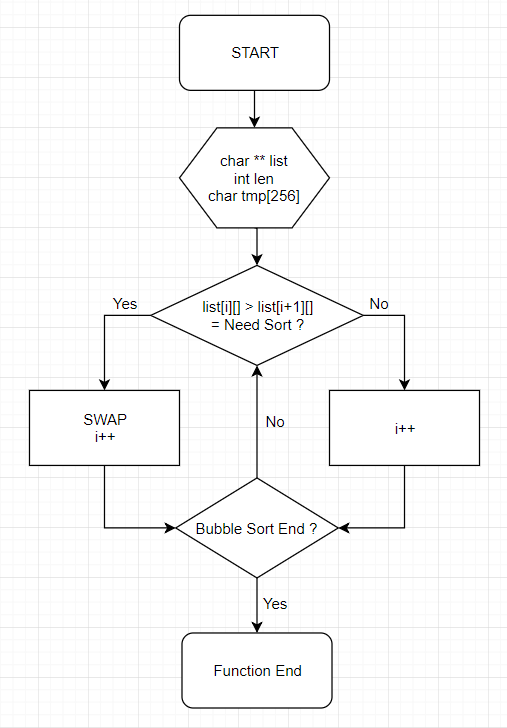
그리고 Fnmatch()를 이용한 wild card matching 기능을 추가하고, ‘\*’를 사용하여 디렉토리의 내의 모든 디렉토리를 오픈하는 기능을 추가 구현하는 것이 이번 2-3 과제이다.

**Flow Chart**

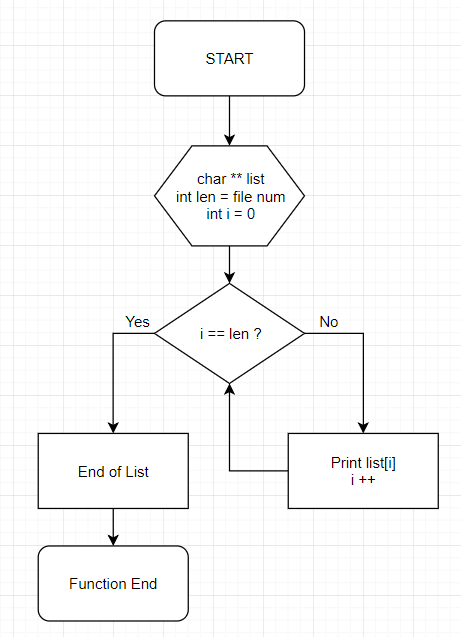
-int strcmp\_i



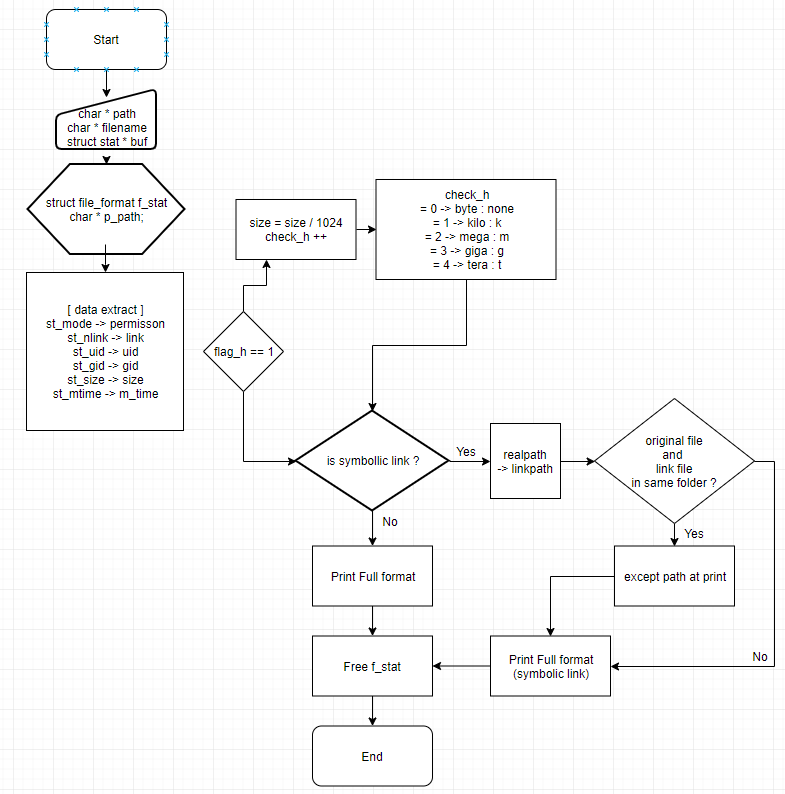
-void sort\_list



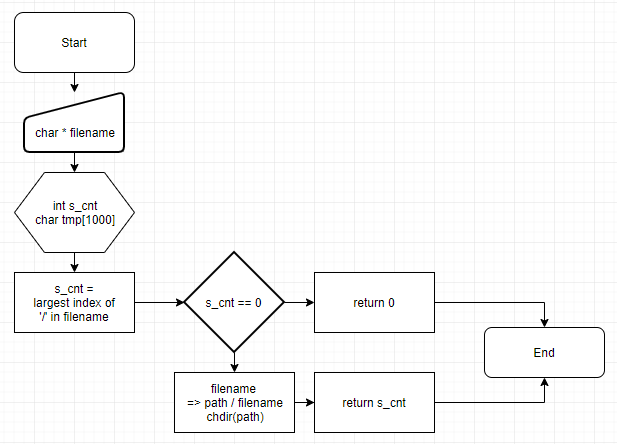
-void view\_list



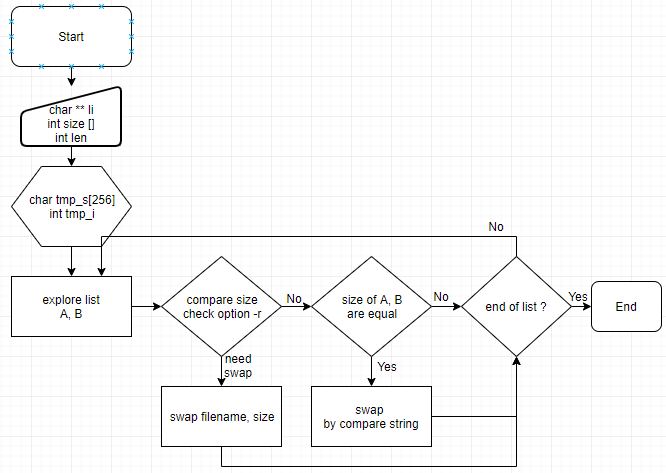
-void view\_advanced\_list



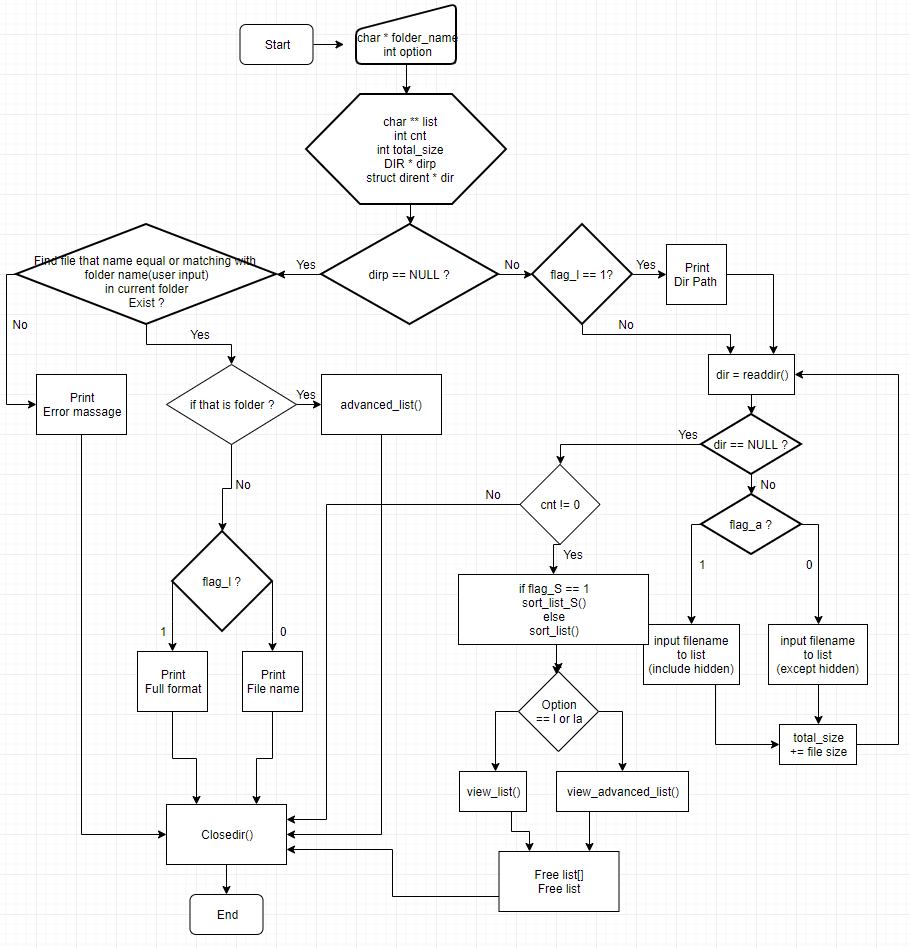
-int check\_real\_path



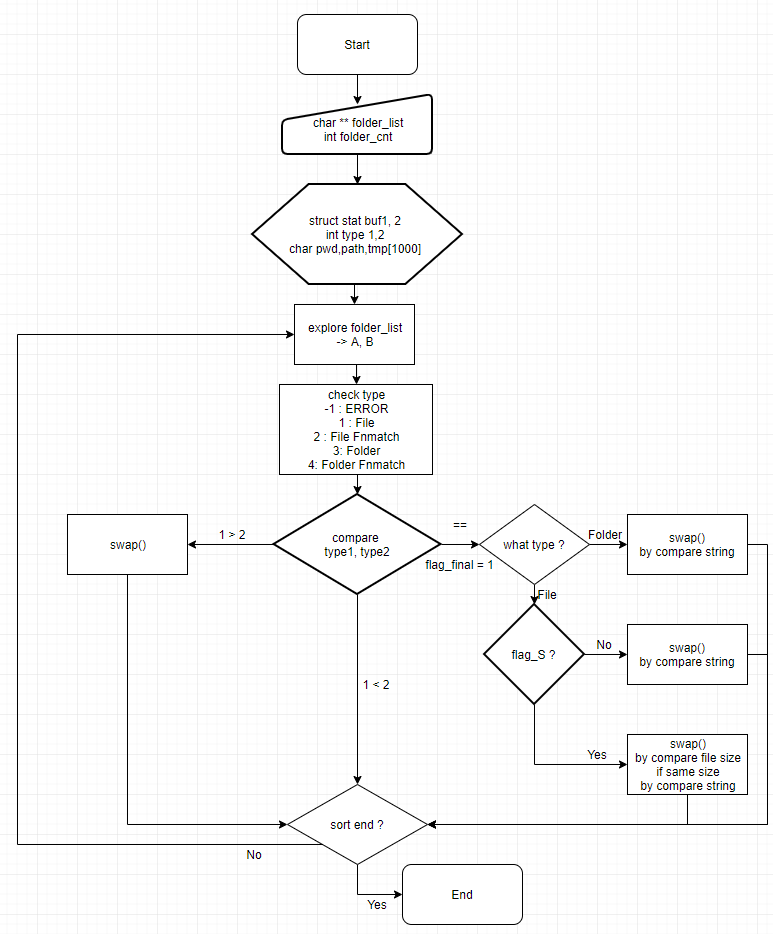
-void sort\_list\_S



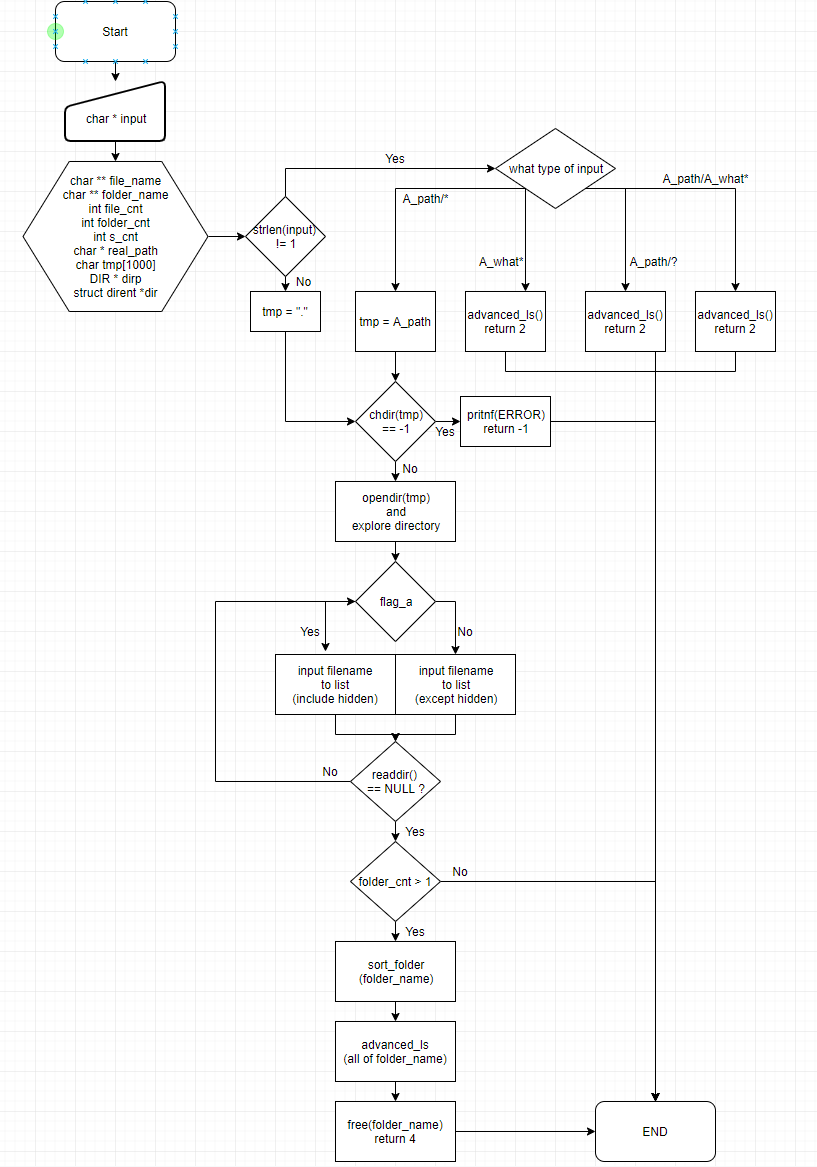
-int advanced\_ls



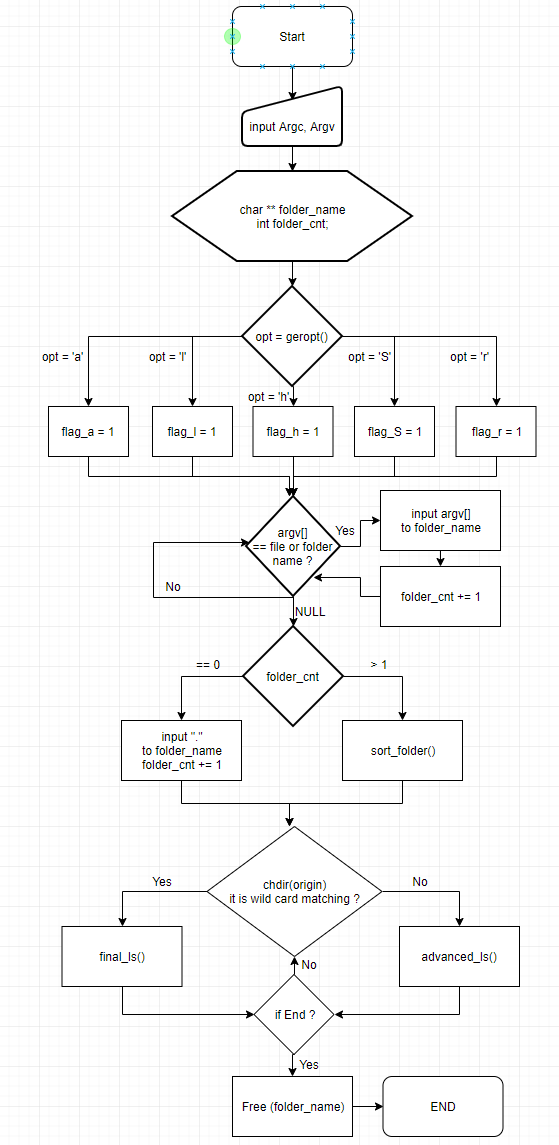
-void sort\_folder



-int final\_ls



-int main



**Pseudo code**

int check\_real\_path(char \* filename)

{

for (int i = 0; i < strlen(filename); i++) // check directory.

{

입력된 파일 이름에 '/' 가 있을 경우 해당 index => s\_cnt

}

if 파일 이름에 '/' 가 있다면 -> real path

{

devide < path > / < file >

<path> 로 작업중인 디렉토리 변경

s\_cnt 값 반환

}

else real path 가 아니라면

0 반환

}

void swap(char \* str1, char \* str2)

{

str1과 str2 swap

}

int strcmp\_i ( char \* 비교할 첫 번째 문자열 A, char \* 비교할 두 번째 문자열 B )

{

<예외 처리> : r 옵션이면 거꾸로.

A랑 B가 “.” , “..” 이면 “..” 이 뒤로,

둘 중 하나만 “.” 이면 “.”이 앞으로

둘 중 하나만 “..” 이면 “..”이 앞으로

if( A 가 B 보다 길면 )

len = B의 길이;

else

len = A의 길이

for(len 만큼 반복)

{

if(소문자라면)

대문자로 바꾼다

}

If 옵션이 –r 이라면

if( A > B ) return = -1;

else if( B > A ) return = 1;

else return = 0;

else

if( A > B ) return = 1;

else if( B > A ) return = -1;

else return = 0;

}

void sort\_list(char \*\* li, int len)

{

for (저장된 파일이름 개수-1 만큼 반복)

{

if (더 이상 내용이 없다)

break;

for (저장된 파일이름 개수-1 만큼 반복)

{

if ( 앞에 저장된 파일이름 > 뒤에 저장된 파일 이름)

두 파일 이름을 swap

}

};

}

void sort\_list\_S(char \*\* li, int size[], int len)

{

// sort like bublle sort

for (int i = 0; i < len - 1; i++) // total cycle

{

if 리스트가 끝났다면

break;

for (int j = 0; j < len - 1; j++) // one cycle

{

if 파일 사이즈가 다르다면, 정렬 필요 -r 옵션에 따라서.

{

swap file name

swap file size

}

else if 파일 사이즈가 같다면, -> 문자열로 정렬

{

if 정렬이 필요하다면

{

swap file name

}

}

}

}

return;

}

void view\_list(char \*\* li, int len)

{

for (파일이름 개수 만큼 반복)

{

for (파일 이름 다 출력할 때까지 반복)

{

파일 이름의 알파벳 하나씩 출력

}

}

}

void view\_advanced\_list(char \* 파일 경로, char \* 파일 이름, struct stat \* 파일 정보)

{

St\_mode에 저장되어 있는 파일 타입에 따라 permission[0] 결정

St\_mode에 저장되어 있는 user 권한에 따라 permission[1~3] 결정

St\_mode에 저장되어 있는 group 권한에 따라 permission[4~6] 결정

St\_mode에 저장되어 있는 other 권한에 따라 permission[7~9] 결정

* Permission = “- --- --- ---”

st\_nlink -> f\_stat->link

st\_uid -> f\_stat->uid

st\_gid -> f\_stat->gid

st\_size -> f\_stat->size

st\_mtime -> f\_stat->m\_time

if 옵션이 –h 라면

size = size / 1024.0, check\_h++(단위가 몇번 상승했는지 체크)

check\_h 값에 따라서 단위(K, M, G) 결정

// Full Format 출력

if 파일 type 이 symbolic link 라면

{

If 옵션이 –h 라면, 사이즈 출력시 단위와 함께 출력

Symbolic link 파일과 가리키는 경로 파일이 같은 폴더에 있다면 -> 가리키는 파일명만 출력

Symbolic link 파일과 가리키는 경로 파일이 다른 폴더에 있다면 -> 절대 경로 모두 출력

}

If 옵션이 –h 라면, 사이즈 출력시 단위와 함께 출력

else 파일 type 이 symbolic link 가 아니라면

파일 이름만 Full Format 출력

}

int advanced\_ls(char \* 입력한 폴더)

{

if 폴더가 아니라면

{

혹시 절대경로로 입력된 폴더인지 확인 -> 맞으면 해당 디렉토리로 변경

-> 아니면 현재 디렉토리

while 디렉토리 탐색

{

if 파일을 찾았다면

{

If 옵션이 a 나 default 라면

파일 이름만 출력

else if 옵션이 l 이나 la 라면

Full Format 출력

}

If fnmatch 로 매칭이 된다면,

{

폴더라면 advanced\_ls () 폴더 전체 출력

파일이라면

If 옵션이 a 나 default 라면

파일 이름만 출력

else if 옵션이 l 이나 la 라면

Full Format 출력

}

}

If 출력한 파일이 없다면

에러 메시지 출력 -> 반환 -1

else 1 반환

}

else // is folder

{

p 플래그가 표시되어있다면 return 3 => 폴더

if 옵션이 l 이나 la 라면

작업중인 디렉토리 출력

심볼릭 링크로 연결된 폴더인지 확인

-> 심볼릭 링크 폴더이고 옵션이 l 이면 심볼릭 링크만 Full Format 출력

아니면 탐색 시작

while 입력된 폴더 탐색

{

if 파일이 없다면

반복문 탈출

if 히든 파일이고 옵션이 –l 이나 default 라면

{

List에 파일 이름 입력

Total size += 파일 size

}

else if 옵션이 –a 이나 –la 라면 모든 파일 입력

{

List에 파일 이름 입력

Total size += 파일 size

}

}

if 파일이 존재한다면

{

List 정렬.

if 옵션이 –a 거나 default 라면

모든 파일 이름 출력

else if 옵션이 –l 이거나 –la 라면

{

토탈 사이즈 출력

for (파일 개수만큼 반복)

{

Full Format 으로 파일 이름 출력

}

}

파일 리스트 동적 할당 해제

}

}

폴더 닫기

}

void sort\_folder(char \*\* 폴더 list, int 폴더 개수)

{

int check1;

int check2;

for (폴더 개수 – 1 만큼 반복)

{

for (폴더 개수 – 1 만큼 반복)

{

만약 final\_ls 함수에서 정렬이라면 chdir(final\_wd)

아니라면(main에서 정렬) chdir(origin\_wd)

표시 데이터 전부 리셋

리스트에 연속된 폴더 이름 2개로 폴더 open : A폴더(앞), B폴더(뒤)

If 정렬이 필요하다면,

폴더 이름으로 정렬

else if 같은 타입이고 final 에서 정렬하는 경우

{

if 둘다 폴더일 경우

폴더 이름으로 정렬

else 둘다 파일인경우

if S 옵션이 있을경우

파일 크기로 정렬 -> 같을 경우 이름으로 정렬

else

파일 이름으로 정렬

}

if 마지막 cycle이고, final에서 호출했다면

if 파일이고, origin 디렉토리가 아니라면,

파일 이름앞에 절대 경로 입력

}

}

}

int final\_ls(char \* input)

{

if input의 길이가 1이 아닌경우

if A\_what\* 같이 입력된 경우

advanced\_ls () 호출 -> 2 반환

else if A\_path / A\_What\* 같이 입력된 경우

폴더가 존재하는 지 확인 -> 없으면 에러

존재하면 advanced\_ls () 호출 -> 2 반환

else if A\_path/\* 같이 입력된 경우

존재하는 폴더인지 확인 -> 없으면 에러

else

폴더가 존재하는 지 확인 -> 없으면 에러

존재하면 advanced\_ls () 호출 -> 2 반환

else 길이가 1인 경우

A\_path = 현재 경로

if A\_path 가 폴더가 아닌 경우

에러 메시지 출력 -> -1 반환

else A\_path가 폴더인 경우

{

플래그 p 가 -1 이면 4 반환

A\_path 로 작업중인 디렉토리 변경

final\_wd 에 A\_path 저장

A\_path 디렉토리 opendir()

while 디렉토리 탐색

{

if -a 옵션일 경우 모든 파일을 리스트에 저장, ('.', '..' 제외)

{

폴더 이름 리스트 동적할당

파일 이름을 폴더 이름 리스트에 입력

폴더 개수 카운트 ++

}

else if -a 옵션이 아닐 경우 히든 파일 제외한 파일 리스트에 저장.

{

폴더 이름 리스트 동적할당

파일 이름을 폴더 이름 리스트에 입력

폴더 개수 카운트 ++

}

}

if 파일이 존재할 경우

{

flag\_final 를 1 로 변경. -> final\_ls 에서 정렬을 시작한다고 알림

sort\_folder() -> 리스트 정렬

}

final\_wd 출력, 출력할 폴더, 파일들의 상위 폴더. = final\_ls가 실행되는 디렉토리

for (int i = 0; i < folder\_cnt; i++) // Print folders !

{

final\_wd 로 디렉토리 변경

폴더 혹은 파일 출력 -> advanced\_ls()

}

for (int a = 0; a < folder\_cnt; a++)

{

동적 할당 해제

}

동적 할당 해제

}

flag\_final = 0;

4 반환

}

int main(int 입력한 인자 개수, char \* 입력한 인자 문자열)

{

while 옵션 구하기

{

switch (옵션)

{

case 'a': // include option a : print all file (include hidden)

-a 옵션이 있다면 a 옵션 flag ++

break;

case 'l': // include option l : print all format of file

-l 옵션이 있다면 l 옵션 flag ++

break;

case 'h': // include option h : change format of print file size

-h 옵션이 있다면 l 옵션 flag ++

break;

case 'r': // include option r : reverse sort

-r 옵션이 있다면 l 옵션 flag ++

break;

case 'S': // include option S : sort by file size

-S 옵션이 있다면 l 옵션 flag ++

break; }

}

for(입력한 인자 개수 – 1 만큼 반복 (첫 번째 인자는 파일 실행이므로 제외)

{

// find folder name. (or file)

if (옵션이 아니고 폴더 이름이라면, (“.”, “..” 포함) )

{

폴더 리스트 동적 할당

폴더 리스트에 폴더 이름 추가

폴더 개수 카운트 ++

}

}

if 폴더 이름이 없다면 -> default = 현재 디렉토리

{

폴더 리스트에 현재 디렉토리 추가

폴더 개수 카운트 ++

}

else 폴더 개수가 2개 이상이라면

sort\_folder()

for (폴더 개수 만큼 반복)// Print folders !

{

origin\_wd 로 작업중인 디렉토리 변경

-> '\*' 출력일 경우 -> final\_ls()

-> 아니라면 -> advanced\_ls()

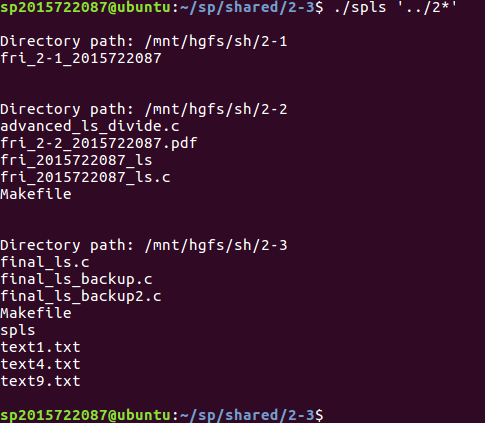
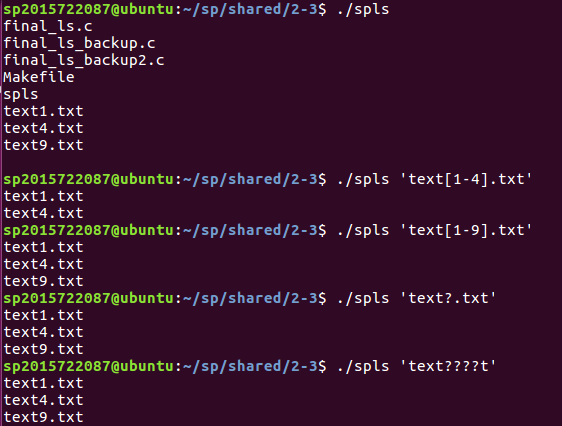
}

폴더 리스트 동적 할당 해제

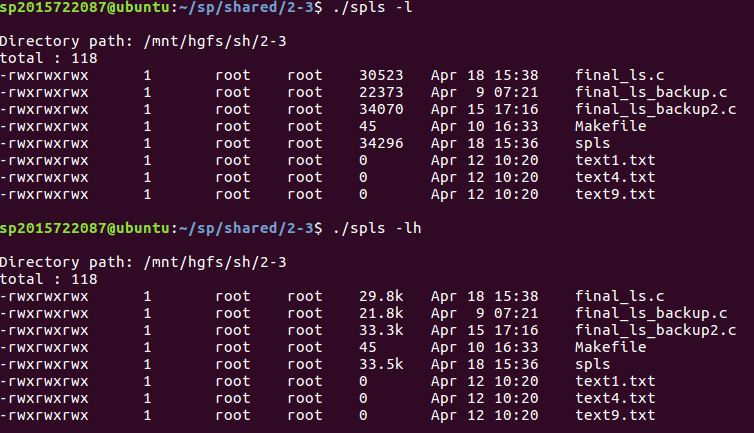
프로그램 종료

}

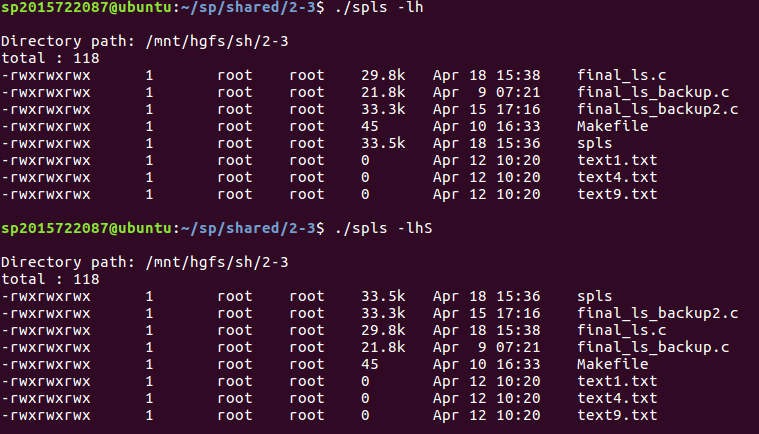
**Result**



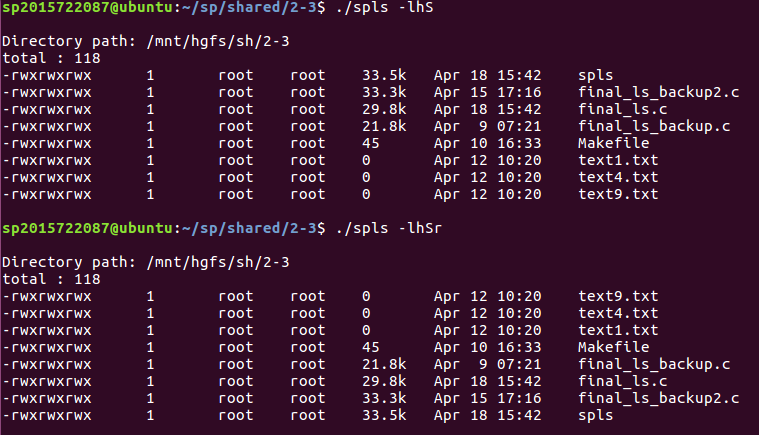
2-2 에 비해서 추가된 기능 중 wild card matching은 구현을 하지 않아도 작동이 되었으나, 이번 과제에서는 작은 따옴표 사이에 입력이 들어가도 기능을 하도록 추가 구현을 하였다. 절대 경로로 입력하여도, 상대 경로로 입력하여도 기능이 동작하기 위해 상대경로인지 절대경로인지 확인하여 폴더를 opendir() 하였고, 해당 디렉토리에서 fnmatch() function을 통해 확인하고 출력하였다.



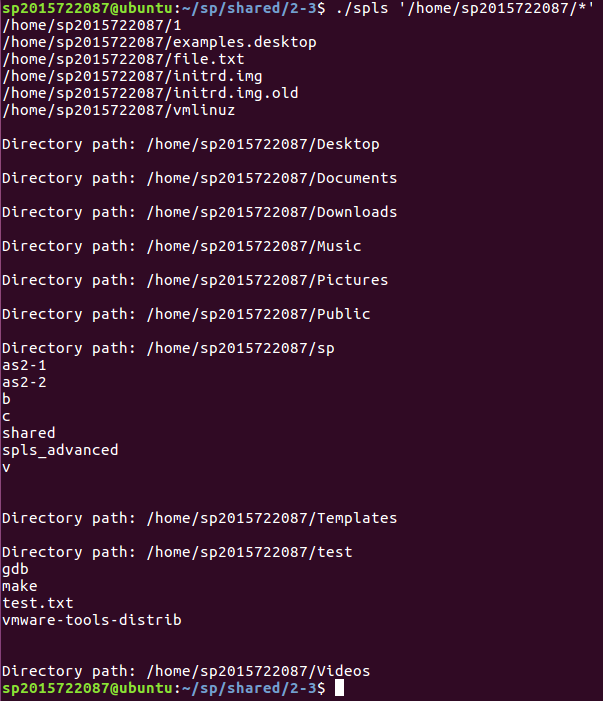
추가된 옵션 –h 는 Full Format으로 출력하는 view\_advanced\_ls()에서 소수점을 출력하기 위해 size 변수를 double 형으로 바꾸었고, -h 옵션인 경우에는 1024 로 나누어 단위 소수점 1자리수 까지 단위와 함께 출력하도록 하였으며, -h 옵션이 아닌 경우에는 int형으로 강제 형 변환하여 출력하였다.



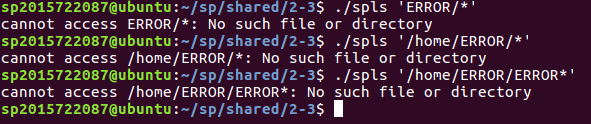
추가된 옵션 –S 는 새로운 함수 sort\_S() 를 만들어 구현하였다. 기존의 sort\_list()는 파일의 이름으로만 비교했으나, sort\_S() 에서는 파일 크기로 리스트를 정렬하도록 하였고, 파일 크기가 같은 경우에는 이름순으로 정렬되도록 구현하였다.



추가된 옵션 –r 은 –S 옵션인 경우와 아닌 경우 모두 정렬을 반대로 하면 되기 때문에, 문자열 비교 함수 strcmp\_i() 에서 리턴 값을 반대로 주어 문자열 기준 정렬을 반대로 하게 하였고, sort\_S() 에서도 비교 부호를 반대로 바꾸어 정렬하였다.



마지막으로 ./spls\_final ‘\*’ 의 기능은 해당 디렉토리 내의 모든 파일을 출력하고, 폴더는 모두 해당 폴더의 내용을 출력하도록 하는 기능인데, 이를 구현하기 위하여 final\_ls() 함수를 구현하였다. 기존에 main()에서 입력 받은 인자들을 folder\_list 리스트에 입력하여 정렬하고, 파일인지 폴더인지 확인하여 출력하였다. 이번에는 그 리스트에 ‘\*’(현재 디렉토리) 혹은 ‘경로/\*’ 의 입력이 있을 경우, final\_ls() 함수를 통하여 처리하게 하였는데, 이때 final\_ls() 함수에서는 기존 main() 의 역할을 한번 더 수행한다. 해당 디렉토리의 모든 파일을(-a 옵션이 아닐 경우 히든 파일 제외, -a 옵션일 경우에도 ‘.’과 ‘..’ 제외) folder\_list 리스트에 입력하여 정렬하고 파일인지 폴더인지 확인하여 출력하게 하였다. 기존의 advanced\_ls 프로그램을 한번 더 수행하는 것과 같은 기능을 하도록 하였다.



예외 처리를 출력한 화면이다.

**Conclusion**

이번 과제는 ls를 구현하는 마지막 과제 인 것 같다. 처음 실습 시간에 강의 자료를 접하고 수업을 들었을 때부터, 기능이 너무 많이 추가되는 것은 아닌가하는 생각이 들었다. Advanced\_ls를 구현하는 과정에서도 옵션을 구현하는 것이 공부도 필요하고 시행착오가 필요했었는데, 이번에는 옵션을 기존 2개에서 3개 추가하여 총 5개의 옵션이 올바르게 동작하도록 해야 했기 때문에, 잘못 건드렸다가 프로그램이 엉망이 될 수도 있겠다는 생각이 들어 이번에 추가로 구현해야 될 기능들을 미리 공부하고 뭐부터 추가해야 될지, 어떤 식으로 업데이트 해야 할지 생각하고 시작하였는데, 이것이 중요했었던 것 같다.

옵션 5개를 구현하고 프로그램 중간 중간에서 옵션에 따른 구분이 필요했었기 때문에 옵션 flag를 전역 변수로 선언하여 사용하는 변화를 주었고, ‘\*’ 의 기능에선 chdir(), getcwd() 함수의 사용이 많아지고 디렉토리의 변경이 복잡해지면서 실제로 구현하는 중에 지금 작업중인 디렉토리가 뭐였는지 헷갈리는 경우가 많이 발생하였다.

결국 처음 프로그램을 실행했던 디렉토리 : origin\_wd,

Final\_ls() 함수가 실행되는 디렉토리 : final\_wd 를 전역 변수로 선언하여 헷갈리지 않도록 해결하였다.

이번 과제도 역시 처음에는 어려워 보이고 언제 해결하나 걱정했지만, 막상 시작하고 강의 자료를 참고하여 함수를 공부하고 사용하다 보니 결과물이 잘 나온 것도 좋고, 중요한 공부가 된 것 같아 시험 기간이지만 전혀 시간을 소비하는 것이 아깝다고 느껴지지 않은 과제였다.