2019. 3. 31.

Assignment # 2-1

금요일 - 이성원 교수님
2015722087 컴퓨터정보공학부
김민철

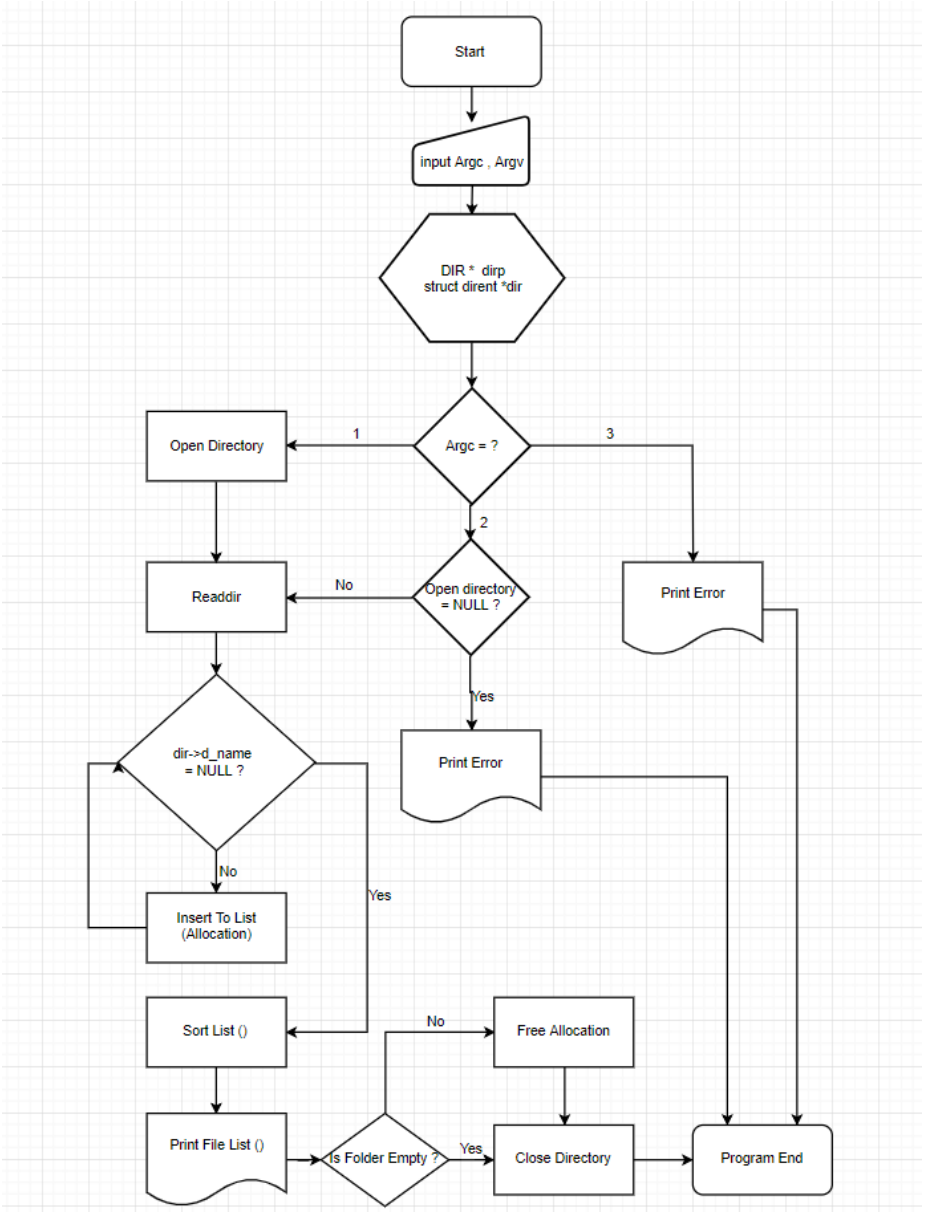
Simple ls 구현

Introduction

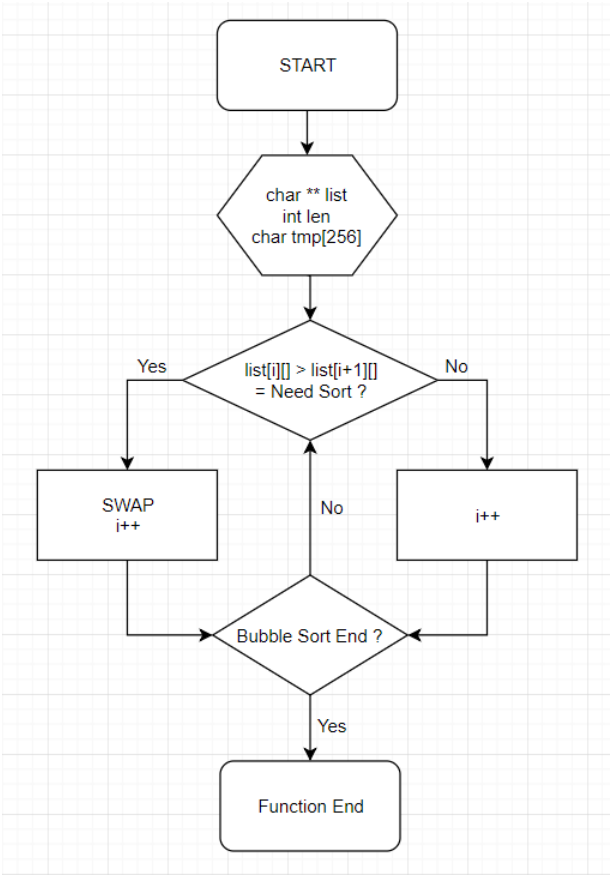
이번 과제는 리눅스에서 많이 이용하는 명령어인 'ls'를 C 언어로 구현하는 과제이다. Dirstream DIR 구조체와 dirent 구조체를 활용하여 구현하며, 주요 함수로는 Opendir(), Readdir(), closedir() 을 이용한다. 히든 파일(. 으로 시작)은 출력하지 않으며, 옵션 없이 파일 명만 출력하는 프로그램을 구현한다. 없는 폴더 혹은 없는 파일인 경우, 폴더 이름을 2 개이상 입력하는 경우 예외 처리를 한다.

Flow Chart

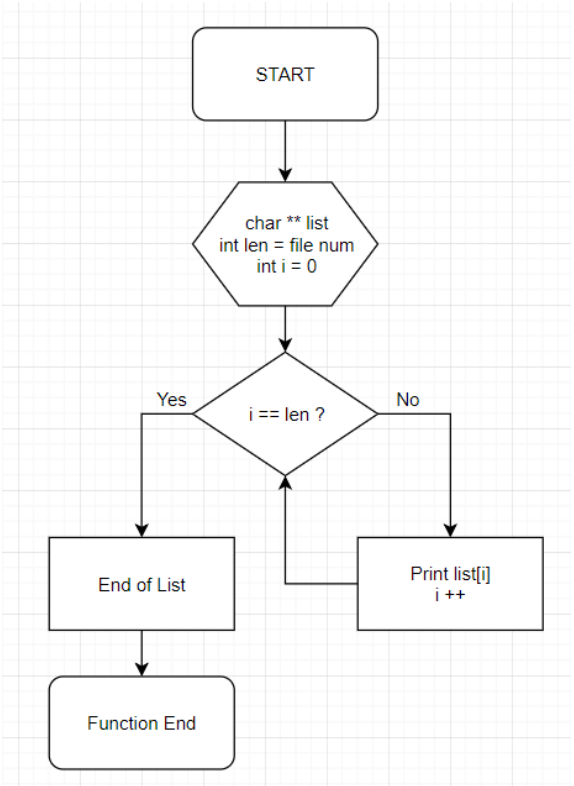
-Main()



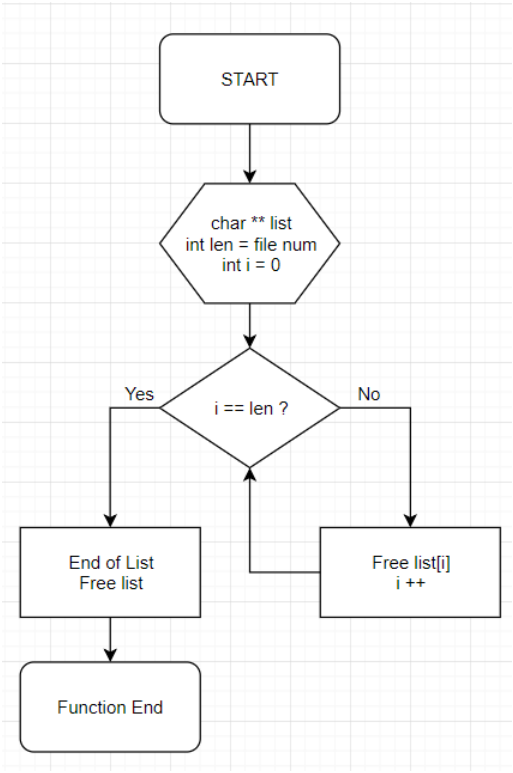
-Sort()


START
char ** list
int len
char tmp[256]
list[i][] > list[i+1][]
= Need Sort ?
Yes
No
SWAP
i++
No
i++
Bubble Sort End ?
Yes
Function End

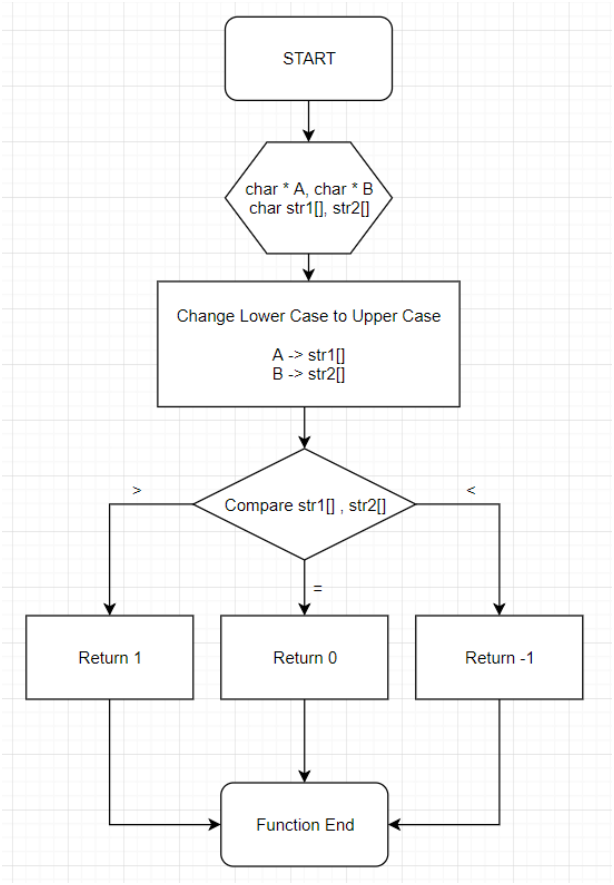
-View()


START
char ** list
int len = file num
int i = 0
i == len ?
Yes
No
End of List
Print list[i]
i ++
Function End

-Free()



-Strcmp_i



Pseudo code

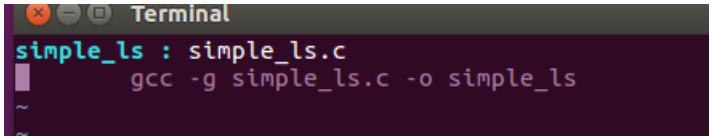
```
void free_list(char ** li, int i)
{
        for( a 가 0 부터 i-1 까지 증가하면서 반복 )
        {
                li[a] 메모리 할당 해제
        }
        li 메모리 할당 해제
}
================================================
int strcmp_i(char * A, char * B)
{
        if( A 가 B 보다 길면 )
                len = B 의 길이;
        else
                len = A 의 길이

        for(둘중 짧은 길이만큼 반복)
        {
                if(소문자라면)
                        대문자로 바꾼다
        }
        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 view_list(char ** li, int len)
{
        for (파일이름 개수 만큼 반복)
        {
                for (파일 이름 다 출력할 때까지 반복)
                {
                        파일 이름의 알파벳 하나씩 출력
                }
        }
}
=================================================
int main (int argc, char * argv[])
{
        char ** 파일이름을 저장할 리스트
        int 파일 개수 카운터=0

        if (특정 폴더를 지정하지 않았다면)
        {
                현재 디렉토리 open

                while (파일을 읽었다면 반복)
                {
                        if (더 이상 파일이 없다면)
```

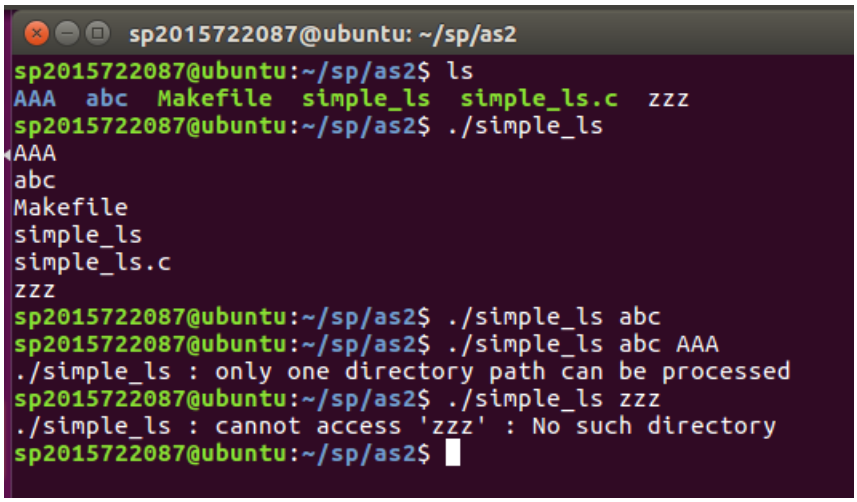
```
                    break;
                else if(히든 파일이 아니라면)
                {
                    if (첫 파일 입력이라면)
                    {
                        리스트를 메모리 동적 할당
                        리스트에 파일이름 입력
                        파일 개수 카운터 + 1
                    }
                    Else 첫 파일 입력이 아니라면
                    {
                        리스트를 메모리 동적 할당(저장 공간 하나 늘림)
                        리스트에 파일이름 입력
                        파일 개수 카운터 + 1
                    }
                }
            }

            sort_list(list, cnt); // sort by filename
            view_list(list, cnt); // print file list to screen

            if(파일개수가 0 개가 아니라면)
                free_list(list, cnt); // if list allocated, free list

            오픈된 폴더 close
    }
    else if (특정 폴더를 입력했다면)
    {
            특정 폴더를 open

            if(폴더를 여는데 실패했다면)
                에러 메시지 출력
            else 폴더를 여는데 성공했다면
            {
                while (파일을 읽었다면 반복)
                {
```

```
                        if (더 이상 파일이 없다면)
                                break;
                        else if(히든 파일이 아니라면)
                        {
                                if (첫 파일 입력이라면)
                                {
                                        리스트를 메모리 동적 할당
                                        리스트에 파일이름 입력
                                        파일 개수 카운터 + 1
                                }
                                Else 첫 파일 입력이 아니라면
                                {
                                        리스트를 메모리 동적 할당(저장 공간 하나
늘림)
                                        리스트에 파일이름 입력
                                        파일 개수 카운터 + 1
                                }
                        }
                }

                sort_list(list, cnt); // sort by filename
                view_list(list, cnt); // print file list to screen

                if(파일개수가 0 개가 아니라면)
                        free_list(list, cnt); // if list allocated, free list
        }
        오픈된 폴더 close
    }
    else 특정 폴더를 2 개 이상 입력했을 경우
        에러메세지 출력
}
```

Result



Makefile 의 내용이다. 파란색의 simple_ls 는 타겟, simple_ls.c 는 타겟명 이다. 아랫줄은 명령어를 입력한 것이며, -g 는 디버깅을 하기 위한 옵션, -o 는 실행 파일 이름을 지정해주는 옵션이다.



컴파일 후 만들어진 실행 파일을 실제로 작동시킨 화면이다. 프로그램을 명령어 없이 실행하였을 때 현재 디렉토리의 파일들을 대,소문자 구분없이 정렬하여 출력하였고, 히든 파일은 출력하지 않았다. 절대 경로, 상대 경로의 폴더 이름을 입력하면 해당 폴더 내부의 파일들을 출력하였고, 입력한 이름의 폴더가 존재하지 않을 경우와 폴더 이름을 두개 이상 입력했을 경우 예외처리를 해주었다.

Conclusion

아직 리눅스 환경에서 작업을 많이 안해봐서 낯설고, shell 에서 명령어를 사용하고, 내가 하고 싶은 명령을 다 못하는 상황에서 막상 가장 대표적인 명령어인 'ls' 를 직접 구현하라고 해서 처음에는 자신이 없었다. 당장 머리 속에 뭐부터 해야 될지 모르겠고, 리눅스에서 하는 거면 윈도우랑 많이 다르지 않을까 하는 걱정이 많이 들었었다.
그런데 막상 코드를 짜려고 해보니 헤더 파일인 <dirent.h>에 필요한 모든게 있어서 걱정했던 것보다 훨씬 수월하게 진행할 수 있었다. 폴더를 열어주고, 파일을 읽어주고, 폴더를 닫아주는 이런 중요한 과정들이 이미 구현이 되어있어서 나는 파일 이름을 모아놓은 배열을 만들고, 그 배열을 정렬시키고, 출력하는 것이 내가 구현 해야할 전부였기에 오랜만에 C 언어의 동적 할당을 다루면서 조금 공부가 필요했던 것 이외에는 어려운 점이 없었다. 오히려 리눅스를 조금은 더 익힐 수 있었던 것 같았다.