

System Programming

(Assignment2-3)

과 목	시스템프로그래밍실습
담당교수	이기훈 교수님
학 과	컴퓨터공학과
학 번	2010720149
성 명	이동현
날 짜	2016. 04. 15 (금)



A. Introduction

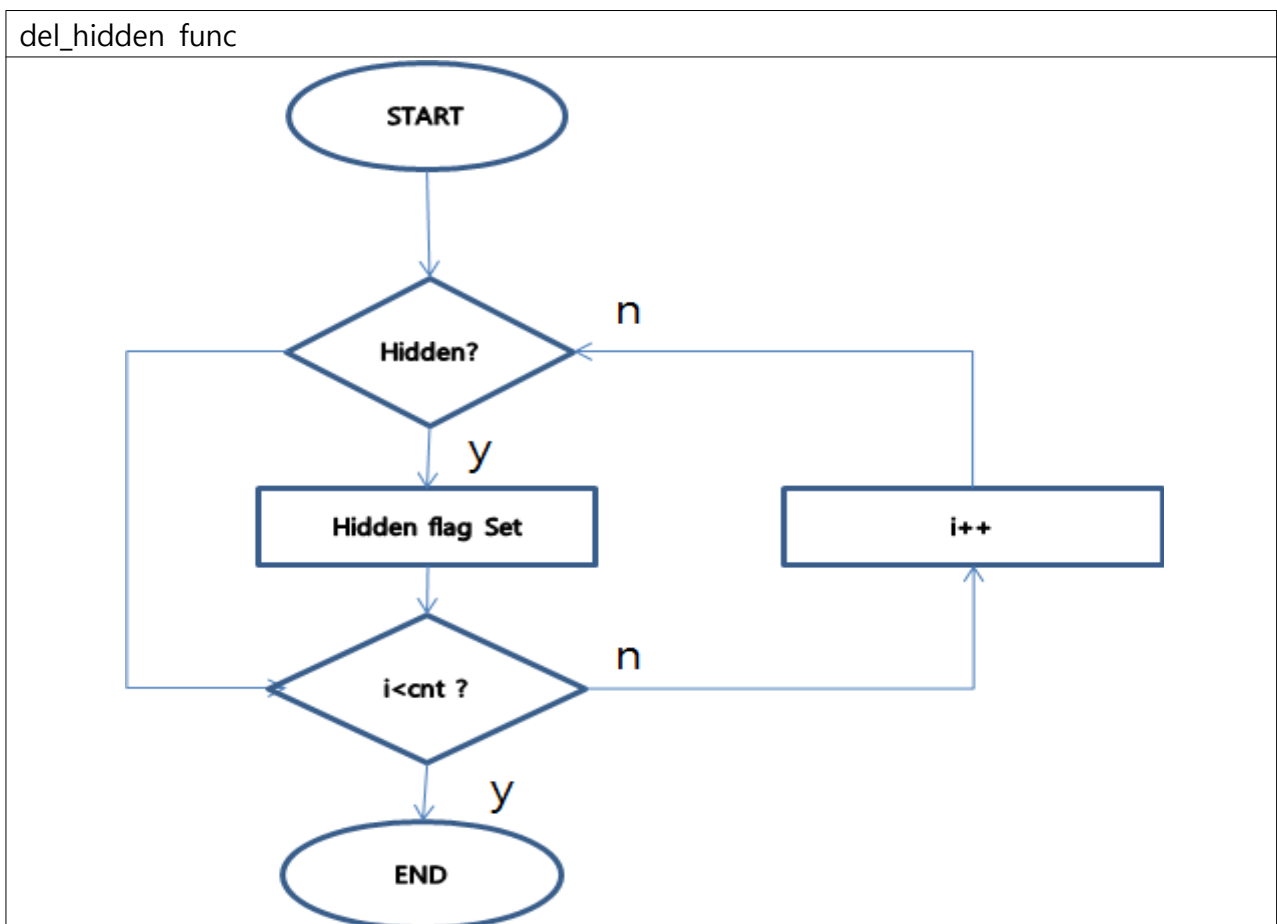
♣ Final ls를 구현하기 ♣

3차 과제에서 Shell에서 현재 Working dirctory list를 출력하는 명령어인 ls를 직접 코드를 구현과 함께 진보 된 -l과 -a 옵션을 포함하여 추가적으로 ls를 구현하였다. final ls는 wildcard matching과 추가적인 -h, -s, -S의 옵션을 구현하여 완성도 높은 명령어 ls를 구현하는데 목적이 있다.

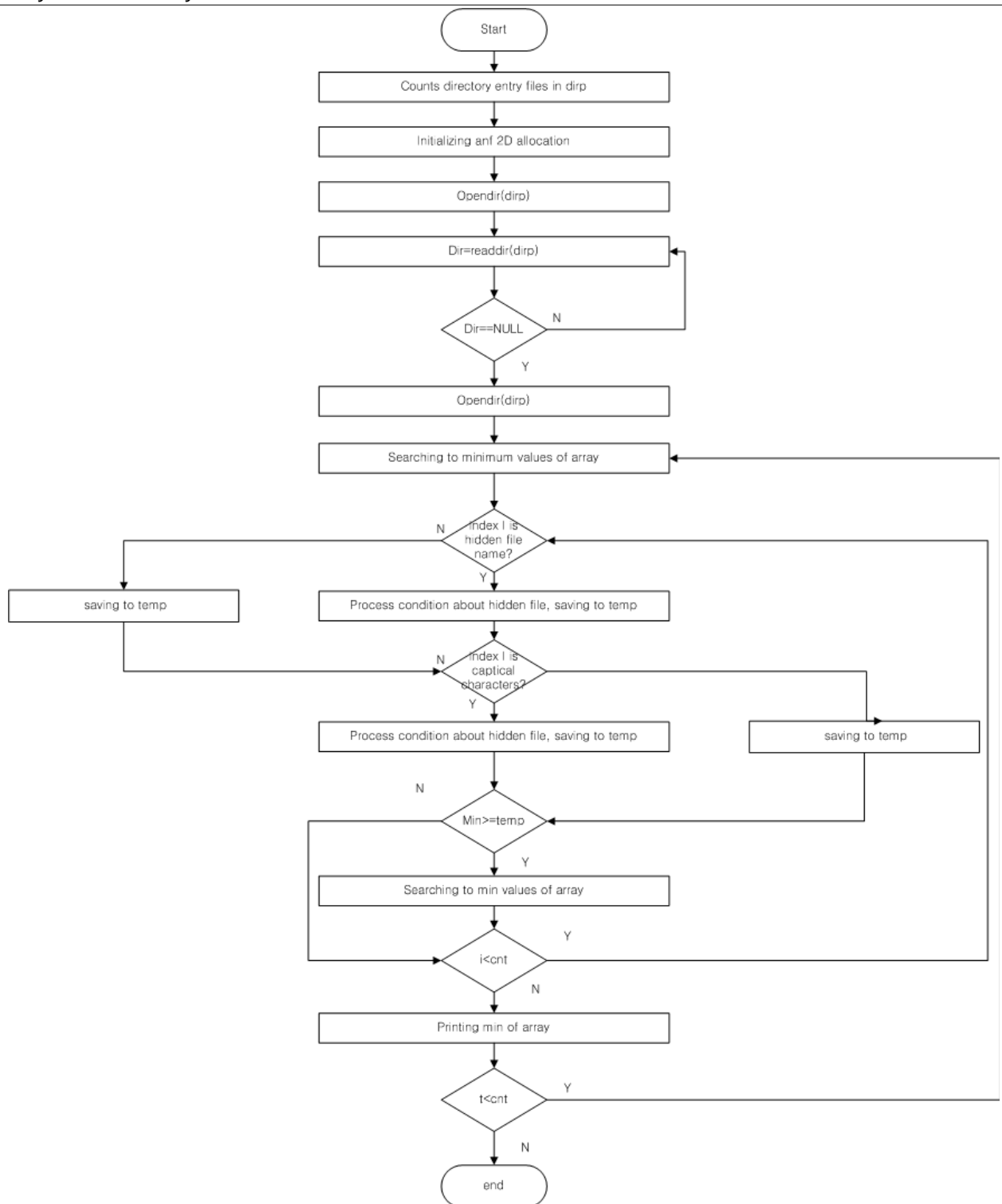
B. Flowchart

함수:

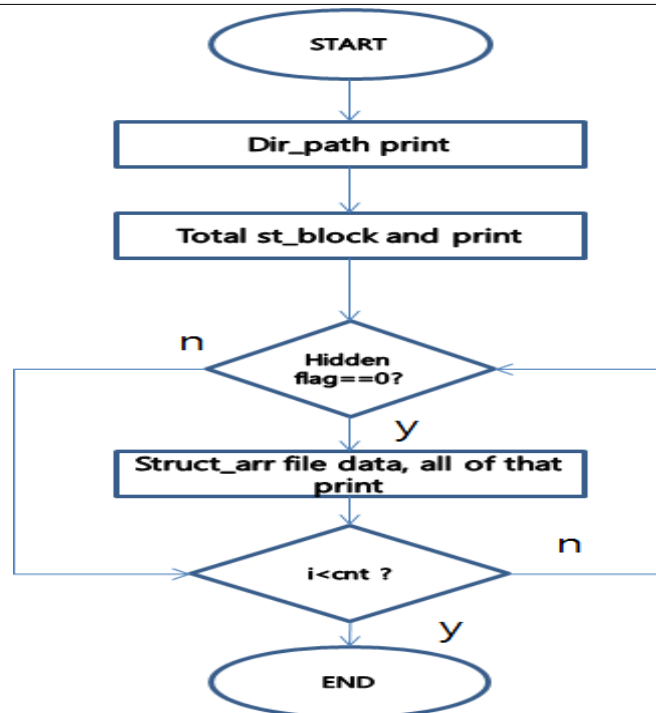
알고리즘 함수	
main func	save func
del_hidden func	array func
S_array func	l_print func
	w_print func
main func	



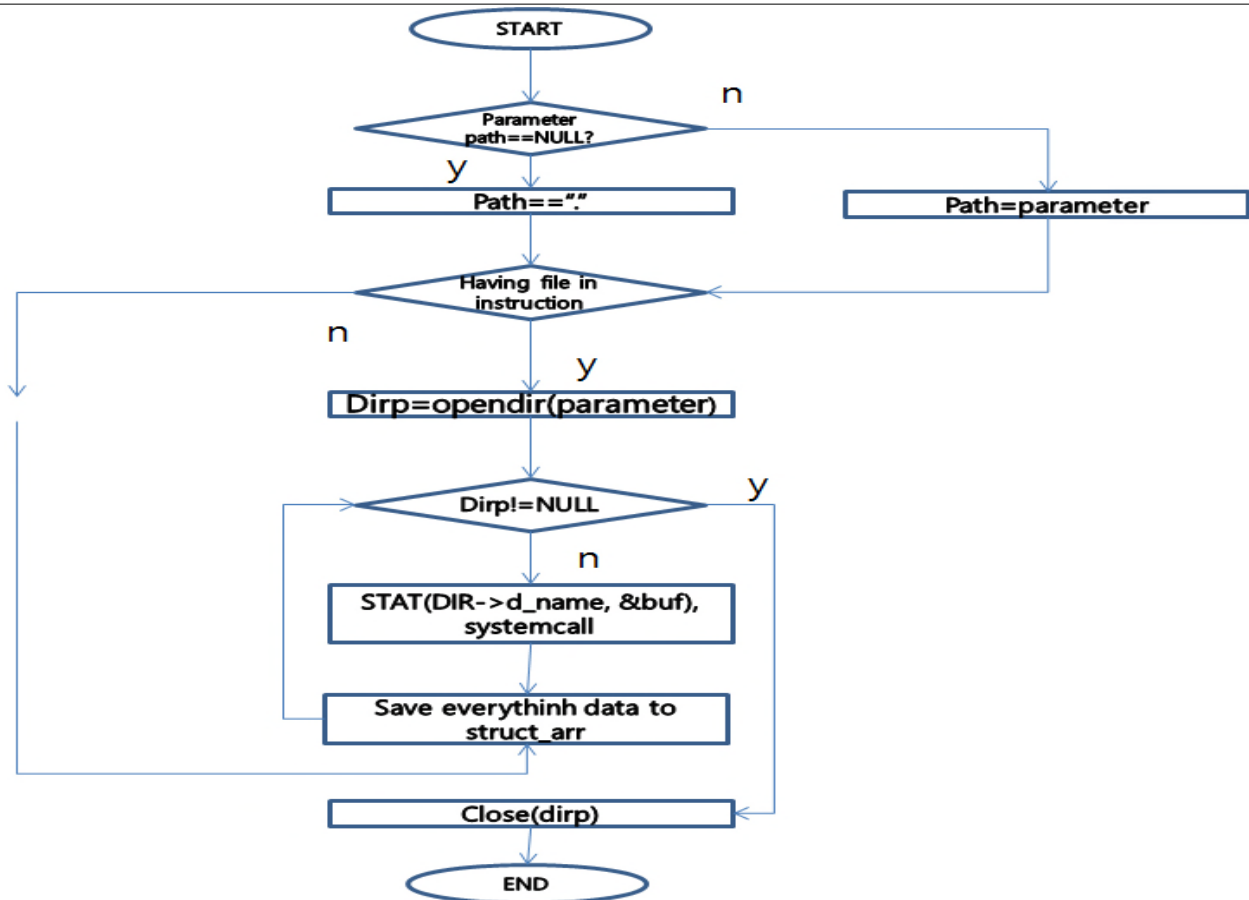
array(sort), S_array와 알고리즘 유사



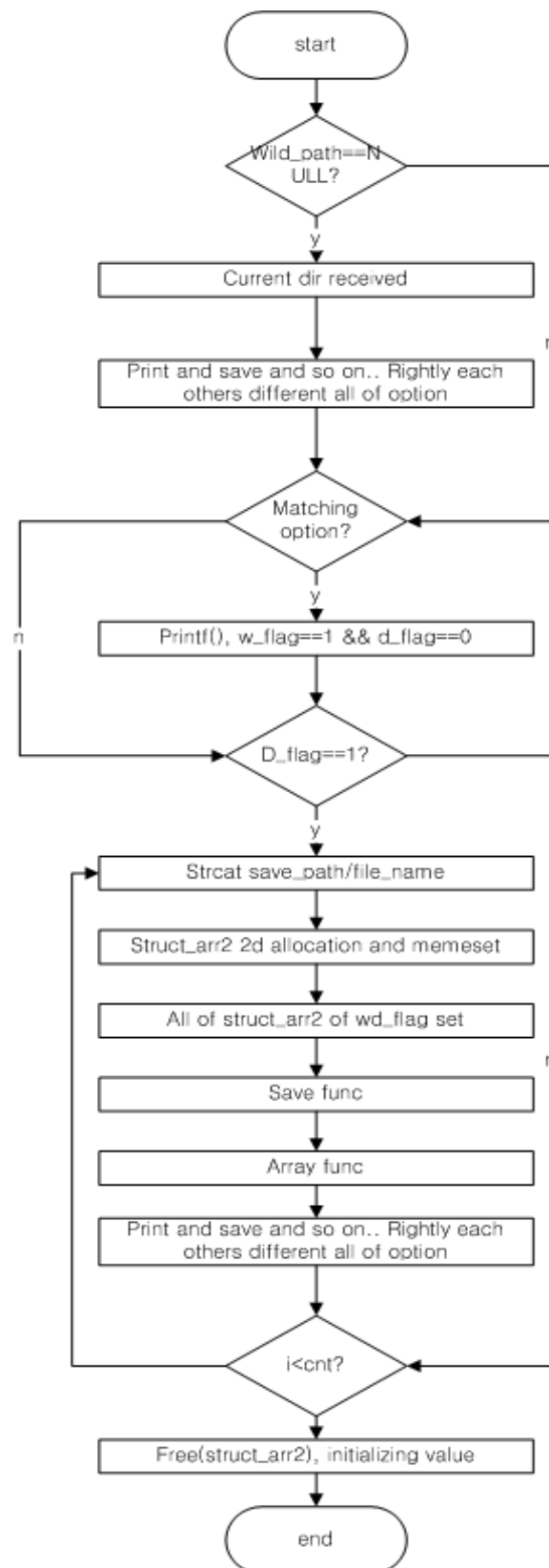
l_print



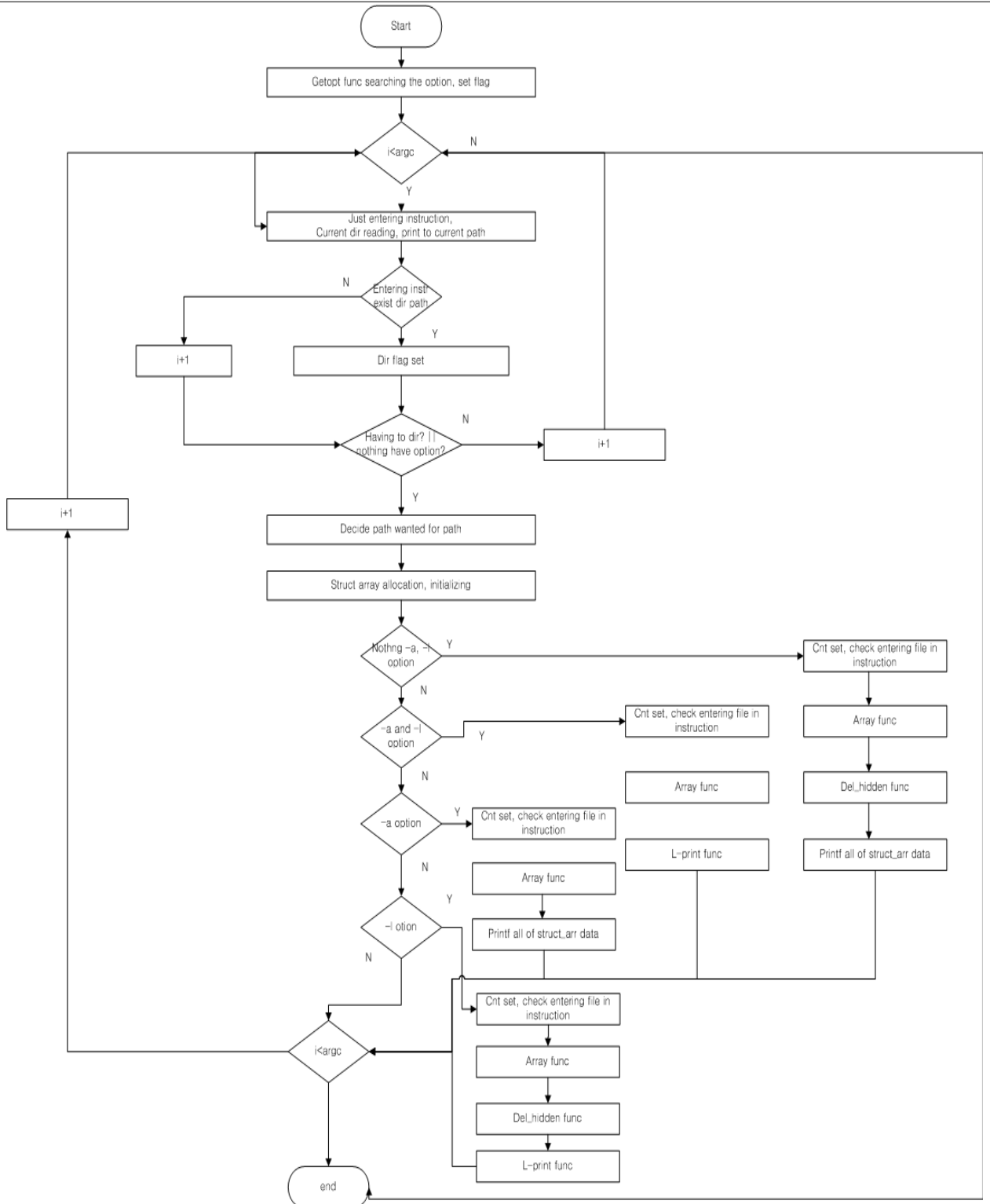
save



w_print()



main()



C. Pseudo code

del_hidden func

```
int del_hidden(struct data* struct_arr, int cnt)
{
    for(i=0 ; i<cnt ; i++)
        if hidden file
            struct_arr[i].hidden flag <-1
    return 0
}
```

array func

```
int main
{
    set file of dir cnt
    allocate struct save[]
    dirp=opendir(".");
    for(dir=readdir(dirp) ; dir ; dir=readdir(dirp))
        save[i], save dir ->d_name;
        i++
    closedir(dirp);
    while(t<cnt)
        for(z=0 ; z<cnt ; z++)
            searching to min
        for(i=0 ; i<cnt ; i++)
            if(save[i]!=NULL)
                if(hidden is)
                    for(j=0 ; save[i][j]!=0 ; j++)
                        temp[j]<=save[i][j+1];
                    for(f=0 ; temp[f]!=0 ; f++)
                        temp[f] change to small letter
                else
                    temp <- save[i]
                    same set, change to small letter
            if(min>=temp)
                min<-temp
                k=i
        printf("%s", save[i]);
        save[k]<- NULL
        t++
        initialzing temp

    free(save)
}
```

l_print func

```
void l_print(struct data* struct_arr, int cnt)
{
    if Dir_path exist
    {
        for(i=0 ; i<cnt ; i++)
            if Dir_path exist
                printf("%s", struct_arr[i].Dir_path);
    }
}
```

```

    printf<- dir_path
else
    cwd <- getcwd(cwd_buf, 1024)
    printf<- cwd, current path
for(i=0 ; i<cnt ; i++)
    if struct_arr[i] of hidden_flag == 0
        total<- total+ struct_arr[i] of st_blocks
printf<- total
if(hflag==1)
    if(total>=G)
        changing number about G and then print
    else if(total<=G && total>=K)
        changing number about K and then print
    else if(total<=K && total>=M)
        changing number about M and then print
    else
        changing number about low num and then print
if wild_flag==0
    for(i=0 ; i<cnt ; i++)
        if struct_arr[i] of hidden_flag == 0
            if sflag==1
                block size print
                printf<- struct_arr[i] of permission by using Mecro code each others
                printf<- else data in struct_arr[i]
                if(h_flag==1)
                    samething print
else
    if struct_arr[i] of hidden_flag == 0
        if struct_arr[i] of hidden_flag == 0
            if struct_arr[i] of hidden_flag == 0
                If sflag==1
                    block size print
                all of samething upper code of if//

```

save func

```

int save(struct data* struct_arr, char* G)
    if G==NULL G= current directory "."
    dirp<-opendir(G) // From NULL read
    cnt++
    close(dirp)
    if cnt==0 printf<- error

    if File_cnt!=0
        dirp<-opendir(G) // From NULL read
        stat(dir->d_name, &buf)

```



```

for(j=0 ; j<File_cnt ; j++)
    if File_temp have anything
        flag<-1;
        struct_arr[i].data<- each of struct. data, everything exe
        i++;
        break;
if i+1==cnt and flag==0
    printf<- no such file or directory
else
    dirp<-opendir(G)
    struct_arr[i].data<- each of struct. data, everything exe
    if wild_flag==1
        if(struct_arr[i].file_name[0]!='.' )
            if(S_ISDIR(stmode))
                d_flag set
            if(fnmatch(wild_str, dir->d_name, 0)==0)
                w_flag set
            if(wdflag==1)
                w_flag set
    closedir(dirp)
return 0;

```

back_strtok()

```

char* back_strtok(char *str)
    length=0
    length=strlen(str)
    for(i=length ; i>=0 ; I--)
        if str[i]=='/'
            break;
    return &str[i+1];

```

w_print()

```

void w_print(struct data* struct_arr, int cnt, char* al_flag)
    if wild_path==NULL
        cwd<-getcwd(cwd_buf, 1024)
        wild_path<-cwd;
    if -l || -al
        Dir_path<-wild_path
        print wild_path
        for(i=0 ; i<cnt ; i++)
            if hidden_flag!=1 && w_flag==1 && d_flag==0
                print file_name
    else if -l || -al

```

```

    Dir_path<-wild_path
    l_print func()
strcpy func save_path<-wild_path

for(i=0 ; i<cnt ; i++)
    if hidden_flag!=1 && w_flag==1 && d_flag==1
        strcat save_path<-save_path+/
        strcat save_path<-save_path+file_name
        save_path open and searching error, w_cnt
        Dir_path<-save_path
        struct_arr2 2D allocation and memset
        for(j=0 ; j<w_cnt ; j++)
            wd_flag set
        save func
        array func
        if nothing op || -l
            del_hidden func
        if nothing op
            for(j=0 ; j<w_cnt ; j++)
                if hidden_flag==0
                    print file name
            else if -a
                for(j=0 ; j<w_cnt ; j++)
                    print file name
            else if -l || -al
                l_print func
        free(struct_arr2)
    initializing value

```

main()

```

int main(int argc, char** argv)
    while((c<-getopt(argc, argv, "alhsS"))!=-1)
        switch(c)
            case a: aflag<-1; break;
            case l: lflag<-1; break;
            case h: hflag<-1; break;
            case s: sflag<-1; break;
            case S: sflag<-1; break;
            case ?: printf<- no option; return 0;
            default: break;

    if argc<2
        dirp<-opendir (".") // from NULL read
        cnt++

```

```
closedir(dirp)
1D allocation struct_arr
initializing struct_arr
if no struct having data
    return 0;
save func;
array func
del_hidden func
for(j=0 ; j<cnt ; j++ )
    if struct_arr[i] of hidden_flag == 0
        printf<- struct_arr[j] of file_name
free(struct_arr)
return
for(i=1 ; i<argc ; i++)
    if wildcard is include in instr
        wild_flag<-1
        wild_str<-argv[i]
        if absolute path
            temp<-"."
        else
            temp<-argv[i]
            back_strtok func call
            cut string saving
    else if Directory existing in instr
        temp<-argv[i]
        dir_flag<-1
    else if file existing in instr
        strcpy file_temp[file_cnt]<- argv[i]
        file_cnt++
if dir_flag==1 or no existing option in instr
    if temp== no option
        dirp<- opendir(".")
    else
        dirp<- opendir(temp)
        dir_path saving
closedir(dirp)
1D allocation struct_arr
initializing struct_arr

if nothing -a and -l
    if no struct having data
        return 0;
    if file_cnt!=0
        cnt=file_cnt
    if Sflag==1
```

```
        S_array func
    if wild_flag==1
        w_print func
    else
        sflag blocksize, filename print code//
        save func;
        array func;
        del_hidden func;
        for(j=0 ; j<cnt ; j++ )
            if struct_arr[i] of hidden_flag == 0
                printf<- struct_arr[j] of file_name
else if -a and -l
    if no struct having data
        return 0;
    if file_cnt!=0
        cnt=file_cnt
    save func;
    array func;
    if Sflag==1
        S_array func;
    if wild_flag==1
        w_print func
    else
        l_print func;
else if -a
    if no struct having data
        return 0;
    if file_cnt!=0
        cnt=file_cnt
    save func;
    array func;
    something aflag=0, lflag=0//
    for(j=0 ; j<cnt ; j++ )
        if struct_arr[i] of hidden_flag == 0
            printf<- struct_arr[j] of file_name
else if -l
    if no struct having data
        return 0;
    if file_cnt!=0
        cnt=file_cnt
    save func;
    array func;
    something aflag=1, lflag=1//
    del_hidden func;
```

```

        l_print func;
        free(struct_arr)

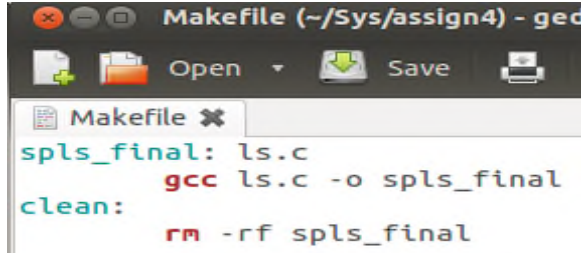
return0;

```

D. Reference

- 강의자료 ' 2016-1_SPLab_06_Final_ls+ls_v2 '
- string.h 함수인 strstr 함수의 설명 // 09학번 임현기

E. Conclusion

♣ 조건 ♣	Makefile
<ul style="list-style-type: none"> - Assignment 2-1 과 2-2 추가 - 와일드카드를 이용해 디렉토리 내용 출력 - ls -h, -s, -S 옵션 추가 구현 	

♣ 결과 화면 1 ♣	h옵션
./spls_final -lh	
<pre>ldh@ubuntu:~/Sys/assign4\$./spls_final /home/ldh -lh Directory path: /home/ldh 433.7M -rw-r--r-- 1 ldh ldh 27.4K 4 11 03:12 core drwxr-xr-x 2 ldh ldh 4.9M 3 12 21:32 Desktop drwxr-xr-x 2 ldh ldh 4.9M 3 12 21:32 Documents drwxr-xr-x 2 ldh ldh 4.9M 3 12 21:32 Downloads drwxrwxr-x 3 ldh ldh 4.9M 3 23 05:57 Embe -rw-r--r-- 1 ldh ldh 8.4M 3 12 21:22 examples.desktop -rw-rw-r-- 1 ldh ldh 0 4 6 06:04 ls.c~ -rw-rw-r-- 1 ldh ldh 50 3 31 06:32 Makefile -rw-rw-r-- 1 ldh ldh 48 3 31 06:32 Makefile~ drwxr-xr-x 2 ldh ldh 4.9M 3 12 21:32 Music drwxr-xr-x 2 ldh ldh 4.9M 3 12 21:32 Pictures drwxr-xr-x 2 ldh ldh 4.9M 3 12 21:32 Public drwxr-xr-x 32 ldh ldh 4.9M 4 4 07:04 qt-everywhere-op source-src-5.4.2 -rw-r--r-- 1 ldh ldh 416.6K 6 1 02:41 qt-everywhere-op source-src-5.4.2.tar.gz drwxrwxr-x 8 ldh ldh 4.9M 4 6 20:14 Sys drwxr-xr-x 2 ldh ldh 4.9M 3 12 21:32 Templates drwxr-xr-x 2 ldh ldh 4.9M 3 12 21:32 Videos</pre>	h 옵션은 st_size를 M, K, G에 맞게 수치화하여 출력하는 것으로, total size 또한 이와 맞게 출력을 한다.

♣ 결과 화면 2 ♣	s옵션
./spls_final -s	
<pre>ldh@ubuntu:~/Sys/assign4\$./spls_final -s total: 172 4 k.c 24 ls 28 ls.c 28 ls.c~ 0 ls.h 4 Makefile 4 Makefile~ 28 spls_final 24 spls_final.c 24 spls_final.c~ 4 thth</pre>	-s 옵션은 그 디렉토리의 파일 이름을 출력하되, block size를 함께 출력하고, total size또한 맨 위에 출력을 한다.

♣ 결과 화면 3♣	s옵션
./spls_final -sl	
<pre>ldh@ubuntu:~/Sys/assign4\$./spls_final -sl)irectory path: /home/ldh/Sys/assign4 total: 172 4 drwxrwxr-x 2 ldh ldh 4096 4 12 08:59 k.c 24 -rwxrwxr-x 1 ldh ldh 20622 4 13 09:50 ls 28 -rw-rw-r-- 1 ldh ldh 26892 4 14 09:26 ls.c 28 -rw-rw-r-- 1 ldh ldh 26893 4 14 09:26 ls.c~ 0 -rw-rw-r-- 1 ldh ldh 0 4 13 10:58 ls.h 4 -rw-rw-r-- 1 ldh ldh 67 4 13 10:03 Makefile 4 -rw-rw-r-- 1 ldh ldh 44 4 12 09:39 Makefile~ 28 -rwxrwxr-x 1 ldh ldh 24786 4 14 09:26 spls_final 24 -rw-rw-r-- 1 ldh ldh 24239 4 13 10:52 spls_final .c 24 -rw-rw-r-- 1 ldh ldh 24065 4 13 10:09 spls_final .c~ 4 drwxrwxr-x 2 ldh ldh 4096 4 12 12:07 thth ldh@ubuntu:~/Sys/assign4\$</pre>	-s 옵션에서 추가적인 l옵션과 함께 예를 보여주었다. -l 옵션의 출력중 맨 앞에 block size를 출력을 하도록 하였다.

♣ 결과 화면 4♣	S옵션
./spls_final -sS ./spls_final -Sl ./spls_final -S	
<pre>ldh@ubuntu:~/Sys/assign4\$./spls_final -sS total: 172 28 spls_final 28 ls.c~ 28 ls.c 24 spls_final.c~ 24 spls_final.c 24 ls 4 thth 4 Makefile~ 4 Makefile 4 k.c 0 ls.h ldh@ubuntu:~/Sys/assign4\$</pre>	S 옵션은 대문자이며, s옵션에서 보여준 block size를 정렬하여 출력하는 명령어이다. 파일 이름의 정렬을 무시하고 block size에 우선이 있다. S 옵션은 보이지 않게 block size만 정렬하기 때문에 화면으로 확인 하기 위해선 -s 옵션이나 -l을 통해 block size를 확인하여 알 맞게 정렬 되었음을 확인 할 수 있다.
<pre>ldh@ubuntu:~/Sys/assign4\$./spls_final -Sl)irectory path: /home/ldh/Sys/assign4 total: 172 -rwxrwxr-x 1 ldh ldh 24786 4 14 09:26 spls_final -rw-rw-r-- 1 ldh ldh 26893 4 14 09:26 ls.c~ -rw-rw-r-- 1 ldh ldh 26892 4 14 09:26 ls.c -rw-rw-r-- 1 ldh ldh 24065 4 13 10:09 spls_final.c~ -rw-rw-r-- 1 ldh ldh 24239 4 13 10:52 spls_final.c -rwxrwxr-x 1 ldh ldh 20622 4 13 09:50 ls drwxrwxr-x 2 ldh ldh 4096 4 12 12:07 thth -rw-rw-r-- 1 ldh ldh 44 4 12 09:39 Makefile~ -rw-rw-r-- 1 ldh ldh 67 4 13 10:03 Makefile drwxrwxr-x 2 ldh ldh 4096 4 12 08:59 k.c -rw-rw-r-- 1 ldh ldh 0 4 13 10:58 ls.h ldh@ubuntu:~/Sys/assign4\$</pre>	
<pre>ldh@ubuntu:~/Sys/assign4\$./spls_final -S spls_final ls.c~ ls.c spls_final.c~ spls_final.c ls thth Makefile~ Makefile k.c ls.h ldh@ubuntu:~/Sys/assign4\$ █</pre>	

♣ 결과 화면 5 ♣	wild_card
<pre>./spls_fianl '*.c'</pre> <pre>ldh@ubuntu:~/Sys/assign4\$./spls_final '*.c'</pre> <pre>Directory path: /home/ldh/Sys/assign4</pre> <pre>ls.c</pre> <pre>spls_final.c</pre> <pre>Directory path: k.c</pre> <pre>hihi</pre> <pre>ldh@ubuntu:~/Sys/assign4\$./spls_final '*.c'</pre>	<p>다음화면은 와일드카드 *로 .c에 해당하는 모든 파일을 찾고 출력하는 명령어이다. 추가적인 옵션은 없고 현재 디렉토리에서 명령어를 수행했기 때문에 ls.c 와 spls_final.c를 찾았음을 확인할 수 있고, Directory는 k.c가 존재함을 확인할 수 있다.(c파일 아님) path와 그 안의 파일을 출력함을 보여주었다.</p>
<pre>./spls_fianl '*.c' -al</pre> <pre>gcc ls.c -o spls_final</pre> <pre>ldh@ubuntu:~/Sys/assign4\$./spls_final '*.c' -al</pre> <pre>Directory path: /home/ldh/Sys/assign4</pre> <pre>total: 168</pre> <pre>-rw-rw-r-- 1 ldh ldh 24241 4 13 10:53 ls.c</pre> <pre>-rw-rw-r-- 1 ldh ldh 24239 4 13 10:52 spls_final.c</pre> <pre>Directory path: k.c</pre> <pre>Directory path: /home/ldh/Sys/assign4/k.c</pre> <pre>total: 8</pre> <pre>drwxrwxr-x 2 ldh ldh 4096 4 12 08:59 .</pre> <pre>drwxrwxr-x 4 ldh ldh 4096 4 13 10:53 ..</pre> <pre>-rw-rw-r-- 1 ldh ldh 0 4 12 08:59 hihi</pre> <pre>ldh@ubuntu:~/Sys/assign4\$</pre>	<p>이 것은 위에 동일하지만 a와 l옵션을 추가한 것으로 .. 히든파일과 데이터 정보를 보여주는 것과 함께 아래와 같이 와일드카드로 찾은 디렉토리에선 확인할 수 있다.</p>
<pre>ldh@ubuntu:~/Sys/assign4\$ ls *.c -al</pre> <pre>-rw-rw-r-- 1 ldh ldh 26892 Apr 14 09:26 ls.c</pre> <pre>-rw-rw-r-- 1 ldh ldh 24239 Apr 13 10:52 spls_final.c</pre> <pre>k.c:</pre> <pre>total 8</pre> <pre>drwxrwxr-x 2 ldh ldh 4096 Apr 12 08:59 .</pre> <pre>drwxrwxr-x 4 ldh ldh 4096 Apr 14 09:26 ..</pre> <pre>-rw-rw-r-- 1 ldh ldh 0 Apr 12 08:59 hihi</pre> <pre>ldh@ubuntu:~/Sys/assign4\$</pre>	
< 비교 실제 ls 화면 >	
♣ 결과 화면 6 ♣	
<pre>./spls_final 'ls??'</pre> <pre>ldh@ubuntu:~/Sys/assign4\$./spls_final 'ls??'</pre> <pre>Directory path: /home/ldh/Sys/assign4</pre> <pre>ls.c</pre> <pre>ldh@ubuntu:~/Sys/assign4\$ ls ls??</pre> <pre>ls.c</pre> <pre>ldh@ubuntu:~/Sys/assign4\$</pre>	<p>이번에는 다른 와일드카드를 찾는 명령어로 ?는 어떤 문자든 상관없음은 *와 동일하나 글자의 수에 영향을 미친다. 따라서 ls.c만 나옴을 실제 ls명령어와 동일함을 알 수 있다.</p>

♣ 결과 화면 7 ♣

./spls_final 'ls.[^c]'

```
ldh@ubuntu:~/Sys/assign4$ ls ls.[^c]
ls.h
ldh@ubuntu:~/Sys/assign4$ ./spls_final 'ls.[^c]'
```

Directory path: /home/ldh/Sys/assign4
ls.h
ldh@ubuntu:~/Sys/assign4\$

이번 와일드 카드는 [str]의 박스에 담긴 것으로 ^c는 ^d을 not이라고 생각하고 c가 아닌 파일을 찾아낸다. 그래서 다음 과 같은 출력을 보여준다.

♣ 결과 화면 8 ♣

./spls_final ~ '*'

```
ldh@ubuntu:~/Sys/assign4$ ./spls_final ~ '*'
core
Desktop
Documents
Downloads
Embe
examples.desktop
ls.c~
Makefile
Makefile~
Music
Pictures
Public
qt-everywhere-opensource-src-5.4.2
qt-everywhere-opensource-src-5.4.2.tar.gz
Sys
Templates
Videos
```

```
Directory path: /home/ldh/Sys/assign4
ls
ls.c
ls.c~
ls.h
Makefile
Makefile~
spls_final
spls_final.c
spls_final.c~

Directory path: k.c
hihi

Directory path: thth
```

이번은 두가지를 수행하는 결과를 보여준다. ~는 로그인시 현재 디렉토리의 path를 가진 문자이다. 우선 위의 문자대로 먼저 출력됨을 볼 수 있고, 그다음 '*'에 대한 와일드 카드를 찾아 해당 디렉토리의 path에 대해서 모든 파일을 출력함을 확인 할 수 있다. directory가 k.c와 thth가 존재했기 때문에 따로 아래에 path와 함께 출력이 되었고, 그 안에 파일 내용들을 모두 출력할 수 있게 한다. 이 경우는 절대 경로와 상대 경로를 모두 보여준 예 이다.

♣ 결과 화면 9 ♣

`./spls_final /home/ldh/'*'`

```
ldh@ubuntu:~/Sys/assign4$ ./spls_final /home/ldh/'*'
```

```
Directory path: /home/ldh
core
examples.desktop
ls.c~
Makefile
Makefile~
qt-everywhere-opensource-src-5.4.2.tar.gz
```

```
Directory path: Desktop
```

```
Directory path: Documents
```

```
Directory path: Downloads
```

```
Directory path: Embe
assignment1
```

```
Directory path: Music
```

```
Directory path: Pictures
```

```
Directory path: Public
```

```
Directory path: qt-everywhere-opensource-src-5.4.2
configure
configure.bat
cscope.out
gnuwin32
LICENSE_EXCEPTION.txt
LICENSE.FDL
LICENSE.GPLv2
LICENSE.LGPLv21
LICENSE.LGPLv3
LICENSE.PREVIEW.COMMERCIAL
Makefile
qt.pro
qtactiveqt
qtactiveqt.exe
```

```
...
```

이번은 절대 경로에서의 와일드카드 '*'을 출력하는 예이다.

해당 path /home/ldh에서 모든 파일을 출력하되 directory도 그리고 그 안의 파일도 모두 잘 출력 되었음을 확인할 수 있다.

너무 많은 파일을 소유하고 있기 때문에 ... 으로 표현하였다.

♣ 고찰 ♣

마지막 ls의 과제를 마치면서 완성에 대한 즐거움과 간단한 shell 명령어를 구현할 수 있음에 좋았다. 이번과제는 여태 1, 2차를 수행하는 과제에서 추가적으로 wildcard를 이용한 데이터를 찾거나, -h, -s, -S의 옵션을 통해 print할 시 사용자가 더 필요한 정보를 얻는데 효율적인 옵션을 추가하는 것이었다. 2차 과제에서 코드가 완벽하고 깔끔하지 못해서 추가적인 구현하기에 너무 어렵다고 판단하여, 2차 과제부터 다시 시작하는 느낌으로 시작했다. 더 큰 그림을 보고 나중의 추가적인 옵션을 구현할 시 더 간단하게 구현을 하기 위해 함수의 사용과 전역변수의 사용을 하여 가시성을 높이는데 중점을 두었다. 그래서 3차를 구현을 할 때, flag와 fnmatch 시스템 콜 함수를 적용하기가 수월했고, -h -s, -S 또한 마찬가지였다. 과제 진행 중에 오류를 범했던 것 중 하나는 DIR의 구조체를 사용하여 파일을 Open하였을 때, close를 빼먹곤하는 일이 종종 있다. 사용자에게 에러 발생을 찾아주는 데 부족한 Linux에서는 c++과 다르게 더욱 찾기가 어려웠다. 메모리 문제로 stack.. 관하여 매우 복잡한 언어가 발생을 했다. 뿐 만아니라 동적할당 등 메모리 접근에 더 유의할 필요가 있다고 생각이 들었다. coding 부분에서는 string 문자열 중에서 선택된 문자를 찾기 위한 방법에서 시간이 걸렸었는데, 그 것을 쉽게 해결한 함수가 존재 했다. string.h에 가진 strstr함수였다. 이 함수는 해당 문자열에서 원하는 문자를 찾으면 해당 문자가 존재하면 1을 반환하기 때문에, wildcard 문자 '*', '[str]', '?'을 찾아내기 수월했고 이어 프로그래밍을 진행하는데 큰 도움이 되었다.

그리고 이번 과제는 하나 하나의 과제를 추가적으로 구현을 하는 것이기 때문에, 어떤 구조를 짜냐에 따라서 추가적인 구현이 쉽게 가능하였다. 처음 과제는 함수를 이용하지 않고 이용을 했다가 2차과제에서 큰 봉변을 맞았다. 함수를 이용하여 중복처리가 가능한 것을 최대한 활용하고, main함수는 최대한 깔끔하게 인자를 넘겨주고 받아 처리하는 시스템이 가장 중요할 것이라고 생각했다. 그것이 앞으로 추가적인 구현이나 남들이 확인할 때 인터페이스가 훨씬 유용할 것이라는 생각이 든다.

ls은 디렉토리의 list를 보여주는 shell 명령어로 간단한 기능을 수행한다고 생각을 했지만, 막상 구현을 하게 되면 매우 복잡하면서도 많은 system call을 활용한 복잡한 명령어라고 생각이 들었다. 그래서 예외처리와 에러처리할 것이 너무나 많았고, 계속 수정해도 코드가 꼬이는 경우도 많았다. 한가지 과제의 아쉬운 점이 한 차례씩 과제를 수행할 때, 모든 과제에 대해서 임의적인 설명이 있었다면, 초반에 기반을 세우고 예외처리 하기 위한 섬세한 코드가 더욱 가능 할 것이라는 생각이 들었다.

이번 과제를 수행하면서 리눅스의 c언어 기반의 인터페이스를 익힐 수 있었고, 기본 명령어를 구현함으로써 리눅스 체계의 커널에 연관된 시스템 콜과 프로세스에 해당하는 명령어들의 수행 결과 등 여러 가지를 확인 할 수 있었고, 다음 과제에도 큰 도움이 될 발판의 과제였다.