I. 개요

리눅스/유닉스에서 cp(copy) 명령어는 파일(또는 디렉토리)을 복사시킨다.
cp [*options*] *source dest*의 형식으로 명령어를 쓸 수 있다. 복사된 파일은 사용자의 umask 값에 따라서 권한이 결정되고, 옵션(-p)으로 원본파일과 동일하게 변경 할 수 있다. 또한 dest가 이미 존재하는 경우에는 옵션에 따라서 덮어쓰거나, 백업파일을 생성하는 등 다양한 방식으로 source 파일을 복사할 수 있다.

II. 설계

가) 목표설정

・ssu_cp : 기본 리눅스 명령어 cp와 같이 source를 dest의 이름으로 복사하는 기능이 구현되어 있도록 한다. 형식은 cp [*options*] *source dest*이고, source에는 파일과 디렉터리를 모두 입력할 수 있는데 디렉터리는 디렉터리 관련 옵션이 있어야 실행을 할 수 있다. ssu_cp를 실행하는 과정에서 발생하는 오류는 오류의 원인과 ssu_cp의 명령어 사용법을 출력해서 처리한다. ssu_cp의 옵션은 [*source*]파일(또는 디렉터리) 내용 출력, 덮어쓰기, 디렉터리 복사, [*source*]파일에 대한 심볼링 링크 생성등 다양하게 사용 할 수 있다.

나) 구조 설계

main 함수 :

① option( ) : main함수의 인자들을 받아 [*source*]와 [*target*] 파일을 지정한 후, option()함수를 통해 명령 줄(command line)에서 받은 옵션들을 처리한다.

② ssu_copy( ), option_r( ) : 옵션에 맞게 ssu_cp를 실행하는데, [*source*]의 파일 모드가 디렉터리인 경우 디렉터리 관련 옵션 함수들을 호출하여 복사하고, [*source*]의 파일 모드가 일반 파일인 경우 ssu_copy( )함수를 호출하여 파일을 복사한다. [*target*]에 해당하는 파일이 이미 있는 경우 기본적으로 덮어쓰기를 한다.

[*option*] 처리 함수 :

① ssu_option_s( ) : [*source*]에 대한 심볼링 링크를 생성한다.

② ssu_option_l( ) : [*source*] 파일의 소유주, 그룹, 권한, 시간 정보를 [*target*]에 같이 복사한다.

③ ssu_option_r( ) : [*source*]가 디렉터리인 경우에만 실행 될 수 있다. [*source*] 디렉터리 뿐만 아니라 하위 디렉터리와 파일들 모두 [*target*]에 복사한다.

④ ssu_option_r( ) : [*source*] 파일의 파일 이름, 데이터의 마지막 읽은 시간, 수정 시간, 변경 시간, owner 아이디 이름, group 아이디 이름, 파일 크기를 출력 후 [*target*]에 복사한다.

⑤ ssu_option_d( ) : [*source*]가 디렉터리인 경우에 쓸 수 있는 옵션함수이다. [*source*] 디렉터리를 [*target*]에 복사한다.

III. 구현 (각 함수별 기능)

< ssu_cp.c >

가) 전역 변수

① int on : 옵션이 하나도 적용되지 않은 경우 on = 1, 옵션이 추가될 때마다 +1씩 증가한다.

② int s, i, l, n, p, r, d : 옵션 s, I, l, n, p, r, d를 의미

나) int main(int argc, char *argv[])

① 매개 변수 : argc, argv[]

i. argc : 명령 줄에서 입력된 명령행 옵션들의 개수를 저장한다.

ii. argv[] : 명령 줄에서 입력된 옵션들과 *source*, *target*을 받아온다.

② 기능

source 파일과 *target* 파일을 명령 줄에서 입력받고, *ssu*_cp의 전반적인 함수들을 호출하는 main 함수이다.

option()함수를 실행 후, '-i',이 적용되고 해당 *target* 파일이 이미 존재하는 경우 사용자에게 덮어쓸 지 물어본다. 덮어쓰기를 입력한 경우('y'), ssu_copy()함수를 호출하여 진행하고, 그렇지 않은 경우 종료한다.

옵션 '-l'이 적용된 경우 ssu_copy() 함수를 복사가 이루어진 후 option_l()을 호출하여 source 파일의 파일 권한, 사용자 id, 그룹 id, 시간 정보 등을 target에 지정한다.

다) void option(int OptNum, char *Option[])

① 매개 변수 : OptNum, Option[]

i. OptNum, Option[] : 명령 줄에서 받은 여러 명령어들과 그 명령어들의 개수를 인자로 받아 옵션들을 구분하고 처리한다.

② 기능

getopt()함수를 통해 옵션들을 차례대로 처리한다. 옵션이 적용 될 때 그에 해당하는 전역변수를 1로 설정하고, 해당 옵션이 중복되어서 입력되었는지 확인해 중복되었을 경우, 에러 처리를 한다. '-s'인 경우, 단독으로만 쓰일 수 옵션이므로 OptNum이 4개가 아닌 경우(./ssu_cp, [-s] [*source*] [*target*]) 에러처리를 한다.

라) void ssu_copy(char *source, char *target)

① 매개 변수 : source, target

i. source, target : 사용자가 입력한 [*source*]파일과 [*target*]파일을 매개변수로 받아온다.

② 기능

인자로 받은 source파일을 target의 이름으로 read()함수를 이용하여 1byte 씩 복사한다. source의 파일 모드가 디렉터리인데 '-r' 또는 '-d'의 옵션이 적용되지 않았을 경우 에러처리를 한다. '-i'의 옵션이 적용되고 사용자가 'y'를 입력했을 경우를 위해 파일 플래그 O_TRUNC를 포함해서 target을 open()함수를 통해 생성한다(또는 연다).

마) void print_usage()

① 기능 : 에러 처리 시 사용법 출력

< ssu_option_l.c >

가) void option_l(char* target, struct stat src_stat)

① 매개 변수 : target, src_stat

i. src_stat : *source* 파일의 파일 정보가 담긴 구조체를 인자로 받는다.

② 기능 : 옵션 '-l' 처리

src_stat 내에 저장된 st_mode, st_atime, st_mtime, st_uid, st_gid를 이용하여 target의 파일정보를 변경한다. target의 파일 권한은 chmod() 함수를 이용하여 변경한다. 시간 정보는 utimebuf 구조체에 시간 정보를 저장한 후, utime() 함수를 이용하고, 사용자 id와

그룹 id는 chown()함 수를 통해 변경한다.

옵션 '-l'이 적용 된 경우 main() 함수 내에서 파일 복사 또는 디렉터리 복사를 모두 끝낸 후 option_l()을 호출을 하여 target 정보를 변경한다.

< ssu_option_p.c >

가) void option_p(char* src)

① 매개 변수 : src

i. src : source 파일을 인자로 받아와 함수 내에서 이 파일의 정보를 읽어낸다.

② 기능 : 옵션 '-p' 처리

stat() 함수를 통해 src의 파일 정보를 stat 구조체에 저장한다. 구조체 내 멤버 변수st_atime, st_mtime, st_ctime, st_uid, st_gid에 저장된 값을 이용하여 파일 정보를 출력한다. 사용자 id와 그룹 id를 출력할 때는 stat 구조체에 저장된 uid와 gid는 정수형이므로 getpwuid()함수와 getgrgid()함수를 이용하여 문자열의 형태로 출력한다.

<ssu_option_r.c>

가) void option_r(char* src, char* tar)

① 매개 변수 : src, tar

② 기능 : 옵션 '-r' 처리

readdir()함수를 통해 source 디렉터리 내 하위 파일들을 차례대로 받고 디렉터리의 끝까지 읽은 경우 종료한다. 일반 파일의 경우 ssu_copy()함수를 호출하여 target 디렉터리에 복사한다. 그리고 해당 파일이 디렉터리인 경우 해당 파일의 이름으로 option_r()을 재귀 호출하여 모든 하위 파일들과 디렉터리들을 복사한다. 또한 target 디렉터리가 source 디렉터리 내에 있을 경우 그곳에 복사한다.

<ssu_option_s.c>

가) void option_s(char* src, char* tar)

① 매개 변수 : src, tar

② 기능 : 옵션 '-s' 처리

symlinke()함수를 이용하여 source 파일에 대한 심볼링 링크를 생성한다.

IV. 테스트 및 결과

가) ssu_cp [SOURCE] [TARGET] : source 파일을 target 이름으로 복사

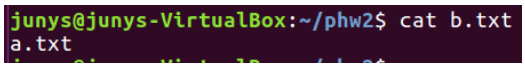

그림 1 – a.txt의 내용이 복사된 b.txt

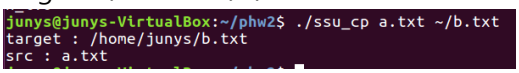

그림 2 – a.txt를 절대경로 ~/b.txt에 복사한다.

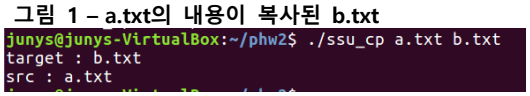

그림 3 – a.txt를 b.txt에 복사한다.

나) ssu_cp [OPTION] [SOURCE] [TARGET]

① 옵션이 적용된 경우 해당 옵션을 출력함

② -s

```
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -s a.txt a_symlink.txt
target : a_symlink.txt
src : a.txt
s option is on
junys@junys-VirtualBox:~/phw2$ ls -l a_symlink.txt
lrwxrwxrwx 1 junys junys 5  4월 23 22:52 a_symlink.txt -> a.txt
```

```
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -s a_dir dir_symlink
target : dir_symlink
src : a_dir
s option is on
junys@junys-VirtualBox:~/phw2$ ls -l dir_symlink
ls: cannot access 'dir_symlink': No such file or directory
```

그림 4 - -s 옵션 적용이 된 경우 옵션 메시지를 출력 후, a.txt에 대한 심볼릭 링크 파일 a_symlink.txt를 생그림 5 - [*source*]가 디렉터리인 경우 생략한다.

③ -i

```
junys@junys-VirtualBox:~/phw2$ cat b.txt
a.txt
junys@junys-VirtualBox:~/phw2$ cat c.txt
c.txt
junys@junys-VirtualBox:~/phw2$ ./ssu_cp c.txt b.txt
target : b.txt
src : c.txt
junys@junys-VirtualBox:~/phw2$ cat b.txt
c.txt
```

```
junys@junys-VirtualBox:~/phw2$ cat b.txt
a.txt
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -i c.txt b.txt
target : b.txt
src : c.txt
i option is on
overwrite b.txt (y/n)? y
junys@junys-VirtualBox:~/phw2$ cat b.txt
c.txt
```

그림 6 – 기존의 파일이 있는 경우 기본적으로 덮어쓴 그림 7 – 옵션 i가 적용되었을 때 기존 파일이 있는 경우 사용자에게 확인한다.

④ -n

```
junys@junys-VirtualBox:~/phw2$ ls -l a.txt
-rw-rw-rw- 1 junys junys 6  4월 20 21:04 a.txt
junys@junys-VirtualBox:~/phw2$ cat b.txt
c.txt
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -n a.txt b.txt
target : b.txt
src : a.txt
n option is on
junys@junys-VirtualBox:~/phw2$ cat b.txt
c.txt
```

그림 8 – 기존 파일이 존재하는 경우 덮어쓰기를 수행하지 않는

⑤ -l

```
junys@junys-VirtualBox:~/phw2$ chmod 0666 a.txt
junys@junys-VirtualBox:~/phw2$ ls -l a.txt
-rw-rw-rw- 1 junys junys 6  4월 20 21:04 a.txt
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -l a.txt a_l.txt
target : a_l.txt
src : a.txt
l option is on
junys@junys-VirtualBox:~/phw2$ ls -l a_l.txt
-rw-rw-rw- 1 junys junys 6  4월 20 21:04 a_l.txt
```

그림 7 – [*source*] 파일의 소유주, 그룹, 권한, 시간정보를 보존하여 [*target*]에 복사

⑥ -p

```
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -p a.txt a_p.txt
target : a_p.txt
src : a.txt
p option is on
**********file info**********
파일 이름 : a.txt
데이터의 마지막 읽은 시간 : 2018.4.23 23:49:55
데이터의 마지막 수정 시간 : 2018.4.23 23:49:55
데이터의 마지막 변경 시간 : 2018.4.23 23:49:55
OWNER : junys
GROUP : junys
file size : 6
```

그림 8 – [*source*]파일에 대한 정보를 출력하고 ssu_cp를 수행한다.

⑦ -r

```
junys@junys-VirtualBox:~/phw2$ ls -al ~/phw_cp
total 148
drwxrwxr-x  7 junys junys  4096  4월 17 23:32 .
drwxr-xr-x 35 junys junys  4096  4월 24 00:12 ..
-rw-rw-r--  1 junys junys     0  4월 13 19:06 a
drwxrwxr-x  3 junys junys  4096  4월 13 19:06 abc
-rwxrwxr-x  1 junys junys 13752  4월 13 19:06 a.out
-rw-rw-r--  1 junys junys    11  4월 13 19:06 a.txt
lrwxrwxrwx  1 junys junys     5  4월 13 19:06 b -> a.txt
drwxrw-r--  2 junys junys  4096  4월 13 19:06 cp2
drwxrw-r--  4 junys junys  4096  4월 13 19:06 dir
-rw-rw-r--  1 junys junys   516  4월 13 19:06 Makefile
-rw-rw-r--  1 junys junys    22  4월 13 19:06 src_link.c
-rwxrwxr-x  1 junys junys 18584  4월 17 23:27 ssu_cp
-rw-rw-r--  1 junys junys  4523  4월 13 19:06 ssu_cp_1.c
-rw-rw-r--  1 junys junys  9552  4월 17 23:27 ssu_cp_1.o
-rw-rw-r--  1 junys junys  1119  4월 13 19:06 ssu_option_p.c
-rw-rw-r--  1 junys junys  2912  4월 17 23:27 ssu_option_p.o
-rw-rw-r--  1 junys junys  3519  4월 13 19:06 ssu_option_r.c
-rw-r--r--  1 junys junys 12288  4월 13 19:06 .ssu_option_r.c.swp
-rw-rw-r--  1 junys junys  3656  4월 17 23:27 ssu_option_r.o
-rw-rw-r--  1 junys junys   456  4월 13 19:06 ssu_option_s.c
-rw-rw-r--  1 junys junys  2024  4월 17 23:27 ssu_option_s.o
drwxrwxr-x  4 junys junys  4096  4월 13 19:06 ssu_osdir
drwxrwxr-x  3 junys junys  4096  4월 13 19:06 ssu_oslab
-rwxrwxr-x  1 junys junys  8664  4월 13 19:06 ssu_symlink
-rw-rw-r--  1 junys junys   124  4월 13 19:06 ssu_symlink.c
lrwxrwxrwx  1 junys junys    10  4월 13 19:06 symlink_tar -> src_link.c
-rw-rw-r--  1 junys junys     0  4월 13 19:06 TARGET_symlink_tar
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -r ~/phw_cp copy_dir
target : copy_dir
src : /home/junys/phw_cp
r option is on
junys@junys-VirtualBox:~/phw2$ ls -al copy_dir
total 156
drwxrw-r-- 7 junys junys  4096  4월 24 00:42 .
drwxrwxr-x 4 junys junys  4096  4월 24 00:42 ..
-rw-rw-r-- 1 junys junys     0  4월 24 00:42 a
drwxrw-r-- 3 junys junys  4096  4월 24 00:42 abc
-rw-rw-r-- 1 junys junys 13752  4월 24 00:42 a.out
-rw-rw-r-- 1 junys junys    11  4월 24 00:42 a.txt
-rw-rw-r-- 1 junys junys    11  4월 24 00:42 b
drwxrw-r-- 2 junys junys  4096  4월 24 00:42 cp2
drwxrw-r-- 4 junys junys  4096  4월 24 00:42 dir
-rw-rw-r-- 1 junys junys   516  4월 24 00:42 Makefile
-rw-rw-r-- 1 junys junys    22  4월 24 00:42 src_link.c
-rw-rw-r-- 1 junys junys 18584  4월 24 00:42 ssu_cp
-rw-rw-r-- 1 junys junys  4523  4월 24 00:42 ssu_cp_1.c
-rw-rw-r-- 1 junys junys  9552  4월 24 00:42 ssu_cp_1.o
-rw-rw-r-- 1 junys junys  1119  4월 24 00:42 ssu_option_p.c
-rw-rw-r-- 1 junys junys  2912  4월 24 00:42 ssu_option_p.o
-rw-rw-r-- 1 junys junys  3519  4월 24 00:42 ssu_option_r.c
-rw-rw-r-- 1 junys junys 12288  4월 24 00:42 .ssu_option_r.c.swp
-rw-rw-r-- 1 junys junys  3656  4월 24 00:42 ssu_option_r.o
-rw-rw-r-- 1 junys junys   456  4월 24 00:42 ssu_option_s.c
-rw-rw-r-- 1 junys junys  2024  4월 24 00:42 ssu_option_s.o
drwxrw-r-- 4 junys junys  4096  4월 24 00:42 ssu_osdir
drwxrw-r-- 3 junys junys  4096  4월 24 00:42 ssu_oslab
-rw-rw-r-- 1 junys junys  8664  4월 24 00:42 ssu_symlink
-rw-rw-r-- 1 junys junys   124  4월 24 00:42 ssu_symlink.c
-rw-rw-r-- 1 junys junys    22  4월 24 00:42 symlink_tar
```

그림 8 – [*source*] 디렉터리의 하위 디렉터리와 파일들을 모두 복사한다.

⑧ -d

⑨ 옵션 중복 사용 및 대소문자 구분 없음

```
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -p -r ~/ssu_oslab cp_ssu_oslab
target : cp_ssu_oslab
src : /home/junys/ssu_oslab
p option is on
**********file info**********
파일 이름 : /home/junys/ssu_oslab
데이터의 마지막 읽은 시간 : 2018.4.24 0:50:16
데이터의 마지막 수정 시간 : 2018.4.24 0:50:16
데이터의 마지막 변경 시간 : 2018.4.24 0:50:16
OWNER : junys
GROUP : junys
file size : 4096
r option is on
```

그림 9 – '-p'와 '-r' 사용하여 [*source*]디렉터리의 정보를 출력하고 하위 모든 파일과 디렉터리 복사한

```
junys@junys-VirtualBox:~/phw2$ ls -al filep
-rw-rw-r-- 1 junys junys 0  4월 20 19:46 filep
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -n -r ../filep cp_ssu_oslab
target : cp_ssu_oslab
src : ../filep
n option is on
r option is on
junys@junys-VirtualBox:~/phw2$ ls -al cp_ssu_oslab
total 28
drwxrw-r-- 3 junys junys 4096  4월 24 02:31 .
drwxrwxr-x 3 junys junys 4096  4월 24 02:31 ..
-rw-rw-r-- 1 junys junys    0  4월 24 02:31 a.txt
-rw-rw-r-- 1 junys junys    0  4월 24 02:31 b.c
drwxrw-r-- 2 junys junys 4096  4월 24 02:31 ssu_dir
-rw-rw-r-- 1 junys junys   98  4월 24 02:31 ssu_oslab_1.c
-rw-rw-r-- 1 junys junys 8656  4월 24 02:31 ssu_oslab_exec
```

그림 10 – '-n'이 적용 되었을 때, 이미 존재하는 디렉터리를 복사하려는 경우 덮어쓰지 않고 종료한

```
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -P -I -R ~/ssu_oslab copy_dir
target : copy_dir
src : /home/junys/ssu_oslab
p option is on
**********file info**********
파일 이름 : /home/junys/ssu_oslab
데이터의 마지막 읽은 시간 : 2018.4.24 0:50:16
데이터의 마지막 수정 시간 : 2018.4.24 0:50:16
데이터의 마지막 변경 시간 : 2018.4.24 0:50:16
OWNER : junys
GROUP : junys
file size : 4096
i option is on
r option is on
overwrite directory copy_dir (y/n)?y
```

그림 11 – 옵션의 대/소문자는 구분하지 않는다.

다) 오류 처리

① -s

i. -s 옵션은 단독적으로만 쓰인다

```
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -s -p a.txt a_symlink.txt
target : a_symlink.txt
src : a.txt
ssu_cp: -s
ssu_cp error
usage : in case of file
        cp [-i/n][-l][-p]       [source][target]
        or cp [-s][source][target]
        in case of directory cp [-i/n][-l][-p][-r][-d][N]
```

ii. 같은 옵션이 두 번 이상 나온 경우 오류 처리를 한다

```
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -l -L a.txt a_error.txt
target : a_error.txt
src : a.txt
l option is on
OPTION is repeated
ssu_cp error
usage : in case of file
        cp [-i/n][-l][-p]       [source][target]
        or cp [-s][source][target]
        in case of directory cp [-i/n][-l][-p][-r][-d][N]
```

iii. -d 옵션

```
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -l -d ~/ssu_osdir dir_error
target : dir_error
src : /home/junys/ssu_osdir
l option is on
ssu_cp : -d [NUMBER]
ssu_cp error
usage : in case of file
        cp [-i/n][-l][-p]       [source][target]
        or cp [-s][source][target]
        in case of directory cp [-i/n][-l][-p][-r][-d][N]
```

그림 12 - -d 옵션 바로 뒤에는 1~10의 숫자가 입력되어야 한다

```
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -r -d 3 ../ssu_oslab dir_error
target : dir_error
src : ../ssu_oslab
r option is on
sus_cp : cannot use option d with option r
ssu_cp : -d [NUMBER]
ssu_cp error
usage : in case of file
        cp [-i/n][-l][-p]       [source][target]
        or cp [-s][source][target]
        in case of directory cp [-i/n][-l][-p][-r][-d][N]
```

그림 13 - -d 옵션과 -s 옵션 중 하나만 적용되어야 한다

iv. 일반파일 복사인 경우 -d 또는 -r 옵션이 올 수 없다

```
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -r a.txt a_error.txt
target : a_error.txt
src : a.txt
ssu_cp : source file must be directory with option r
ssu_cp error
usage : in case of file
        cp [-i/n][-l][-p]       [source][target]
        or cp [-s][source][target]
        in case of directory cp [-i/n][-l][-p][-r][-d][N]
```

```
junys@junys-VirtualBox:~/phw2$ ./ssu_cp -d3 a.txt a_error.txt
target : a_error.txt
src : a.txt
ssu_cp : source file must be directory with option d
ssu_cp error
usage : in case of file
        cp [-i/n][-l][-p]       [source][target]
        or cp [-s][source][target]
        in case of directory cp [-i/n][-l][-p][-r][-d][N]
```

V. 소스 코드

가) ssu_cp.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <dirent.h>

#define FILE_LENGTH 256
#define BUFFER_SIZE       1024
```

```
typedef int OPTION;
OPTION on = 0;
OPTION S = 0;
OPTION I = 0;
OPTION L = 0;
OPTION N = 0;
OPTION P = 0;
OPTION R = 0;
OPTION D = 0;

void option(int, char *argv[]); //process Option
void option_s(char* src, char* tar); //Option -s
void option_p(char* src); //Option -p
void option_r(char* src, char* tar); //Option -r
void option_l(char* tar, struct stat stat_src); //Option -l
void ssu_copy(char *, char *);
void print_usage();
int main(int argc, char *argv[])
{
        int fd;
        char *src;
        char *tar;
        char opt_i;
        char new_tar[FILE_LENGTH];
        struct stat src_stat;
        if (argc < 3) {
                print_usage();
                exit(1);
        }

        tar = argv[argc-1];
        src = argv[argc-2];
        if(strcmp(tar, src) == 0) {
                fprintf(stderr, "ssu_cp : %s: source file and directory cannot be the same₩n", src);
                print_usage();
                exit(1);
        }

        chdir(".");
        printf("target : %s₩n", tar);
        printf("src : %s₩n", src);

        option(argc, argv);
        if(R) {
                if(mkdir(tar, 0766) < 0) { // 이미 존재하는 같은 이름을 가진  디렉터리를 만들려고 할 때
에러 발생-> 별도의 옵션이 없을 경우 덮어쓰기 진행
                        if(I) { //사용자에게 덮어쓰기 여부를 물어보는 옵션 -i
                                printf("overwrite directory %s (y/n)?", tar);
                                scanf("%c", &opt_i);
                                if(opt_i == 'y') N = 0;
                                else N = 1;
                        }
                        if(N) {
                                exit(0);
                        }
                        else {
                                opendir(tar); //open [target] directory
                        }
                }
                option_r(src, tar); //하위 디렉터리와 모든 파일들을 복사하는 함수
        }
        else {
                if(open(tar, O_RDWR | O_CREAT | O_EXCL, 0664) < 0) {
                        //이미 존재하는 [target] 파일인 경우
                        if(I) {
                                printf("overwrite %s (y/n)? ", tar);
                                scanf("%c", &opt_i);
                                if(opt_i == 'y') {
                                        N = 0;
                                }
                                else N = 1;
                        }
                        if(N) exit(0);
```

```
                }
                ssu_copy(src, tar);
        }
        //option 'l' is on
        if (L == 1) {
                stat(src, &src_stat);
                option_l(tar, src_stat);
        }
}

void option(int OptNum, char *Option[])
{
        int c;
        int repeat_option = 0; //옵션 중복 처리
        char *src = Option[OptNum-2];
        char *tar = Option[OptNum-1];
        char opt_i;
        int optD_N;
        struct stat src_stat;
        int error_d = 0;

        stat(src, &src_stat);

        while((c = getopt(OptNum, Option, "sSilILnNpPrRdD:")) != -1) {
                switch(c) {
                        case 's' : //symoblic link 생성
                        case 'S':
                                S++;
                                on++;
                                if(OptNum != 4) {
                                        fprintf(stderr, "ssu_cp: -sWn");
                                        print_usage();
                                        exit(1);
                                }
                                        printf("s option is onWn");
                                        option_s(src, tar);
                                        exit(0);
                                return;
                        case 'i' : //강제로 덮어쓰기
                        case 'I' :
                                I++;
                                on++;
                                if(S == 1) {
                                        fprintf(stderr,"ssu_cp : cannot use option %c with option sWn",
c);
                                        print_usage();
                                        exit(1);
                                }
                                if(I == 1) {
                                        printf("i option is onWn");
                                }
                                else repeat_option = 1;
                                break;
                        case 'l' : //파일 정보 [TARGET]에 복사
                        case 'L' :
                                L++;
                                on++;
                                if(S == 1) {
                                        fprintf(stderr, "ssu_cp : cannot use option %c with option sWn",
c);
                                        print_usage();
                                        exit(1);
                                }
                                if(L == 1) {
                                        printf("l option is onWn");

                                }
                                else repeat_option = 1;
                                break;
                        case 'n' : //기존 파일이 있는 경우 덮어쓰지 않음
                        case 'N' :
                                N++;
                                on++;
```

```
                        if(S == 1) {
                                fprintf(stderr, "ssu_cp : cannot use option %c with option s₩n",
c);
                                print_usage();
                                exit(1);
                        }
                        if (N == 1) {
                                printf("n option is on₩n");
                        }
                        else repeat_option = 1;
                        break;
                case 'p' : //파일 정보 출력 후 cp 수행
                case 'P' :
                        P++;
                        on++;
                //      printf("ON : %d₩n", on);
                        if(S == 1) {
                                fprintf(stderr, "ssu_cp : cannot use option %c with option s₩n",
c);
                                print_usage();
                                exit(1);
                        }
                        if (P == 1) {
                                printf("p option is on₩n");
                                option_p(src); //파일의 정보를 출력하는 함수 호출
                        }
                        else repeat_option = 1;
                        break;
                case 'r' :
                case 'R' :
                        R++;
                        on++;
                        //-s 또는 -d 옵션이 중복된 경우, [SOURCE]파일이 디렉터리가 아닌경우
에러처리
                        if(S == 1 || D == 1 || !S_ISDIR(src_stat.st_mode)) {
                                if(S)
                                        fprintf(stderr, "ssu_cp : cannot use option %c with
option s₩n", c);
                                if(D)
                                        fprintf(stderr, "ssu_cp : cannot use option %c with
option d₩n", c);
                                if(!S_ISDIR(src_stat.st_mode))
                                        fprintf(stderr, "ssu_cp : source file must be directory
with option %c₩n", c);
                                print_usage();
                                exit(1);
                        }
                        if (R == 1) {
                                printf("r option is on₩n");

                        }
                        else repeat_option = 1;
                        break;

                case 'd' :
                case 'D' :
                        D++;
                        on++;
                        //-s 또는 -r 옵션이 중복된 경우, [SOURCE]파일이 디렉터리가 아닌경우
에러처리
                        if(S == 1 || R == 1) {
                                if(S)
                                        fprintf(stderr, "ssu_cp : cannot use option %c with
option s₩n", c);
                                if(R)
                                        fprintf(stderr, "sus_cp : cannot use option %c with
option r₩n", c);

                                if(optarg == NULL) {
                                        fprintf(stderr, "ssu_cp : -d [NUMBER]₩n");
                                }
                                if(!S_ISDIR(src_stat.st_mode))
                                        fprintf(stderr, "ssu_cp : source file must be directory
```

```
with option %c₩n", c);
                                        print_usage();
                                        exit(1);
                                }
                                if (D == 1) {
                                        //if(optarg == NULL) {
                                        //      fprintf(stderr, "ssu_cp : -d [NUMBER] ₩n");
                                        //      error_d = 1;
                                        //}
                                        if(!S_ISDIR(src_stat.st_mode)) {
                                                fprintf(stderr, "ssu_cp : source file must be directory
with option %c₩n", c);
                                                error_d = 1;
                                        }
                                        if (error_d) {
                                                print_usage();
                                                exit(1);
                                        }
                                        printf("d option is on₩n");
                                        optD_N = atoi(optarg);
                                        printf("N : %d₩n", optD_N);
                                }
                                else repeat_option = 1;
                                break;
                        }
                        if (repeat_option) {
                                printf("OPTION is repeated₩n");
                                print_usage();
                                exit(0);
                        }
                }
        }

void ssu_copy(char *source, char *target) {
        struct stat src_stat;
        char opt_i;
        int sym = 0;
        int fd_src, fd_tar; //source file descriptor, target file descriptor
        int length;
        char c;
        int i, a;

        //source파일 정보 src_stat stat구조체에 저장
        stat(source, &src_stat);

        //source 파일이 directory인 경우
        if(S_ISDIR(src_stat.st_mode) && (!R || !D)) {
                printf("ssu_cp : source file cannot be directory file₩n"); print_usage(); exit(1);
                print_usage();
                exit(1);
        }

        if ((fd_src = open(source, O_RDONLY)) < 0) { //읽기 권한으로 [SOURCE]파일 open
                fprintf(stderr, "ssu_cp : %s: No such file or directory₩n", source);
                exit(1);
        }

        if((fd_tar = open(target, O_RDWR | O_CREAT | O_TRUNC, 0664)) < 0) { //[target]파일 생성(또는
덮어쓰기)
                fprintf(stderr, "target %s open error₩n", target);
        }

        //source 파일을 target 파일로 복사
        while(read(fd_src, &c, 1) > 0) {
                write(fd_tar, &c, 1);
        }
}

void print_usage() {
        fprintf(stderr, "ssu_cp error₩n");
        fprintf(stderr, "usage : in case of file₩n");
        fprintf(stderr, "        cp [-i/n][-l][-p]        [source][target]₩n");
        fprintf(stderr, "        or cp [-s][source][target]₩n");
```

```
        fprintf(stderr, "        in case of directory cp [-i/n][-l][-p][-r][-d][N]₩n");
}
```

나) ssu_option_s.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <fcntl.h>

void print_usage();
void option_s(char *src, char *tar) {
        struct stat stat_src;

        stat(src, &stat_src); //source 파일 정보 구조체 stat_src에 저장

        if(S_ISDIR(stat_src.st_mode)); //source 파일이 directory인 경우 생략

        else {
                if(symlink(src, tar) < 0) {
                        fprintf(stderr, "ssu_cp : target file already exists!₩n");
                        print_usage();
                        exit(1);
                }
        }
}
```

다) ssu_option_l.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <utime.h>

void option_l(char *target, struct stat src_stat) {
        struct utimbuf time_buf;
                //change mode
                if(chmod(target, src_stat.st_mode) < 0) {
                        fprintf(stderr, "chmod : error₩n");
                        exit(1);
                }

                //change utime
                time_buf.actime = src_stat.st_atime;
                time_buf.modtime = src_stat.st_mtime;
                if(utime(target, &time_buf) < 0) {
                        fprintf(stderr, "utime error₩n");
                        exit(1);
                }

                //change pid, gid
                if(chown(target, src_stat.st_uid, src_stat.st_gid) < 0) {
                        fprintf(stderr, "chown error₩n");
                        exit(1);
                }
}
```

라) ssu_option_p.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <time.h>
#include <pwd.h>
#include <grp.h>

void print_usage();
void option_p(char* src) {
        struct stat stat_src;
```

```
        struct tm *atime;
        struct tm *mtime;
        struct tm *ctime;
        struct passwd *uid;
        struct group *gid;
        lstat(src, &stat_src); //파일의 정보를 stat 구조체에 저장
        atime = localtime(&stat_src.st_atime);
        mtime = localtime(&stat_src.st_mtime);
        ctime = localtime(&stat_src.st_ctime);
        uid = getpwuid(stat_src.st_uid);
        gid = getgrgid(stat_src.st_gid);
        printf("**********file info**********₩n");
        printf("파일 이름 : %s₩n", src);
        printf("데이터의 마지막 읽은 시간 : %d.%d.%d %d:%d:%d₩n", atime->tm_year+1900, atime->tm_mon+1,
atime->tm_mday, atime->tm_hour, atime->tm_min, atime->tm_sec);
        printf("데이터의 마지막 수정 시간 : %d.%d.%d %d:%d:%d₩n", mtime->tm_year+1900, mtime->tm_mon+1,
mtime->tm_mday, mtime->tm_hour, mtime->tm_min, mtime->tm_sec );
        printf("데이터의 마지막 변경 시간 : %d.%d.%d %d:%d:%d₩n", ctime->tm_year+1900, ctime->tm_mon+1,
ctime->tm_mday, ctime->tm_hour, ctime->tm_min, ctime->tm_sec);
        printf("OWNER : %s₩n", uid -> pw_name);
        printf("GROUP : %s₩n", gid -> gr_name);
        printf("file size : %ld₩n", stat_src.st_size);
}
```

마) ssu_option)r.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <dirent.h>

#define FILE_LENGTH 256

char TARGET[FILE_LENGTH];

void print_usage();
void option_s(char *src, char *tar);
void ssu_copy(char *src, char *tar);
void option_r(char *src, char *tar) {
        struct stat stat_src; //source 파일의 정보가 담긴 구조체
        struct dirent *dent_src; //source directory 정보가 담긴 구조체
        DIR *dp_src; //source 디렉터리 가리키는 포인터
        char fname_src[FILE_LENGTH] = {'₩0'}; //source 파일의 하위 파일들의 경로 이름 저장
        char fname_tar[FILE_LENGTH] = {'₩0'}; //source 파일의 하위 파일들의 경로 target 파일에 저장
        char buf[FILE_LENGTH] = {'₩0'};
        int len_src, len_tar;
        int fd_src, fd_tar; //source 파일과 target 파일의 파일 디스크립터
        int i;
        char c;

        dp_src = opendir(src);

        while(1) {
                //src 디렉터리에 하위 파일이 있는 경우
                if((dent_src = readdir(dp_src)) != NULL) {
                        if(dent_src->d_ino == 0) {
                                continue;
                        }
                        //src 디렉터리의 하위 파일 경로 이름 저장
                        len_src = strlen(src);
                        strcpy(fname_src, src);
                        fname_src[len_src] = '/';
                        fname_src[len_src+1] = '₩0';
                        strcat(fname_src, dent_src->d_name);

                        //tar 디렉터리에 src 하위 파일 경로 이름 저장
                        strcpy(fname_tar, tar);
                        len_tar = strlen(tar);
                        fname_tar[len_tar] = '/';
```

```
                        fname_tar[len_tar+1] = '₩0';
                        strcat(fname_tar, dent_src->d_name);
                        if(stat(fname_src, &stat_src) < 0) {
                                continue;
                        }
                        if(S_ISDIR(stat_src.st_mode)) { //하위 파일이 디렉터리인 경우
                                if(strcmp(".", dent_src->d_name) == 0) { // '.' directory
                                        continue;
                                }
                                if(strcmp("..", dent_src->d_name) == 0) { //'..' directory
                                        continue;
                                }
                                mkdir(fname_tar, 0766); //tar 디렉터리에 src의 하위 디렉터리 생성
                                option_r(fname_src, fname_tar);
                        }
                        else if (S_ISREG(stat_src.st_mode)) { //하위 파일이 일반 파일인 경우
                                fd_src = open(fname_src, O_RDONLY);

                                ssu_copy(fname_src, fname_tar);
                        }

                        else;

                        for (i = 0; i < FILE_LENGTH; i++) {
                                fname_tar[i] = '₩0';
                                fname_src[i] = '₩0';
                                buf[i] = '₩0';
                        }
                }
                //src 디렉터리의 끝
                else {
                        break;
                }
        }

        return;
}
```