1. rm 명령어를 구현하라. 이 명령어는 rm filename 또는 rm dirname의 형태를 이루며, 주어진 인자가 파일 이름이면 파일을 삭제하고 디렉토리 이름이면 디렉토리를 삭제해야 한다.

 1) 찾기 기능으로 기존에 cmd.c에 이미 작성되어 있는 rm() 함수를 찾아 간다. (/rm)

 2) 강의노트 4장 pp.42~43, 교재 p.144(130), pp.160(142)을 참조하여 rm() 함수를 구현하자. 강의노트를 보고 입력하면 에러 발생하니 참조만 하라.

  i) 강의노트처럼 buf 변수를 선언하라.

  ii) API 함수인 lstat() 함수(함수1)를 호출하고, 에러가 발생했다면 PRINT_ERR_RET()를 호출하라.

```
if (함수1호출 < 0)
   PRINT_ERR_RET();
```

  함수의 첫 인자는(사용자가 준 명령어 인자)? 두번째 인자는? 이 함수는 파일 또는 디렉토리에 대한 정보를 얻어 구조체 변수 buf에 저장한다.

  iii) 강의노트를 참조하여 명령어 인자로 준 이름이 디렉토리인지 아닌지를 판단하는 방법을 확인하라. 디렉토리면 rmdir() 함수(함수2:디렉토리 삭제)를 호출하고, 아니면 (파일)이면 unlink() 함수(함수3:파일 삭제)를 호출한다. 두 함수 모두 호출 후 에러가 생기면 에러 메시지 출력하고 리턴 하게 하라.

  여러분은 이렇게 구현했을 것이다.

```
---------------------------------------
int ret;
if (디렉토리면)
   ret = 함수2호출;
else // 파일이면
   ret = 함수3호출;
if (ret < 0) // 에러이면
   PRINT_ERR_RET();
---------------------------------------
```

  그러나 아래처럼 한 문장으로 줄이면 더 효율적이다.

```
---------------------------------------
if (((디렉토리면)? 함수2호출: 함수3호출) < 0)
   PRINT_ERR_RET();
---------------------------------------
```

  iv) ii)와 iii)의 문장을 정리하면 다음과 같다.

```
---------------------------------------
if (함수1호출 < 0)
   PRINT_ERR_RET();
if ( ( (디렉토리면)? 함수2호출: 함수3호출 ) < 0)
   PRINT_ERR_RET();
---------------------------------------
```

  그러나 이것도 아래처럼 간단히 수정하라.

```
---------------------------------------
if ( (함수1호출 < 0) ||
     (((디렉토리면)? 함수2호출: 함수3호출) < 0) )
   PRINT_ERR_RET();
---------------------------------------
```

  여기서 함수1이 에러가 생기면(파일이 존재하지 않을 경우) 함수2와 3은 호출되지 않는다. 지금은 복잡해 보여도 자주 사용하면 복잡하지 않고 당연한 것처럼 보인다.

 3) 이미 입력되어는 cmd_tbl[]의 rm 원소를 확인하라.

 4) 세 함수의 헤드 파일을 $ man으로 찾아 include시킨다.

 5) make하라. cmd를 실행하기 전에, 먼저 **$ touch f1**하여 새로운 빈 파일을 만들어라. 그런 후 cmd를 실행시키고 명령어로 다음을 테스트하라. "**>** "는 cmd의 명령 프롬프트다.

```
> mkdir dir          // dir 디렉토리 생성
> ls -l     // 생성되었는지 확인. f1도 존재 해야 함
> rm  dir            // 디렉토리 삭제
> ls -l              // dir 삭제되었는지 확인
> rm  f1             // 파일 삭제
> ls -l              // f1 삭제되었는지 확인
> rm  f2             // 파일 없음 에러
> rm  /              // 접근권한 에러
> rm  /usr/include/stdio.h  // 접근권한 에러
```

2. mv 명령어를 구현하라.

 1) 먼저 mv() 함수를 정의하자.

```
---------------------------------------
// 파일 이름을 다른 이름으로 변경하는 명령어
// 사용법: mv  원본파일  바뀐이름
// argv[0] -> "원본파일이름"
// argv[1] -> "변경된파일이름"
void mv(void) {}//기존 함수 형태로 할 것
---------------------------------------
```

 2) 강의노트 4장 pp.41~42, 교재 pp.143(129)를 참조하여 mv() 함수를 구현하자. UNIX에는 파일 이름을 변경하는 API 함수가 없다. 따라서 다음과 같이 두 개의 함수를 이용하여 동일한 효과를 내게 해야 한다.

  i) 먼저 link() 함수(함수1)를 호출하여 기존 argv[0]와 동일한 데이터를 포인터하는 새로운 argv[1] 파일 이름을 만든다. 에러가 발생했다면 에러 원인 출력하고 리턴 하라.

  ii) 에러가 없다면 unlink()를(함수2) 호출하여 기존 argv[0] 파일을 삭제한다. 이 역시 에러가 발생했다면 에러 원인 출력하고 리턴 하라.

iii) 위 두 함수가 정상 수행되었다면 argv[0]는 없어지고 argv[1] 파일만 남게 된다. (이름 변경 효과와 동일)

iii) mv() 함수 구조를 보통은

```
-------------------------------------
if (함수1호출 < 0)
   PRINT_ERR_RET();
if (함수2호출 < 0)
   PRINT_ERR_RET();
-------------------------------------
```

처럼 구현하기 쉽다. 그러나 아래처럼 한 문장으로 줄여 구현하는 것이 더 효율적이다.

```
-------------------------------------
if ((함수1호출 < 0) || (함수2호출 < 0))
   PRINT_ERR_RET();
-------------------------------------
```

3) cmd_tbl[] 배열에 mv 관련 배열 원소를 추가하라. 위 1)번의 주석문 중 사용법을 참조하기 바란다.

4) 위 두 함수의 헤드 파일을 찾아 include시킨다.

5) make하라. cmd를 실행하기 전에, 먼저 **$ touch f1**하여 빈 파일을 만들어라. 그런 후 cmd를 실행시키고 명령어로 다음을 테스트하라.

```
> mv f1 f2          // f1을 f2로 이름변경
> ls                // 변경되었는지 확인
> mv f1 f2          // f1 파일 없음 에러
> mkdir dir         // dir 디렉토리 생성
> mv f2 dir         // 이미 존재한다는 에러 발생
// 기존 Unix 명령처럼 끝에 디렉토리만 줄 수 없고, 끝에 파일이름(dir/f1)을 반드시 주어야함. 즉,
> mv f2 dir/f1      // dir 밑에 f1 이름으로 옮김
> ls                // f2 없음
> ls -l dir         // f1 존재
> mv dir/f1 f1      //dir밑에 f1을 .밑에 f1로 변경
> ls                // f1 존재
> ls dir            // f1 없음
```

3. ln 명령어를 구현하라.

1) 먼저 ln() 함수를 정의하자.

```
----------------------------------------
// 하나의 파일을 다른 이름으로 링크를 만들어 주는 명령어
// 사용법: ln -s 원본파일  링크파일
//   "-s" 옵션이 없으면 hard link,
//              있으면 symbolic link(바로가기파일)
//   hard link:  디스크에 파일 데이터는 하나만 존재하는데
//            이 데이터를 지칭하는 파일이름이 여러 개인 경우
// argv[0] -> "원본파일이름"
// argv[1] -> "링크파일이름"
// optc = "-s" 옵션을 준 경우 1, 주지 않았을 경우 0
void ln(void) {}//기존 함수 형태로 할 것
----------------------------------------
```

2) 강의노트 4장 pp.40~41, pp.47~52, 교재 p.143(129), p.152(137)을 참조하여 ln() 함수를 구현하자.

i) 옵션을 주었다면 symlink() 함수(함수1: 바로가기 파일 생성)를 호출하고, 주지 않았다면 link() 함수(함수2: hard link 파일 생성)를 호출한다. 두 함수 모두 호출 후 에러가 생기면 에러 메시지 출력하고 리턴 한다. 옵션을 주었는지 여부 체크는 위 1)번 주석문의 전역변수 optc 값을 참조하면 된다. 여러분은 이렇게 구현할 것이다.

```
----------------------------------------
int ret;
if (옵션을 주었다면)
   ret = 함수1호출;
else // 옵션을 주지 않았다면
   ret = 함수2호출;
if (ret < 0) // 에러이면
   PRINT_ERR_RET();
----------------------------------------
```

그러나 아래처럼 한 문장으로 줄이면 더 효율적이다.

```
----------------------------------------
if (((옵션을 주었다면)? 함수1호출: 함수2호출) < 0)
   PRINT_ERR_RET();
----------------------------------------
```

3) cmd_tbl[] 배열에 ln 관련 배열 원소를 추가하라. 위 1)번의 주석문 중 사용법을 참조하기 바란다.

4) 위 두 함수의 헤드 파일을 찾아 include시킨다.

5) make하라. cmd를 실행하기 전에, 먼저 **$ touch f1**하여 새로운 빈 파일을 만들어라. 그런 후 cmd를 실행시키고 명령어로 다음을 테스트하라.

```
> ls -l     // f1 링크 수 1
> ln f2 f1  // 파일 f2 없음 에러
> ln f1 f2
> ls -l     // f2가 생성되고, f1과 f2의 링크 수 2
> ln f2 f3
> ls -l     // f3가 생성되고, f1, f2, f3의 링크 수 3
> rm f1
> ls -l     // f1이 삭제되고, f3과 f2의 링크 수 2
> rm f3
> ls -l     // f3이 삭제되고, f2의 링크 수 1
> ln -s f4 f5       // 에러 없음
> ls -l     // f4가 없음에도 불구하고 바로가기 파일 f5가 생성되어 있음 (여기까진 정상임)
// 그러나 다른 명령 창에서 ~/up/cmd로 가서, $ cat f5을 실행시켜 보라. f5가 포인터하는 f4가 없다는 에러
```

메시지가 나올 것임.

```
> ln -s f2 f1
> ls -l     // f1 바로가기 파일 생성: f1 -> f2
> ln -s f2 f3
> ls -l     // f3 바로가기 파일 생성: f3 -> f2
> rm f2
> ls -l     // f1, f3 바로가기 파일은 존재하지만
            // 이들이 포인터하는 f2는 존재하지 않음
>rm f1  >rm f3  >rm f5  >ls  //모두 삭제
```

4. chmod 명령어를 구현하라.

1) 기존에 chmod()라는 API 함수가 이미 존재하므로 우리는 changemod() 함수를 정의하자

```
-------------------------------------
// 파일의 접근권한을 변경하는 명령어
// 사용법: chmod  8진수모드값  파일이름
// argv[0] -> "8진수모드값"
// argv[1] -> "파일이름"
void changemod(void) {}//기존 함수 형태로 할 것
-------------------------------------
```

2) 강의노트 4장 p.28, 교재 p.131(118)을 참조하여 changemod() 함수를 구현하자.

i) 사용자가 명령어 첫 인자로 준 argv[0]는 8진수 문자열이다. (예, “755”). 그런데 나중에 우리는 API 함수 chmod()를 호출해야 하고, 이 함수의 두 번째 인자인 mode는 정수 값으로 주어야 한다. 따라서 argv[0]의 문자열로 된 숫자를 정수형 값으로 변환해야 한다. 아래 코드를 삽입하라.

```
-------------------------------------
int mode;// “%o”, 영어 소문자 o임(8진수 octal의 약어)
sscanf(argv[0], "%o", &mode);
-------------------------------------
```

이 함수는 argv[0]가 포인터하는 8진수(“%o”) 문자열("755")을 읽어 이를 정수형 값으로 변환하여 mode에 저장한다. (키보드가 아닌 argv[0] 문자열에서 읽어 들임)

ii) chmod() API 함수를 호출하여 파일의 접근권한을 변경한다. 에러가 발생했다면 에러 원인 출력하고 리턴 하라.

3) cmd_tbl[] 배열에 chmod 관련 배열 원소를 추가하라. 위 1)번의 주석문 중 사용법을 참조하기 바란다.

4) chmod() 함수의 헤드 파일을 찾아 include시킨다.

5) make하라. cmd를 실행하기 전에, 먼저

$ touch f1 하여 빈 파일을 만들고,

$ umask 0 로 설정하라.

// 사용자가 원하는 데로 접근권한을 설정 가능하게 함

그런 후 cmd를 실행시키고 명령어로 다음을 테스트하라.

```
> ls -l             // f1 접근권한을 확인
> chmod 777 f1
> ls -l             // f1 접근권한 -rwxrwxrwx
> chmod 432 f1
> ls -l             // f1 접근권한 -r---wx-w-
```

5. 명령 창에서 다음을 실행하여 모든 것이 정상인지 확인하라.

```
$ eshw 7
$ progtest 7
```