1. touch 명령어를 구현하라.

1) 먼저 touch() 함수를 정의하자.

```
-------------------------------------
// 존재하지 않는 파일이름일 경우 크기가 0인 새로운 파일 생성
// 존재하는 파일인 경우 파일의 수정시간을 현재시간으로 변경함
// 사용법: touch  파일이름
// argv[0] -> "파일이름"
void touch(void) {}//기존 함수 형태로 할 것
-------------------------------------
```

2) 먼저 <errno.h> 헤드 파일을 include시킨다. 이 파일에는 extern int errno라는 전역변수가 extern 선언되어 있다. (진짜 변수 선언은 시스템 라이브러리 속에 정의되어 있음) 이 변수에는 API 함수 호출 시 에러(-1 리턴)가 발생했을 때, 그 에러 원인의 번호가 저장되어 있다. 사실 이 변수(errno)를 이용해 perror() 함수도 에러 원인을 문자열로 출력해 준다. 에러 번호들은 모두 errno.h에 상수로 define 되어 있다. (예: ENOENT)
$ more /usr/include/asm-generic/errno-base.h 해 보라.

3) 강의노트 4장 p.54, 교재 pp.138~140, p.73을 참조하여 touch() 함수를 구현하자.

i) 파일 핸들용으로 정수형 변수 fd를 선언하라.

ii) 파일의 마지막 수정시간과 접근시간을 현재시간으로 변경하는 utime() 함수를 호출한다. 그러기 위해선 함수의 두 번째 인자는 NULL을 주어야 한다. 에러가 발생하지 않았으면, 즉 utime() 함수의 리턴 값이 0이면 touch() 함수에서 바로 리턴 한다. 파일이 이미 존재하고 수정시간도 변경되었으므로 더 이상 할 일이 없기 때문이다.

```
if (utime(?, NULL) == 0)
    return; // 정상적으로 실행
```

iii) 이제 위 ii)의 에러 원인, 즉 errno가 ENOENT(파일이 존재하지 않음)가 아니라면, 에러 원인을 출력하고 바로 리턴 한다. 이는 다른 원인에 의해 에러가 발생했음을 의미한다. (파일접근권한 문제)

```
// 에러가 발생한 경우
if (errno != ?) // 다른 에러 원인이면
    에러 원인 출력 후 리턴
```

iv) 이제 크기가 0인 파일을 생성하면 된다. 강의노트 3장 p.8, 교재 p.83을 참조하라. 먼저 creat() 함수(creat**e**가 아님)를 호출하고 리턴 값을 fd에 저장하되 에러가 발생했다면 원인을 출력하고 리턴 하라. (하나의 if 문장으로 처리하라.) creat() 함수의 두 번째 인자는 생성될 파일의 접근권한으로 8진수 0644(rw-r--r--)로 지정하라.

```
if ((.. = creat(...)) < 0)//파일생성에러
    에러 원인 출력 후 리턴
```

v) 위 iii)과 iv) 모두 에러를 출력하고 리턴 한다. 이를 하나의 if 문장으로 통합하라.

```
if ( (...) ||
     (...) )
    에러 원인 출력 후 리턴
```

vi) 위 creat() 함수는 새로운 파일을 생성하고 여전히 파일이 열린 상태로 존재하게 하므로 close() 함수를 호출하여 닫아 주어야 한다. 에러 체크는 하지 않아도 된다. 파일을 생성하고 write() 함수는 호출하지 않았기에 크기는 0이다.

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

5) 위 세 함수의 헤드 파일을 include시킨다.

6) make하라. 그런 후 cmd를 실행시키고 명령어로 다음을 테스트하라. "**>** "는 cmd의 명령 프롬프트다.

```
> rm f1      // 있으면 삭제, 에러면 무시하라.
> touch f1 // 크기가 0인 f1 생성
> ls -l      // f1 존재해야 함
// cmd.c의 수정시간을 확인해 두라.
> touch cmd.c       // cmd.c 수정시간->현재로
> ls -l      // cmd.c 수정시간 변경되었는지 확인
> touch f2 // 크기가 0인 f2 생성
> touch f1 // f1 수정시간->현재로
> ls -l      // f1 수정시간, f2 생성여부 확인
> rm f1    > rm f2          // 모두 삭제
```

2. cat 명령어를 구현하라.

1) cmd.c의 앞쪽에 파일 I/O 시 필요한 배열(버퍼)의 크기로 사용될 상수 SZ_FILE_BUF를 1024로 define하라.

2) 먼저 cat() 함수를 정의하자.

```
-------------------------------------
// 파일 내용을 화면에 보여 주는 명령어
// 사용법: cat 파일이름
// argv[0] -> "파일이름"
void cat(void) {}//기존 함수 형태로 할 것
-------------------------------------
```

3) 강의노트 3장 p.3, 교재 pp.70~80를 참조하여 cat() 함수를 구현하자.

i) 강의노트를 보고 변수를 선언하라. 읽어 올 파일이름은 전역변수 argv[0]를 이용하라.

ii) 나머지는 노트를 보고 입력하되, open() 함수 호출 후 에러가 생기면 바로 에러 출력하고 리턴 하게 하라.

4) cmd_tbl[] 배열에 cat 관련 배열 원소를 추가하라. 위

2)번의 주석문 중 사용법을 참조하기 바란다.

5) 위 모든 함수들의 헤드 파일을 찾아 include시킨다.

6) make하라. 그런 후 cmd를 실행시키고 명령어로 다음을 테스트하라. "**>** "는 cmd의 명령 프롬프트다.

```
> cat cmd.c
> cat aaa  // aaa 존재하지 않으므로 에러 출력
> cat /home/jhshim/util/chk.c
            // 접근권한이 없으므로 에러 출력
```

3. cp 명령어를 구현하라.

1) 강의노트 3장 p.4, 4장 p.3 교재 pp.83(70~80), pp.115(103~104)를 참조하여, 이미 삽입되어 있는 cp() 함수를 구현하자.

i) 강의노트를 보고 변수를 선언하라. 명령어 인자인 두 파일 이름은 전역변수 argv[0](원본파일이름), argv[1](복사된 파일 이름)이 포인터하고 있다.

ii) 노트를 보고, stat(), open(), creat() 함수 호출 후 에러가 생기면 바로 에러 출력하고 리턴 하게 하라.

ii) 그런데 위처럼 연속으로 바로 호출한 후 에러 체크하고 리턴하게 되면 문제가 발생한다. 만약 creat() 함수 호출 시 에러(접근권한)가 발생했다면, 앞의 open()에 의해 열린 파일 rfd를 close()하지 않은 채 리턴 한다. 따라서 두 단계로 분리해야 한다.

iii) 먼저 stat()와 open()을 호출한 후 에러가 발생했다면 앞에서처럼 리턴 하게 한다.

iv) 그 다음에 별도로 creat()를 호출하여 에러가 발생했다면, 에러 원인을 출력하고 그 전에 open한 파일을 닫고 리턴 하게 하라. 즉,

if (creat() 호출 후 에러면**) {**
    먼저 **perror(cmd)**를 호출하여 에러 원인 출력하고,
    rfd를 닫고,
    return;
}

그런데 여러분은 아래처럼 하고 싶을 것이다.

{ rfd를 먼저 닫고, **PRINT_ERR_RET(); }**

하면 더 간단해 질 거라고 생각할 것이다.

만약 이렇게 한다면, creat() 직후 에러 원인을 저장하고 있던 전역변수 errno 값이 그 다음의 close() 함수가 정상적으로 수행되면 errno 값이 0으로 변경된다. 그러면 PRINT_ERR_RET() 매크로 함수 내에 있는 perror(cmd)를 호출해도 에러 원인을 출력할 수 없게 된다. 따라서 에러가 발생한 API 함수를 호출한 후에 바로 perror() 함수를 호출해야 한다.

v) 그 다음은 강의노트처럼 while 문장과 그 이후를 동일하게 입력하라.

2) 위 모든 함수들의 헤드 파일을 찾아 include시킨다.

3) make하라. 그런 후 cmd를 실행시키고 명령어로 다음을 테스트하라. "**>** "는 cmd의 명령 프롬프트다.

```
> cp cmd.c cmd.old
> ls -l // cmd.old 확인
      // 파일크기와 접근권한이 cmd.c와 같아야 함
> cp cmd cmd.exe
> ls -l // cmd.exe 확인
        // 파일크기와 접근권한이 cmd와 같아야 함
> cp aaa b  // aaa 존재하지 않으므로 에러 출력
> cp cmd.exe cmd.old // 기존 cmd.old에 덮어 씀
> ls -l     // cmd.old의 파일크기가
            // cmd.exe와 같아야 함
> cp /home/jhshim/util/chk.c f1
   // chk.c에 대해 읽기 권한이 없으므로 에러 출력
> cp cmd.exe /home/jhshim/util/cmd.exe
   // util 디렉토리에 쓰기권한이 없으므로 에러 출력
   // 새로운 파일을 만들려면 해당 디렉토리에
   // 쓰기 권한이 있어야 함
> rm cmd.exe   > rm cmd.old  // 모두 삭제
```

4. 명령 창에서 다음을 실행하여 정상임을 확인하라.

```
$ eshw 9
$ progtest 9
```