1. cd 명령어를 구현해 보자.

1) 먼저 cd() 함수를 정의하자.

```
-------------------------------------
// 현재 작업 디렉토리를 변경하는 명령어
// 사용법: cd [디렉토리이름]
//        []는 명령어 인자를 주어도 되고 안 주어도 됨을 의미함
// argv[0] -> "디렉토리이름"; [디렉토리 이름]을 준 경우
// argc = [디렉토리 이름]을 준 경우 1, 주지 않았을 경우 0
void cd(void) {} // 기존 함수 형태로 할 것
-------------------------------------
```

2) 강의노트 4장 p.62~64, 교재 pp.167~168(149~151)를 참조하여 cd() 함수를 구현해 보자.

i) 먼저 사용자가 [디렉토리이름]을 지정하지 않았다면, 아래처럼 자신의 홈을 디폴트로 설정하는 문장을 삽입하라.

**if** (argc가 0)이면, // 명령어 인자를 주지 않았을 경우

**argv[0] = "/home/자신의홈";**

추후에 자신의 홈 디렉토리를 API 함수를 이용해 구할 것이나 지금은 위처럼 프로그램에서 직접 지정하라.

ii) chdir(argv[0]) API 함수를 호출하여 현재 작업 디렉토리를 변경한다. argv[0]는 사용자가 지정한 이동할 디렉토리 문자열의 시작 주소이다. 즉,

```
-------------------------------------
  if (chdir(argv[0]) < 0)
     PRINT_ERR_RET(); // 에러 발생
-------------------------------------
```

처럼 코딩하면 된다. 위처럼 함수를 호출했는데 에러가 발생했다면(함수의 리턴 값이 0보다 적을 경우), 에러 원인을 출력하고 cd() 함수에서 리턴 하게 하는 PRINT_ERR_RET() 매크로 함수를 호출하라.

**위와 같은 문장 방식은 앞으로 매번 등장하니 if 문장의 구조를 잘 파악해 둬라.** 정리하면 다음과 같다.

```
-------------------------------------
  if (API함수호출 < 0)
     PRINT_ERR_RET();
-------------------------------------
```

iii) **else** (위에서 에러가 발생하지 않은 경우임), getcwd() 함수를 호출한다(main() 함수 참조). 현재 작업 디렉토리가 변경되었기 때문에, 이 함수를 호출하여 변경된 디렉토리 이름을 구해 와 cur_work_dir[]에 다시 저장하는 것이다. 그러면 다음 명령 프롬프트 출력 시 바뀐 디렉토리(cur_work_dir[])가 출력될 것이다.

3) cmd_tbl[ ]에 cd 관련 배열원소를 추가하라. 이때 명령어 인자 개수는 0 또는 1임을 의미하는 AC_LESS_1, 명령어 인자는 위 1)에 있는 cd() 함수 사용법처럼 "[디렉토리이름]"를 지정하라.

4) chdir ()이 선언된 헤드파일을 include시킨다.

5) make, cmd 실행하여 아래처럼 cd를 테스트하라. "**>** "는 cmd의 명령 프롬프트다.

| | | |
|---|---|---|
| **> cd** | **> pwd** | (자신의 홈) |
| **> cd up/cmd** | **> pwd** | (up/cmd) |
| **> cd ../..** | **> pwd** | (자신의 홈) |

2. uname 명령어를 구현해 보자.

1) 기존에 uname()이라는 API 함수가 이미 존재하므로 우리는 unixname() 함수를 정의하자.

```
-------------------------------------
// 운영체제 이름 및 버전 등 시스템 정보를 출력하는 명령어
// "-a" 옵션을 주면 상세정보를, 안 주면 시스템 이름만 출력함
// 사용법: uname -a
// optc = "-a" 옵션 주면 1, 안 주면 0
void unixname(void) {} // 기존 함수 형태로 할 것
-------------------------------------
```

2) 강의노트 6장 p.14, 교재 p.232(205)를 참조하여 unixname () 함수를 구현하자.

i) 먼저 구조체 변수 un을 선언하자

**struct utsname un;** //시스템 정보를 저장하는 구조체 변수

ii) uname() API 함수를 호출하라. 호출 시 함수 인자로 구조체 변수 un의 주소를 주어라. 그러면 이 함수가 시스템의 관련 정보를 un 구조체에 저장해 줄 것이다. 명령어 uname과 동일한 이름의 API 함수인 uname()이 이미 있기 때문에, 충돌을 피하기 위해 우리 함수 이름을 unixname()으로 지정하였다.

iii) 이제 un.sysname에 저장된 시스템 이름(문자열)을 출력하는 printf() 문장을 삽입하라.

vi) if ("-a"옵션을 주었다면) un 구조체의 나머지 멤버들을 하나의 printf()로 출력하라. 옵션을 주었는지는 optc 값을 참조하면 된다. (위 1)번의 주석문 참조)

vi) 마지막에 줄 바꾸기("\n") 문자를 출력하라.

3) cmd_tbl[] 배열에 uname 관련 배열원소를 추가하라. 주의할 것은 명령어 이름은 "uname"이지만, 이 명령어를 처리하는 함수이름은 unixname이다. 배열원소의 나머지 항목은 위 1)번의 주석문 중 사용법을 참조하기 바란다.

4) API 함수인 uname()이 선언된 헤드파일을 명령어 $ man 으로 확인하고 include시킨다.

5) make, cmd 실행; cmd의 명령어로

**> uname** **> uname -a**

를 입력해 보자. cmd를 종료하고, 명령 창에서도 두 개의 명령을 실행해 보라. cmd와 결과가 동일한가? 상세정보에서 약간 차이가 있어도 괜찮다.

3. mkdir 명령어를 구현하라.

1) 기존에 mkdir()이라는 API 함수가 이미 존재하므로 우리는 makedir() 함수를 정의하자.

```
---------------------------------------
// 새로운 디렉토리를 생성하는 명령어
// 사용법: mkdir  디렉토리이름
// argv[0] -> "디렉토리이름"
void makedir(void) {} // 기존 함수 형태로 할 것
---------------------------------------
```

2) 강의노트 4장 p.57, 교재 pp.159(141)를 참조하여 makedir() 함수를 구현하자.

i) 디렉토리를 생성하는 API 함수인 mkdir() 함수를 호출하고, 에러가 발생했다면 PRINT_ERR_RET();를 호출하라. 문장 형태는 위 1. 2) ii)의 if 문장 형태이다. mkdir() 함수호출 시 첫 인자는 생성할 디렉토리 이름의 시작주소인 argv[0]를(위 주석문 참조), 두 번째 인자는 디렉토리 접근권한인데 8진수 0755를 지정하라.

ii) 명령어 mkdir과 동일한 이름의 API 함수인 mkdir()가 이미 있기 때문에, 충돌을 피하기 위해 우리 함수 이름을 makedir()로 지정하였다.

iii) PRINT_ERR_RET(); 문장 아래에 다음의 주석문을 추가하라.

```
---------------------------------------
// 0755: rwxr-xr-x
// 이는 이 디렉토리를 만든 사람은 이 디렉토리에
// 읽고(r: ls 실행 가능), 쓰고(w: 파일 생성 및 삭제 가능),
// 옮겨 갈 수 있음(x: cd로 갈 수 있음).
// 그러나 그룹 멤버(r-x)나 다른 제3의 사용자(r-x)는
// cd 명령어로 옮겨(x) 가서 ls를 할 수만 있고(r),
// 그 디렉토리에 파일을 생성하거나 삭제(w)는 할 수 없다.
---------------------------------------
```

3) cmd_tbl[] 배열에 mkdir 관련 배열 원소를 추가하라. 주의할 것은 명령어 이름은 “mkdir”, 이 명령어를 처리하는 함수 이름은 makedir이다. 배열 원소의 나머지 항목은 위 1) 번의 주석문 중 사용법을 참조하기 바란다.

4) API 함수인 mkdir()이 선언된 헤드파일을 명령어 $ man으로 확인하고 include시킨다.

5) make하고 cmd 실행한 후 cmd의 명령어로 다음을 실행하라. “**>** “는 cmd의 명령 프롬프트다.

```
> mkdir dir       // 정상적으로 생성
> ls -l           // 생성되었는지 확인
> mkdir  /dir     // 에러(접근권한문제)
> mkdir dir       // 에러(이미 존재)
```

cmd를 종료하고, 명령 창에서도 **$ ls -l** 하여 디렉토리 dir이 정상적으로 생성되었는지 다시 한번 확인하라. 그런 후 **$ rm -r dir**로 삭제하라.

4. rmdir 명령어를 구현하라.

1) 기존에 rmdir()이라는 API 함수가 이미 존재하므로 우리는 removedir() 함수를 정의하자.

```
---------------------------------------
// 빈 디렉토리($ls -l 때 ..과 .만 존재)를 삭제하는 명령어
// 빈 디렉토리가 아닌 경우 에러 발생
// 사용법: rmdir  디렉토리이름
// argv[0] -> "디렉토리이름"
void removedir(void) {}//기존 함수 형태로 할 것
---------------------------------------
```

2) 강의노트 4장 p.57, 교재 pp.160(142)를 참조하여 removedir () 함수를 구현하자.

i) 디렉토리를 삭제하는 API 함수인 rmdir() 함수를 호출하고, 에러가 발생했다면 PRINT_ERR_RET();를 호출하라. 위 1. 2) ii)의 if 문장 형태를 참조하라.

ii) 명령어 rmdir과 동일한 이름의 API 함수인 rmdir() 가 이미 있기 때문에, 충돌을 피하기 위해 우리 함수 이름을 removedir()로 지정하였다.

3) cmd_tbl[] 배열에 rmdir관련 배열 원소를 추가하라. 주의할 것은 명령어 이름은 “rmdir”, 이 명령어를 처리하는 함수 이름은 removedir이다. 배열 원소의 나머지 항목은 위 1)번의 주석문 중 사용법을 참조하기 바란다.

4) API 함수인 rmdir()이 선언된 헤드파일을 명령어 $ man으로 확인하고 include시킨다.

5) make하고 cmd 실행한 후 cmd의 명령어로 다음을 실행하라. “**>** “는 cmd의 명령 프롬프트다.

```
> mkdir dir       // 정상적으로 생성
> ls -l           // 생성되었는지 확인
```

// 다른 명령 창을 띄운 뒤 로그인하여 **$ cd ~/up/cmd/dir**로 가서 **$ touch f1**하여 새로운 파일을 만든다. 그런 후 cmd로 와서

```
> cd dir          $ pwd
> ls -l           // f1 생성되었는지 확인
> rmdir  dir      // 에러(빈 디렉토리가 아님)
```

// 다른 명령창에서 **$rm f1**하여 파일 삭제 후, cmd에서

```
> ls -l           // f1 삭제되었는지 확인
> cd ..           // cmd 디렉토리
> rmdir  dir      // 정상적으로 삭제
> ls -l           // 삭제되었는지 확인
> rmdir  dir      // 에러(디렉토리가 없음)
```

5. 명령 창에서 다음을 실행하라.

```
$ eshw 6          // 에러가 발생한 항목만 보여 줌
$ progtest 6 2    // 실습 6의 2번 문제만 테스트함
$ progtest 6      // 실습 6 전체를 테스트함
```