디렉토리 관리 툴 만들기

디렉토리 내용 읽기

```
struct dirent {
    ino_t          d_ino;
    off_t          d_off;
    unsigned short d_reclen;
    unsigned char  d_type;
    char           d_name[256];
};
```

- **d_ino** : 해당 항목의 inode 번호
- **d_off** : 디렉터리 오프셋의 위치
- **d_reclen** : 해당 항목의 레코드 길이
- **d_type** : 파일의 종류
- **d_name** : 항목의 이름

예제 2-8 디렉터리 내용 읽기 ch2_8.c

```
01 #include <stdio.h>
02 #include <stdlib.h>
03 #include <dirent.h>
04
05 int main() {
06     DIR *dp;
07     struct dirent *dent;
08
09     dp = opendir(".");
10
11     while ((dent = readdir(dp))) {
12         printf("Name : %s ", dent->d_name);
13         printf("Inode : %d\n", (int)dent->d_ino);
14     }
15
16     closedir(dp);
17 }
```

실행

```
$ ls
bit     ch2_2.c  ch2_4.c  ch2_6.c  ch2_8.c
ch2_1.c  ch2_3.c  ch2_5.c  ch2_7.c  ch2_8.out
$ ch2_8.out
Name : ch2_2.c Inode : 1057556
Name : ch2_1.c Inode : 1056653
Name : ch2_5.c Inode : 1057566
Name : ch2_7.c Inode : 1056650
(생략)
```

기능 요구 사항 (1)

1. 프로그램을 실행하면 사용자 명령을 입력 받을 수 있는 prompt를 표시한다.
   - 예: <현재 디렉토리 경로> $ _
2. 사용자가 명령을 입력하면 명령에 따라 아래 기능이 동작한다.
   - help : 명령어 목록과 기능 설명을 보여준다.
   - cd <path> : 현재 디렉토리를 path 경로로 이동한다.
   - mkdir <path> : path에 해당하는 디렉토리를 생성한다.
   - rmdir <path> : path에 해당하는 디렉토리를 삭제한다.
   - rename <source> <target> : source 디렉토리를 target 이름으로 변경한다.
   - ls : 현재 디렉토리의 내용(파일 및 sub 디렉토리 목록)을 보여준다. 이 때 파일/디렉토리명과 함께 종류(파일/디렉토리/기타 등등)도 알 수 있도록 표시한다.
3. 모든 기능은 수행이 완료된 후 성공/실패 여부를 출력한다. 실패시 어떤 에러가 원인인지 perror()를 사용하여 에러 메시지를 출력한다.

기능 요구 사항 (2)

1. 이 프로그램내에서 최 상위 디렉토리(/)가 실제로는 /tmp/test 디렉토리가 된다.
   a. 프로그램내에서 “cd /” 명령이 수행되면 실제로(프로그램 밖에서)는 /tmp/test 폴더로 이동한다.
   b. 프로그램내에서 “cd /test2” 명령이 수행되면 실제로는 /tmp/test/test2 폴더로 이동한다.
   c. 프로그램내에서 표시되는 현재 디렉토리 경로가 “/test2/test3” 라면 실제로는 “/tmp/test/test2/test3” 디렉토리에 위치한다.
   d. 즉, 프로그램내에서는 실제 /tmp/test 디렉토리보다 상위 디렉토리로 이동 불가, 상위 디렉토리의 디렉토리 생성/삭제/읽기 불가 하도록 기능을 제한해야 한다.

2. 만약 프로그램 실행시, /tmp/test 디렉토리가 존재하지 않는다면 /tmp/test 디렉토리를 생성해야 한다.

code template

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX_CMD_SIZE    (128)

int main(int argc, char **argv)
{
    char *command, *tok_str;
    char *current_dir = "/";

    command = (char*) malloc(MAX_CMD_SIZE);
    if (command == NULL) {
        perror("malloc");
        exit(1);
    }

    do {
        printf("%s $ ", current_dir);
        if (fgets(command, MAX_CMD_SIZE-1, stdin) == NULL) break;

        tok_str = strtok(command, " \n");
        if (tok_str == NULL) continue;

        if (strcmp(tok_str, "quit") == 0) {
            break;
        } else {
            // TODO: implement functions
            printf("your command: %s\n", tok_str);
            printf("and argument is ");

            tok_str = strtok(NULL, " \n");
            if (tok_str == NULL) {
                printf("NULL\n");
            } else {
                printf("%s\n", tok_str);
            }
        }
    } while (1);

    free(command);

    return 0;
}
```