※ 이번 실습부터는 자신의 shell 프로그램인 cmd를 작성하면서 테스트 해 나가도록 한다.

1. 자신의 홈 디렉토리 밑에 up 밑에 cmd 디렉토리를 만든 후, cmd 디렉토리 밑에 cmd.c와 Makefile을 복사한다.
   $ cd ~/up **$ ls** (cmd 있으면 아래 mkdir 무시)
   $ mkdir cmd **$ cd cmd**
   $ cp /home/jhshim/up/cmd/Makefile .
   (**.** 주의할 것)
   $ cp /home/jhshim/up/cmd/cmd.c .
   $ ls (Makefile과 cmd.c가 있어야 함)

2. **$ make**를 실행시켜 cmd라는 실행파일을 만든다. 이 cmd가 명령어를 입력 받고 실행시켜 주는 shell 프로그램이다.
   $ cmd를 실행시킨 후 명령어로 "**help**"를 입력하라. 지원되는 명령어 리스트가 나올 것이다. 명령어 이름을 입력하고 [엔터]하라. **exit**를 입력하면 cmd가 종료된다. **help**와 **exit**를 제외하고는 아직 해당 명령어가 구현되지 않았다고 나올 것이다. 향후 여러분이 명령어들을 순서적으로 구현할 계획이다.

Homework

---

1. 다음 페이지의 함수들을 참조하여 cmd.c 내의 기존 함수들을 수정하라. 이 함수들은 명령어 별로 지정된 명령어 인자의 개수와 옵션이 맞지 않으면, 에러 메시지와 함께 명령어 사용법을 출력해 준다. 기존 cmd.c를 수정하라.

~~2. 리눅스 서버에 있는 자신의 소스 파일 cmd.c를 자신의 PC로 다운 받고, 이 파일을 PC의 편집기를 사용하여 수정한 후, 다시 리눅스 서버로 업로드시키는 방법은 "리눅스 서버접속 및 로그인" 실습 노트의 "파일 다운로드 및 업로드" 부분을 참조하기 바란다. (ftp 프로그램을 활용함)~~
~~주의: 반드시 다운로드 받은 파일로 수정해야 한다. 메모장이나 Visual Studio에선 자동으로 리눅스의 유니코드(utf-8)로된 파일을 인식한다. 그러나 PC에서 파일을 새로 처음부터 입력하여 리눅스로 전송할 경우(한글완성형 KS5601 코드체계로 전송) 한글코드체계가 맞지 않아 리눅스에서 인식할 수 없게 될 수도 있다. 만약 어쩔 수 없이 파일을 처음부터 새로 입력했다면 PC에 저장된 파일을 메모장에서 불러와 [다른 이름 저장]을 띄운 후 아래쪽의 인코딩을 "utf-8"로 설정하고 저장하면 리눅스에서 정상으로 인식할 수 있다.~~

~~3. 2번 문제의 수정을 Putty 프로그램에서 직접 수정해도 되고, 1번 문제에서 설명한 것처럼 자신의 PC로 cmd.c를 다운받은 후 Visual Studio를 이용하여 수정해도 된다. 수정이 완료되면 cmd.c를 다시 서버로 업로드하기 바란다. 서버에 전송된 파일의 한글이 깨져 보인다면 PC의 메모장에서 해당 파일을 불러와 [다른 이름 저장]을 띄운 후 아래쪽의 인코딩을 "utf-8"로 설정하고 저장하라. 그런 후 다시 서버로 전송하라.~~

~~만약 VisualStudio가 없을 경우 메모장을 이용하여 cmd.c를 수정할 수 있다. 그런데 처음 서버에서 다운받아 메모장으로 파일을 불러 오면 여러 줄이 겹쳐 모두 한 줄에 보여진다. 이 경우~~
~~1) 일단 메모장에서 [다른 이름 저장]을 띄운 후 아래쪽의 인코딩을 "utf-8"로 설정하고 저장하라. (한글을 유니코드 utf-8형식으로 저장)~~
~~2) 그런 후 워드패드에서 해당 파일을 불러 오면 여러 줄이 정상적으로 보여지고 한글도 정상적으로 보여진다. (만약 위 1)과정을 생략했다면 한글이 정상적으로 보이지 않을 것이다.) 이때 또 무조건 한번 저장한다. (이때 자동으로 한글 완성형 KS5601으로 저장됨)~~
~~3) 워드패드가 한글완성형으로 저장했기 때문에 메모장에서 다시 해당 파일을 불러와 [다른 이름 저장]을 띄운 후 아래쪽의 인코딩을 "utf-8"로 설정하고 저장한다.~~
~~4) 이후 메모장에서 수정하고 저장하면 계속 한글 완성형이 아니라 유니코드 utf-8형식으로 자동 저장된다.~~
5) ~~수정이 완료되면 서버로 업로드하면 된다. 리눅스 서버에서 정상으로 한글을 인식할 수 있다.~~

<<<<<<<<<< 주의 >>>>>>>>>>

아래 코드의 인덴트(여백공간)는 탭 글자이다. cmd.c에 추가할 때는 인덴트(행의 앞쪽 공백) 부분을 **반드시 탭 키**를 이용하라. 프로그래머들끼리의 묵시적인 상호 약속이다. 이러한 룰들을 여러분들도 지키기 바란다.

```
static int
check_arg(int count)
{
    기존에 있던 return(0); 문장을 삭제한다.

    if (count < 0) { // AC_ANY(-100), AC_LESS_1(-1)인 경우
        // 인자 개수 가변: 인자를 주어도 되고 안 주어도 되는 경우
        count = -count;   // 음수를 양수로 변환
        if (argc <= count)// 입력된 인자 개수(argc)가 최대로 가능한
           return(0);     // 개수(count)보다 같거나 적으면 OK
    }
    if (argc == count)    // 인자 수가 일치함
        return(0);
    if (argc > count)
       // printf("Too many arguments.\n");
        printf("불필요한 명령어 인자가 있습니다.\n");
    else // (argc < count))
       // printf("Insufficient arguments.\n");
        printf("명령어 인자의 수가 부족합니다.\n");
    return(-1);
}


static int
check_opt(char *opt)
{
    기존에 있던 return(0); 문장을 삭제한다.

    int i, err = 0;

    for (i = 0; i < optc; ++i) {
        if (NOT_EQUAL(opt, optv[i]))    {
           // printf("Not supported option(%s).\n", optv[i]);
           printf("지원되지 않는 명령어 옵션(%s)입니다.\n", optv[i]);
           err = -1;
        }
    }
    return(err);   // 주어진 옵션이 하나라도 틀리면 에러(-1)
}

// 다음 페이지에 계속 . . .
```

```
static char *
get_argv_optv(char *cmd_line)
{
    기존에 있던 return(cmd = strtok(cmd_line, " \t\n")); 문장을 삭제한다.
    // 기존의 주석문 밑에 아래 코드를 삽입하라.

    char *tok; // 잘라낸 토근(단어) 문자열의 시작주소(첫 글자의 주소)

    argc = optc = 0;
    // 주의: 아래 두 개의 " \t\n"에서 첫번째 \ 앞에 빈 공백 " " 문자가 들어감
    if ((cmd = strtok(cmd_line, " \t\n")) == NULL) // 첫 토큰(단어) 잘라 냄
        return(NULL); // 명령어 입력 없이 그냥 엔터만 친 경우
    // cmd는 명령 문자열의 시작주소임. 예) "ln"
    for ( ; (tok = strtok(NULL, " \t\n")) != NULL; ) { // 다음 토큰 잘라 냄
        // tok[0]: 잘라된 토큰의 첫 글자
        if (tok[0] == '-')       // 토큰이 옵션임을 의미. 예) optv[0]->"-s"
           optv[optc++] = tok;   // optv[optc]에 토큰 주소 저장하고 optc++
        else                     // 토큰이 명령어 인자임을 의미
           argv[argc++] = tok;   // argv[argc]에 토큰 주소 저장하고 argc++
    }                            // 예) argv[0]->"file1", argv[1]->"file2"
    return(cmd);
    // 이 함수가 리턴해도 각 문자열은 여전히 cmd_line[]에 저장되어 있고
    // 각 문자열의 시작주소만 cmd, optv[], argv[]에 저장되어 있음
}
```

4. cmd.c를 위처럼 수정한 후 리눅스에서 make를 해 보라. 많은 에러가 발생할 것이다. 나도 30년이상 프로그래밍 했지만, 한번도 한번에 컴파일된 적이 없다. 영원한 로망이라 할까나? 어쨌든 인내를 가지고 모든 컴파일 에러를 잡도록 하라. 성공했으면 이제 cmd를 실행시켜라. 그리고 각 명령어 별로 인자 수를 0~3개까지 아무 이름으로 입력해 보고 옵션도 주어 보라. 예를 들어, 명령어가 ls 인 경우 다음처럼 해 보라. "**>** "는 cmd의 명령 프롬프트다.

```
> ls                    // 인자가 없는 경우, 정상
> ls ../pr4             // 인자가 하나인 경우, 정상
> ls -l                 // 옵션 있고, 인자가 없는 경우, 정상
> ls -l ../jhshim       // 옵션 있고, 인자가 하나인 경우, 정상
> ls a b                // 인자가 둘인 경우, 에러 발생
> ls -l a b             // 정상 옵션 있고, 인자가 둘인 경우, 에러 발생
> ls -l -a              // 옵션 에러
> ls -l -o a b          // 옵션 에러, 인자가 둘인 경우, 에러 발생
```

이런 테스트를 각 명령어 별로 모두 테스트 해 보라.

5. 명령 창에서 다음을 실행하여 모든 것이 정상인지 확인하라.

```
$ eshw 5 0   // 에러가 발생한 항목에 대해서만 보여 주면서 확인
$ fshw 5 0   // 모든 항목을 보여 주면서 확인
$ progtest 5 0       // 실습 5의 cmd-0에 대해 테스트함
```