REPORT

명령어 프로젝트

rmdir_최종보고서

과목명	리눅스 프로그래밍
담당 교수님	김영주
학과	컴퓨터공학과
학번	173312
이름	박건웅
작성일	2020.05.29
제출일	2020.06.21

MOKPO NATIONAL UNIVERSITY

목 차

1. 설계 요소

(1) 설계 주제

: 기존 Linux 명령어와 동일한 기능과 옵션을 수행하는 명령어를 설계하라

(2) 설계 목표

① 프로젝트 목적 및 목표를 설정하고 이에 따른 요구사항 분석하고 현실적 제약 조건을 고려한 주제 선정

• 프로젝트 목적

: Linux 명령어 rmdir의 기능과 옵션을 수행하는 명령어 설계, 구현

• 프로젝트 목표

- 리눅스의 커널과 시스템 콜의 상호작용을 이해한다.
- 리눅스 시스템 프로그래밍을 설계하는 능력과 조건에 따른 구현을 배양
- 이론이나 알고리즘을 수식 또는 프로그래밍 등을 통해 검증할 수 있는 능력 증진

② 프로젝트의 필요성, 활용방안 등을 고려한 설계 제안

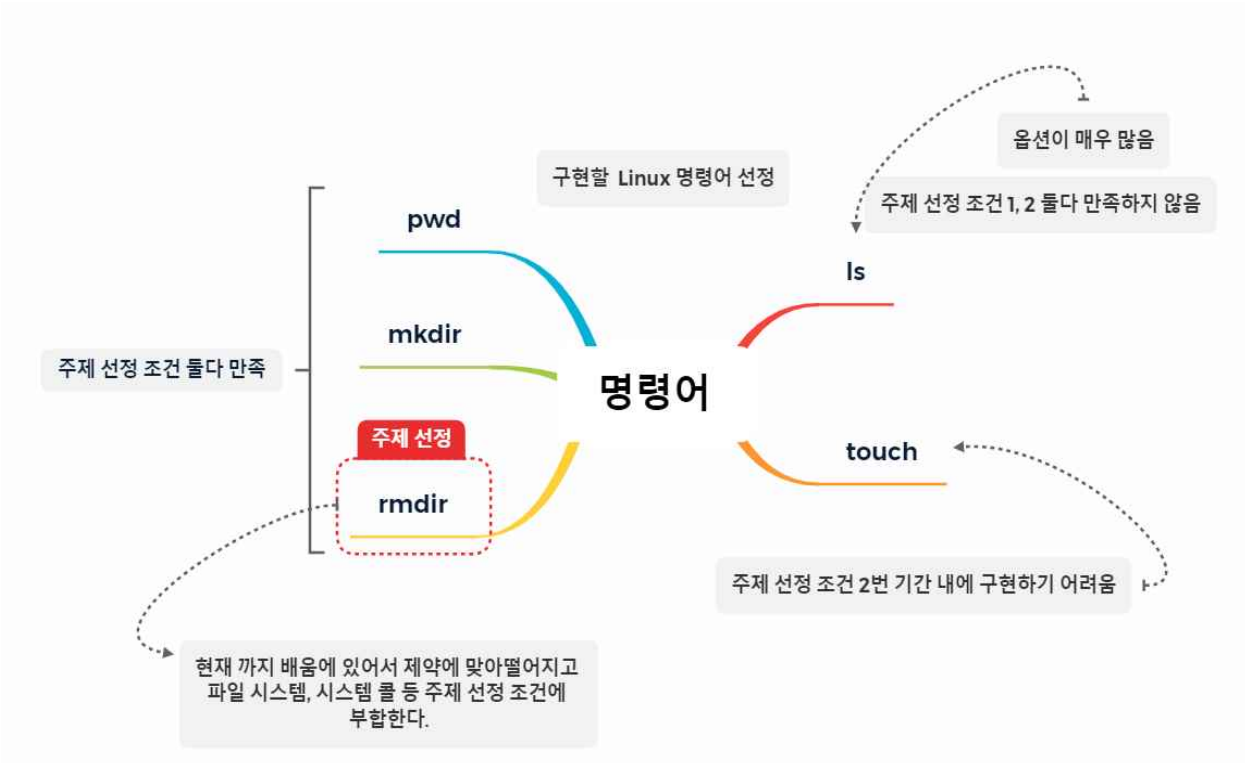
• **프로젝트의 필요성** : 본 교과목을 통해 시스템 소프트웨어의 기본원리를 이해하고 실제적인 프로그램을 작성할 수 있도록 리눅스 시스템 프로그래밍에서 사용하는 시스템 호출이나 라이브러리 함수를 다루기 때문에 이를 응용한 프로젝트가 필요함

(3) 주제 선정

① 현재 이 프로젝트를 진행함으로써 리눅스 시스템 프로그래밍을 설계하는 능력에 맞는가

② 코딩과 설계에 있어서 기간 내에 구현할 수 있을지 고려한다.

• 주제 선정 마인드 맵



2. 현실적 제한 조건

(1) 제품화

- 기존 리눅스 명령어와 동일한 기능을 옵션까지 완벽하게 수행하는 명령어를 1개를 설계하고 제작
- 리눅스 시스템 콜을 최대한 사용하여 구현

(2) 환경

- 실행 환경 : Linux
- 사용자 인터페이스 : 콘솔(consol)에서 실행 가능한 프로그램

3. 인원구성 및 추진 일정

(1) 인원구성 : 개인

(2) 기간 : 2020. 05. 29. ~2020. 06. 21

일정 / 목록	05.29~05.30	05.31~06.07	06.08~06.14	06.15~06.21
자료조사, 명령어 선택				
계획서 작성				
코딩 및 디버그				
결과 보고서 작성				
최종 결과서 제출				

4. 설계 내용

(1) 시스템 호출

시스템 호출이란 프로그래밍 언어에서 지원하지 않는 기능에 대하여 운영체제의 루틴을 호출하여 이용하는 것을 말한다.

① 시스템 호출의 기능

- 사용자 모드에 있는 응용 프로그램이 커널의 기능을 사용할 수 있도록 한다.
- 시스템 호출을 하면 사용자 모드에서 커널 모드로 바뀐다.
- 커널에서 시스템 호출을 처리하면 커널 모드에서 사용자 모드로 돌아가 작업을 계속한다.

② 시스템 호출의 유형

1) 프로세서 제어 (Process Control)
2) 파일 조작 (File Manipulation)
3) 장치 관리 (Device Management)
4) 정보 유지 (Information maintenance)
5) 통신 (Communication)

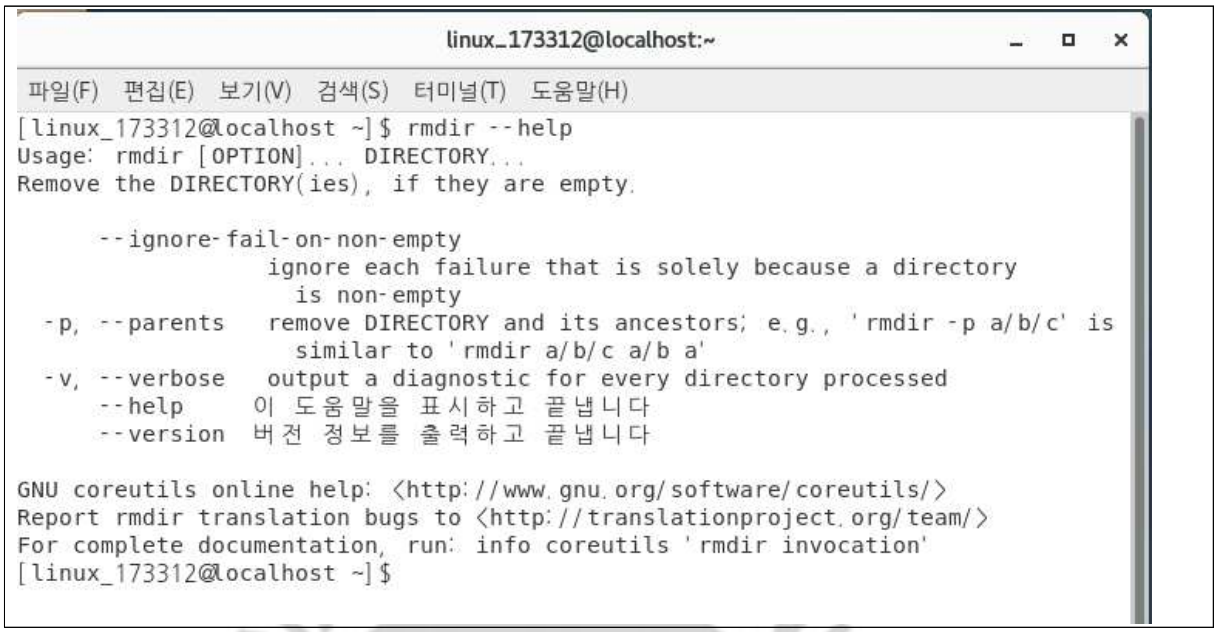
(2) 명령어 소개 : rmdir

remove directory의 준말로, 디렉터리를 삭제하는 명령어이다. 하지만 빈 디렉터리일 경우에만 가능하다.

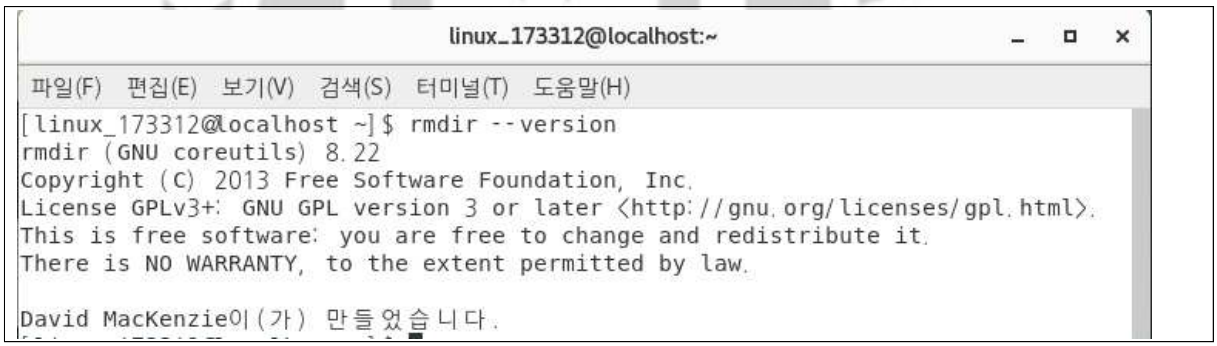
• 형식 : rmdir - [option] [디렉터리명]

(3) 명령어 옵션

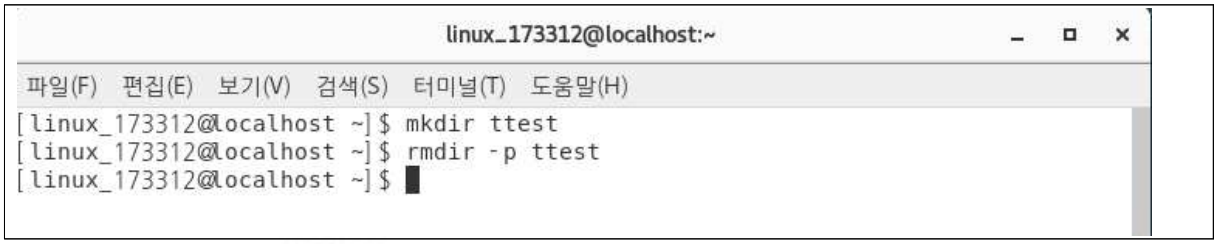
① rmdir —help



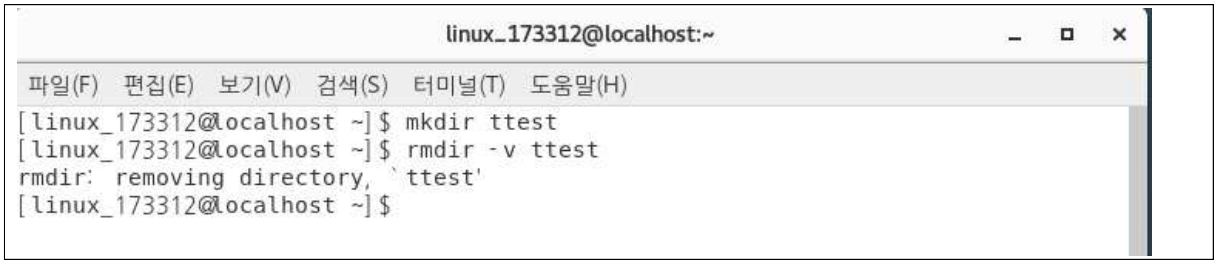
② rmdir —version



③ rmdir –p

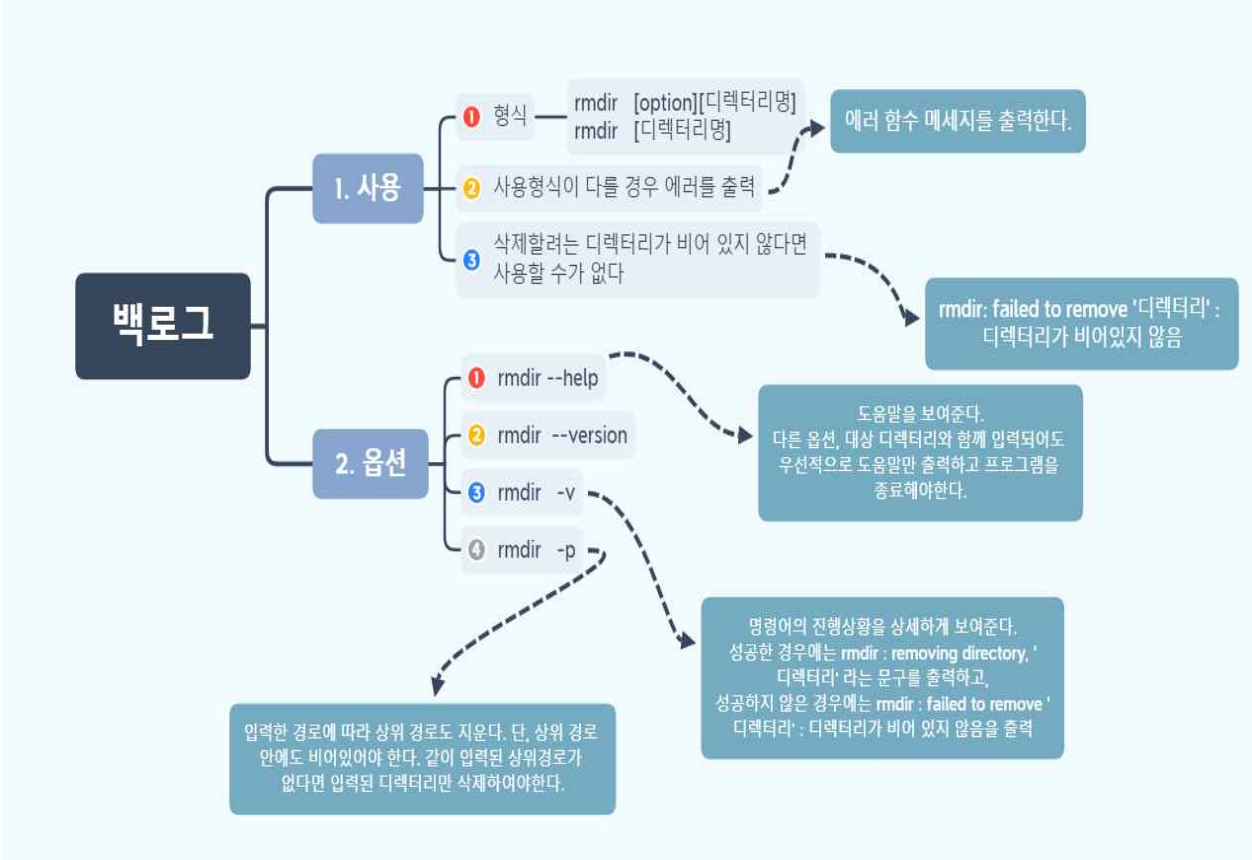


④ rmdir –v



(4) 제품 백로그 (Product Backlog)

- 논리 흐름도



5. 제작 및 구현

(1) 선정한 시스템 콜

① int unlink(char *path)	path : 삭제할 파일의 경로 path로 파일을 삭제한다. 삭제 성공 : 0, 실패시 : –1
unlink() 함수는 시스템 호출을 사용해 파일에 대한 디렉터리 항목을 지우고 링크 개수를 감소시킨다. unlink()를 사용하려면 파일의 디렉터리 항목에 포함된 모든 디렉터리에 대해 쓰기와 실행 권한이 필요하다.	
② int remove(char *path)	path : 삭제할 파일의 경로 path로 파일을 삭제한다. 삭제 성공 : 0, 실패시 : –1
remove 함수는 삭제 대상이 파일이면 unlink 함수와 동일하고, 삭제 대상이 디렉터리면 rmdir과 동일하다.	
차이점	unlink()는 비어있는 디렉터리는 삭제할 수 없고, remove()는 비어있는 디렉터리도 삭제가 가능하다. 하지만, 둘 다 파일을 가지고 있는 디렉터리는 삭제할 수 없다.
③ rmdir(char *path)	비어 있는 디렉터리를 삭제하는 시스템 호출이다.
∴ **결론**	unlink()는 비어있는 디렉터리를 삭제할 수 없고, rmdir()은 현재 구현하고자 하는 명령어이기 때문에 remove()로 구현하기로 결정하였다.

(2) 함수 정의

함수 정의	① int argv_check(char *argv1, char *argv2)
설명	: 명령 인자를 판단하는 함수로서 main 함수에 스위치문에 적용되어 함수들을 호출 할 수 있게 해준다. 조건문으로 작동하는 시스템으로 NULL값일 경우 False, 나머지 조건들은 옵션에 작동한다. 그리고 조건이 없을 경우 else 로 빠져나가게 된다.

함수 정의	② void help()
설명	: argv[1]에 '--help' 가 입력되었을 시, help()문을 출력해주는 함수 이다. '--help' 옵션 이후 어떠한 입력이 들어와도 도움말을 출력하고 종료한다.

함수 정의	③ void version()
설명	: argv[1]에 '--version'이 입력되었을 시, version()문을 출력해주는 함수이다. '--version' 옵션 이후 어떠한 입력이 들어와도 버전정보를 출력하고 종료한다.

함수 정의	④ void error_m(char *argv1)
설명	: 존재하지 않은 옵션이 입력되었을 시, error_m()문을 출력해주는 함수이다. ' [옵션] [존재하지 않는 옵션] ' 인 경우에도 에러 메세지를 출력하고 종료한다.

함수 정의	⑤ void Remove(char *path)
설명	1) 디렉터리가 존재하는지 판단 2) 디렉터리가 비었는지 확인하고 삭제한다. : 디렉터리 구조체에서 디렉터리를 열고 각 디렉터리 엔트리를 반복해서 읽어온다. 현재 디렉터리 조건으로 비교하고 경로명을 읽어서 remove을 이용하여 삭제 시킨다.

함수 정의	⑥ void option_v(char *argv1, char *path)
설명	⑤ 함수를 이용해 주는데 'rmdir –v'처럼 제거하고 제거된 경로를 출력해 제거된 것을 보여준다. : argv[1]에 '-v'을 입력 받았을 시 option_v() 함수를 호출한다.

함수 정의	⑦ void option_p(char *argv1, char *path)
설명	⑤ 함수를 이용해 주는데 'rmdir –p'처럼 경로를 읽어 부모 디렉터리 까지 삭제시켜주는 함수이다. 1) path 경로를 삭제시켜 줌으로써 자식 경로를 없애고, 2) path 경로를 새로운 변수 포인트 함수(*e_path)에 받아와서 3) strtok()를 이용하여 자식 경로를 지워 부모 경로인 (strpath) 4) remove(strpath) 함으로써 부모 디렉터리를 제거한다.

(3) 동작 프로세스, 순서도

① 입력 받은 인자가 판단해주는 함수 (int argv_check)로 들어가 [옵션], [디렉토리] 기능을 확인해 준다.

② 판단을 내린 값으로 메인 함수로 들어가 스위치 구문 (switch) 을 작동시킨다.

③ 스위치 구문에서 받은 함수 호출을 통해 함수가 작동하게 된다.

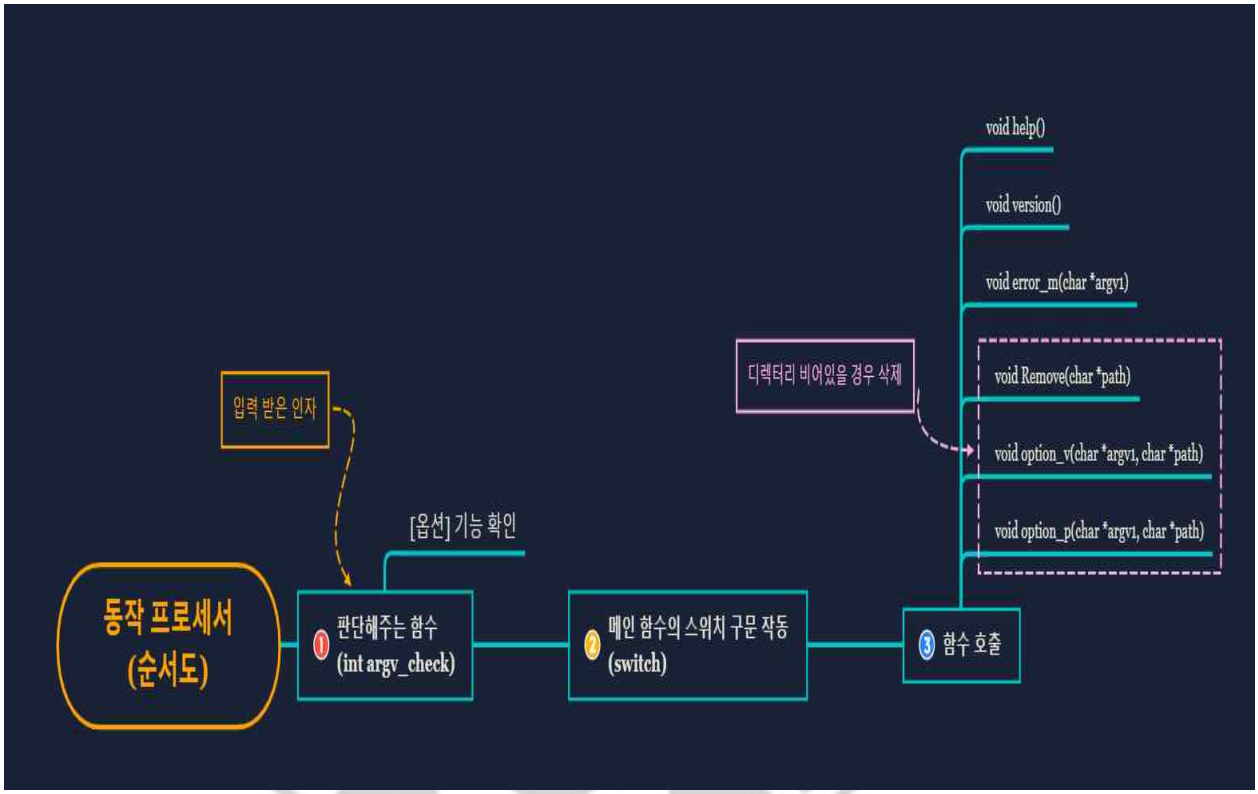
- 옵션 인자가 없을 경우 함수 (void error_m)을 호출하게 된다.

- 옵션 인자가 '--help' 일 경우 도움말 정보를 출력한다. (void help)

- 옵션 인자가 '--version' 일 경우 버전 정보를 출력한다. (void version)

- 옵션 인자가 '-v' , '-p'일 경우 각 인자에 맞게 함수를 호출한다. (void option_v, void option_p)

- Remove()에서 디렉터리를 읽어올 때 비어있는지 분석하고 비어있을 경우에만 삭제 하고 아닐 경우 비어있지 않은 경고문 발생. ('-v', '-p' 옵션 경우 동일)



(4) 소스 코드

제목	rmdIr.c
소스 코드 (Source Code)	#include <stdio.h> //표준 입출력 #include <stdlib.h> //표준 라이브러리 #include <string.h> //문자열 #include <unistd.h> //파일 입출력 라이브러리 #include <fcntl.h> //파일 입출력 #include <sys/types.h> //디렉토리 라이브러리, readdir #include <sys/stat.h> #include <dirent.h> // 디렉토리 라이브러리 #define False -1; int argv_check(char *argv1 , char *argv2) ; //명령 인자 판단 함수 void help(); //도움말 함수 void version(); // 버젼 함수 void error_m(char *argv1); // 에러 함수 void Remove(char *path); // 지우는 함수 void option_v(char *argv1, char *path); // -v 옵션 함수 void option_p(char *argv1, char *path); // -p 옵션 함수 /* void print_argv(char *argv[]){ int k=0; for(k=0; !(argv[k]==NULL); k++) { printf("argv[%d]%s₩n",k,argv[k]); } } */

제목	rmdlr.c
소스 코드 (Source Code)	int main(int argc, char *argv[]) { //print_argv(argv); int num; num = argv_check(argv[1], argv[2]); // argv 확인 //printf("num: %d₩n", num); enum exam {help_n = 1 , version_n, option_v_n, option_p_n, Remove_n}; //////// 스위치 ////////// switch (num) { case help_n: help(); // --help 함수 호출 break; case version_n: version(); // --version 함수 호출 break; case option_v_n: // -v 옵션 함수 호출 option_v(argv[1], argv[2]); break; case option_p_n: // -p 옵션 함수 호출 option_p(argv[1], argv[2]); break; case Remove_n: // 제거하는 함수 호출 Remove(argv[1]); break; default: error_m(argv[1]); // 에러 함수 호출 } }

제목	rmdlr.c
소스 코드 (Source Code)	``` ////////////// 명령 인자 판단 함수////////// int argv_check(char *argv1 , char *argv2) { if (argv1 == NULL) { //변수 값이 없을때 argv1의 값 NULL return False; } else if (!strcmp(argv1, "--help")) { return 1; } else if(!strcmp(argv1, "--version")){ return 2; } else if(!strcmp(argv1, "-v")) { return 3; } else if(!strcmp(argv1, "-p")){ return 4; } else if (!strcmp(argv1, ".") || (argv1, "..")) { return 5; } else{ }; } ```

제목	rmdIr.c
소스 코드 (Source Code)	////// 도움말 메세지 함수 ///// void help () { printf("Usage: ./rmdir [OPTION]... DIRECTORY...₩n"); printf("Remove the DIRECTORY(ies), if they are empty.₩n₩n"); printf(" --ignore-fail-on-non-empty₩n"); printf(" ignore each failure that is solely because a directory₩n"); printf(" is non-empty₩n"); printf(" -p, --parents remove DIRECTORY and its ancestors; e.g., 'rmdir -p a/b/c' is₩n"); printf(" similar to 'rmdir a/b/c a/b a'₩n"); printf(" -v, --verbose output a diagnostic for every directory processed₩n"); printf(" --help 이 도움말을 표시하고 끝냅니다₩n"); printf(" --version 버전 정보를 출력하고 끝냅니다₩n₩n"); printf("GNU coreutils online help: <http://www.gnu.org/software/coreutils/>₩n"); printf("Report rmdir translation bugs to <http://translationproject.org/team/>₩n"); printf("For complete documentation, run: info coreutils 'rmdir invocation'₩n"); } //////// 버젼 메세지 함수 ///// void version() { printf("rmdir (GNU coreutils) 8.22₩n"); printf("Copyright (C) 2013 Free Software Foundation, Inc.₩n"); printf("License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>.₩n"); printf("This is free software: you are free to change and redistribute it.₩nThere is NO WARRANTY, to the extent permitted by law.₩n"); printf("₩n박건웅이 만들었습니다.₩n"); } /////// 에러 메세지 함수 /////// void error_m(char *argv1) { printf("./rmdir : 인식할수 없는 옵션 '%s' ₩n", argv1); printf("Try 'rmdir --help' for more information.₩n"); }

제목	rmdlr.c
소스 코드 (Source Code)	``` /////// 디렉터리 삭제하는 함수 //////// void Remove(char *path) { struct stat statbuf; if(stat(path, &statbuf)<0) //stat() : 파일에 대한 상태 정보 , 파일의 상태 정보를 stat 구조체 buf에 저장 { printf("rmdir: failed to remove '%s' : 그런 파일이나 디렉터리가 없습니다. ₩n", path); error_m(path); exit(1); } /// S_ISDIR : 디렉터리 파일 검사하는 매크로 함수, st_mode : stat구조체의 필드 내부 if(S_ISDIR(statbuf.st_mode)) { // DIR : 디렉터리 구조체 , opendir : 디렉터리 열기 DIR *dir = opendir(path); struct dirent *de=NULL; while((de = readdir(dir))!=NULL) //각 디렉터리 엔트리를 반복해서 읽음 { //strcmp : 두 문자열 비교 (엔트리 내 파일 이름 과 현재 디렉터리 대상) if(strcmp(de->d_name,".")==0) { continue; } if(strcmp(de->d_name,"..")==0) { continue; } char npath[PATH_MAX]; sprintf(npath,"%s/%s",path, de->d_name); Remove(npath); } closedir(dir); remove(path); } else { printf("./rmdir: failed to remove `%s': 디렉터리가 비어있지 않음 ₩n", path); } } ```

제목	rmdIr.c
소스 코드 (Source Code)	``` ///////// rmdir -v 옵션 함수 /////////// void option_v(char *argv1, char *path) { struct stat statbuf; if(stat(path, &statbuf)<0) //stat() : 파일에 대한 상태 정보 , 파일의 상태 정보를 stat 구조체 buf에 저장 { error_m(path); exit(1); } /// S_ISDIR : 디렉터리 파일 검사하는 매크로 함수, st_mode : stat구조체의 필드 내부 if(S_ISDIR(statbuf.st_mode)) { // DIR : 디렉터리 구조체 , opendir : 디렉터리 열기 DIR *dir = opendir(path); struct dirent *de=NULL; while((de = readdir(dir))!=NULL) //각 디렉터리 엔트리를 반복해서 읽음 { //strcmp : 두 문자열 비교 (엔트리 내 파일 이름 과 현재 디렉터리 대상) if(strcmp(de->d_name,".")==0) { continue; } if(strcmp(de->d_name,"..")==0) { continue; } char npath[PATH_MAX]; sprintf(npath,"%s/%s",path, de->d_name); Remove(npath); } closedir(dir); remove(path); printf("./rmdir: removing directory, '%s'₩n", path); } else { printf("./rmdir: failed to remove `%s': 디렉터리가 비어있지 않음 ₩n", path); } } ```

제목	rmdIr.c

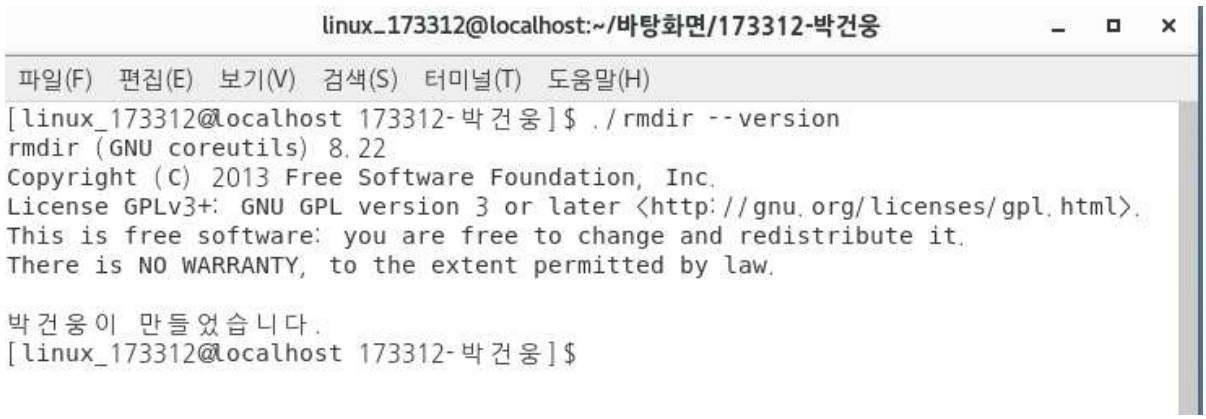
소스 코드 (Source Code)

```
//////// rmdir -p 옵션 함수 ////////
void option_p(char *argv1, char *path)
{
    struct stat statbuf;
     if(stat(path, &statbuf)<0)       //stat() : 파일에 대한 상태 정보 , 파일의 상태 정보를 stat 구조체  buf에 저장
    {
        error_m(path);
        exit(1);
    }
    /// S_ISDIR : 디렉터리 파일 검사하는 매크로 함수, st_mode : stat구조체의 필드 내부
    if(S_ISDIR(statbuf.st_mode))
    {
        // DIR : 디렉터리 구조체 , opendir : 디렉터리 열기
        DIR *dir = opendir(path);
        struct dirent *de=NULL;
        while((de = readdir(dir))!=NULL)     //각 디렉터리 엔트리를 반복해서 읽음
        {
            //strcmp : 두 문자열 비교 (엔트리 내 파일 이름 과 현재 디렉터리 대상)
            if(strcmp(de->d_name,".")==0)
            {
                continue;
            }
            if(strcmp(de->d_name,"..")==0)
            {
                continue;
            }
            char npath[PATH_MAX];
            sprintf(npath,"%s/%s",path, de->d_name);
            Remove(npath);
        }
        closedir(dir);
        printf("%s ₩n", path);
        remove(path);               // 자식 디렉터리 제거 , path = 부모 경로 + 자식 경로
        ///////  부모 경로 (상위 경로) ///////
        char *e_path = path;
        char *strpath = strtok(e_path, "/");          //strtok : 문자열 제거 함수
        printf("%s ₩n", strpath);
        remove(strpath);          // 부모 디렉터리 제거
    }
    else
    {
       printf("./rmdir: failed to remove `%s': 디렉터리가 비어있지 않음 ₩n", path);
    }
}
```

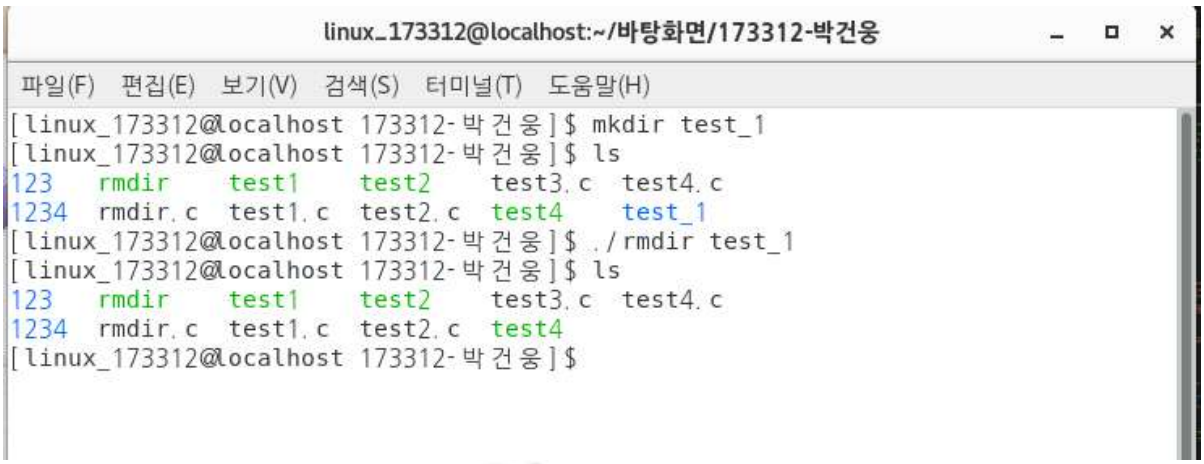
(5) 실행 결과

① '--help' 구현 / 다른 옵션도 붙어도 '--help' 구현

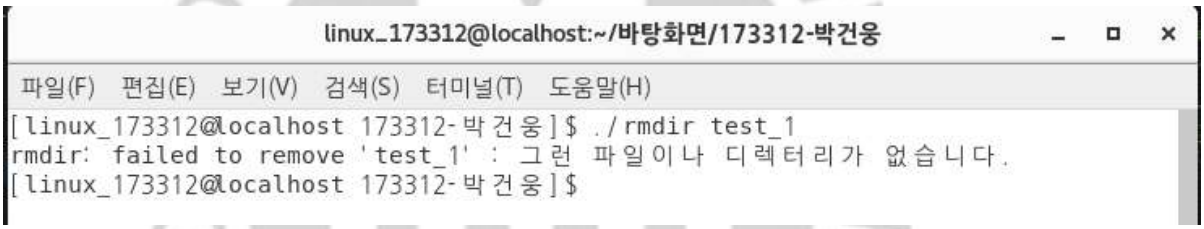
② '--version' 구현



③ 디렉토리 제거

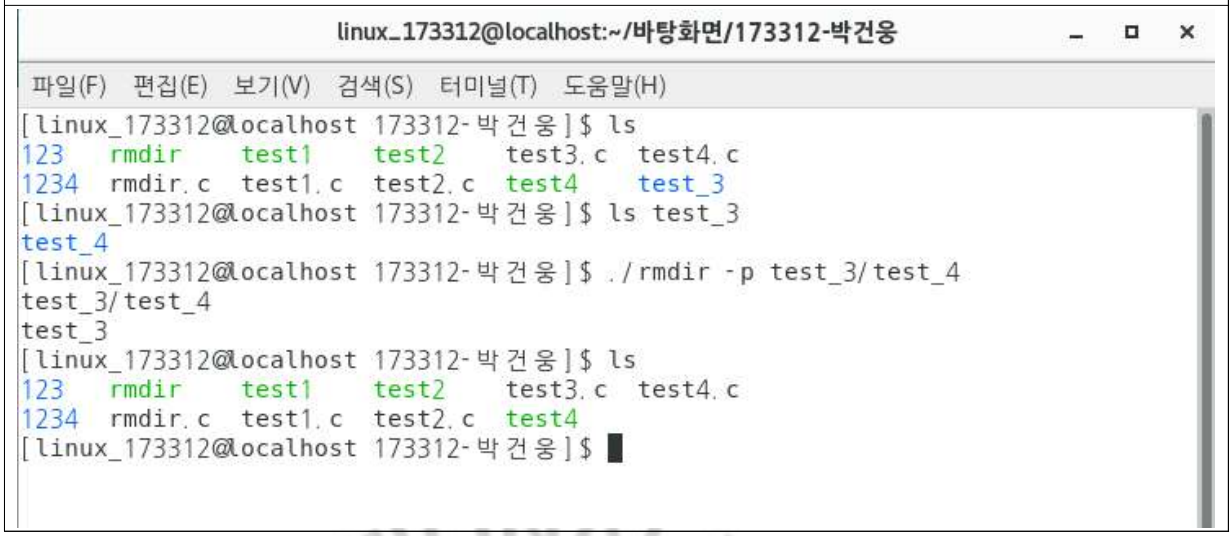


④ 제거할려는 디렉터리가 없을 경우

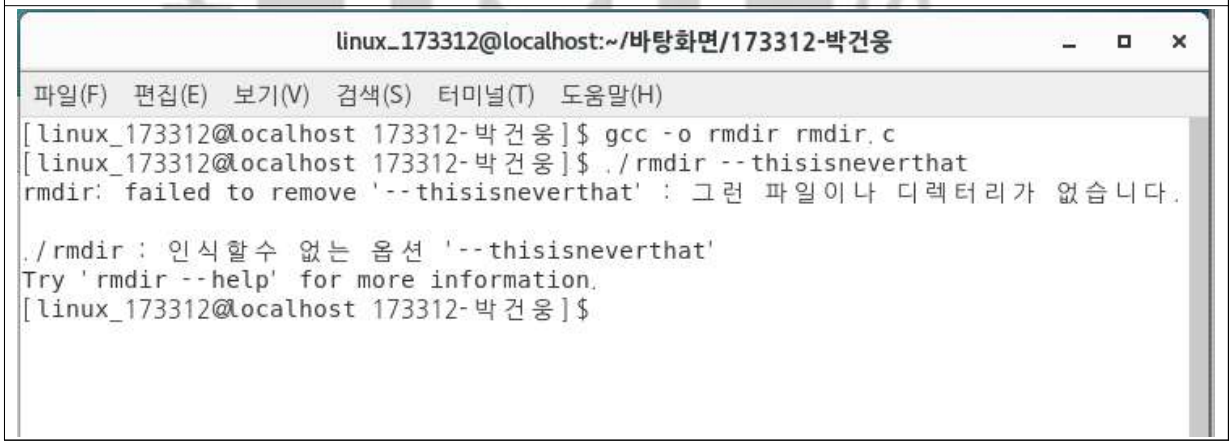


⑤ '-v' 옵션 구현

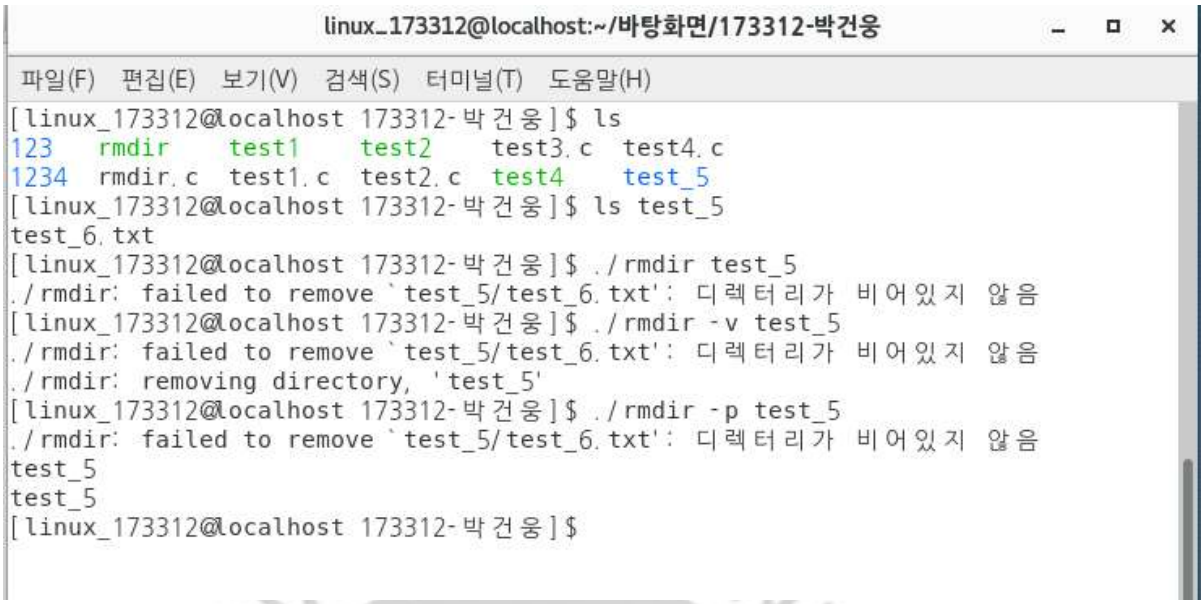
⑥ '-p' 옵션 구현



⑦ 잘못된 옵션 인자 구현시



⑧ 디렉터리가 비어있지 않을 경우



6. 문제점 및 개선

<table>
<tr><td rowspan="2">문제점</td><td>① Remove() 함수가 나오기 전 함수 형태 (프로토타입)</td></tr>
<tr><td>int Remove(char *dir_name) {
char strPath[10] = "./";
strcat(strPath,dir_name);
remove(strPath);
: 1) 이 형태로 strcat을 사용해 디렉터리 와 경로를 합치는 함수를 이용해 remove 시스템 콜을 사용하였으나, 현재 경로만 삭제하는 기능이 구현됨.
2) 또한 디렉터리가 비어있는지에 대한 판단 유무가 불가능함</td></tr>
<tr><td>개선 방향성</td><td>위의 문제를 보완할 디렉터리 존재 여부 판단 함수 생성</td></tr>
<tr><td rowspan="2">문제점</td><td>② 디렉터리 존재 여부 함수 생성과 결과</td></tr>
<tr><td>

```
///// 디렉터리 존재 여부 판단. 존재 True, 아니면 False /////
int d_exist(char path[], char *argv2)
{
    int boolean = 0;
    DIR *d_info;
    struct dirent *d_space;
    d_info = opendir(path);      //path 디렉터리 정보를 d_info에 저장
    if(d_info != NULL)
    {
        while(d_space = readdir(d_info))
        {   //path에 argv2 디렉터리가  있다면 boolean 1 바뀜
            if(!strcmp(d_space->d_name, argv2))
            {
                boolean = 1;
                break;
            }
        }
        closedir(d_info);
    }
    if(boolean == 1) {
        return True;
    }
    else
        return False;
}
```

</td></tr>
</table>

	1) #define True 0; #define False –1; 을 이용하여 path의 정보를 이용해 d_info에 저장하고 디렉터리가 존재하면 True , 디렉터리가 존재하지 않으면 False로 판별하였다. 2) 하지만 위의 식의 함수대로 만들면 문제점이 발생한다. 3) 디렉터리가 존재 여부로 판단을 내리고 다시 또 디렉터리가 비었는지 함수를 만들어야 한다. 4) 또한 입력된 디렉터리 인자가 경로명인지 디렉터리명인지 다시 구분하여야 하고 5) main 함수에서 불러일으킬 때 조건이 까다로워진다. 6) int 형 변수로 return 값을 모두 True, False로 나뉠 수 없다. ∴ 1번 째 문제점 현재 경로 뿐 아니라 디렉터리 판단 유무까지는 파악할 수 있으나 조건문이 까다롭게 들어간다는 불편함이 있다.
개선 방향성	1) void 함수로 써서 ruturn 값이 꼬이지 않게 보기 편하게 설정 2) 조건이 까다롭지 않게 함수 호출로서 이용하게 설정 3) 디렉터리 존재 유무함수가 계속 이어지는 형태가 아니라 한 함수에 담아서 직관적으로 보이는 형태를 설정 4) 인자 형태를 간소화
개선	위의 문제점들을 보완해 현재의 함수들의 형태로 나타내서 main 함수에 스위치 구문으로 직관적이고 간소화 됨
문제점	③ '-p' 의 옵션 구현 중 자식 경로 삭제는 가능하나 부모 디렉터리 제거하는 기능이 문제점이 발생
	이를 해결하기위해 문자열 길이를 축소 하기위해 (strlen) 함수 이용 path[strlen(path)-strlen(de->d_name)] = '₩0'; 를 이용하여 반복문에 작동하였으나 위의 식대로 호출 할 수가 없다. 위의 식대로면 구조체로 표현해야 되고 argv인자를 쪼개야 된다. 함수를 바꿔야 되기에 실현 불가능
개선	문자열을 자르는 'strtok' 라이브러리를 부른다. 1) path 경로를 새로운 포인터 *e_path에 넣고 2) 자식 경로를 strtok(e_path, "/")를 이용하여 제거한다.

참고자료

• 리눅스 프로그래밍 책

-https://www.it-note.kr/217

-https://shaeod.tistory.com/320

• 시스템 콜, 함수 자료

-https://m.blog.naver.com/PostView.nhn?blogId=cache798&logNo=1300802822 21&proxyReferer=https:%2F%2Fwww.google.com%2F

• 문자열 자르기 자료

-https://dojang.io/mod/page/view.php?id=376

-http://forum.falinux.com/zbxe/?mid=C_LIB&sort_index=voted_count&order_type=desc&listStyle=list&page=3&document_srl=408105

-http://ehpub.co.kr/c%EC%96%B8%EC%96%B4-%EC%86%8C%EC%8A%A4-%EB%AC%B8%EC%9E%90%EC%97%B4%EC%97%90%EC%84%9C-%ED%8A%B9%EC%A0%95-%EB%AC%B8%EC%9E%90-%EC%A0%9C%EA%B1%B0%ED%95%98%EA%B8%B0/