

리눅스 명령어 구현 및 깃허브 정리

깃허브 : <https://github.com/PARK-DOHYUN/SystemProgramming>
2021663028 박도현



contents

01

명령어 목록

02

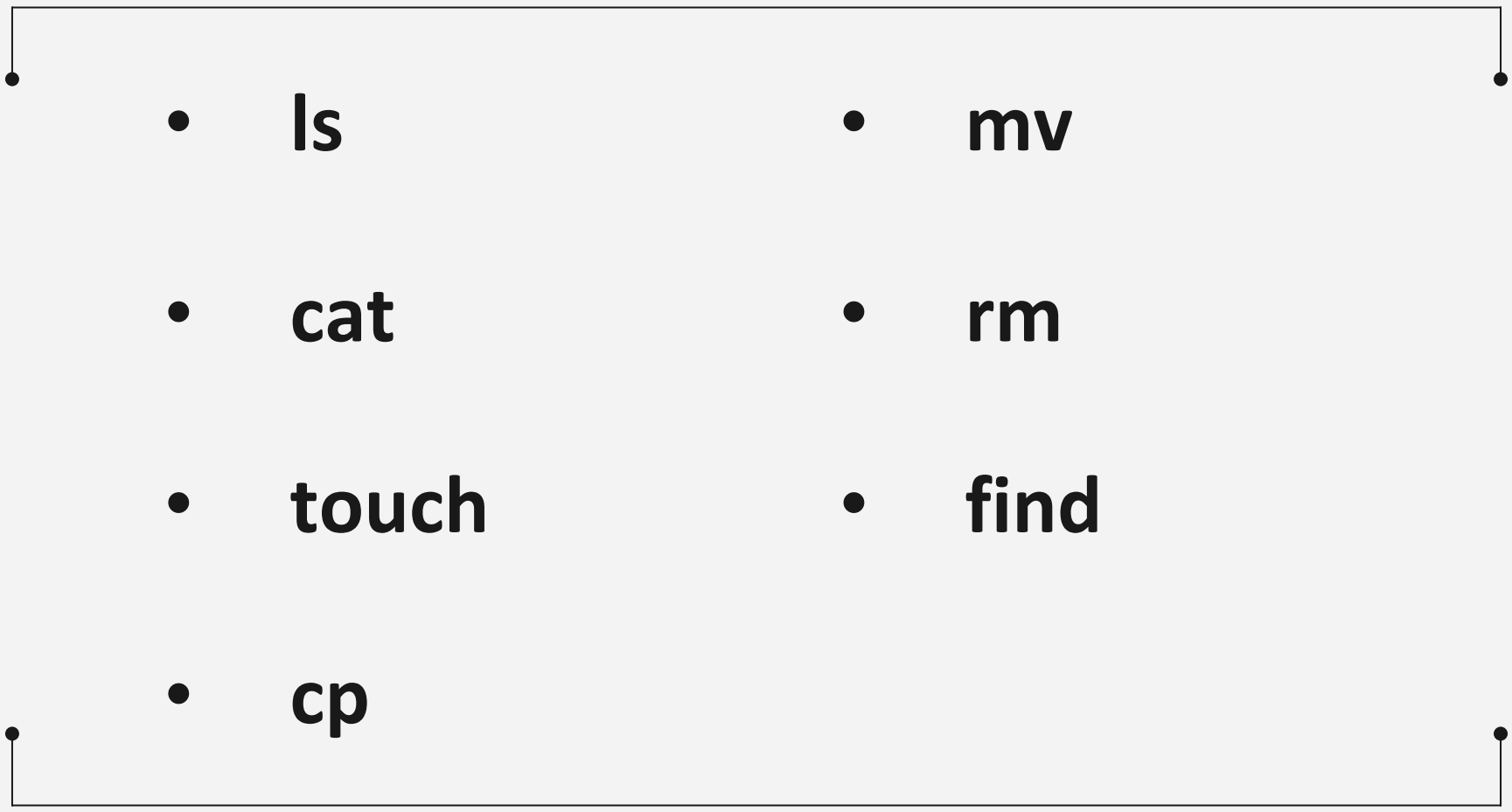
깃허브 정리





01

명령어 목록



- **ls**

- **mv**

- **cat**

- **rm**

- **touch**

- **find**

- **cp**

ls

• -a	: 숨김 파일 포함	./ls -a
• -l	: 상세 정보 표시	./ls -l
• -lh	: 사람이 읽기 쉬운 크기 (K,M)	./ls -lh
• -lt	: 수정 시간 기준 정렬	./ls -lt
• -s	: 블록 단위 용량 표시	./ls -s
• -r	: 역순 정렬	./ls -ltr
• -R	: 재귀적 하위 디렉토리 표시	./ls -R
• -i	: inode 번호 출력	./ls -i
• -e	: 확장자 별로 묶기	./ls -e
• -d	: 디렉토리 우선 정렬	./ls -d
• -f [확장자]	: 특정 확장자만 표시	./ls -f .txt

옵션

예시

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run
```

```
Makefile
```

```
cat
```

```
cat_run
```

```
cp
```

```
cp_run
```

```
find
```

```
find_run
```

```
ls
```

```
ls_run
```

```
mv
```

```
mv_run
```

```
rm
```

```
rm_run
```

```
touch
```

```
touch_run
```

```
// === 4단계: 파일 정보 출력 ===
```

```
// 정렬된 순서대로 각 파일의 정보를 출력
```

```
for (int i = 0; i < file_count; i++) {
```

```
    print_file_info(&files[i], options);
```

```
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -a
```

```
.  
..  
Makefile  
cat  
cat_run  
cp  
cp_run  
find  
find_run  
ls  
ls_run  
mv  
mv_run  
rm  
rm_run  
touch  
touch_run
```

```
int should_show_file(const char *filename, ls_options_t *options) {  
    // 숨김 파일 체크: 점으로 시작하는 파일 처리  
    if (!options->show_all && filename[0] == '.') {  
        return 0; // 숨김 파일 표시 옵션이 없으면 숨김  
    }  
}
```

```
// 상세 정보 출력 (long format 옵션이 설정된 경우)
if (options->long_format) {
    // 파일 권한 문자열 생성 및 출력
    char permissions[11];
    format_permissions(file->stat_info.st_mode, permissions);
    printf("%s ", permissions);

    // 하드링크 수 출력
    printf("%3lu ", (unsigned long)file->stat_info.st_nlink);

    // 소유자 및 그룹 정보 조회 및 출력
    struct passwd *pwd = getpwuid(file->stat_info.st_uid);
    struct group *grp = getgrgid(file->stat_info.st_gid);

    printf("%-8s %-8s ",
           pwd ? pwd->pw_name : "unknown", // 소유자명 (또는 "unknown")
           grp ? grp->gr_name : "unknown"); // 그룹명 (또는 "unknown")

    // 파일 크기 포매팅 및 출력
    char size_str[20];
    format_size(file->stat_info.st_size, size_str, options->human_readable);
    printf("%8s ", size_str);

    // 수정 시간 포매팅 및 출력
    char *time_str = ctime(&file->stat_info.st_mtime);
    time_str[strlen(time_str) - 1] = '\0'; // 개행 문자 제거
    printf("%.12s ", time_str + 4); // "Mon DD HH:MM" 형식으로 출력
}
}
```

```
void format_permissions(mode_t mode, char *buffer) {
    // 첫 번째 문자: 파일 타입 결정
    buffer[0] = S_ISDIR(mode) ? 'd' : // 디렉토리
               S_ISLNK(mode) ? 'l' : // 심볼릭 링크
               S_ISCHR(mode) ? 'c' : // 문자 디바이스
               S_ISBLK(mode) ? 'b' : // 블록 디바이스
               S_ISFIFO(mode) ? 'p' : // 파이프
               S_ISSOCK(mode) ? 's' : '-'; // 소켓 또는 일반 파일

    // 소유자 권한 (2-4번째 문자)
    buffer[1] = (mode & S_IRUSR) ? 'r' : '-'; // 읽기
    buffer[2] = (mode & S_IWUSR) ? 'w' : '-'; // 쓰기
    buffer[3] = (mode & S_IXUSR) ? 'x' : '-'; // 실행

    // 그룹 권한 (5-7번째 문자)
    buffer[4] = (mode & S_IRGRP) ? 'r' : '-'; // 읽기
    buffer[5] = (mode & S_IWGRP) ? 'w' : '-'; // 쓰기
    buffer[6] = (mode & S_IXGRP) ? 'x' : '-'; // 실행

    // 기타 사용자 권한 (8-10번째 문자)
    buffer[7] = (mode & S_IROTH) ? 'r' : '-'; // 읽기
    buffer[8] = (mode & S_IWOTH) ? 'w' : '-'; // 쓰기
    buffer[9] = (mode & S_IXOTH) ? 'x' : '-'; // 실행

    buffer[10] = '\0'; // 문자열 종료
}
```

```
qkrehgus@LAPTOP-52P9NSOL:~/commands$ ./ls_run -l
total 240
-rw-r--r-- 1 qkrehgus qkrehgus 4488 May 29 13:31 Makefile
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:28 cat
-rwxr-xr-x 1 qkrehgus qkrehgus 23000 May 29 13:28 cat_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:29 cp
-rwxr-xr-x 1 qkrehgus qkrehgus 25088 May 29 13:29 cp_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:31 find
-rwxr-xr-x 1 qkrehgus qkrehgus 30640 May 29 13:31 find_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:26 ls
-rwxr-xr-x 1 qkrehgus qkrehgus 32544 May 29 13:26 ls_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:31 mv
-rwxr-xr-x 1 qkrehgus qkrehgus 25160 May 29 13:31 mv_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:32 rm
-rwxr-xr-x 1 qkrehgus qkrehgus 26088 May 29 13:32 rm_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:32 touch
-rwxr-xr-x 1 qkrehgus qkrehgus 30656 May 29 13:32 touch_run
```



```

void format_size(off_t size, char *buffer, int human_readable) {
    // 사람이 읽기 쉬운 형식이 요청되고 크기가 1KB 이상인 경우
    if (human_readable && size >= 1024) {
        const char *units[] = {"", "K", "M", "G", "T"}; // 단위 배열
        int unit_index = 0;
        double size_d = (double)size;

        // 적절한 단위까지 1024로 나누기 반복
        while (size_d >= 1024.0 && unit_index < 4) {
            size_d /= 1024.0;
            unit_index++;
        }

        // 크기에 따라 소수점 표시 여부 결정
        if (size_d < 10.0) {
            snprintf(buffer, 20, "%.1f%s", size_d, units[unit_index]); // 소수점 1자리
        } else {
            snprintf(buffer, 20, "%.0f%s", size_d, units[unit_index]); // 정수
        }
    } else {
        // 기본: 바이트 단위로 표시
        snprintf(buffer, 20, "%ld", (long)size);
    }
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lh
total 240
-rw-r--r-- 1 qkrehgus qkrehgus 4.4K May 29 13:31 Makefile
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:28 cat
-rwxr-xr-x 1 qkrehgus qkrehgus 22K May 29 13:28 cat_run
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:29 cp
-rwxr-xr-x 1 qkrehgus qkrehgus 24K May 29 13:29 cp_run
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:31 find
-rwxr-xr-x 1 qkrehgus qkrehgus 30K May 29 13:31 find_run
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:26 ls
-rwxr-xr-x 1 qkrehgus qkrehgus 32K May 29 13:26 ls_run
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:31 mv
-rwxr-xr-x 1 qkrehgus qkrehgus 25K May 29 13:31 mv_run
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:32 rm
-rwxr-xr-x 1 qkrehgus qkrehgus 25K May 29 13:32 rm_run
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:32 touch
-rwxr-xr-x 1 qkrehgus qkrehgus 30K May 29 13:32 touch_run

```

```
// 3단계: 시간 기준 정렬 vs 이름 기준 정렬
```

```
if (options->sort_by_time) {
```

```
    // 수정 시간 비교 (최신 파일이 앞에 오도록)
```

```
    if (file_a->stat_info.st_mtime < file_b->stat_info.st_mtime) {
```

```
        result = -1;
```

```
    } else if (file_a->stat_info.st_mtime > file_b->stat_info.st_mtime) {
```

```
        result = 1;
```

```
    } else {
```

```
        result = 0;
```

```
    }
```

```
    // 시간이 같으면 이름으로 보조 정렬
```

```
    if (result == 0) {
```

```
        result = strcmp(file_a->name, file_b->name);
```

```
    }
```

```
} else {
```

```
    // 기본: 파일명 기준 알파벳 순 정렬
```

```
    result = strcmp(file_a->name, file_b->name);
```

```
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lt
```

```
total 240
```

drwxr-xr-x	2	qkrehgus	qkrehgus	4096	May	29	13:26	ls
-rwxr-xr-x	1	qkrehgus	qkrehgus	32544	May	29	13:26	ls_run
drwxr-xr-x	2	qkrehgus	qkrehgus	4096	May	29	13:28	cat
-rwxr-xr-x	1	qkrehgus	qkrehgus	23000	May	29	13:28	cat_run
drwxr-xr-x	2	qkrehgus	qkrehgus	4096	May	29	13:29	cp
-rwxr-xr-x	1	qkrehgus	qkrehgus	25088	May	29	13:29	cp_run
-rw-r--r--	1	qkrehgus	qkrehgus	4488	May	29	13:31	Makefile
drwxr-xr-x	2	qkrehgus	qkrehgus	4096	May	29	13:31	find
-rwxr-xr-x	1	qkrehgus	qkrehgus	30640	May	29	13:31	find_run
drwxr-xr-x	2	qkrehgus	qkrehgus	4096	May	29	13:31	mv
-rwxr-xr-x	1	qkrehgus	qkrehgus	25160	May	29	13:31	mv_run
drwxr-xr-x	2	qkrehgus	qkrehgus	4096	May	29	13:32	rm
-rwxr-xr-x	1	qkrehgus	qkrehgus	26088	May	29	13:32	rm_run
drwxr-xr-x	2	qkrehgus	qkrehgus	4096	May	29	13:32	touch
-rwxr-xr-x	1	qkrehgus	qkrehgus	30656	May	29	13:32	touch_run

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -s
```

```
16 Makefile
 8 cat
48 cat_run
 8 cp
56 cp_run
 8 find
64 find_run
 8 ls
64 ls_run
 8 mv
56 mv_run
 8 rm
56 rm_run
 8 touch
64 touch_run
```

```
// long format에서는 총 블록 수를 먼저 출력
if (options->long_format) {
    long total_blocks = 0;
    for (int i = 0; i < file_count; i++) {
        total_blocks += files[i].stat_info.st_blocks;
    }
    printf("total %ld\n", total_blocks / 2); // 512바이트 블록을 1K 블록으로 변환
}
```

```
// 2단계: 확장자별 그룹화 (옵선이 설정된 경우)
if (options->group_by_ext) {
    char *ext_a = get_file_extension(file_a->name);
    char *ext_b = get_file_extension(file_b->name);

    // 둘 다 확장자가 있는 경우: 확장자 비교
    if (ext_a && ext_b) {
        result = strcmp(ext_a, ext_b);
        if (result != 0) {
            return options->reverse_sort ? -result : result;
        }
    }

    // 한쪽만 확장자가 있는 경우: 확장자 있는 것을 뒤로
    else if (ext_a && !ext_b) {
        return options->reverse_sort ? -1 : 1;
    } else if (!ext_a && ext_b) {
        return options->reverse_sort ? 1 : -1;
    }

    // 둘 다 확장자가 없으면 다음 단계로
}
}
```

```
OP-52P9N5OL:~/commands$ ./ls_run -ltr
```

```
1 qkrehgus qkrehgus 30656 May 29 13:32 touch_run
2 qkrehgus qkrehgus 4096 May 29 13:32 touch
1 qkrehgus qkrehgus 26088 May 29 13:32 rm_run
2 qkrehgus qkrehgus 4096 May 29 13:32 rm
1 qkrehgus qkrehgus 25160 May 29 13:31 mv_run
2 qkrehgus qkrehgus 4096 May 29 13:31 mv
1 qkrehgus qkrehgus 30640 May 29 13:31 find_run
2 qkrehgus qkrehgus 4096 May 29 13:31 find
1 qkrehgus qkrehgus 4488 May 29 13:31 Makefile
1 qkrehgus qkrehgus 25088 May 29 13:29 cp_run
2 qkrehgus qkrehgus 4096 May 29 13:29 cp
1 qkrehgus qkrehgus 23000 May 29 13:28 cat_run
2 qkrehgus qkrehgus 4096 May 29 13:28 cat
1 qkrehgus qkrehgus 32544 May 29 13:26 ls_run
2 qkrehgus qkrehgus 4096 May 29 13:26 ls
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -R
```

```
Makefile
```

```
cat
```

```
cat_run
```

```
cp
```

```
cp_run
```

```
find
```

```
find_run
```

```
ls
```

```
ls_run
```

```
mv
```

```
mv_run
```

```
rm
```

```
rm_run
```

```
touch
```

```
touch_run
```

```
./cat:
```

```
cat.c
```

```
cat.o
```

```
cat_options.c
```

```
cat_options.h
```

```
cat_options.o
```

```
// === 5단계: 재귀 처리 ===
```

```
// 재귀 옵션이 설정된 경우, 하위 디렉토리들을 재귀적으로 처리
```

```
if (options->recursive) {
```

```
    for (int i = 0; i < file_count; i++) {
```

```
        // 디렉토리이면서 현재/상위 디렉토리가 아닌 경우만 재귀 호출
```

```
        if (S_ISDIR(files[i].stat_info.st_mode) &&
```

```
            strcmp(files[i].name, ".") != 0 &&
```

```
            strcmp(files[i].name, "..") != 0) {
```

```
                list_directory(files[i].path, options, 1); // is_recursive=1로 호출
```

```
        }
```

```
    }
```

```
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -i
```

```
42229 Makefile
41123 cat // inode 번호 출력 (옵션이 설정된 경우)
42238 cat_run if (options->show_inode) {
42189 cp printf("%8lu ", (unsigned long)file->stat_info.st_ino);
42241 cp_run }
42225 find
42244 find_run
12890 ls
42235 ls_run
42215 mv
42247 mv_run
42220 rm
42250 rm_run
41129 touch
42253 touch_run
```

```
// 2단계: 확장자별 그룹화 (옵션이 설정된 경우)
```

```
if (options->group_by_ext) {  
    char *ext_a = get_file_extension(file_a->name);  
    char *ext_b = get_file_extension(file_b->name);  
  
    // 둘 다 확장자가 있는 경우: 확장자 비교  
    if (ext_a && ext_b) {  
        result = strcmp(ext_a, ext_b);  
        if (result != 0) {  
            return options->reverse_sort ? -result : result;  
        }  
    }  
  
    // 한쪽만 확장자가 있는 경우: 확장자 있는 것을 뒤로  
    else if (ext_a && !ext_b) {  
        return options->reverse_sort ? -1 : 1;  
    } else if (!ext_a && ext_b) {  
        return options->reverse_sort ? 1 : -1;  
    }  
  
    // 둘 다 확장자가 없으면 다음 단계로  
}
```

```
qkrehgus@LAPTOP-52P9N5OL:~/commands$ ./ls_run -le ./ls  
total 68  
-rw-r--r-- 1 qkrehgus qkrehgus 5878 May 29 12:41 ls.c  
-rw-r--r-- 1 qkrehgus qkrehgus 11586 May 29 12:41 ls_options.c  
-rw-r--r-- 1 qkrehgus qkrehgus 5004 May 29 12:41 ls_options.h  
-rw-r--r-- 1 qkrehgus qkrehgus 16248 May 29 13:26 ls.o  
-rw-r--r-- 1 qkrehgus qkrehgus 20672 May 29 13:26 ls_options.o
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -d
```

```
cat
```

```
cp
```

```
find
```

```
ls
```

```
mv
```

```
rm
```

```
touch
```

```
Makefile
```

```
cat_run
```

```
cp_run
```

```
find_run
```

```
ls_run
```

```
mv_run
```

```
rm_run
```

```
touch_run
```

```
// 1단계: 디렉토리 우선 정렬 (옵션이 설정된 경우)
```

```
if (options->dirs_first) {
```

```
    // 각 파일이 디렉토리인지 확인
```

```
    int a_is_dir = S_ISDIR(file_a->stat_info.st_mode);
```

```
    int b_is_dir = S_ISDIR(file_b->stat_info.st_mode);
```

```
    // 한쪽은 디렉토리, 다른 쪽은 일반 파일인 경우
```

```
    if (a_is_dir && !b_is_dir) return -1; // a가 앞에 옴
```

```
    if (!a_is_dir && b_is_dir) return 1;  // b가 앞에 옴
```

```
}
```



```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -f txt  
a.txt  
b.txt
```

```
// 확장자 필터 체크: 특정 확장자만 표시하는 옵션 처리  
if (options->filter_ext) {  
    char *ext = get_file_extension(filename);  
    // 확장자가 없거나 지정된 확장자와 다르면 숨김  
    if (!ext || strcmp(ext, options->filter_ext) != 0) {  
        return 0;  
    }  
}
```

cat

- | | | |
|-------------------------|-------------------|-----------------------------------|
| • -b | : 공백이 아닌 줄에만 줄 번호 | <code>./cat -b file.txt</code> |
| • -n | : 모든 줄에 줄 번호 | <code>./cat -n file.txt</code> |
| • -s | : 빈 줄 압축 | <code>./cat -s file.txt</code> |
| • -h <small>[N]</small> | : 처음 N만 출력 | <code>./cat -h 10 file.txt</code> |

옵션

예시

```

// 파일을 한 줄씩 읽어서 처리
while ((read = getline(&line, &len, fp)) != -1) {
    // -h 옵션 처리: 지정된 줄 수만큼 출력했으면 중단
    if (opts->head_lines > 0 && lines_printed >= opts->head_lines) {
        break;
    }

    // 현재 줄이 빈 줄인지 확인 (공백, 탭, 개행만 있는 줄)
    int is_blank = is_blank_line(line);

    // -s 옵션 처리: 연속된 빈 줄 중 두 번째부터는 건너뛰기
    if (opts->squeeze_blank && is_blank && prev_blank) {
        line_num++;           // 줄 번호는 증가시키되
        continue;           // 출력하지 않고 다음 줄로
    }

    // 줄 번호 출력 처리
    if (opts->number_nonblank && !is_blank) {
        // -b 옵션: 빈 줄이 아닌 경우에만 줄 번호 출력
        printf("%6d\t", output_line_num++);
    } else if (opts->number_all) {
        // -n 옵션: 모든 줄에 줄 번호 출력
        printf("%6d\t", line_num);
    }

    // 실제 줄 내용 출력 (개행 문자 포함)
    printf("%s", line);

    // 다음 반복을 위한 상태 업데이트
    prev_blank = is_blank; // 현재 줄의 빈 줄 여부를 저장
    line_num++;           // 파일의 실제 줄 번호 증가
    lines_printed++;      // 출력한 줄 수 증가
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./cat_run c.txt
Hello!!

```

```

This test file!!!

```

```

안녕하세요!
이건 테스트입니다.

```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./cat_run -b c.txt
```

```
1 Hello!!
```

```
2 This test file!!!
```

```
3 안녕하세요 !
```

```
4 이건 테스트입니다 .
```

```
// 줄 번호 출력 처리
```

```
if (opts->number_nonblank && !is_blank) {
```

```
    // -b 옵션: 빈 줄이 아닌 경우에만 줄 번호 출력
```

```
    printf("%6d\t", output_line_num++);
```

```
} else if (opts->number_all) {
```

```
    // -n 옵션: 모든 줄에 줄 번호 출력
```

```
    printf("%6d\t", line_num);
```

```
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./cat_run -n c.txt
```

```
1 Hello!!
```

```
2
```

```
3 This test file!!!
```

```
4
```

```
5
```

```
6
```

```
7 안녕하세요 !
```

```
8 이건 테스트입니다 .
```

```
// 줄 번호 출력 처리
```

```
if (opts->number_nonblank && !is_blank) {
```

```
    // -b 옵션: 빈 줄이 아닌 경우에만 줄 번호 출력
```

```
    printf("%6d\t", output_line_num++);
```

```
} else if (opts->number_all) {
```

```
    // -n 옵션: 모든 줄에 줄 번호 출력
```

```
    printf("%6d\t", line_num);
```

```
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./cat_run -s c.txt  
Hello!!
```

This test file!!!

안녕하세요!
이건 테스트입니다.

```
int is_blank_line(const char *line) {  
    // 문자열의 각 문자를 검사  
    while (*line) {  
        // 공백, 탭, 개행, 캐리지 리턴이 아닌 문자가 있으면 빈 줄이 아님  
        if (*line != ' ' && *line != '\t' && *line != '\n' && *line != '\r') {  
            return 0;  
        }  
        line++;  
    }  
    return 1; // 모든 문자가 공백문자이거나 빈 문자열  
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./cat_run -h 3 c.txt  
Hello!!
```

```
This test file!!!
```

```
// -h 옵션 처리: 지정된 줄 수만큼 출력했으면 중단  
if (opts->head_lines > 0 && lines_printed >= opts->head_lines) {  
    break;  
}
```

touch

- | | | |
|--|-----------------|---|
| • -a | : 접근 시간 변경 | <code>./touch -a file.txt</code> |
| • -m | : 수정 시간 변경 | <code>./touch -m file.txt</code> |
| • -c | : 파일이 없으면 생성 안함 | <code>./touch -c example.txt</code> |
| • -t <small>[[CC]YY]MMDDhhmm[.ss]</small> | : 시간 설정 | <code>./touch -t 202505291545
file.txt</code> |
| • -p | : 중간 디렉토리 자동 생성 | <code>./touch -p ex/file.txt</code> |

옵션

예시


```

int process_file(const char *filename, const touch_options *opts) {
    struct stat file_stat;
    // 파일 존재 여부 확인
    int file_exists = (stat(filename, &file_stat) == 0);

    // 파일이 존재하지 않는 경우의 처리
    if (!file_exists) {
        if (opts->no_create) {
            // -c 옵션: 파일을 생성하지 않고 종료
            return 0;
        }

        // -p 옵션: 필요한 중간 디렉토리들을 생성
        if (opts->create_path) {
            char *filename_copy = strdup(filename);
            if (!filename_copy) {
                perror("메모리 할당 실패");
                return -1;
            }

            // 파일의 디렉토리 부분만 추출하여 디렉토리 생성
            char *dir_path = dirname(filename_copy);

            // 현재 디렉토리가 아닌 경우에만 디렉토리 생성
            if (strcmp(dir_path, ".") != 0) {
                // 전체 디렉토리 경로를 다시 구성해서 재귀적으로 생성
                char *full_dir_copy = strdup(filename);
                if (!full_dir_copy) {
                    perror("메모리 할당 실패");
                    free(filename_copy);
                    return -1;
                }

                char *full_dir_path = dirname(full_dir_copy);
                if (create_directories(full_dir_path) != 0) {
                    free(filename_copy);
                    free(full_dir_copy);
                    return -1;
                }
                free(full_dir_copy);
            }
            free(filename_copy);
        }

        // 새 파일 생성
        if (create_file(filename) != 0) {
            perror(filename);
            return -1;
        }

        // 파일의 접근/수정 시간 설정
        if (set_file_times(filename, opts) != 0) {
            perror(filename);
            return -1;
        }

        return 0;
    }
}

```

```

qkrehgus@LAPTOP-52P9N50L: ~/commands$ ./cs_run -lf txt
total 12
-rw-r--r--  1 qkrehgus qkrehgus   15 May 29 16:53 a.txt
-rw-r--r--  1 qkrehgus qkrehgus   13 May 29 16:54 b.txt
-rw-r--r--  1 qkrehgus qkrehgus   74 May 29 17:00 c.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./touch_run a.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 12
-rw-r--r--  1 qkrehgus qkrehgus   15 May 29 17:05 a.txt
-rw-r--r--  1 qkrehgus qkrehgus   13 May 29 16:54 b.txt
-rw-r--r--  1 qkrehgus qkrehgus   74 May 29 17:00 c.txt

```

```

int set_file_times(const char *filename, const touch_options *opts) {
    struct stat file_stat;

    // 파일의 현재 정보를 가져온다 (기존 시간 정보 필요)
    if (stat(filename, &file_stat) != 0) {
        return -1;
    }

    struct utimbuf times;

    if (opts->use_custom_time) {
        // 사용자가 지정한 시간 사용
        if (opts->access_time && opts->modify_time) {
            // 둘 다 변경
            times.actime = opts->custom_time.tv_sec;
            times.modtime = opts->custom_time.tv_sec;
        } else if (opts->access_time) {
            // 접근 시간만 변경, 수정 시간은 기존 값 유지
            times.actime = opts->custom_time.tv_sec;
            times.modtime = file_stat.st_mtime;
        } else if (opts->modify_time) {
            // 수정 시간만 변경, 접근 시간은 기존 값 유지
            times.actime = file_stat.st_atime;
            times.modtime = opts->custom_time.tv_sec;
        }
    } else {
        // 현재 시간 사용
        time_t now = time(NULL);
        if (opts->access_time && opts->modify_time) {
            // 둘 다 현재 시간으로 설정
            times.actime = now;
            times.modtime = now;
        } else if (opts->access_time) {
            // 접근 시간만 현재 시간으로 설정
            times.actime = now;
            times.modtime = file_stat.st_mtime;
        } else if (opts->modify_time) {
            // 수정 시간만 현재 시간으로 설정
            times.actime = file_stat.st_atime;
            times.modtime = now;
        }
    }

    // 파일 시간 변경 적용
    return utime(filename, &times);
}

```

./touch_run -a b.txt

Access: 2025-05-29 16:53:57.391443994 +0900
 Modify: 2025-05-29 16:54:13.439444578 +0900
 Change: 2025-05-29 16:54:13.439444578 +0900
 Birth: 2025-05-29 16:53:57.391443994 +0900



Access: 2025-05-29 17:08:31.000000000 +0900
 Modify: 2025-05-29 16:54:13.000000000 +0900
 Change: 2025-05-29 17:08:31.711450153 +0900
 Birth: 2025-05-29 16:53:57.391443994 +0900

```

int set_file_times(const char *filename, const touch_options *opts) {
    struct stat file_stat;

    // 파일의 현재 정보를 가져온다 (기본 시간 정보 필요)
    if (stat(filename, &file_stat) != 0) {
        return -1;
    }

    struct utimbuf times;

    if (opts->use_custom_time) {
        // 사용자가 지정한 시간 사용
        if (opts->access_time && opts->modify_time) {
            // 둘 다 변경
            times.actime = opts->custom_time.tv_sec;
            times.modtime = opts->custom_time.tv_sec;
        } else if (opts->access_time) {
            // 접근 시간만 변경, 수정 시간은 기존 값 유지
            times.actime = opts->custom_time.tv_sec;
            times.modtime = file_stat.st_mtime;
        } else if (opts->modify_time) {
            // 수정 시간만 변경, 접근 시간은 기존 값 유지
            times.actime = file_stat.st_atime;
            times.modtime = opts->custom_time.tv_sec;
        }
    } else {
        // 현재 시간 사용
        time_t now = time(NULL);
        if (opts->access_time && opts->modify_time) {
            // 둘 다 현재 시간으로 설정
            times.actime = now;
            times.modtime = now;
        } else if (opts->access_time) {
            // 접근 시간만 현재 시간으로 설정
            times.actime = now;
            times.modtime = file_stat.st_mtime;
        } else if (opts->modify_time) {
            // 수정 시간만 현재 시간으로 설정
            times.actime = file_stat.st_atime;
            times.modtime = now;
        }
    }

    // 파일 시간 변경 적용
    return utime(filename, &times);
}

```

```

Access: 2025-05-29 17:08:31.000000000 +0900
Modify: 2025-05-29 16:54:13.000000000 +0900
Change: 2025-05-29 17:08:31.711450153 +0900
Birth: 2025-05-29 16:53:57.391443994 +0900

```



```

Access: 2025-05-29 17:08:31.000000000 +0900
Modify: 2025-05-29 17:09:43.000000000 +0900
Change: 2025-05-29 17:09:43.295475567 +0900
Birth: 2025-05-29 16:53:57.391443994 +0900

```

```
// 파일 존재 여부 확인
int file_exists = (stat(filename, &file_stat) == 0);

// 파일이 존재하지 않는 경우의 처리
if (!file_exists) {
    if (opts->no_create) {
        // -c 옵션: 파일을 생성하지 않고 종료
        return 0;
    }
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./touch_run -c d.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 12
-rw-r--r--  1 qkrehgus qkrehgus      15 May 29 17:05 a.txt
-rw-r--r--  1 qkrehgus qkrehgus      13 May 29 17:09 b.txt
-rw-r--r--  1 qkrehgus qkrehgus     74 May 29 17:00 c.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./touch_run -c c.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 12
-rw-r--r--  1 qkrehgus qkrehgus      15 May 29 17:05 a.txt
-rw-r--r--  1 qkrehgus qkrehgus      13 May 29 17:09 b.txt
-rw-r--r--  1 qkrehgus qkrehgus     74 May 29 17:10 c.txt
```

```
// 옵션인지 확인 ('-'로 시작하고 빈 문자열이 아님)
if (argv[i][0] == '-' && argv[i][1] != '\0') {
    // -t 옵션 특별 처리 (시간 지정 옵션)
    if (strncmp(argv[i], "-t", 2) == 0) {
        char *time_str = NULL;
        if (argv[i][2] != '\0') {
            // "-t202312251430" 형태 (붙여서 작성)
            time_str = &argv[i][2];
        } else if (i + 1 < argc) {
            // "-t 202312251430" 형태 (공백으로 분리)
            time_str = argv[++i];
        } else {
            fprintf(stderr, "옵션 -t에는 시간 인수가 필요합니다\n");
            return -1;
        }

        // 시간 문자열을 파싱하여 timespec 구조체로 변환
        if (parse_time_string(time_str, &opts->custom_time) != 0) {
            fprintf(stderr, "잘못된 시간 형식: %s\n", time_str);
            return -1;
        }
        opts->use_custom_time = 1;
        continue;
    }
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./touch_run -t 202202222222 a.
txt
```

```
Access: 2022-02-22 22:22:00.000000000 +0900
Modify: 2022-02-22 22:22:00.000000000 +0900
Change: 2025-05-29 17:12:28.563425091 +0900
Birth: 2025-05-29 16:53:26.323440091 +0900
```

```
if (opts->use_custom_time) {
    // 사용자가 지정한 시간 사용
    if (opts->access_time && opts->modify_time) {
        // 둘 다 변경
        times.actime = opts->custom_time.tv_sec;
        times.modtime = opts->custom_time.tv_sec;
    } else if (opts->access_time) {
        // 접근 시간만 변경, 수정 시간은 기존 값 유지
        times.actime = opts->custom_time.tv_sec;
        times.modtime = file_stat.st_mtime;
    } else if (opts->modify_time) {
        // 수정 시간만 변경, 접근 시간은 기존 값 유지
        times.actime = file_stat.st_atime;
        times.modtime = opts->custom_time.tv_sec;
    }
}
```

```

// 파일이 존재하지 않는 경우의 처리
if (!file_exists) {
    if (opts->no_create) {
        // -c 옵션: 파일을 생성하지 않고 종료
        return 0;
    }

    // -p 옵션: 필요한 중간 디렉토리들을 생성
    if (opts->create_path) {
        char *filename_copy = strdup(filename);
        if (!filename_copy) {
            perror("메모리 할당 실패");
            return -1;
        }

        // 파일의 디렉토리 부분만 추출하여 디렉토리 생성
        char *dir_path = dirname(filename_copy);

        // 현재 디렉토리가 아닌 경우에만 디렉토리 생성
        if (strcmp(dir_path, ".") != 0) {
            // 전체 디렉토리 경로를 다시 구성해서 재귀적으로 생성
            char *full_dir_copy = strdup(filename);
            if (!full_dir_copy) {
                perror("메모리 할당 실패");
                free(filename_copy);
                return -1;
            }

            char *full_dir_path = dirname(full_dir_copy);
            if (create_directories(full_dir_path) != 0) {
                free(filename_copy);
                free(full_dir_copy);
                return -1;
            }
            free(full_dir_copy);
        }
        free(filename_copy);
    }

    // 새 파일 생성
    if (create_file(filename) != 0) {
        perror(filename);
        return -1;
    }
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./touch_run -p ex/file.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ls
Makefile  c.txt      cp          find        ls_run      rm          touch_run
a.txt     cat        cp_run      find_run    mv          rm_run
b.txt     cat_run    ex          ls          mv_run      touch
qkrehgus@LAPTOP-52P9N50L:~/commands$ cd ex
qkrehgus@LAPTOP-52P9N50L:~/commands/ex$ ls
file.txt

```

cp

- | | | |
|------|------------------|----------------------------------|
| • -i | : 덮어쓰기 전 확인 | <code>./cp -i a.txt b.txt</code> |
| • -f | : 강제 복사 | <code>./cp -f a.txt b.txt</code> |
| • -u | : 새 파일만 복사 | <code>./cp -u a.txt b.txt</code> |
| • -p | : 권한, 소유자, 시간 유지 | <code>./cp -p a.txt b.txt</code> |

옵션

예시

```

int copy_file_content(const char *src_path, const char *dst_path) {
    int src_fd, dst_fd;           // 소스, 대상 파일 디스크립터
    char buffer[BUFFER_SIZE];    // 데이터 복사용 버퍼
    ssize_t bytes_read, bytes_written; // 읽은/쓴 바이트 수
    int result = 0;              // 함수 반환값

    // 소스 파일을 읽기 전용으로 열기
    src_fd = open(src_path, O_RDONLY);
    if (src_fd == -1) {
        fprintf(stderr, "cp: cannot open '%s': %s\n", src_path, strerror(errno));
        return -1;
    }

    // 대상 파일을 쓰기용으로 생성/열기
    // O_CREAT: 파일이 없으면 생성, O_TRUNC: 기존 내용 삭제
    // 0644: 소유자는 읽기/쓰기, 그룹/기타는 읽기만 가능
    dst_fd = open(dst_path, O_WRONLY | O_CREAT | O_TRUNC, 0644);
    if (dst_fd == -1) {
        fprintf(stderr, "cp: cannot create '%s': %s\n", dst_path, strerror(errno));
        close(src_fd);
        return -1;
    }

    // 파일 내용을 버퍼 단위로 복사
    while ((bytes_read = read(src_fd, buffer, BUFFER_SIZE)) > 0) {
        bytes_written = write(dst_fd, buffer, bytes_read);
        if (bytes_written != bytes_read) {
            // 쓰기 실패 또는 부분 쓰기 발생
            fprintf(stderr, "cp: error writing to '%s': %s\n", dst_path, strerror(errno));
            result = -1;
            break;
        }
    }

    // 읽기 에러 확인
    if (bytes_read == -1) {
        fprintf(stderr, "cp: error reading from '%s': %s\n", src_path, strerror(errno));
        result = -1;
    }

    // 파일 디스크립터 닫기
    close(src_fd);
    close(dst_fd);

    return result;
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 4
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 17:58 a.txt
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 17:55 b.txt
-rw-r--r--  1 qkrehgus qkrehgus    74 May 29 17:10 b_1.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./cp_run a.txt c.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 4
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 17:58 a.txt
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 17:55 b.txt
-rw-r--r--  1 qkrehgus qkrehgus    74 May 29 17:10 b_1.txt
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 17:58 c.txt

```



```
qkrehgus@LAPTOP-52P9N50L:~/commands$ cat b_1.txt
Hello!!
```

This test file!!!

안녕하세요!
이건 테스트입니다.

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./cp_run -i b_1.txt b.txt
cp: overwrite 'b.txt'? y
qkrehgus@LAPTOP-52P9N50L:~/commands$ cat b.txt
Hello!!
```

This test file!!!

안녕하세요!
이건 테스트입니다.

```
// -i 옵션 처리: 덮어쓰기 전 사용자 확인 (-f 옵션이 없는 경우)
if (opts->interactive && !opts->force && dst_exists) {
    if (!ask_user_confirmation(dst_path)) {
        return 0; // 사용자가 거부했으므로 복사 중단
    }
}
```

```
int ask_user_confirmation(const char *dst_path) {
    printf("cp: overwrite '%s'? ", dst_path);
    fflush(stdout); // 즉시 출력되도록 버퍼 플러시

    char response[10]; // 사용자 응답을 저장할 버퍼
    if (fgets(response, sizeof(response), stdin) == NULL) {
        return 0; // EOF나 읽기 에러 시 거부로 처리
    }

    // 첫 번째 문자가 'y' 또는 'Y'인지 확인
    return (response[0] == 'y' || response[0] == 'Y');
}
```

```
// -f 옵션 처리: 대상 파일이 쓰기 금지되어 있어도 강제 삭제
if (opts->force && dst_exists) {
    if (unlink(dst_path) != 0 && errno != ENOENT) {
        fprintf(stderr, "cp: cannot remove '%s': %s\n", dst_path, strerror(errno));
        return -1;
    }
}
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 8
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 17:58 a.txt
-----  1 qkrehgus qkrehgus    74 May 29 18:00 b.txt
-rw-r--r--  1 qkrehgus qkrehgus    74 May 29 17:10 b_1.txt
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 17:58 c.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./cp_run -f b_1.txt b.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 8
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 17:58 a.txt
-rw-r--r--  1 qkrehgus qkrehgus    74 May 29 18:01 b.txt
-rw-r--r--  1 qkrehgus qkrehgus    74 May 29 17:10 b_1.txt
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 17:58 c.txt
```

```
// 대상 파일의 존재 여부 확인
int dst_exists = (stat(dst_path, &dst_stat) == 0);

// -u 옵션 처리: 소스가 더 새로운 경우만 복사
if (opts->update && dst_exists) {
    int newer = is_source_newer(src_path, dst_path);
    if (newer == -1) {
        return -1; // 시간 비교 실패
    }
    if (newer == 0) {
        return 0; // 소스가 더 새롭지 않으므로 복사 생략
    }
}
```

```
int is_source_newer(const char *src_path, const char *dst_path) {
    struct stat src_stat, dst_stat;

    // 소스 파일의 상태 정보 가져오기
    if (stat(src_path, &src_stat) != 0) {
        return -1; // 소스 파일 stat 실패
    }

    // 대상 파일의 상태 정보 가져오기
    if (stat(dst_path, &dst_stat) != 0) {
        return 1; // 대상 파일이 없으면 무조건 복사해야 함
    }

    // 수정 시간 비교 (초 단위)
    return (src_stat.st_mtime > dst_stat.st_mtime);
}
```

```
./ls_run -lf txt
```

```
0 May 29 17:58 a.txt
74 May 29 18:01 b.txt
74 May 29 17:10 b_1.txt
0 May 29 17:58 c.txt
./cp_run -u b.txt a.txt
./ls_run -lf txt
```

```
74 May 29 18:03 a.txt
74 May 29 18:01 b.txt
74 May 29 17:10 b_1.txt
0 May 29 17:58 c.txt
```

```
./ls_run -lf txt
```

```
74 May 29 18:03 a.txt
74 May 29 18:01 b.txt
74 May 29 17:10 b_1.txt
0 May 29 17:58 c.txt
./cp_run -u b.txt a.txt
./ls_run -lf txt
```

```
74 May 29 18:03 a.txt
74 May 29 18:01 b.txt
74 May 29 17:10 b_1.txt
0 May 29 17:58 c.txt
```

```
// 실제 파일 내용 복사 실행
if (copy_file_content(src_path, dst_path) != 0) {
    return -1;
}

// -p 옵션 처리: 파일 속성 보존
if (opts->preserve) {
    if (preserve_attributes(src_path, dst_path) != 0) {
        // 속성 보존 실패는 경고만 출력하고 성공으로 처리
        fprintf(stderr, "cp: warning: failed to preserve some attributes for '%s'\n", dst_path);
    }
}
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 12
-rwxr-xr-x  1 qkrehgus qkrehgus    74 May 29 18:03 a.txt
-rw-r--r--  1 qkrehgus qkrehgus    74 May 29 18:01 b.txt
-rw-r--r--  1 qkrehgus qkrehgus    74 May 29 17:10 b_1.txt
-rw-r--r--  1 qkrehgus qkrehgus     0 May 29 17:58 c.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./cp_run -p a.txt b.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 12
-rwxr-xr-x  1 qkrehgus qkrehgus    74 May 29 18:03 a.txt
-rwxr-xr-x  1 qkrehgus qkrehgus    74 May 29 18:03 b.txt
-rw-r--r--  1 qkrehgus qkrehgus    74 May 29 17:10 b_1.txt
-rw-r--r--  1 qkrehgus qkrehgus     0 May 29 17:58 c.txt
```

```
int preserve_attributes(const char *src_path, const char *dst_path) {
    struct stat src_stat;

    // 소스 파일의 속성 정보 가져오기
    if (stat(src_path, &src_stat) != 0) {
        perror("stat");
        return -1;
    }

    // 파일 권한 설정 (rwxrwxrwx 형태의 모드)
    if (chmod(dst_path, src_stat.st_mode) != 0) {
        perror("chmod");
        return -1;
    }

    // 소유자 및 그룹 설정 (root 권한이 필요할 수 있음)
    if (chown(dst_path, src_stat.st_uid, src_stat.st_gid) != 0) {
        // 소유자 변경 실패는 권한 부족인 경우가 많으므로 경고만 출력
        if (errno != EPERM) {
            perror("chown");
        }
    }

    // 파일 시간 설정 (접근 시간, 수정 시간)
    struct utimbuf times;
    times.actime = src_stat.st_atime; // 마지막 접근 시간
    times.modtime = src_stat.st_mtime; // 마지막 수정 시간

    if (utime(dst_path, &times) != 0) {
        perror("utime");
        return -1;
    }

    return 0;
}
```

mv

- | | | |
|------|------------------|------------------------------------|
| • -i | : 덮어쓰기 전 확인 | <code>./mv -i a.txt b.txt</code> |
| • -s | : 중복 시 이름에 숫자 추가 | <code>./mv -s a.txt b.txt</code> |
| • -n | : 덮어쓰기 방지 | <code>./mv -n a.txt b.txt</code> |
| • -v | : 작업 내용 출력 | <code>./mv -v a.txt b.txt</code> |
| • -f | : 강제 이동 | <code>./mv -f a.txt folder/</code> |

옵션

예시

```

int perform_move(const char *src, const char *dest, mv_options_t *opts) {
    char *final_dest = NULL; // 최종 대상 경로
    char *unique_dest = NULL; // -s 옵션으로 생성된 고유 경로
    bool allocated_dest = false; // 메모리 할당 여부 추적 플래그
    int result = 0; // 함수 반환값

    // 대상이 디렉토리인 경우, 원본 파일명을 유지하여 디렉토리 내부로 이동
    // 예: mv file.txt /home/user/ -> /home/user/file.txt
    if (is_directory(dest)) {
        char *src_basename = basename((char*)src); // 원본 파일의 기본 이름 추출
        // 디렉토리 경로 + '/' + 파일명 + null terminator를 위한 메모리 할당
        final_dest = malloc(strlen(dest) + strlen(src_basename) + 2);
        snprintf(final_dest, strlen(dest) + strlen(src_basename) + 2, "%s/%s", dest, src_basename);
        allocated_dest = true; // 메모리를 할당했음을 표시
    } else {
        // 대상이 파일인 경우 그대로 사용
        final_dest = (char*)dest;
    }

    // -s 옵션 처리: 종속 파일이 있을 때 고유한 이름 생성 (예: file_1.txt)
    if (opts->suffix) {
        unique_dest = generate_unique_name(final_dest);
        if (allocated_dest) {
            free(final_dest); // 이전에 할당한 메모리 해제
        }
        final_dest = unique_dest;
        allocated_dest = true;
    } else {
        // -s 옵션이 없는 경우 덮어쓰기 여부 확인
        if (isshould_overwrite(final_dest, opts)) {
            if (opts->verbose) {
                printf("'Xs' 이동이 취소되었습니다.\n", src);
            }
            // 메모리 정리 후 함수 종료
            if (allocated_dest) {
                free(final_dest);
            }
            return 0;
        }
    }

    // rename() 시스템 콜을 사용하여 실제 파일 이동 수행
    // 같은 파일시스템 내에서는 빠른 이동, 다른 파일시스템 간에는 복사 후 삭제
    if (rename(src, final_dest) != 0) {
        fprintf(stderr, "mv: '%s'에서 '%s'로 이동할 수 없습니다: %s\n",
            src, final_dest, strerror(errno));
        result = -1;
    } else {
        // -v 옵션: 이동 성공 시 상세 정보 출력
        if (opts->verbose) {
            printf("'Xs' -> '%s'\n", src, final_dest);
        }
    }

    // 할당한 메모리가 있다면 해제하여 메모리 누수 방지
    if (allocated_dest) {
        free(final_dest);
    }

    return result;
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./mv_run a.txt b.txt c.txt ex
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run ex
a.txt
b.txt
c.txt
file.txt

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./mv_run a.txt newa.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ls
Makefile  cat      cp_run   find_run  mv      rm      touch_run
b.txt     cat_run  ex       ls        mv_run   rm_run
c.txt     cp       find     ls_run    newa.txt touch

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./mv_run ex sample
qkrehgus@LAPTOP-52P9N50L:~/commands$ ls
Makefile  cat      cp_run   ls        mv_run   rm_run   touch_run
b.txt     cat_run  find     ls_run    newa.txt sample
c.txt     cp       find_run mv         rm        touch

```

```

// -s 옵션이 없는 경우 덮어쓰기 여부 확인
if (!should_overwrite(final_dest, opts)) {
    if (opts->verbose) {
        printf("' %s' 이동이 취소되었습니다.\n", src);
    }
    // 메모리 정리 후 함수 종료
    if (allocated_dest) {
        free(final_dest);
    }
    return 0;
}

// -i 옵션: interactive, 사용자에게 덮어쓰기 여부를 직접 확인
if (opts->interactive) {
    char response; // 사용자 입력을 저장할 변수
    printf("' %s'를 덮어쓰시겠습니까? (y/n): ", dest);
    scanf(" %c", &response); // 앞의 공백 문자는 이전 입력의 개행문자 무시
    // 'y' 또는 'Y' 입력시에만 덮어쓰기 허용
    return (response == 'y' || response == 'Y');
}

```

```

qkrengus@LAPTOP-52P9N5OL:~/commands$ ./mv -i b.txt d.txt
-bash: ./mv: Is a directory
qkrengus@LAPTOP-52P9N5OL:~/commands$ ./mv_run -i b.txt d.txt
'd.txt'를 덮어쓰시겠습니까? (y/n): y

```

```
// -s 옵션 처리: 중복 파일이 있을 때 고유한 이름 생성 (예: file_1.txt)
if (opts->suffix) {
    unique_dest = generate_unique_name(final_dest);
    if (allocated_dest) {
        free(final_dest); // 이전에 할당한 메모리 해제
    }
    final_dest = unique_dest;
    allocated_dest = true;
} else {
    // -s 옵션이 없는 경우 덮어쓰기 여부 확인
    if (!should_overwrite(final_dest, opts)) {
        if (opts->verbose) {
            printf("%s' 이동이 취소되었습니다.\n", src);
        }
        // 메모리 정리 후 함수 종료
        if (allocated_dest) {
            free(final_dest);
        }
        return 0;
    }
}
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 4
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 17:55 b.txt
-rw-r--r--  1 qkrehgus qkrehgus    74 May 29 17:10 newa.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./mv_run -s newa.txt b.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 4
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 17:55 b.txt
-rw-r--r--  1 qkrehgus qkrehgus    74 May 29 17:10 b_1.txt
```

```
char* generate_unique_name(const char *dest) {
    struct stat st; // 파일 존재 확인용 구조체
    char *new_name; // 생성된 새 파일명을 저장할 포인터
    char *base_name, *extension; // 기본 파일명과 확장자
    char *dot_pos; // 확장자 구분점(마지막 '.')의 위치
    int counter = 1; // 고유 번호 생성용 카운터

    // 원본 파일명이 존재하지 않으면 중복이 아니므로 그대로 사용 가능
    if (stat(dest, &st) != 0) {
        new_name = malloc(strlen(dest) + 1); // null terminator 포함
        strcpy(new_name, dest);
        return new_name;
    }

    // 파일명과 확장자를 분리하기 위해 원본 문자열 복사
    base_name = malloc(strlen(dest) + 1);
    strcpy(base_name, dest);

    // strrchr()로 마지막 '.' 위치를 찾아 확장자 분리
    // 예: "file.txt" -> base_name="file", extension=".txt"
    dot_pos = strrchr(base_name, '.');
    if (dot_pos != NULL) {
        *dot_pos = '\0'; // '.' 위치에 null 문자 삽입하여 기본명 분리
        extension = dot_pos + 1; // '.' 다음부터가 확장자
    } else {
        // 확장자가 없는 파일 (예: "README")
        extension = "";
    }

    // 고유한 이름을 찾기 위한 충분한 메모리 할당
    // 원본 길이 + 숫자 부분 + 구분자들을 위한 여유 공간
    new_name = malloc(strlen(dest) + 20);

    // 중복되지 않는 파일명을 찾을 때까지 반복
    do {
        if (strlen(extension) > 0) {
            // 확장자가 있는 경우: base_name_counter.extension
            snprintf(new_name, strlen(dest) + 20, "%s_%d.%s", base_name, counter, extension);
        } else {
            // 확장자가 없는 경우: base_name_counter
            snprintf(new_name, strlen(dest) + 20, "%s_%d", base_name, counter);
        }
        counter++; // 다음 시도를 위해 카운터 증가
    } while (stat(new_name, &st) == 0); // 파일이 존재하는 동안 계속 반복

    // 임시로 할당한 base_name 메모리 해제
    free(base_name);
    return new_name; // 호출자가 free()로 해제해야 함
}
```



```
// -n 옵션: no-clobber, 기존 파일을 절대 덮어쓰지 않음
if (opts->no_clobber) {
    return false;
}
```

```
// -s 옵션이 없는 경우 덮어쓰기 여부 확인
if (!should_overwrite(final_dest, opts)) {
    if (opts->verbose) {
        printf("'%' 이동이 취소되었습니다.\n", src);
    }
    // 메모리 정리 후 함수 종료
    if (allocated_dest) {
        free(final_dest);
    }
    return 0;
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./mv_run -n newa.txt d.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 8
-rw-r--r--  1 qkrehgus qkrehgus    74 May 29 17:10 d.txt
-rw-r--r--  1 qkrehgus qkrehgus    15 Feb 22 22:22 newa.txt
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./mv_run -v d.txt newa.txt  
'd.txt' -> 'newa.txt'
```

```
// -v 옵션: 이동 성공 시 상세 정보 출력  
if (opts->verbose) {  
    printf("`%s' -> '%s'\n", src, final_dest);  
}
```

```

// -s 옵션이 없는 경우 덮어쓰기 여부 확인
if (!should_overwrite(final_dest, opts)) {
    if (opts->verbose) {
        printf("'%' 이동이 취소되었습니다.\n", src);
    }
    // 메모리 정리 후 함수 종료
    if (allocated_dest) {
        free(final_dest);
    }
    return 0;
}

```

```

bool should_overwrite(const char *dest, mv_options_t *opts) {
    struct stat st; // 파일 상태 정보를 저장할 구조체

    // stat()로 대상 파일의 존재 여부 확인
    // 파일이 존재하지 않으면 (stat 실패) 덮어쓰기 문제가 없으므로 허용
    if (stat(dest, &st) != 0) {
        return true;
    }

    // -n 옵션: no-clobber, 기존 파일을 절대 덮어쓰지 않음
    if (opts->no_clobber) {
        return false;
    }

    // -f 옵션: force, 어떤 경우에도 강제로 덮어쓰기
    // 파일 권한이나 다른 제약과 관계없이 시도
    if (opts->force) {
        return true;
    }
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 12
-rw-r--r--  1 qkrehgus qkrehgus   74 May 29 17:10 c.txt
-----  1 qkrehgus qkrehgus   13 May 29 17:09 d.txt
-rw-r--r--  1 qkrehgus qkrehgus  15 Feb 22 22:22 newa.txt

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./mv_run -f c.txt d.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 8
-rw-r--r--  1 qkrehgus qkrehgus   74 May 29 17:10 d.txt
-rw-r--r--  1 qkrehgus qkrehgus   15 Feb 22 22:22 newa.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ cat d.txt
Hello!!

```

This test file!!!

안녕하세요!
이건 테스트입니다.

rm

- | | | |
|------|-----------------|-------------------------------|
| • -r | : 디렉토리 포함 재귀 삭제 | <code>./rm -r folder/</code> |
| • -f | : 강제 삭제 | <code>./rm -f file.txt</code> |
| • -i | : 삭제 전 사용자 확인 | <code>./rm -i file.txt</code> |
| • -v | : 삭제되는 항목 출력 | <code>./rm -v file.txt</code> |
| • -z | : 0바이트 파일만 삭제 | <code>./rm -z file.txt</code> |

옵션

예시

```

int remove_file(const char *filepath, const rm_options_t *opts) {
    // 1단계: 파일/디렉토리 존재 여부 확인
    if (access(filepath, F_OK) != 0) {
        // 파일이 존재하지 않는 경우
        if (lopts->force) {
            // force 모드가 아니면 에러 메시지 출력
            fprintf(stderr, "rm: cannot remove '%s': No such file or directory\n", filepath);
        }
        // force 모드에서는 존재하지 않는 파일을 무시하고 성공으로 처리
        return opts->force ? 0 : -1;
    }

    // 2단계: -z 옵션 처리 (0바이트 파일만 삭제)
    if (opts->zero_only && !is_zero_byte_file(filepath)) {
        if (opts->verbose) {
            printf("skipped '%s' (not a zero-byte file)\n", filepath);
        }
        return 0; // 0바이트가 아닌 파일은 건너뛰고 성공으로 처리
    }

    // 3단계: 디렉토리 처리
    if (is_directory(filepath)) {
        if (lopts->recursive) {
            // recursive 옵션 없이 디렉토리를 삭제하려는 경우 에러
            if (lopts->force) {
                fprintf(stderr, "rm: cannot remove '%s': Is a directory\n", filepath);
            }
            return -1;
        }
        // recursive 옵션이 있으면 재귀적으로 디렉토리 삭제
        return remove_directory_recursive(filepath, opts);
    }

    // 4단계: 대화형 확인 (-i 옵션)
    if (opts->interactive && !get_user_confirmation(filepath)) {
        return 0; // 사용자가 삭제를 거부하면 성공으로 처리 (에러가 아님)
    }

    // 5단계: 실제 파일 삭제
    if (unlink(filepath) != 0) {
        // unlink() 시스템 콜 실패시
        if (lopts->force) {
            fprintf(stderr, "rm: cannot remove '%s': %s\n", filepath, strerror(errno));
        }
        // force 모드에서는 삭제 실패도 성공으로 처리
        return opts->force ? 0 : -1;
    }

    // 6단계: verbose 출력 (-v 옵션)
    if (opts->verbose) {
        printf("removed '%s'\n", filepath);
    }

    return 0; // 성공
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 12
----- 1 qkrehgus qkrehgus      74 May 29 18:03 a.txt
-rwxr-xr-x 1 qkrehgus qkrehgus      74 May 29 18:03 b.txt
-rw-r--r-- 1 qkrehgus qkrehgus      74 May 29 17:10 b_1.txt
-rw-r--r-- 1 qkrehgus qkrehgus       0 May 29 18:13 c.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./rm_run c.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 12
----- 1 qkrehgus qkrehgus      74 May 29 18:03 a.txt
-rwxr-xr-x 1 qkrehgus qkrehgus      74 May 29 18:03 b.txt
-rw-r--r-- 1 qkrehgus qkrehgus      74 May 29 17:10 b_1.txt

```

```

// 3단계: 디렉토리 처리
if (is_directory(filepath)) {
    if (!opts->recursive) {
        // recursive 옵션 없이 디렉토리를 삭제하려는 경우 에러
        if (!opts->force) {
            fprintf(stderr, "rm: cannot remove '%s': Is a directory\n", filepath);
        }
        return -1;
    }

    // recursive 옵션이 있으면 재귀적으로 디렉토리 삭제
    return remove_directory_recursive(filepath, opts);
}

```

```

// 디렉토리 내의 모든 항목을 순회
while ((entry = readdir(dir)) != NULL) {
    // 현재 디렉토리(.)와 부모 디렉토리(..) 건너뛰기
    // 이를 삭제하려고 하면 무한 루프나 시스템 오류가 발생할 수 있음
    if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0) {
        continue;
    }

    // 디렉토리 경로와 항목 이름을 결합하여 전체 경로 생성
    char *filepath = join_path(dirpath, entry->d_name);
    if (filepath == NULL) {
        fprintf(stderr, "rm: memory allocation failed\n");
        result = -1;
        break;
    }

    // 각 항목을 재귀적으로 삭제 (파일이면 파일 삭제, 디렉토리면 재귀 호출)
    if (remove_file(filepath, opts) != 0) {
        result = -1; // 하나라도 실패하면 전체 실패로 간주
    }

    free(filepath); // 동적 할당된 경로 메모리 해제
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ls
Makefile  cat      cp_run  ls      mv_run  sample
b.txt     cat_run  find    ls_run  rm      touch
b_1.txt   cp       find_run mv      rm_run  touch_run
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./rm_run -r sample
qkrehgus@LAPTOP-52P9N50L:~/commands$ ls
Makefile  cat      cp_run  ls      mv_run  touch
b.txt     cat_run  find    ls_run  rm      touch_run
b_1.txt   cp       find_run mv      rm_run

```

```

// 1단계: 파일/디렉토리 존재 여부 확인
if (access(filepath, F_OK) != 0) {
    // 파일이 존재하지 않는 경우
    if (!opts->force) {
        // force 모드가 아니면 에러 메시지 출력
        fprintf(stderr, "rm: cannot remove '%s': No such file or directory\n", filepath);
    }
    // force 모드에서는 존재하지 않는 파일을 무시하고 성공으로 처리
    return opts->force ? 0 : -1;
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 12
----- 1 qkrehgus qkrehgus      74 May 29 18:03 a.txt
-rwxr-xr-x 1 qkrehgus qkrehgus      74 May 29 18:03 b.txt
-rw-r--r-- 1 qkrehgus qkrehgus      74 May 29 17:10 b_1.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./rm_run -f a.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 8
-rwxr-xr-x 1 qkrehgus qkrehgus      74 May 29 18:03 b.txt
-rw-r--r-- 1 qkrehgus qkrehgus      74 May 29 17:10 b_1.txt

```

```
// 4단계: 대화형 확인 (-i 옵션)
if (opts->interactive && !get_user_confirmation(filepath)) {
    return 0; // 사용자가 삭제를 거부하면 성공으로 처리 (에러가 아님)
}
```

```
static bool get_user_confirmation(const char *filepath) {
    printf("rm: remove '%s'? ", filepath);
    fflush(stdout); // 출력 버퍼를 즉시 비워서 프롬프트가 바로 표시되도록 함

    char response[10];
    if (fgets(response, sizeof(response), stdin) == NULL) {
        return false; // 입력 오류 시 삭제하지 않음
    }

    // 첫 번째 문자가 'y' 또는 'Y'인 경우에만 삭제 승인
    return (response[0] == 'y' || response[0] == 'Y');
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 8
-rwxr-xr-x  1 qkrehgus qkrehgus      74 May 29 18:03 b.txt
-rw-r--r--  1 qkrehgus qkrehgus      74 May 29 17:10 b_1.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./rm_run -i b_1.txt
rm: remove 'b_1.txt'? y
```



```
// 6단계: verbose 출력 (-v 옵션)
if (opts->verbose) {
    printf("removed '%s'\n", filepath);
}
```

```
total 4
-rwxr-xr-x  1 qkrehgus qkrehgus      74 May 29 18:03 b.txt
-rw-r--r--  1 qkrehgus qkrehgus       0 May 29 18:16 c.txt
-rw-r--r--  1 qkrehgus qkrehgus       0 May 29 18:16 d.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./rm_run -v c.txt
removed 'c.txt'
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 4
-rwxr-xr-x  1 qkrehgus qkrehgus      74 May 29 18:03 b.txt
-rw-r--r--  1 qkrehgus qkrehgus       0 May 29 18:16 d.txt
```

```
static bool is_zero_byte_file(const char *filepath) {
    struct stat st;
    if (stat(filepath, &st) != 0) {
        return false; // stat 실패시 0바이트 파일이 아닌 것으로 간주
    }
    // 일반 파일이면서 크기가 0인 경우에만 true 반환
    return (S_ISREG(st.st_mode) && st.st_size == 0);
}
```

```
// 2단계: -z 옵션 처리 (0바이트 파일만 삭제)
if (opts->zero_only && !is_zero_byte_file(filepath)) {
    if (opts->verbose) {
        printf("skipped '%s' (not a zero-byte file)\n", filepath);
    }
    return 0; // 0바이트가 아닌 파일은 건너뛰고 성공으로 처리
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 4
-rw-r--r-- 1 qkrehgus qkrehgus 0 May 29 18:16 a.txt
-rwxr-xr-x 1 qkrehgus qkrehgus 74 May 29 18:03 b.txt
-rw-r--r-- 1 qkrehgus qkrehgus 0 May 29 18:16 c.txt
-rw-r--r-- 1 qkrehgus qkrehgus 0 May 29 18:16 d.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./rm_run -z a.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 4
-rwxr-xr-x 1 qkrehgus qkrehgus 74 May 29 18:03 b.txt
-rw-r--r-- 1 qkrehgus qkrehgus 0 May 29 18:16 c.txt
-rw-r--r-- 1 qkrehgus qkrehgus 0 May 29 18:16 d.txt
```

find

• -n [이름]	: 정확한 파일명으로 검색 (대소문자 구분)	<code>./find -n hello.txt</code>
• -i [이름]	: 대소문자 구분 없이 파일명 검색	<code>./find -i hello.txt</code>
• -f	: 일반 파일만 검색	<code>./find -f</code>
• -d	: 디렉토리만 검색	<code>./find -d</code>
• -s [+/- 크기]	: 특정 크기 기준 검색	<code>./find -s +10M</code>
• -u [사용자명]	: 특정 소유자의 파일 검색	<code>./find -u user1</code>
• -p [권한]	: 특정 권한 조건을 만족하는 파일 검색	<code>./find -p 755</code>
• -t [N]	: N일 전에 수정된 파일 검색	<code>./find -t +7</code>
• -e	: 빈 파일 또는 빈 디렉토리 검색	<code>./find -e</code>

옵션

예시

```

void find_recursive(const char *path, const find_options_t *opts) {
    DIR *dir;
    struct dirent *entry;
    struct stat st;
    char filepath[PATH_MAX]; // 전체 경로 저장용

    // 현재 경로 자체도 조건 검사 대상
    if (stat(path, &st) == 0) {
        // 경로에서 파일명만 추출
        const char *filename = strrchr(path, '/');
        filename = filename ? filename + 1 : path; // '/' 이후 부분, 없으면 전체

        // 조건에 맞으면 출력
        if (matches_criteria(path, filename, &st, opts)) {
            printf("%s\n", path);
        }
    }

    // 디렉토리거나 아니면 더 이상 탐색할 필요 없음
    if (!IS_IDIR(st.st_mode)) {
        return;
    }

    // 디렉토리 열기 시도
    dir = opendir(path);
    if (!dir) {
        perror(path); // 디렉토리 열기 실패시 에러 출력
        return;
    }

    // 디렉토리 내의 모든 항목 순회
    while ((entry = readdir(dir)) != NULL) {
        // 현재 디렉토리(.)와 상위 디렉토리(..) 건너뛰기
        if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0) {
            continue;
        }

        // 전체 경로 구성 (부모경로/항목명)
        snprintf(filepath, sizeof(filepath), "%s/%s", path, entry->d_name);

        // 파일 정보 획득 (신분과 링크는 링크 자체 정보)
        if (lstat(filepath, &st) != 0) {
            continue; // 파일 정보 획득 실패시 건너뛰기
        }

        // 검색 조건에 맞는지 확인하고 맞으면 출력
        if (matches_criteria(filepath, entry->d_name, &st, opts)) {
            printf("%s\n", filepath);
        }

        // 디렉토리인 경우 재귀적으로 하위 탐색
        if (S_ISDIR(st.st_mode)) {
            find_recursive(filepath, opts);
        }
    }

    closedir(dir); // 디렉토리 핸들 정리
}

```

```

qkrehgus@LAPTOP-52P9N5OL:~/commands$ ./find_run find
find
find/find_options.o
find/find_options.c
find/find.o
find/find_options.h
find/find.c

```

```

case 'n':
    // 인자가 필요한 옵션: 현재 옵션 문자열의 마지막이고 다음 인자가 있어야 함
    if (j == strlen(arg) - 1 && i + 1 < argc) {
        opts->name_pattern = strdup(argv[++i]); // 다음 인자를 패턴으로 사용
        goto next_arg; // 현재 옵션 문자열 처리 완료
    } else {
        fprintf(stderr, "오류: -n 옵션은 분리해서 사용해야 합니다.\n");
        return -1;
    }
    break;

```

```

// 현재 경로 자체도 조건 검사 대상
if (stat(path, &st) == 0) {
    // 경로에서 파일명만 추출
    const char *filename = strrchr(path, '/');
    filename = filename ? filename + 1 : path; // '/' 이후 부분, 없으면 전체

    // 조건에 맞으면 출력
    if (matches_criteria(path, filename, &st, opts)) {
        printf("%s\n", path);
    }
}

```

```

// 파일명 패턴 매칭 (대소문자 구분)
if (opts->name_pattern && fnmatch(opts->name_pattern, filename, 0) != 0) {
    return false; // 이름 패턴이 일치하지 않음
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./find_run -n find
./find
./find

```

```
// 파일명 패턴 매칭 (대소문자 무시)
if (opts->iname_pattern && fnmatch(opts->iname_pattern, filename, FNM_CASEFOLD) != 0) {
    return false; // 이름 패턴이 일치하지 않음 (대소문자 무시)
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -lf txt
total 4
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 18:26 D.txt
-rwxr-xr-x  1 qkrehgus qkrehgus    74 May 29 18:03 b.txt
-rw-r--r--  1 qkrehgus qkrehgus      0 May 29 18:16 d.txt
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./find_run -i d.txt
./d.txt
./D.txt
```

```

// 디렉토리 내의 모든 항목 순회
while ((entry = readdir(dir)) != NULL) {
    // 현재 디렉토리(.)와 상위 디렉토리(..) 건너뛰기
    if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
        continue;

    // 전체 경로 구성 (부모경로/항목명)
    snprintf(filepath, sizeof(filepath), "%s/%s", path, entry->d_name);

    // 파일 정보 획득 (심볼릭 링크는 링크 자체 정보)
    if (lstat(filepath, &st) != 0) {
        continue; // 파일 정보 획득 실패시 건너뛰기
    }

    // 검색 조건에 맞는지 확인하고 맞으면 출력
    if (matches_criteria(filepath, entry->d_name, &st, opts)) {
        printf("%s\n", filepath);
    }

    // 디렉토리인 경우 재귀적으로 하위 탐색
    if (S_ISDIR(st.st_mode)) {
        find_recursive(filepath, opts);
    }
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./find_run -f
./find/find_options.o
./find/find_options.c
./find/find.o
./find/find_options.h
./find/find.c
./d.txt
./ls/ls_options.o
./ls/ls_options.c
./ls/ls.c
./ls/ls.o
./ls/ls_options.h
./b.txt

```

```

// 파일 타입 필터링
if (opts->type_file && !S_ISREG(st->st_mode)) {
    return false; // 파일만 검색하는데 파일이 아님
}

```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./find_run -d
```

```
.  
./find  
./find  
./ls  
./ls  
./touch  
./touch  
./cat  
./cat  
./cp  
./cp  
./mv  
./mv  
./rm  
./rm
```

```
// 디렉토리 내의 모든 항목 순회  
while ((entry = readdir(dir)) != NULL) {  
    // 현재 디렉토리(.)와 상위 디렉토리(..) 건너뛰기  
    if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)  
        continue;  
}  
  
// 전체 경로 구성 (부모경로/항목명)  
snprintf(filepath, sizeof(filepath), "%s/%s", path, entry->d_name);  
  
// 파일 정보 획득 (심볼릭 링크는 링크 자체 정보)  
if (lstat(filepath, &st) != 0) {  
    continue; // 파일 정보 획득 실패시 건너뛰기  
}  
  
// 검색 조건에 맞는지 확인하고 맞으면 출력  
if (matches_criteria(filepath, entry->d_name, &st, opts)) {  
    printf("%s\n", filepath);  
}  
  
// 디렉토리인 경우 재귀적으로 하위 탐색  
if (S_ISDIR(st.st_mode)) {  
    find_recursive(filepath, opts);  
}
```

```
}  
  
if (opts->type_dir && !S_ISDIR(st->st_mode)) {  
    return false; // 디렉토리만 검색하는데 디렉토리가 아님  
}
```



```

case 's':
    if (j == strlen(arg) - 1 && i + 1 < argc) {
        opts->size_spec = strdup(argv[++i]); // 크기 조건
        goto next_arg;
    } else {
        fprintf(stderr, "오류: -s 옵션은 분리해서 사용해야 합니다.\n");
        return -1;
    }
    break;

// 파일 크기 조건 확인
if (opts->size_spec) {
    long target_size = parse_size_spec(opts->size_spec);
    if (target_size < 0) return false; // 크기 파싱 실패

    // 크기 비교 (+: 보다 큰, -: 보다 작은, 없음: 정확히 같은)
    if (opts->size_spec[0] == '+' && st->st_size <= target_size) {
        return false; // 지정 크기보다 크지 않음
    } else if (opts->size_spec[0] == '-' && st->st_size >= target_size) {
        return false; // 지정 크기보다 작지 않음
    } else if (opts->size_spec[0] != '+' && opts->size_spec[0] != '-' && st->st_size != target_size) {
        return false; // 정확한 크기가 아님
    }
}

```

```

qkrehgus@LAPTOP-52P9N50L:~/commands$ ./ls_run -hl
total 244
-rw-r--r-- 1 qkrehgus qkrehgus 0 May 29 18:26 D.txt
-rw-r--r-- 1 qkrehgus qkrehgus 4.4K May 29 13:31 Makefile
-rwxr-xr-x 1 qkrehgus qkrehgus 74 May 29 18:03 b.txt
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:28 cat
-rwxr-xr-x 1 qkrehgus qkrehgus 22K May 29 13:28 cat_run
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:29 cp
-rwxr-xr-x 1 qkrehgus qkrehgus 24K May 29 13:29 cp_run
-rw-r--r-- 1 qkrehgus qkrehgus 0 May 29 18:16 d.txt
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:31 find
-rwxr-xr-x 1 qkrehgus qkrehgus 30K May 29 13:31 find_run
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:26 ls
-rwxr-xr-x 1 qkrehgus qkrehgus 32K May 29 13:26 ls_run
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:31 mv
-rwxr-xr-x 1 qkrehgus qkrehgus 25K May 29 13:31 mv_run
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 13:32 rm
-rwxr-xr-x 1 qkrehgus qkrehgus 25K May 29 13:32 rm_run
drwxr-xr-x 2 qkrehgus qkrehgus 4.0K May 29 17:24 touch
-rwxr-xr-x 1 qkrehgus qkrehgus 30K May 29 17:24 touch_run
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./find_run -s +30K
./ls_run
./touch_run

```

```
qkrehgus@LAPTOP-52P9N5OL:~/commands$ ./find_run -u qkrehgus
```

```
.  
./find  
./find  
./find/find_options.o  
./find/find_options.c  
./find/find.o  
./find/find_options.h  
./find/find.c  
./d.txt  
./ls  
./ls  
./ls/ls_options.o
```

```
// 파일 소유자 확인  
if (opts->user_name) {  
    uid_t target_uid = get_uid_by_name(opts->user_name);  
    if (target_uid == (uid_t)-1 || st->st_uid != target_uid) {  
        return false; // 지정된 사용자의 파일이 아님  
    }  
}
```

```
qkrehgus@LAPTOP-52P9N50L:~/commands$ ./find_run -fp 755
```

```
./d.txt
```

```
./b.txt
```

```
./mv_run
```

```
./ls_run
```

```
./find_run
```

```
./D.txt
```

```
./rm_run
```

```
./cat_run
```

```
./cp_run
```

```
./touch_run
```

```
// 파일 권한 확인
```

```
if (opts->perm_spec) {
```

```
    mode_t target_perm = parse_perm_spec(opts->perm_spec);
```

```
    // 파일 권한 부분만 비교 (하위 9비트: rwxrwxrwx)
```

```
    if ((st->st_mode & 0777) != target_perm) {
```

```
        return false; // 권한이 일치하지 않음
```

```
    }
```

```
}
```

```

qkrehgus@LAPTOP-52P9N5OL:~/commands$ ls -l
total 244
-rwxr-xr-x 1 qkrehgus qkrehgus  0 Jan  1 00:00 D.txt
-rw-r--r-- 1 qkrehgus qkrehgus 4488 May 29 13:31 Makefile
-rwxr-xr-x 1 qkrehgus qkrehgus  74 Jan  1 00:00 b.txt
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:28 cat
-rwxr-xr-x 1 qkrehgus qkrehgus 23000 May 29 13:28 cat_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:29 cp
-rwxr-xr-x 1 qkrehgus qkrehgus 25088 May 29 13:29 cp_run
-rwxr-xr-x 1 qkrehgus qkrehgus  0 Jan  1 00:00 d.txt
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 18:53 find
-rwxr-xr-x 1 qkrehgus qkrehgus 30776 May 29 18:53 find_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:26 ls
-rwxr-xr-x 1 qkrehgus qkrehgus 32544 May 29 13:26 ls_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:31 mv
-rwxr-xr-x 1 qkrehgus qkrehgus 25160 May 29 13:31 mv_run
-rw-r--r-- 1 qkrehgus qkrehgus  0 Mar  1 2024 old.txt
-rw-r--r-- 1 qkrehgus qkrehgus  0 May 29 13:00 recent.txt
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:32 rm
-rwxr-xr-x 1 qkrehgus qkrehgus 26088 May 29 13:32 rm_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 17:24 touch
-rwxr-xr-x 1 qkrehgus qkrehgus 30760 May 29 17:24 touch_run
qkrehgus@LAPTOP-52P9N5OL:~/commands$ ./find_run -t +30
./d.txt
./b.txt
./D.txt
./old.txt

```

```

// 수정 시간 확인 (+n: n일 이전, -n: n일 이내, n: 정확히 n일 전)
if (opts->mtime_set) {
    time_t now = time(NULL);
    time_t file_time = st->st_mtime;
    // 현재 시간과 파일 수정 시간의 차이를 일 단위로 계산
    int days_diff = (now - file_time) / (24 * 60 * 60);

    // mtime 조건 확인
    if (opts->mtime_prefix == '+' && days_diff <= opts->mtime_days) {
        return false; // +n: n일보다 이전에 수정된 파일이 아님
    } else if (opts->mtime_prefix == '-' && days_diff >= opts->mtime_days) {
        return false; // -n: n일 이내에 수정된 파일이 아님
    } else if (opts->mtime_prefix == 0 && days_diff != opts->mtime_days) {
        return false; // n: 정확히 n일 전에 수정된 파일이 아님
    }
}

```

```

qkrehgus@LAPTOP-52P9N5OL:~/commands$ ls -l
total 244
-rwxr-xr-x 1 qkrehgus qkrehgus  0 Jan  1 00:00 D.txt
-rw-r--r-- 1 qkrehgus qkrehgus 4488 May 29 13:31 Makefile
-rwxr-xr-x 1 qkrehgus qkrehgus  74 Jan  1 00:00 b.txt
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:28 cat
-rwxr-xr-x 1 qkrehgus qkrehgus 23000 May 29 13:28 cat_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:29 cp
-rwxr-xr-x 1 qkrehgus qkrehgus 25088 May 29 13:29 cp_run
-rwxr-xr-x 1 qkrehgus qkrehgus  0 Jan  1 00:00 d.txt
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 18:53 find
-rwxr-xr-x 1 qkrehgus qkrehgus 30776 May 29 18:53 find_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:26 ls
-rwxr-xr-x 1 qkrehgus qkrehgus 32544 May 29 13:26 ls_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:31 mv
-rwxr-xr-x 1 qkrehgus qkrehgus 25160 May 29 13:31 mv_run
-rw-r--r-- 1 qkrehgus qkrehgus  0 Mar  1 2024 old.txt
-rw-r--r-- 1 qkrehgus qkrehgus  0 May 29 13:00 recent.txt
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 13:32 rm
-rwxr-xr-x 1 qkrehgus qkrehgus 26088 May 29 13:32 rm_run
drwxr-xr-x 2 qkrehgus qkrehgus 4096 May 29 17:24 touch
-rwxr-xr-x 1 qkrehgus qkrehgus 30760 May 29 17:24 touch_run
qkrehgus@LAPTOP-52P9N5OL:~/commands$ ./find_run -e
./recent.txt
./d.txt
./D.txt
./old.txt

```

```

// 빈 파일/디렉토리 필터
if (opts->empty_filter && !is_empty(filepath, st)) {
    return false; // 빈 파일/디렉토리가 아님
}

```

```

bool is_empty(const char *path, const struct stat *st) {
    if (S_ISREG(st->st_mode)) {
        // 일반 파일의 경우: 크기가 0이면 빈 파일
        return st->st_size == 0;
    } else if (S_ISDIR(st->st_mode)) {
        // 디렉토리의 경우: 하위 항목이 없으면 빈 디렉토리
        DIR *dir = opendir(path);
        if (!dir) return false;

        struct dirent *entry;
        int count = 0;

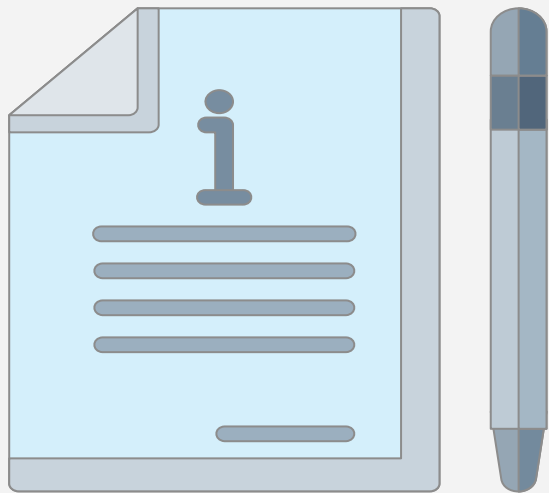
        // 디렉토리 내용을 읽어서 실제 파일/디렉토리가 있는지 확인
        while ((entry = readdir(dir)) != NULL) {
            // "."과 ".." 제외하고 실제 항목이 있는지 확인
            if (strcmp(entry->d_name, ".") != 0 && strcmp(entry->d_name, "..") != 0) {
                count++;
                break; // 하나라도 있으면 빈 디렉토리가 아님
            }
        }
        closedir(dir);
        return count == 0;
    }
    return false;
}

```

• 점수 : 15점

이유

주어진 시간 안에 리눅스에서 자주 사용하는 파일 및 디렉토리 관리 명령어들을 핵심 기능과 다양한 옵션까지 충실히 구현해냈고, 실제 사용자가 직관적으로 활용할 수 있도록 명확하고 일관된 구조를 갖추고 옵션별 동작도 예상대로 잘 작동되기 때문입니다.



02

깃허브 정리



PARK-DOHYUN Create README.md

2b03261 · 3 hours ago



 0314	Create README.md	3 hours ago
 0321	Update README.md	3 months ago
 0328	Delete 0328/hw.c	2 months ago
 0404	Create README.md	2 months ago
 0411	Add files via upload	2 months ago
 0418	Update README.md	2 months ago
 0502	Update README.md	last month
 0509	Update README.md	3 weeks ago
 0516	Update README.md	2 weeks ago
 0523	Update README.md	5 days ago
 0530	Update README.md	3 hours ago
 Commands	Add files via upload	5 days ago
 README.md	Initial commit	3 months ago

Commits on Apr 4, 2025

Create mission.c Verified 71ed27f [📄](#) [↗](#)
PARK-DOHYUN authored on Apr 4

Commits on Mar 28, 2025

Add files via upload Verified cc6c44f [📄](#) [↗](#)
PARK-DOHYUN authored on Mar 28

Create README.md Verified 30f1e4a [📄](#) [↗](#)
PARK-DOHYUN authored on Mar 28

Commits on Mar 23, 2025

Update README.md Verified f788469 [📄](#) [↗](#)
PARK-DOHYUN authored on Mar 23

Commits on Apr 24, 2025

Update README.md Verified 3db2f43 [📄](#) [↗](#)
PARK-DOHYUN authored on Apr 24

Add files via upload Verified 3485563 [📄](#) [↗](#)
PARK-DOHYUN authored on Apr 24

Create README.md Verified 9e7c266 [📄](#) [↗](#)
PARK-DOHYUN authored on Apr 24

Commits on Apr 11, 2025

Add files via upload Verified d269dc6 [📄](#) [↗](#)
PARK-DOHYUN authored on Apr 11

Update README.md Verified 79791e8 [📄](#) [↗](#)
PARK-DOHYUN authored on Apr 11

Update and rename README.md to README.md Verified d02ffc9 [📄](#) [↗](#)
PARK-DOHYUN authored 3 weeks ago

Create README.md Verified 36f8c34 [📄](#) [↗](#)
PARK-DOHYUN authored 3 weeks ago

Commits on May 15, 2025

Update README.md Verified 36d25a7 [📄](#) [↗](#)
PARK-DOHYUN authored 3 weeks ago

Add files via upload Verified f2ce6ac [📄](#) [↗](#)
PARK-DOHYUN authored 3 weeks ago

Commits on May 9, 2025

Update README.md Verified 2bea841 [📄](#) [↗](#)
PARK-DOHYUN authored 3 weeks ago

Update README.md Verified b4d0d31 [📄](#) [↗](#)
PARK-DOHYUN authored 4 hours ago

Create README.md Verified d98e0b6 [📄](#) [↗](#)
PARK-DOHYUN authored 4 hours ago

Commits on May 29, 2025

Update README.md Verified 79e4acc [📄](#) [↗](#)
PARK-DOHYUN authored 5 days ago

Add files via upload Verified 949d805 [📄](#) [↗](#)
PARK-DOHYUN authored 5 days ago

Create README.md Verified 1218767 [📄](#) [↗](#)
PARK-DOHYUN authored 5 days ago

Add files via upload Verified 1111111 [📄](#) [↗](#)
PARK-DOHYUN authored 5 days ago

• 점수 : 14.4점

이유

2,4주차 수업을 제외하고 모든 파일을 깃허브에 제시간에 업로드 하였기에 14.4점을 주었습니다.