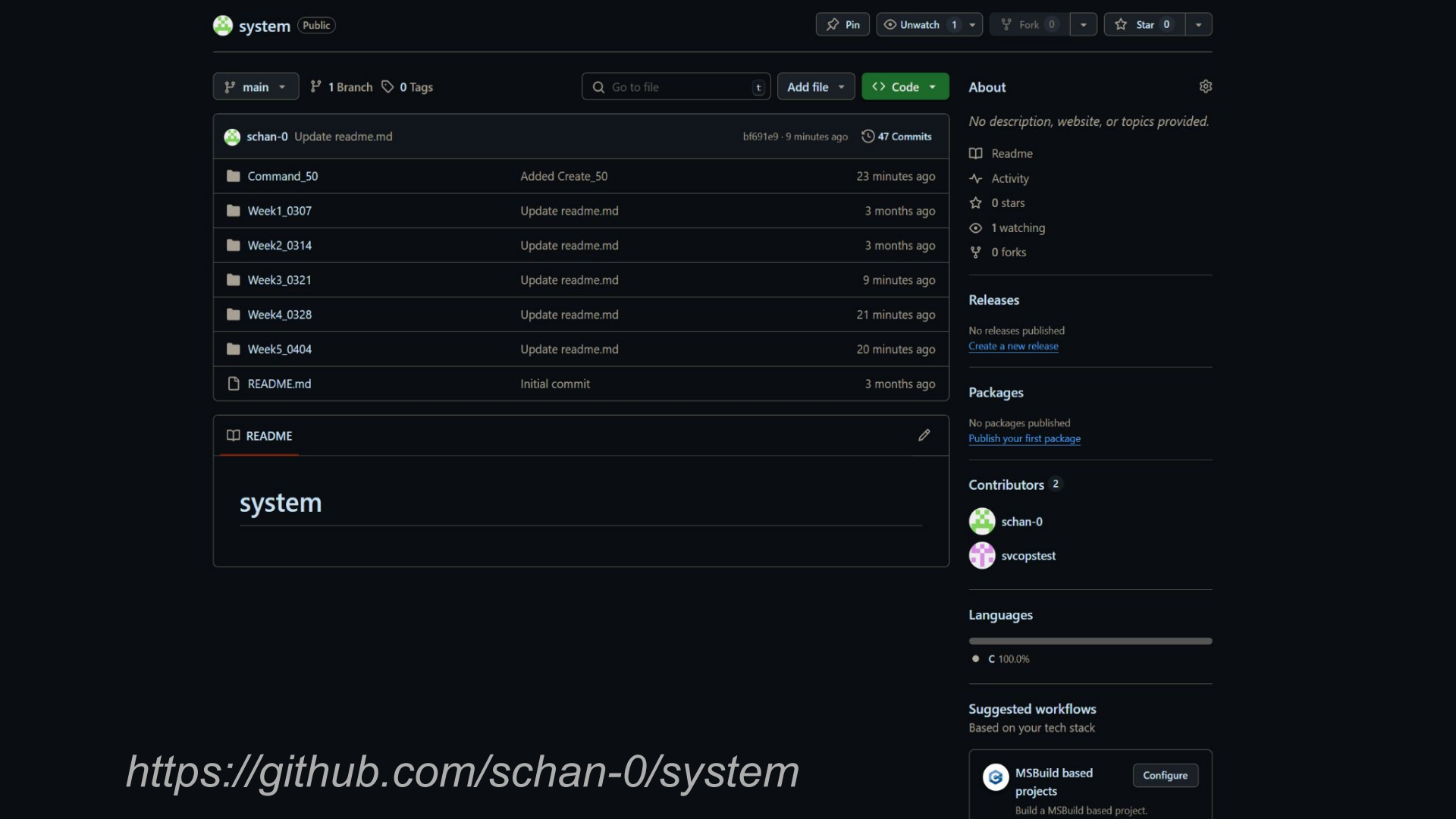
Github

</> System Programming Report

2021563074 박승찬

https://github.com/schan-0/system



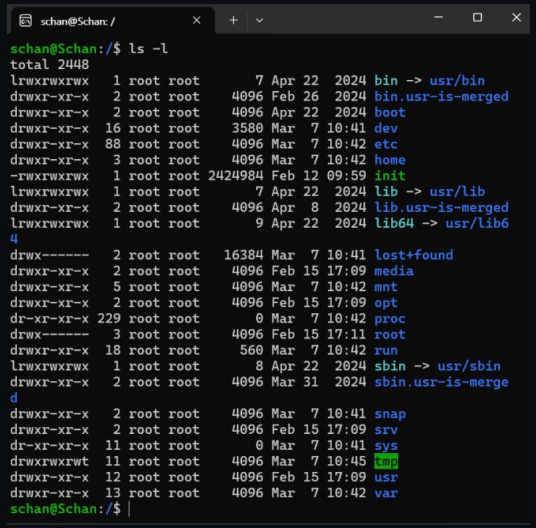
https://github.com/schan-0/system

Name	Last commit message	Last commit date
..		
readme.md	Update readme.md	3 months ago

readme.md

Week1

1. WSL 설치



- 윈도우 파워쉘을 관리자 모드로 실행하고 `wsl --install` 을 입력합니다.
- WSL을 설치합니다. 이 과정에서 리눅스 우분투 배포판이 자동으로 설치되며, 한 번의 재부팅을 요구합니다.
- WSL을 실행하여 우분투에 사용자 계정명, 비밀번호를 입력합니다.
- `cd /` 명령어로 루트 디렉토리로 이동하여 `ls -l` 명령어로 디렉토리 내부를 살펴봅니다.

1주차

Commits

History for system / Week1_0307 on `main`

Commits on Mar 13, 2025

- **Update readme.md** — schan-0 authored on Mar 13
- **Update readme.md** — schan-0 authored on Mar 13
- **Update readme.md** — schan-0 authored on Mar 13
- **Update readme.md** — schan-0 authored on Mar 13
- **Update readme.md** — schan-0 authored on Mar 13
- **Update readme.md** — schan-0 authored on Mar 13
- **Update readme.md** — schan-0 authored on Mar 13
- **Update readme.md** — schan-0 authored on Mar 13
- **Update readme.md**

system / Week2_0314 /

Add file

schan-0 Update readme.md ffb49f7 · 3 months ago History

Name	Last commit message	Last commit date
..		
readme.md	Update readme.md	3 months ago

readme.md

Week2

1. Tree 패키지 설치

`$ apt update`

설치 전 패키지 리포지터리를 업데이트한다.

`$ sudo apt install tree`

tree 패키지를 설치한다.

2. 디렉토리 생성

```
schan@Schan:~/software$ tree
.
├── app
│   └── d.txt
└── system
    ├── LP
    │   ├── compiler
    │   │   └── a.txt
    │   └── interpreter
    │       └── b.txt
    ├── OS
    └── Util
        └── c.txt
```

2주차

Commits

History for system / Week2_0314 on main

Commits on Mar 20, 2025

- **Update readme.md** — schan-0 authored on Mar 20
- **Update readme.md** — schan-0 authored on Mar 20
- **Update readme.md** — schan-0 authored on Mar 20
- **Update readme.md** — schan-0 authored on Mar 20
- **Update readme.md** — schan-0 authored on Mar 20
- **Create readme.md** — schan-0 authored on Mar 20

End of commit history for this file

system / Week3_0321 /

schan-0 Update readme.md

Name	Last commit message
..	
readme.md	Update readme.md
student.c	Add files via upload

readme.md

Week3

```
schan@Schan:~$ ./student
이름을 입력해주세요 : 박승찬

학번을 입력해주세요 : 2021563074

당신의 이름과 학번은 박승찬_2021563074 입니다.
schan@Schan:~$
```

이름 및 학번 출력 프로그램

이 프로그램은 사용자로부터 이름과 학번을 입력받아 "이름_학번" 형식으로 출력합니다.

🚀 기능

- **이름 입력 및 출력**: 사용자에게 이름을 입력받고, 입력받은 이름을 출력 문자열에 포함합니다.
- **학번 입력 및 출력**: 사용자에게 학번을 입력받고, 입력받은 학번을 출력 문자열에 포함합니다.
- **통합 출력**: 입력받은 이름과 학번을 `이름_학번` 형식으로 조합하여 최종 출력합니다.

🛠️ 빌드 및 실행 방법

이 프로그램을 빌드하고 실행하려면 C 컴파일러(예: GCC)가 필요합니다.

1. **소스 코드 저장**: 위의 코드를 `student_info.c` 파일로 저장합니다.

3주차

Commits

History for system / Week3_0321 on `main`

Commits on Jun 13, 2025

- **Update readme.md** — schan-0 authored now
- **Update readme.md** — schan-0 authored 5 hours ago

Commits on Mar 27, 2025

- **Update readme.md** — schan-0 authored on Mar 27
- **Add files via upload** — schan-0 authored on Mar 27
- **Create readme.md** — schan-0 authored on Mar 27

End of commit history for this file

system / Week4_0328 /

schan-0 Update readme.md

Name	Last commit message
..	
converter.c	Add files via upload
readme.md	Update readme.md

readme.md

Week4

```
schan@Schan:~/0328$ cat converter.c
#include <stdio.h>
#include <ctype.h>


int main()
{
        char input;
        char tmp;

        while(1)
        {

                printf("문자 입력 : ");
                input = getchar();
                while ((tmp = getchar()) != '\n');
                if (input == '0')
                        return 1;
                else if (isupper(input))
                        printf("%c의 소문자는 %c입니다.\n", input, tolower(input));
                else if (islower(input))
                        printf("%c의 대문자는 %c입니다.\n", input, toupper(input));
                else
                        printf("유효한 입력이 아닙니다.\n");
        }
        return 0;
}

schan@Schan:~/0328$ ./converter
문자 입력 : a
a의 대문자는 A입니다.
문자 입력 : z
z의 소문자는 z입니다.
문자 입력 : hello
h의 대문자는 H입니다.
문자 입력 : ?hello
유효한 입력이 아닙니다.
문자 입력 : [
유효한 입력이 아닙니다.
문자 입력 : 0
schan@Schan:~/0328$
```

4주차

Commits

History for system / Week4_0328 on main

Commits on Jun 13, 2025

- **Update readme.md** — schan-0 authored 14 minutes ago
- **Update readme.md** — schan-0 authored 5 hours ago
- **Update readme.md** — schan-0 authored 5 hours ago
- **Update readme.md** — schan-0 authored 5 hours ago

Commits on Apr 3, 2025

- **Add files via upload** — schan-0 authored on Apr 3
- **Create readme.md** — schan-0 authored on Apr 3

End of commit history for this file

system / Week5_0404 /

schan-0 Update readme.md

Name	Last commit message
..	
binary.c	Add files via upload
readme.md	Update readme.md

readme.md

Week5

```
schan@Schan:~/0404$ vim binary.c
schan@Schan:~/0404$ gcc binary.c -o binary
schan@Schan:~/0404$ ls
binary  binary.c
schan@Schan:~/0404$ ./binary
0부터 255 사이의 정수를 입력하세요: 355
입력 값의 범위가 벗어났습니다.
schan@Schan:~/0404$ ./binary
0부터 255 사이의 정수를 입력하세요: 27
입력 값의 2진수 값 A: 00011011
A가 가지는 1의 개수: 4
A의 상위 4비트: 0001
schan@Schan:~/0404$ ./binary
0부터 255 사이의 정수를 입력하세요: 255
입력 값의 2진수 값 A: 11111111
A가 가지는 1의 개수: 8
A의 상위 4비트: 1111
schan@Schan:~/0404$
```

이진수 변환 및 비트 연산 프로그램

이 프로그램은 사용자가 0부터 255 사이의 정수를 입력하면, 해당 정수를 2진수로 변환하고, 2진수 값에서 1의 개수
니다.

🚀 기능

- **10진수를 2진수로 변환:** 입력된 정수를 8비트 2진수 문자열로 변환합니다.

Commits

History for system / Week5_0404 on main

Commits on Jun 13, 2025

- **Update readme.md** — schan-0 authored 17 minutes ago
- **Update readme.md** — schan-0 authored 18 minutes ago
- **Update readme.md** — schan-0 authored 34 minutes ago
- **Update readme.md** — schan-0 authored 34 minutes ago

Commits on Apr 10, 2025

- **Update readme.md** — schan-0 authored on Apr 10
- **Add files via upload** — schan-0 authored on Apr 10
- **Create readme.md** — schan-0 authored on Apr 10

End of commit history for this file

5주차

총 점수 : 6 / 15

- Week07 [누락] . . . - 1.5
- Week09 [누락] . . . - 1.5
- Week10 [누락] . . . - 1.5
- Week11 [누락] . . . - 1.5
- Week12 [누락] . . . - 1.5
- Week13 [누락] . . . - 1.5

cat.c	- n b e t E T A s v …	10
clear.c	- …	1
date.c	- d u R I …	5
head.c	- n c q v …	5
id.c	- u g G n r a …	7
mkdir.c	- p v m …	4
pwd.c	- …	1
rmdir.c	- p v …	3
sleep.c	- …	1
tail.c	- n c q v …	5
touch.c	- c a m d t r …	7
whoami.c	- …	1

: 50

$ pwd

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <limits.h>    // PATH_MAX

int main() {
    char path[PATH_MAX];
    getcwd(path, PATH_MAX);
    printf("%s\n", path);
    return 0;
}
```

```
❯ ./pwd
/home/schan/PROJECT
```

```c
#define _XOPEN_SOURCE
#include <sys/time.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <utime.h>
#include <sys/stat.h>
#include <time.h>    // time_t, time, strptime, mktime
#include <string.h>  // strcmp, memset
#include <getopt.h>  // getopt

extern char *optarg;
extern int optind, opterr, optopt;

void print_usage(const char *prog_name);

int main(int argc, char *argv[]) {
    int opt;
    int set_atime = 1; // Default to update both
    int set_mtime = 1; // Default to update both
    int no_create = 0;
    time_t specific_time = 0;
    char *ref_file = NULL;
    int use_specific_time = 0;
    int use_ref_file = 0;
    int no_dereference = 0;

    while ((opt = getopt(argc, argv, "acmd:r:t:h")) != -1) {
        switch (opt) {
            case 'a':
                set_mtime = 0;
                break;
            case 'c':
                no_create = 1;
                break;
            case 'm':
                set_atime = 0;
                break;
            case 'd': {
                struct tm tm_info;
                memset(&tm_info, 0, sizeof(struct tm));
                if (strptime(optarg, "%Y-%m-%d %H:%M:%S", &tm_info) == NULL &&
                    strptime(optarg, "%Y-%m-%d", &tm_info) == NULL) { // Simplified date parsing
                    fprintf(stderr, "touch: invalid date format: '%s'\n", optarg);
                    print_usage(argv[0]);
                    return EXIT_FAILURE;
                }
                specific_time = mktime(&tm_info);
                use_specific_time = 1;
                break;
            }
            case 'r':
                ref_file = optarg;
                use_ref_file = 1;
                break;
            case 't': {
                struct tm tm_info;
                memset(&tm_info, 0, sizeof(struct tm));
                if (strptime(optarg, "%Y%m%d%H%M.%S", &tm_info) == NULL && // YYYYMMDDhhmm.ss
                    strptime(optarg, "%Y%m%d%H%M", &tm_info) == NULL) {     // YYYYMMDDhhmm
                    // Add more formats as needed for full compliance, this is simplified.
                    fprintf(stderr, "touch: invalid timestamp format: '%s'\n", optarg);
                    print_usage(argv[0]);
                    return EXIT_FAILURE;
                }
                specific_time = mktime(&tm_info);
                use_specific_time = 1;
                break;
            }
            case 'h':
                no_dereference = 1;
                break;
            default:
                print_usage(argv[0]);
                return EXIT_FAILURE;
        }
    }

    if (optind >= argc) {
        fprintf(stderr, "touch: missing file operand\n");
        print_usage(argv[0]);
        return EXIT_FAILURE;
    }

    for (int i = optind; i < argc; i++) {
        const char *filename = argv[i];
        struct utimbuf new_times;
        struct stat st;
        int stat_result;

        if (no_dereference) {
            stat_result = lstat(filename, &st);
        } else {
            stat_result = stat(filename, &st);
        }

        if (stat_result != 0) { // File does not exist
            if (no_create) {
                continue; // -c option, do not create
            }
            // Create empty file
            FILE *f = fopen(filename, "w");
            if (f == NULL) {
                perror(filename);
                continue;
            }
            fclose(f);
        }

        if (use_specific_time) {
            new_times.actime = specific_time;
            new_times.modtime = specific_time;
        } else if (use_ref_file) {
            struct stat ref_st;
            if (stat(ref_file, &ref_st) != 0) {
                perror(ref_file);
                return EXIT_FAILURE;
            }
            new_times.actime = ref_st.st_atime;
            new_times.modtime = ref_st.st_mtime;
        } else {
            new_times.actime = time(NULL);
            new_times.modtime = time(NULL);
        }

        // Adjust times based on -a and -m
        if (stat_result == 0) { // If file existed before (or just created)
            if (!set_atime) { // -m was given
                new_times.actime = st.st_atime;
            }
            if (!set_mtime) { // -a was given
                new_times.modtime = st.st_mtime;
            }
        }

        if (no_dereference) {
            if (utimes(filename, (struct timeval *)&new_times) != 0) { // utimes for symlinks
                perror(filename);
```

$ touch -camdtr

$ touch -camdtr

```
❯ ./touch 1.txt
❯ ./touch 2.txt
❯ ./touch 3.txt
❯ ./touch 4.txt
❯ ./touch 5.txt
❯ ./touch 6.txt
❯ ./touch 7.txt
❯ ./touch 8.txt
❯ ./touch 9.txt
❯ ./touch -c 10.txt
```

```
❯ ./touch 1.txt
❯ ./touch -a 2.txt
❯ ./touch -m 3.txt
❯ ./touch -d "2025-01-01 00:00:00" 4.txt
❯ ./touch -t 20250101000000 5.txt
❯ ./touch -r 5.txt 6.txt
```

```
❯ ll
합계 20K
-rw-rw-r-- 1 schan schan    0  6월 13 04:29 1.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:29 2.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:29 3.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:29 4.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:29 5.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:29 6.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:30 7.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:30 8.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:30 9.txt
-rwxrwxr-x 1 schan schan 17K  6월 13 04:29 touch
```

```
❯ ls -lu
합계 20
-rw-rw-r-- 1 schan schan     0  6월 13 04:29 1.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:29 2.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:29 3.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:29 4.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:29 5.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:29 6.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:30 7.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:30 8.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:30 9.txt
-rwxrwxr-x 1 schan schan 16696  6월 13 04:29 touch
```

```
❯ ll
합계 20K
-rw-rw-r-- 1 schan schan    0  6월 13 04:36 1.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:29 2.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:37 3.txt
-rw-rw-r-- 1 schan schan    0  1월  1 00:00 4.txt
-rw-rw-r-- 1 schan schan    0  1월  1 00:00 5.txt
-rw-rw-r-- 1 schan schan    0  1월  1 00:00 6.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:30 7.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:30 8.txt
-rw-rw-r-- 1 schan schan    0  6월 13 04:30 9.txt
-rwxrwxr-x 1 schan schan 17K  6월 13 04:29 touch
```

```
❯ ls -lu
합계 20
-rw-rw-r-- 1 schan schan     0  6월 13 04:36 1.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:37 2.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:29 3.txt
-rw-rw-r-- 1 schan schan     0  1월  1 00:00 4.txt
-rw-rw-r-- 1 schan schan     0  1월  1 00:00 5.txt
-rw-rw-r-- 1 schan schan     0  1월  1 00:00 6.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:30 7.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:30 8.txt
-rw-rw-r-- 1 schan schan     0  6월 13 04:30 9.txt
-rwxrwxr-x 1 schan schan 16696  6월 13 04:29 touch
```

$ id -ugGnra

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <pwd.h>
#include <grp.h>
#include <string.h>

#define OPT_U (1 << 0)
#define OPT_G (1 << 1)
#define OPT_CAP_G (1 << 2)
#define OPT_R (1 << 3)
#define OPT_N (1 << 4)
#define OPT_A (1 << 5)

void print_user_info(uid_t uid, gid_t gid, int options) {
    struct passwd *pw = getpwuid(uid);
    struct group *gr = getgrgid(gid);

    if (!pw) {
        fprintf(stderr, "id: cannot find name for user ID %d\n", uid);
        return;
    }

    if (options == 0 || (options & OPT_A)) {
        printf("uid=%d(%s) gid=%d(%s)", pw->pw_uid, pw->pw_name, gr ? gr->gr_gid : gid, gr ? gr->gr_name : "unknown");

        int ngroups = 0;
        gid_t *groups = NULL;
        if (getgrouplist(pw->pw_name, pw->pw_gid, NULL, &ngroups) == -1) {
            groups = malloc(sizeof(gid_t) * ngroups);
            if (groups) {
                if (getgrouplist(pw->pw_name, pw->pw_gid, groups, &ngroups) == -1) {
                }
            }
        }

        if (ngroups > 0) {
            printf(" groups=");
            for (int i = 0; i < ngroups; i++) {
                struct group *g = getgrgid(groups[i]);
                printf("%d(%s)%s", groups[i], g ? g->gr_name : "unknown", (i == ngroups - 1) ? "" : ",");
            }
        }
        printf("\n");
        free(groups);
        return;
    }

    if (options & OPT_U) {
        if (options & OPT_N) {
            printf("%s\n", pw->pw_name);
        } else {
            printf("%d\n", pw->pw_uid);
        }
    } else if (options & OPT_G) {
        if (options & OPT_N) {
            printf("%s\n", gr ? gr->gr_name : "unknown");
        } else {
            printf("%d\n", gr ? gr->gr_gid : gid);
        }
    } else if (options & OPT_CAP_G) {
        int ngroups = 0;
        gid_t *groups = NULL;
        if (getgrouplist(pw->pw_name, pw->pw_gid, NULL, &ngroups) == -1) {
            groups = malloc(sizeof(gid_t) * ngroups);
            if (groups) {
                if (getgrouplist(pw->pw_name, pw->pw_gid, groups, &ngroups) == -1) {
                }
            }
        }
        for (int i = 0; i < ngroups; i++) {
            if (options & OPT_N) {
                struct group *g = getgrgid(groups[i]);
                printf("%s%s", g ? g->gr_name : "unknown", (i == ngroups - 1) ? "" : " ");
            } else {
                printf("%d%s", groups[i], (i == ngroups - 1) ? "" : " ");
            }
        }
        printf("\n");
        free(groups);
    } else if (options & OPT_R) {
        printf("%d\n", geteuid());
    }
}

int main(int argc, char *argv[]) {
    uid_t target_uid = geteuid();
    gid_t target_gid = getegid();
    struct passwd *pw_lookup = NULL;
    int options = 0;
    int opt;

    while ((opt = getopt(argc, argv, "ugGrna")) != -1) {
        switch (opt) {
            case 'u': options |= OPT_U; break;
            case 'g': options |= OPT_G; break;
            case 'G': options |= OPT_CAP_G; break;
            case 'r': options |= OPT_R; break;
            case 'n': options |= OPT_N; break;
            case 'a': options |= OPT_A; break;
            case '?':
                fprintf(stderr, "Usage: %s [-ugGrna] [username]\n", argv[0]);
                return 1;
            default:
                abort();
        }
    }

    if (optind < argc) {
        char *username = argv[optind];
        pw_lookup = getpwnam(username);
        if (pw_lookup) {
            target_uid = pw_lookup->pw_uid;
            target_gid = pw_lookup->pw_gid;
        } else {
            fprintf(stderr, "id: '%s': no such user\n", username);
            return 1;
        }
    }

    print_user_info(target_uid, target_gid, options);

    return 0;
}
```

$ id -ugGnra

```
❯ ./id -u
1000
❯ ./id -un
schan
❯ ./id -ur
1000
❯ ./id -urn
schan
```

```
❯ ./id -g
1000
❯ ./id -gn
schan
❯ ./id -gr
1000
❯ ./id -grn
schan
```

```
❯ ./id -G
4 24 27 30 46 100 114 1000
❯ ./id -Gn
adm cdrom sudo dip plugdev users lpadmin schan
❯ ./id -Gr
4 24 27 30 46 100 114 1000
❯ ./id -Grn
adm cdrom sudo dip plugdev users lpadmin schan
```

```
❯ ./id
uid=1000(schan) gid=1000(schan) groups=1000(schan),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),100(users),114(lp
admin)
❯ ./id -a
uid=1000(schan) gid=1000(schan) groups=1000(schan),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),100(users),114(lp
admin)
❯ ./id root
uid=0(root) gid=0(root) groups=0(root)
```

$ head -ncqv

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

#define DEFAULT_LINES 10

typedef enum {
    MODE_LINES,
    MODE_BYTES
} OutputMode;

typedef enum {
    FLAG_QUIET = 1 << 0,
    FLAG_VERBOSE = 1 << 1
} HeadFlags;

void process_file(const char *filename, int lines_to_read, long bytes_to_read, OutputMode mode, int flags, int num_files) {
    FILE *fp;
    char buffer[4096];
    size_t bytes_read;
    int lines_count = 0;
    long total_bytes_read = 0;

    if (strcmp(filename, "-") == 0) {
        fp = stdin;
    } else {
        fp = fopen(filename, "r");
        if (fp == NULL) {
            fprintf(stderr, "head: cannot open '%s' for reading: %s\n", filename, strerror(errno));
            return;
        }
    }

    if (num_files > 1 && !(flags & FLAG_QUIET)) {
        if (flags & FLAG_VERBOSE) {
            printf("==> %s <==\n", filename);
        } else {
            printf("==> %s <==\n", filename);
        }
    } else if (num_files == 1 && (flags & FLAG_VERBOSE)) {
        printf("==> %s <==\n", filename);
    }

    if (mode == MODE_LINES) {
        while (lines_count < lines_to_read && (bytes_read = fread(buffer, 1, sizeof(buffer), fp)) > 0) {
            for (size_t i = 0; i < bytes_read; ++i) {
                putchar(buffer[i]);
                if (buffer[i] == '\n') {
                    lines_count++;
                    if (lines_count >= lines_to_read) {
                        break;
                    }
                }
            }
        }
    } else { // MODE_BYTES
        while (total_bytes_read < bytes_to_read && (bytes_read = fread(buffer, 1, sizeof(buffer), fp)) > 0) {
            long remaining_bytes = bytes_to_read - total_bytes_read;
            size_t write_bytes = (bytes_read < remaining_bytes) ? bytes_read : (size_t)remaining_bytes;
            fwrite(buffer, 1, write_bytes, stdout);
            total_bytes_read += write_bytes;
        }
    }

    if (fp != stdin) {
        fclose(fp);
    }
}
```

```c
int main(int argc, char *argv[]) {
    int lines_to_read = DEFAULT_LINES;
    long bytes_to_read = -1;
    OutputMode mode = MODE_LINES;
    int flags = 0;
    int opt;

    while ((opt = getopt(argc, argv, "n:c:qv")) != -1) {
        switch (opt) {
            case 'n':
                lines_to_read = atoi(optarg);
                if (lines_to_read < 0) {
                    fprintf(stderr, "head: invalid number of lines: '%s'\n", optarg);
                    return 1;
                }
                mode = MODE_LINES;
                break;
            case 'c':
                bytes_to_read = atol(optarg);
                if (bytes_to_read < 0) {
                    fprintf(stderr, "head: invalid number of bytes: '%s'\n", optarg);
                    return 1;
                }
                mode = MODE_BYTES;
                break;
            case 'q':
                flags |= FLAG_QUIET;
                break;
            case 'v':
                flags |= FLAG_VERBOSE;
                break;
            case '?':
                fprintf(stderr, "Usage: %s [-n lines | -c bytes] [-qv] [FILE...]\n", argv[0]);
                return 1;
            default:
                abort();
        }
    }

    int num_files = argc - optind;

    if (num_files == 0) {
        process_file("-", lines_to_read, bytes_to_read, mode, flags, 1);
    } else {
        for (int i = optind; i < argc; i++) {
            process_file(argv[i], lines_to_read, bytes_to_read, mode, flags, num_files);
        }
    }

    return 0;
}
```

$ head -ncqv

```
❯ ./head head.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

#define DEFAULT_LINES 10

typedef enum {
    MODE_LINES,
```

```
❯ ./head -v head.c
==> head.c <==
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

#define DEFAULT_LINES 10

typedef enum {
    MODE_LINES,
```

```
❯ ./head -n 3 head.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
❯ ./head -c 8 head.c
#include%
```

```
❯ ./head -n 3 -q head.c id.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
```

$ tail -ncqv

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

#define DEFAULT_LINES 10
#define BUFFER_SIZE 4096

typedef enum {
    MODE_LINES,
    MODE_BYTES
} OutputMode;

typedef enum {
    FLAG_QUIET = 1 << 0,
    FLAG_VERBOSE = 1 << 1
} TailFlags;

void process_file(const char *filename, int lines_to_read, long bytes_to_read, OutputMode mode, int flags, int num_files) {
    FILE *fp;
    long file_size;
    long start_offset;
    char buffer[BUFFER_SIZE];
    size_t bytes_read;
    int lines_found = 0;

    if (strcmp(filename, "-") == 0) {
        fp = stdin;
        if (mode == MODE_BYTES) {
            long total_read = 0;
            while ((bytes_read = fread(buffer, 1, sizeof(buffer), fp)) > 0) {
                total_read += bytes_read;
            }
            if (bytes_to_read > total_read) bytes_to_read = total_read;
            fprintf(stderr, "tail: -c option on standard input is not fully supported in this simplified version.\n");
            return;
        }
    } else {
        fp = fopen(filename, "rb");
        if (fp == NULL) {
            fprintf(stderr, "tail: cannot open `%s' for reading: %s\n", filename, strerror(errno));
            return;
        }

        fseek(fp, 0, SEEK_END);
        file_size = ftell(fp);
    }

    if (num_files > 1 && !(flags & FLAG_QUIET)) {
        if (flags & FLAG_VERBOSE) {
            printf("==> %s <==\n", filename);
        } else {
            printf("==> %s <==\n", filename);
        }
    } else if (num_files == 1 && (flags & FLAG_VERBOSE)) {
        printf("==> %s <==\n", filename);
    }
```

```c
    if (mode == MODE_LINES) {
        if (fp == stdin) {
            while ((bytes_read = fread(buffer, 1, sizeof(buffer), fp)) > 0) {
                fwrite(buffer, 1, bytes_read, stdout);
            }
        } else {
            start_offset = file_size;
            for (long i = file_size - 1; i >= 0 && lines_found <= lines_to_read; --i) {
                fseek(fp, i, SEEK_SET);
                char c = fgetc(fp);
                if (c == '\n') {
                    lines_found++;
                }
                start_offset = i;
            }

            if (lines_found > lines_to_read) {
                start_offset++;
            }

            fseek(fp, start_offset, SEEK_SET);

            while ((bytes_read = fread(buffer, 1, sizeof(buffer), fp)) > 0) {
                fwrite(buffer, 1, bytes_read, stdout);
            }
        }
    } else {
        if (fp == stdin) {
            fprintf(stderr, "tail: -c option on standard input is not fully supported in this simplified version.\n");
            return;
        }
        start_offset = file_size - bytes_to_read;
        if (start_offset < 0) {
            start_offset = 0;
        }
        fseek(fp, start_offset, SEEK_SET);

        while ((bytes_read = fread(buffer, 1, sizeof(buffer), fp)) > 0) {
            fwrite(buffer, 1, bytes_read, stdout);
        }
    }

    if (fp != stdin) {
        fclose(fp);
    }
}
```

```c
int main(int argc, char *argv[]) {
    int lines_to_read = DEFAULT_LINES;
    long bytes_to_read = -1;
    OutputMode mode = MODE_LINES;
    int flags = 0;
    int opt;

    while ((opt = getopt(argc, argv, "n:c:qv")) != -1) {
        switch (opt) {
            case 'n':
                lines_to_read = atoi(optarg);
                if (lines_to_read < 0) {
                    fprintf(stderr, "tail: invalid number of lines: `%s'\n", optarg);
                    return 1;
                }
                mode = MODE_LINES;
                break;
            case 'c':
                bytes_to_read = atol(optarg);
                if (bytes_to_read < 0) {
                    fprintf(stderr, "tail: invalid number of bytes: `%s'\n", optarg);
                    return 1;
                }
                mode = MODE_BYTES;
                break;
            case 'q':
                flags |= FLAG_QUIET;
                break;
            case 'v':
                flags |= FLAG_VERBOSE;
                break;
            case '?':
                fprintf(stderr, "Usage: %s [-n lines | -c bytes] [-qv] [FILE...]\n", argv[0]);
                return 1;
            default:
                abort();
        }
    }

    int num_files = argc - optind;

    if (num_files == 0) {
        process_file("-", lines_to_read, bytes_to_read, mode, flags, 1);
    } else {
        for (int i = optind; i < argc; i++) {
            process_file(argv[i], lines_to_read, bytes_to_read, mode, flags, num_files);
        }
    }

    return 0;
}
```

$ tail -ncqv

```
❯ ./tail tail.c
    if (num_files == 0) {
        process_file("-", lines_to_read, bytes_to_read, mode, flags, 1);
    } else {
        for (int i = optind; i < argc; i++) {
            process_file(argv[i], lines_to_read, bytes_to_read, mode, flags, num_files);
        }
    }

    return 0;
}
❯ ./tail -v tail.c
==> tail.c <==
    if (num_files == 0) {
        process_file("-", lines_to_read, bytes_to_read, mode, flags, 1);
    } else {
        for (int i = optind; i < argc; i++) {
            process_file(argv[i], lines_to_read, bytes_to_read, mode, flags, num_files);
        }
    }

    return 0;
}
```

```
❯ ./tail -n 3 tail.c

    return 0;
}
❯ ./tail -c 12 tail.c
return 0;
}
❯ ./tail -n 3 -q tail.c head.c

    return 0;
}

    return 0;
}
```

$ sleep

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

int main(int argc, char *argv[]) {
    if (argc < 2) {
        fprintf(stderr, "Usage: %s <number>\n", argv[0]);
        return 1;
    }

    errno = 0;
    long seconds = strtol(argv[1], NULL, 10);

    if (errno != 0 || seconds < 0) {
        fprintf(stderr, "sleep: invalid time interval '%s'\n", argv[1]);
        return 1;
    }

    if (sleep(seconds) != 0) {
        fprintf(stderr, "sleep: sleep interrupted\n");
        return 1;
    }

    return 0;
}
```

```
❯ ./sleep 3
```

$ whoami

```c
#include <stdio.h>
#include <unistd.h>
#include <pwd.h>
#include <sys/types.h>
#include <errno.h>
#include <string.h>

int main(void) {
    uid_t euid = geteuid();
    struct passwd *pw = getpwuid(euid);

    if (pw == NULL) {
        fprintf(stderr, "whoami: cannot find name for user ID %d: %s\n", euid, strerror(errno));
        return 1;
    }

    printf("%s\n", pw->pw_name);

    return 0;
}
```

```
❯ ./whoami
schan
```

$ cat -nbetETAsv

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <ctype.h>

#define BUFFER_SIZE 4096

#define OPT_N (1 << 0)
#define OPT_B (1 << 1)
#define OPT_E (1 << 2)
#define OPT_T (1 << 3)
#define OPT_A (1 << 4)
#define OPT_S (1 << 5)
#define OPT_V (1 << 6)

void process_file(const char *filename, int flags) {
    FILE *fp;
    char buffer[BUFFER_SIZE];
    size_t bytes_read;
    int line_number = 1;
    int prev_char = '\n';
    int blank_lines_count = 0;

    if (strcmp(filename, "-") == 0) {
        fp = stdin;
    } else {
        fp = fopen(filename, "r");
        if (fp == NULL) {
            fprintf(stderr, "cat: %s: %s\n", filename, strerror(errno));
            return;
        }
    }

    while ((bytes_read = fread(buffer, 1, sizeof(buffer), fp)) > 0) {
        for (size_t i = 0; i < bytes_read; ++i) {
            unsigned char current_char = buffer[i];

            if (flags & OPT_S) {
                if (current_char == '\n') {
                    blank_lines_count++;
                } else {
                    blank_lines_count = 0;
                }
                if (blank_lines_count > 1) {
                    prev_char = current_char;
                    continue;
                }
            }

            if (prev_char == '\n') {
                if ((flags & OPT_N) && !(flags & OPT_B)) {
                    printf("%6d\t", line_number++);
                } else if ((flags & OPT_B) && current_char != '\n') {
                    printf("%6d\t", line_number++);
                }
            }

            if (current_char == '\t') {
                if (flags & OPT_T) {
                    fputs("^I", stdout);
                } else {
                    putchar(current_char);
                }
            } else if (current_char == '\n') {
                if (flags & OPT_E) {
                    putchar('$');
                }
                putchar(current_char);
            } else if (flags & OPT_V) {
                if (iscntrl(current_char)) {
                    if (current_char == 127) {
                        fputs("^?", stdout);
                    } else {
                        putchar('^');
                        putchar(current_char + '@');
                    }
                } else if (current_char >= 128) {
                    putchar('M');
                    putchar('-');
                    if (iscntrl(current_char - 128)) {
                        if ((current_char - 128) == 127) {
                            fputs("^?", stdout);
                        } else {
                            putchar('^');
                            putchar((current_char - 128) + '@');
                        }
                    } else {
                        putchar(current_char - 128);
                    }
                } else {
                    putchar(current_char);
                }
            } else {
                putchar(current_char);
            }
            prev_char = current_char;
        }
    }

    if (fp != stdin) {
        fclose(fp);
    }
}

int main(int argc, char *argv[]) {
    int flags = 0;
    int opt;

    while ((opt = getopt(argc, argv, "nbETAsvet")) != -1) {
        switch (opt) {
            case 'n': flags |= OPT_N; break;
            case 'b': flags |= OPT_B; break;
            case 'E': flags |= OPT_E; break;
            case 'T': flags |= OPT_T; break;
            case 'A': flags |= OPT_A; break;
            case 's': flags |= OPT_S; break;
            case 'v': flags |= OPT_V; break;
            case 'e': flags |= (OPT_E | OPT_V); break;
            case 't': flags |= (OPT_T | OPT_V); break;
            case '?':
                fprintf(stderr, "Usage: %s [-nbeTAs] [FILE ...]\n", argv[0]);
                return 1;
            default:
                abort();
        }
    }

    if (flags & OPT_A) {
        flags |= (OPT_V | OPT_E | OPT_T);
    }
    if ((flags & OPT_B) && (flags & OPT_N)) {
        flags &= ~OPT_N;
    }

    if (optind == argc) {
        process_file("-", flags);
    } else {
        for (int i = optind; i < argc; i++) {
            process_file(argv[i], flags);
        }
    }

    return 0;
}
```

$ cat -nbetETAsv

```
❯ ./cat pwd.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <limits.h>    // PATH_MAX

int main() {


        char path[PATH_MAX];
        getcwd(path, PATH_MAX);
        printf("%s\n", path);
        return 0;


}
```

```
❯ ./cat -n pwd.c
     1  #include <stdio.h>
     2  #include <stdlib.h>
     3  #include <unistd.h>
     4  #include <limits.h>    // PATH_MAX
     5
     6  int main() {
     7
     8
     9          char path[PATH_MAX];
    10          getcwd(path, PATH_MAX);
    11          printf("%s\n", path);
    12          return 0;
    13
    14
    15  }
    16
```

```
❯ ./cat -b pwd.c
     1  #include <stdio.h>
     2  #include <stdlib.h>
     3  #include <unistd.h>
     4  #include <limits.h>    // PATH_MAX

     5  int main() {


     6          char path[PATH_MAX];
     7          getcwd(path, PATH_MAX);
     8          printf("%s\n", path);
     9          return 0;


    10  }
```

```
❯ ./cat -T pwd.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <limits.h>    // PATH_MAX

int main() {


^Ichar path[PATH_MAX];
^Igetcwd(path, PATH_MAX);
^Iprintf("%s\n", path);
^Ireturn 0;


}
```

```
❯ ./cat -E pwd.c
#include <stdio.h>$
#include <stdlib.h>$
#include <unistd.h>$
#include <limits.h>    // PATH_MAX$
$
int main() {$
$
$
        char path[PATH_MAX];$
        getcwd(path, PATH_MAX);$
        printf("%s\n", path);$
        return 0;$
$
$
}$
$
```

```
❯ ./cat -A pwd.c
#include <stdio.h>$
#include <stdlib.h>$
#include <unistd.h>$
#include <limits.h>    // PATH_MAX$
$
int main() {$
$
$
^Ichar path[PATH_MAX];$
^Igetcwd(path, PATH_MAX);$
^Iprintf("%s\n", path);$
^Ireturn 0;$
$
$
}$
$
```

```
❯ ./cat -s pwd.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <limits.h>    // PATH_MAX
int main() {
        char path[PATH_MAX];
        getcwd(path, PATH_MAX);
        printf("%s\n", path);
        return 0;
}
```

```
❯ ./cat -v pwd ¦ tail -c 30
^@^@^@^@^@^@^@^@^@^@^@^@^@^@%
❯ ./cat -e pwd ¦ tail -c 30
^@^@^@^@^@^@^@^@^@^@^@^@^@^@%
❯ ./cat -t pwd ¦ tail -c 30
^@^@^@^@^@^@^@^@^@^@^@^@^@^@%
```

$ mkdir -pvm

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/stat.h>
#include <errno.h>

int verbose_flag = 0;

int make_dir(char *path, mode_t mode, int parents) {
    if (parents) {
        char *p = path;
        char *slash = NULL;
        mode_t old_umask = umask(0);

        while (*p != '\0') {
            if (*p == '/') {
                slash = p;
                *slash = '\0';
                if (strlen(path) > 0) {
                    if (mkdir(path, mode) == -1 && errno != EEXIST) {
                        fprintf(stderr, "mkdir: cannot create directory '%s': %s\n", path, strerror(errno));
                        umask(old_umask);
                        return -1;
                    } else if (errno != EEXIST && verbose_flag) {
                        fprintf(stdout, "mkdir: created directory '%s'\n", path);
                    }
                }
                *slash = '/';
            }
            p++;
        }

        if (mkdir(path, mode) == -1 && errno != EEXIST) {
            fprintf(stderr, "mkdir: cannot create directory '%s': %s\n", path, strerror(errno));
            umask(old_umask);
            return -1;
        } else if (errno != EEXIST && verbose_flag) {
            fprintf(stdout, "mkdir: created directory '%s'\n", path);
        }
        umask(old_umask);
    } else {
        if (mkdir(path, mode) == -1 && errno != EEXIST) {
            fprintf(stderr, "mkdir: cannot create directory '%s': %s\n", path, strerror(errno));
            return -1;
        } else if (verbose_flag) {
            fprintf(stdout, "mkdir: created directory '%s'\n", path);
        }
    }
    return 0;
}
```

```c
int main(int argc, char *argv[]) {
    int opt;
    int parents_flag = 0;
    mode_t mode = S_IRWXU | S_IRGRP | S_IXGRP | S_IROTH | S_IXOTH;

    while ((opt = getopt(argc, argv, "pm:v")) != -1) {
        switch (opt) {
            case 'p':
                parents_flag = 1;
                break;
            case 'm':
                mode = (mode_t)strtol(optarg, NULL, 8);
                break;
            case 'v':
                verbose_flag = 1;
                break;
            default:
                fprintf(stderr, "Usage: %s [-p] [-m mode] [-v] directory ...\n", argv[0]);
                exit(EXIT_FAILURE);
        }
    }

    if (optind >= argc) {
        fprintf(stderr, "Usage: %s [-p] [-m mode] [-v] directory ...\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    for (int i = optind; i < argc; i++) {
        make_dir(argv[i], mode, parents_flag);
    }

    return EXIT_SUCCESS;
}
```

$ mkdir -pvm

```
❯ ll
합계 20K
-rwxrwxr-x 1 schan schan 17K  6월 13 06:54 mkdir
❯ ./mkdir A
❯ ./mkdir -m 700 B
❯ mkdir -pv C/D/E
mkdir: 'C' 디렉터리를 만들었습니다
mkdir: 'C/D' 디렉터리를 만들었습니다
mkdir: 'C/D/E' 디렉터리를 만들었습니다
```

```
❯ tree C
C
└── D
    └── E

3 directories, 0 files
❯ ll
합계 32K
drwxr-xr-x 2 schan schan 4.0K  6월 13 06:55 A
drwx------ 2 schan schan 4.0K  6월 13 06:55 B
drwxrwxr-x 3 schan schan 4.0K  6월 13 06:55 C
-rwxrwxr-x 1 schan schan  17K  6월 13 06:54 mkdir
```

$ rmdir -pv

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

int verbose_flag = 0;

int remove_dir(char *path, int parents) {
    if (rmdir(path) == -1) {
        if (errno != ENOTEMPTY && errno != EEXIST) {
            fprintf(stderr, "rmdir: failed to remove '%s': %s\n", path, strerror(errno));
            return -1;
        } else if (errno == ENOTEMPTY) {
            fprintf(stderr, "rmdir: failed to remove '%s': Directory not empty\n", path);
            return -1;
        }
    } else {
        if (verbose_flag) {
            fprintf(stdout, "rmdir: removed directory '%s'\n", path);
        }
    }

    if (parents) {
        char *parent_path = strdup(path);
        if (parent_path == NULL) {
            fprintf(stderr, "rmdir: memory allocation failed\n");
            return -1;
        }

        char *last_slash = strrchr(parent_path, '/');
        if (last_slash != NULL) {
            *last_slash = '\0';
            if (strlen(parent_path) > 0) {
                if (remove_dir(parent_path, 1) != 0) {
                    free(parent_path);
                    return -1;
                }
            }
        }
        free(parent_path);
    }
    return 0;
}
```

```c
int main(int argc, char *argv[]) {
    int opt;
    int parents_flag = 0;

    while ((opt = getopt(argc, argv, "pv")) != -1) {
        switch (opt) {
            case 'p':
                parents_flag = 1;
                break;
            case 'v':
                verbose_flag = 1;
                break;
            default:
                fprintf(stderr, "Usage: %s [-p] [-v] directory ...\n", argv[0]);
                exit(EXIT_FAILURE);
        }
    }

    if (optind >= argc) {
        fprintf(stderr, "Usage: %s [-p] [-v] directory ...\n", argv[0]);
        exit(EXIT_FAILURE);
    }

    for (int i = optind; i < argc; i++) {
        remove_dir(argv[i], parents_flag);
    }

    return EXIT_SUCCESS;
}
```

$ rmdir -pv

```
❯ tree .
.
├── A
├── B
├── C
│   └── D
│       └── E
├── mkdir
└── rmdir

6 directories, 2 files
```

```
❯ ./rmdir A
❯ ./rmdir -v B
rmdir: removed directory 'B'
❯ ./rmdir -pv C/D/E
rmdir: removed directory 'C/D/E'
rmdir: removed directory 'C/D'
rmdir: removed directory 'C'
❯ tree .
.
├── mkdir
└── rmdir

1 directory, 2 files
```

$ clear

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

int main() {
    printf("\033[H\033[2J");
    return EXIT_SUCCESS;
}
```

$ clear

```
❯ l
.  cat    clear    head    id    mkdir    mkdir_test  pwd.c  rmdir.c  sleep.c  tail.c  touch.c     whoami
.. cat.c  clear.c  head.c  id.c  mkdir.c  pwd                rmdir    sleep    tail    touch       touch_test  whoami.c
❯ l
.  cat    clear    head    id    mkdir    mkdir_test  pwd.c  rmdir.c  sleep.c  tail.c  touch.c     whoami
.. cat.c  clear.c  head.c  id.c  mkdir.c  pwd                rmdir    sleep    tail    touch       touch_test  whoami.c
❯ l
.  cat    clear    head    id    mkdir    mkdir_test  pwd.c  rmdir.c  sleep.c  tail.c  touch.c     whoami
.. cat.c  clear.c  head.c  id.c  mkdir.c  pwd                rmdir    sleep    tail    touch       touch_test  whoami.c
❯ l
.  cat    clear    head    id    mkdir    mkdir_test  pwd.c  rmdir.c  sleep.c  tail.c  touch.c     whoami
.. cat.c  clear.c  head.c  id.c  mkdir.c  pwd                rmdir    sleep    tail    touch       touch_test  whoami.c

 ~/z  ./clear                                                          ✔  at 07:19:17
```

```
 ~/z                                                                    ✔  at 07:19:59
```

$ date -duRI

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <unistd.h>
#include <sys/stat.h>
#include <errno.h>

#define DEFAULT_FORMAT "%a %b %d %H:%M:%S %Z %Y"
#define RFC2822_FORMAT "%a, %d %b %Y %H:%M:%S %z"
#define ISO8601_FORMAT "%Y-%m-%d"
#define ISO8601_FORMAT_DATE_TIME "%Y-%m-%dT%H:%M:%S%z"

int main(int argc, char *argv[]) {
    int opt;
    char *format_string = DEFAULT_FORMAT;
    time_t current_time;
    struct tm *timeinfo;
    int use_utc = 0;
    int use_date_string = 0;
    int use_reference_file = 0;
    char *date_string_arg = NULL;
    char *reference_file_arg = NULL;
    int iso_format_flag = 0;

    while ((opt = getopt(argc, argv, "d:r:uRI")) != -1) {
        switch (opt) {
            case 'd':
                use_date_string = 1;
                date_string_arg = optarg;
                break;
            case 'r':
                use_reference_file = 1;
                reference_file_arg = optarg;
                break;
            case 'u':
                use_utc = 1;
                break;
            case 'R':
                format_string = RFC2822_FORMAT;
                break;
            case 'I':
                iso_format_flag = 1;
                format_string = ISO8601_FORMAT;
                break;
            default:
                fprintf(stderr, "Usage: %s [-u] [-R] [-I] [-d STRING] [-r FILE] [+FORMAT]\n", argv[0]);
                exit(EXIT_FAILURE);
        }
    }

    if (optind < argc && argv[optind][0] == '+') {
        format_string = argv[optind] + 1;
        iso_format_flag = 0;
    }

    if (use_date_string) {
        time(&current_time);

        if (strcmp(date_string_arg, "yesterday") == 0) {
            current_time -= (24 * 60 * 60);
        } else if (strcmp(date_string_arg, "tomorrow") == 0) {
            current_time += (24 * 60 * 60);
        } else {
            fprintf(stderr, "date: invalid date string '%s' (simple parser only supports 'yesterday' and 'tomorrow')\n", date_string_arg);
            exit(EXIT_FAILURE);
        }
    } else if (use_reference_file) {
        struct stat file_stat;
        if (stat(reference_file_arg, &file_stat) == -1) {
            fprintf(stderr, "date: cannot stat '%s': %s\n", reference_file_arg, strerror(errno));
            exit(EXIT_FAILURE);
        }
        current_time = file_stat.st_mtime;
    } else {
        time(&current_time);
    }

    if (use_utc) {
        timeinfo = gmtime(&current_time);
    } else {
        timeinfo = localtime(&current_time);
    }

    char buffer[256];
    strftime(buffer, sizeof(buffer), format_string, timeinfo);
    printf("%s\n", buffer);

    return EXIT_SUCCESS;
}
```

$ date -duRI

```
❯ ./date
Fri Jun 13 07:41:18 KST 2025
❯ ./date -I
2025-06-13
❯ ./date -R
Fri, 13 Jun 2025 07:41:25 +0900
❯ ./date -u
Thu Jun 12 22:41:31 GMT 2025
❯ ./date -d yesterday
Thu Jun 12 07:41:59 KST 2025
```

총 점수 : 15 / 15

- 누락 X

깃허브 + 명령어 = 6 + 15 = 21