

깃 허브 정리 및 리눅스 명령어 구현

- <https://github.com/8352341/systempgm>
- 2020875015
- 김상훈


















목차

깃허브 정리

리눅스
명령어 구현

깃 허브 정리

 8352341 Update README.md bffe162 · 4 days ago 		
 0307	Update readme.md	3 months ago
 0314	Create readme.md	3 months ago
 0321	Update readme.md	2 months ago
 0328	Update readme.md	2 months ago
 0404	Update readme.md	2 months ago
 0411	Update readme.md	2 months ago
 0418	Update readme.md	last month
 0502	Update readme.md	last month
 0509	Update readme.md	last month
 0516	Update readme.md	3 weeks ago
 0523	Update readme.md	2 weeks ago
 0530	Update readme.md	last week
 README.md	Update README.md	4 days ago

History for [systempgm](#) / 0314 on [main](#) All users All time

Commits on Mar 27, 2025

Create readme.md

8352341 authored on Mar 27

Verified

51120c8



History for [systempgm](#) / 0321 on [main](#) All users All time

Commits on Apr 3, 2025

Update readme.md

8352341 authored on Apr 3

Verified

07fa66b



History for [systempgm](#) / 0307 on [main](#) All users All time

Commits on Mar 27, 2025

Update readme.md

8352341 authored on Mar 27

Verified

a2c7875



History for [systempgm](#) / 0404 on [main](#) All users All time

Commits on Apr 10, 2025

Update readme.md

8352341 authored on Apr 10

Verified

5b1d680



History for [systempgm](#) / 0411 on [main](#) All users All time

Commits on Apr 17, 2025

Update readme.md

8352341 authored on Apr 17

Verified

f914a7a



History for [systempgm](#) / 0328 on [main](#) All users All time

Commits on Apr 3, 2025

Update readme.md

8352341 authored on Apr 3

Verified

40f90a8







0307~0411

History for [systempgm](#) / 0509 on [main](#) [All users](#) [All time](#)

Commits on May 11, 2025

Update readme.md




Verified 76f37c5   


 8352341 authored on May 11

History for [systempgm](#) / 0502 on [main](#) [All users](#) [All time](#)

Commits on May 8, 2025

Update readme.md




Verified d32d54b   


 8352341 authored on May 8

History for [systempgm](#) / 0530 on [main](#) [All users](#) [All time](#)

Commits on Jun 5, 2025

Update readme.md




Verified 4d153aa   


 8352341 authored last week

History for [systempgm](#) / 0523 on [main](#) [All users](#) [All time](#)

Commits on May 28, 2025

Update readme.md




Verified 5f9c81a   


 8352341 authored 2 weeks ago

History for [systempgm](#) / 0418 on [main](#) [All users](#) [All time](#)

Commits on May 8, 2025

Update readme.md




Verified f155f75   


 8352341 authored on May 8

History for [systempgm](#) / 0516 on [main](#) [All users](#) [All time](#)

Commits on May 22, 2025

Update readme.md

Verified 9ce8105   

 8352341 authored 3 weeks ago

0418~0530

점수:13점

- 2주차(0314) 내용을 업로드 하지 못했습니다.
- 3주차(0321),7주차(0418) 업로드를 늦게 하였습니다.
- 이런 이유로 각각 -1,-0.5*2해서 -2 점을 주었습니다.

리눅스 명령어 구현

- 1.Hostname
- 2.Uname
- 3.Ls
- 4.Cat
- 5.Ps
- 6.Df
- 7.who
- 7.rm
- 8.echo
- 9.Grep
- 10.sort

hostname

- 현재 접속한 컴퓨터(서버)의 이름을 출력하는 명령어입니다.
- Gethostname()을 통해 호스트 이름을 읽어와 출력합니다.

```
ubuntu > home > kim > c c_hostname.c
1  #include <stdio.h>
2  #include <unistd.h> // for gethostname
3  #include <limits.h> // for HOST_NAME_MAX
4
5  int main() {
6      char host_name[HOST_NAME_MAX + 1]; // null문자를 저장하기 위한 공간 생성
7
8      // gethostname 시스템 호출을 사용하여 호스트 이름을 얻어옴
9      if (gethostname(host_name, sizeof(host_name)) == -1) {
10         perror("gethostname"); // 에러 발생 시 메시지 출력
11         return 1;
12     }
13
14     printf("%s\n", host_name);
15     return 0;
16 }
```

```
kim@DESKTOP-00NR3HJ:~$ ./hostname
DESKTOP-00NR3HJ
```


Uname

- Uname: 현재 운영체제의 정보를 표시합니다.
- -a : 모든 정보를 표시합니다.
- -s : 시스템 이름을 출력합니다.
- -n : 네트워크 호스트 이름을 출력합니다.
- -r : 커널 릴리즈 버전을 출력합니다.
- -v : 커널 버전을 출력합니다.
- -m : 머신 하드웨어 이름을 출력합니다.

```
kim@DESKTOP-00NR3HJ:~$ ./uname -a
Linux DESKTOP-00NR3HJ 5.15.167.4-microsoft-standard-WSL2 #1 SMP Tue Nov 5 00:21:55 UTC 2024 x86_64
kim@DESKTOP-00NR3HJ:~$ ./uname -s
Linux
kim@DESKTOP-00NR3HJ:~$ ./uname -n
DESKTOP-00NR3HJ
kim@DESKTOP-00NR3HJ:~$ ./uname -r
5.15.167.4-microsoft-standard-WSL2
kim@DESKTOP-00NR3HJ:~$ ./uname -v
#1 SMP Tue Nov 5 00:21:55 UTC 2024
kim@DESKTOP-00NR3HJ:~$ ./uname -m
x86_64
```

Uname-a,-s,-n,-r,-v,-m

- s,n,r,v,m의 변수를 생성후
- Uname()로 정보를 가져오고a,s,n,r,v,m 옵션을 만들어 옵션에 따라 변수들을 활성화 합니다.
- 활성화 된 변수에 따라 uname()으로 가져온 정보를 출력합니다..

```
24 // ... getopt을 활용한 특별한 옵션 처리 ...
25 // "asnrvm"은 -a, -s, -n, -r, -v, -m 옵션을 모두 허용한다는 의미
26 while ((opt = getopt(argc, argv, "asnrvm")) != -1) {
27     switch (opt) {
28         case 'a': // -a 옵션은 모든 플래그를 활성화
29             show_s = 1;
30             show_n = 1;
31             show_r = 1;
32             show_v = 1;
33             show_m = 1;
34             break;
35         case 's':
36             show_s = 1;
37             break;
38         case 'n':
39             show_n = 1;
40             break;
41         case 'r':
42             show_r = 1;
43             break;
44         case 'v':
45             show_v = 1;
46             break;
47         case 'm':
48             show_m = 1;
49             break;
50         case '?': // 인식할 수 없는 옵션 처리
51             fprintf(stderr, "사용법: %s [-a | -s | -n | -r | -v | -m]... \n", a
52             exit(EXIT_FAILURE);
53         default:
54             exit(EXIT_FAILURE);
55     }
56 }
```

```
1 #include <stdio.h>
2 #include <sys/utsname.h> // for uname
3 #include <unistd.h> // for getopt
4 #include <stdlib.h> // for exit
5
6 int main(int argc, char *argv[]) {
7     struct utsname uts_info; // uname() 함수로 채워질 시스템
8
9     // 출력할 정보를 결정하는 플래그 변수들
10    int show_s = 0; // sysname
11    int show_n = 0; // nodename
12    int show_r = 0; // release
13    int show_v = 0; // version
14    int show_m = 0; // machine
15
16    int opt;
17
18    // 먼저 uname 시스템 호출로 모든 정보를 가져옴
19    if (uname(&uts_info) == -1) {
20        perror("uname");
21        return 1;
22    }
23 }
```

```
68 printf("%s", uts_info.sysname);
69 space_needed = 1;
70 }
71 if (show_n) {
72     if (space_needed) printf(" ");
73     printf("%s", uts_info.nodename);
74     space_needed = 1;
75 }
76 if (show_r) {
77     if (space_needed) printf(" ");
78     printf("%s", uts_info.release);
79     space_needed = 1;
80 }
81 if (show_v) {
82     if (space_needed) printf(" ");
83     printf("%s", uts_info.version);
84     space_needed = 1;
85 }
86 if (show_m) {
87     if (space_needed) printf(" ");
```

Ls

- ls : 현재 디렉토리의 파일 이름을 출력합니다.
- -a : 숨김 파일까지 모두 출력합니다.
- -l : 파일의 상세 정보를 출력합니다.
- -R : 파일의 하위 디렉토리까지 출력합니다.
- -t : 수정 시간 기준으로 정렬합니다.
- -S : 파일 크기 기준으로 정렬합니다.

```
DESKTOP-00NR3HJ: $ ./ls -R
test.c test123 hostname swap uname
forkwait practice ls waitpid c_hostname
fork3 fork2.c c_uname.c list.c waitpid
hello.c fork3.c mode mode.c practice.c
swap.c hello listc fork1.c fork2 fork2.c
DESKTOP-00NR3HJ: $ ./ls -t
test.c test123 hostname swap uname
forkwait practice ls waitpid c_hostname
fork3 fork2.c c_uname.c list.c waitpid
hello.c fork3.c mode mode.c practice.c
swap.c hello listc fork1.c fork2 fork2.c
DESKTOP-00NR3HJ: $ ./ls -S
test.c test123 hostname swap uname
forkwait practice ls waitpid c_hostname
fork3 fork2.c c_uname.c list.c waitpid
hello.c fork3.c mode mode.c practice.c
swap.c hello listc fork1.c fork2 fork2.c
```

```
im@DESKTOP-00NR3HJ:~$ ./ls
fork1 test.c test123 hostname swap uname
practices forkwait practice ls waitpid c_hostname
fork3 fork2.c c_uname.c list.c waitpid
hello.c fork3.c mode mode.c practice.c
ls.c swap.c hello listc fork1.c fork2 fork2.c
im@DESKTOP-00NR3HJ:~$ ./ls -a
fork1 test.c .viminfo .test.swp .test1.s
test123 hostname swap .landscape .. .bas
ory uname practices .test.swn . .bashr
kwait practice ls waitpid c_hostname.c
file fork3 fork2.c c_uname.c list.c waitpid
hello.c fork3.c .motd.shown mode mode
ash_logout practice.c c_ls.c swap.c hel
list2.c.swp .cache listc fork1.c fork2
it.c .sudo_as_admin_successful
im@DESKTOP-00NR3HJ:~$ ./ls -l
-rwxr-xr-x 1 kim kim 16048 May 28 15:48 fork
-rw-r--r-- 1 kim kim 1730 Apr 10 09:51 test
-rwxr-xr-x 1 kim kim 16144 Apr 10 09:52 test
-rwxr-xr-x 1 kim kim 16104 Jun 12 12:36 host
-rwxr-xr-x 1 kim kim 16200 Apr 03 11:25 swap
-rwxr-xr-x 1 kim kim 16296 Jun 12 13:44 unam
-rwxr-xr-x 1 kim kim 16064 Mar 29 18:59 prac
-rwxr-xr-x 1 kim kim 16184 May 28 15:49 fork
-rwxr-xr-x 1 kim kim 16064 Mar 29 18:57 prac
-rwxr-xr-x 1 kim kim 16648 Jun 12 14:48 ls
```

ls, ls-a, ls-l

- 옵션이 없으면 기본출력
- -a는 "." 으로 시작하는 파일까지 출력하고
-l는 권한, 링크 수, 소유자 이름 등을 출력

```
93 // -a 옵션이 없으면 '.'으로 시작하는 숨김 파일은 건너뛴다.  
94 if (!show_all && entry->d_name[0] == '.') {  
95     continue;  
96 }
```

```
131 for (int i = 0; i < count; i++) {  
132     if (long_format) { // -l 옵션: 긴 형식으로 출력  
133         print_long_format(&files[i]);  
134     }
```

```
135 } else { // 기본 출력  
136     printf("%s\t", files[i].name);  
137 }
```

```
37 void print_permissions(mode_t mode) {  
38     // 1. 파일 종류 출력 (디렉토리, 일반 파일 등)  
39     printf(S_ISDIR(mode) ? "d" : "-");  
40  
41     // 2. 소유자(User)의 권한 (읽기, 쓰기, 실행)  
42     printf((mode & S_IRUSR) ? "r" : "-");  
43     printf((mode & S_IWUSR) ? "w" : "-");  
44     printf((mode & S_IXUSR) ? "x" : "-");  
45  
46     // 3. 그룹(Group)의 권한  
47     printf((mode & S_IRGRP) ? "r" : "-");  
48     printf((mode & S_IWGRP) ? "w" : "-");  
49     printf((mode & S_IXGRP) ? "x" : "-");  
50  
51     // 4. 그 외 사용자(Others)의 권한  
52     printf((mode & S_IROTH) ? "r" : "-");  
53     printf((mode & S_IWOTH) ? "w" : "-");  
54     printf((mode & S_IXOTH) ? "x" : "-");  
55 }  
56  
57 // ls -l 형식에 맞춰 파일의 상세 정보를 출력하는 함수  
58 void print_long_format(FileInfo *f) {  
59     // 권한 출력  
60     print_permissions(f->st.st_mode);  
61  
62     // 링크 수, 소유자 이름, 그룹 이름, 파일 크기 출력  
63     printf(" %2lu", f->st.st_nlink);  
64     printf(" %-8s", getpwuid(f->st.st_uid)->pw_name);  
65     printf(" %-8s", getgrgid(f->st.st_gid)->gr_name);  
66     printf(" %8ld", f->st.st_size);  
67  
68     // 최종 수정 시간을 보기 좋은 형식으로 변환하여 출력  
69     char timebuf[20];  
70     strftime(timebuf, sizeof(timebuf), "%b %d %H:%M", localtime(&f->st.st_mtime));  
71     printf(" %s", timebuf);  
72  
73     // 파일 이름 출력  
74     printf(" %s\n", f->name);  
75 }  
76
```

ls-R,ls-t,ls-S

```
143     if (recursive) {
144         for (int i = 0; i < count; i++) {
145             // 현재 항목이 디렉터리이고, 자기 자신('.')이나 부모('.')가 아닐 경우
146             if (S_ISDIR(files[i].st.st_mode) &&
147                 strcmp(files[i].name, ".") != 0 &&
148                 strcmp(files[i].name, "..") != 0) {
149
150                 // 해당 하위 디렉터리에 대한 전체 경로 생성
151                 char sub_path[512];
152                 snprintf(sub_path, sizeof(sub_path), "%s/%s", path, files[i].name);
153
154                 // 자기 자신(list_directory 함수)을 다시 호출하여 재귀적으로 탐색
155                 list_directory(sub_path, show_all, long_format, recursive, sort_time, sort_size);
156             }
157         }
158     }
```

```
125     // 실제 출력 부분
126     // -R 옵션 사용 시, 어느 디렉터리에 대한 출력인지 명시해준다.
127     if (recursive) {
128         printf("\n%s:\n", path);
129     }
130
131     // qsort를 위한 비교 함수: 파일 크기(size) 기준 내림차순 정렬
132     // 크기가 큰 파일이 먼저 오도록 정렬한다.
133     int compare_size(const void *a, const void *b) {
134         FileInfo *fileA = (FileInfo *)a;
135         FileInfo *fileB = (FileInfo *)b;
136         // b의 크기가 더 크면 양수를 반환하여 b를 앞으로 보냄
137         return fileB->st.st_size - fileA->st.st_size;
138     }
139
140     else if (sort_size) { // -S 옵션: 크기순 정렬
141         qsort(files, count, sizeof(FileInfo), compare_size);
142     }
143
144     // (아무 옵션도 없으면 기본적으로 이름순으로 readdir이 읽은 순서대로 출력됨)
```

```
18     // qsort를 위한 비교 함수: 수정 시간(mtime) 기준 내림차순 정렬
19     // 최신 파일이 먼저 오도록 정렬한다.
20     int compare_mtime(const void *a, const void *b) {
21         FileInfo *fileA = (FileInfo *)a;
22         FileInfo *fileB = (FileInfo *)b;
23         // b의 시간이 더 크면(최신이면) 양수를 반환하여 b를 앞으로 보냄
24         return fileB->st.st_mtime - fileA->st.st_mtime;
25     }
```

```
116     // 정렬 옵션 처리
117     if (sort_time) { // -t 옵션: 시간순 정렬
118         qsort(files, count, sizeof(FileInfo), compare_mtime);
119     }
```

-R에선 재귀적으로 디렉터리 내부를 탐색해 경로이름까지 출력
-t에선 시간을 비교해 시간순으로 정렬하고
-S에선 파일크기끼리 비교해 정렬한다.

Cat

- Cat:파일의 내용을 출력하고, 입력하지 않을 경우
입력한걸 출력합니다.
- -n : 줄 번호 출력합니다.
- -s : 줄 끝 표시합니다.
- -E : 연속된 빈줄 제거합니다.

```
kim@DESKTOP-00NR3HJ:~$ ./cat cats
asd

asd

asd
kim@DESKTOP-00NR3HJ:~$ ./cat -n cats
 1  asd
 2
 3
 4  asd
 5
 6  asd
kim@DESKTOP-00NR3HJ:~$ cat -E cats
asd$
$
$
asd$
$
asd$
kim@DESKTOP-00NR3HJ:~$ cat -s cats
asd

asd

asd
```

cat, cat -n, cat -s, cat -E

- 기본 Cat을 실행하면 파일을 읽기 모드로 연다.
- -n에선 줄번호를 1씩 증가시키면서 앞에 출력한다
- -E 에선 줄 끝에 \n을 찾아 %s로 변경한다
- -s는 앞에 빈 줄일 경우 플래그를 켜고 다음 줄도 빈 줄일 경우 건너뛴다

```
72 // --- -E 옵션 (show_ends) 처리 및 최종 출력 ---
73 if (show_ends) {
74     // 줄 끝의 개행 문자(\n)를 찾아서 널 문자(\0)로 바꿔 제거한다.
75     size_t len = strlen(line);
76     if (len > 0 && line[len - 1] == '\n') {
77         line[len - 1] = '\0';
78     }
79     // 내용을 뒤에 '$'와 개행 문자를 붙여서 출력한다.
80     printf("%s$\n", line);
81 } else {
82     // -E 옵션이 없으면 읽은 그대로 출력한다.
83     printf("%s", line);
84 }
85 }
```

```
46 // fgets()를 사용하여 파일에서 한 줄씩 읽는다. 파일 끝에 도달하면 NULL을 반환
47 while (fgets(line, sizeof(line), fp) != NULL) {
48     // --- -s 옵션 (squeeze_blank) 처리 ---
49     if (squeeze_blank) {
50         // 현재 줄이 빈 줄("\n")만 있는 줄인지 확인
51         if (strcmp(line, "\n") == 0) {
52             // 이번 줄도 빈 줄이었다면, 이번 줄은 건너뛴다(continue).
53             if (prev_is_blank) {
54                 continue;
55             }
56             // 이번 줄이 첫 번째 빈 줄이라면, 상태 플래그를 켜고.
57             prev_is_blank = 1;
58         } else {
59             // 현재 줄이 빈 줄이 아니라면, 상태 플래그를 끈다.
60             prev_is_blank = 0;
61         }
62     }
63 }
```

```
31 // optind는 getopt가 처리한 마지막 인덱스의 다음을 가리킨다.
32 // 즉, 옵션이 아닌 첫 번째 파일 이름을 가리킴
33 if (optind < argc) {
34     // 파일 이름이 주어졌다면, 해당 파일을 읽기 모드("r")로 연다.
35     fp = fopen(argv[optind], "r");
36     if (!fp) {
37         perror("fopen"); // 파일 열기 실패 시 에러 메시지 출력
38         exit(EXIT_FAILURE);
39     }
40 }
41 }
```

```
65 // --- -n 옵션 (show_line_numbers) 처리 ---
66 // 실제 cat은 빈 줄이 출력되지 않아도 줄 번호는 증가시킨다.
67 // 따라서 여기서 출력 여부와 상관없이 번호를 붙인다.
68 if (show_line_numbers) {
69     printf("%6d\t", line_num++);
70 }
71 }
```

Ps

- ps : 프로세스 관련 정보를 출력합니다.
- -u : 현재 사용자 프로세스를 출력합니다.
- -f : 전체 정보(UID, PID, PPID, CMD 등)를 출력합니다.
- -e : 전체 사용자 프로세스 출력합니다.

```
kim@DESKTOP-0ONR3HJ:~$ ./ps -u
```

```
PID      CMD
286      -bash
338      /usr/lib/systemd/systemd
339      (sd-pam)
350      -bash
902      ./ps
```

```
kim@DESKTOP-0ONR3HJ:~$ ./ps -f
```

```
UID      PID      PPID      CMD
kim      286      285      -bash
kim      338      1        /usr/lib/systemd/systemd
kim      339      338      (sd-pam)
kim      350      287      -bash
kim      903      286      ./ps
```

```
kim@DESKTOP-0ONR3HJ:~$ ./ps -e
```

```
PID      CMD
1        /sbin/init
2        /init
6        plan9
49       /usr/lib/systemd/systemd-journald
93       /usr/lib/systemd/systemd-udev
104      /usr/lib/systemd/systemd-resolved
105      /usr/lib/systemd/systemd-timesyncd
164      /usr/lib/systemd/systemd-udevd
```


ps-u, ps-f, ps-u

- UID로 시작하는 문장을 통해 UID값을 읽어오고 UID를 사용해 사용자 이름을 찾는다
- -f의 옵션을 확인해 출력을 다르게 한다
- -e 옵션을 확인해 프로세스를 어디까지 보여줄지 결정한다

```
84 // --- 옵션에 따른 필터링 ---  
85 // -e 옵션이 없으면, 현재 사용자의 프로세스만 보여준다.  
86 if (!show_all_users) {  
87     uid_t process_uid = (uid_t)atoi(p_info.uid_str);  
88     if (process_uid != my_uid) {  
89         continue;  
90     }  
91 }
```

```
188 // "Uid:"로 시작하는 라인을 찾아 UID 값을 파싱한다.  
189 while (fgets(line, sizeof(line), fp)) {  
190     if (strncmp(line, "Uid:", 4) == 0) {  
191         sscanf(line, "Uid:\t%u", &uid);  
192         break;  
193     }  
194 }  
195 fclose(fp);  
196  
197 // UID를 사용자 이름으로 변환한다.  
198 struct passwd *pw = getpwuid(uid);  
199 if (pw) {  
200     strncpy(username_buf, pw->pw_name, buf_size);  
201 } else { // 해당 UID의 사용자가 없을 경우  
202     snprintf(username_buf, buf_size, "%u", uid); // 그냥 UID 숫자를 쓴다.  
203 }  
204 }
```

```
63  
64 // 헤더 출력: -f 옵션 여부에 따라 다르게 출력  
65 if (full_format) {  
66     printf("%-8s %-5s %-5s %-20s\n", "UID", "PID", "PPID", "CMD");  
67 } else {  
68     printf("%5s %-10s %-5s\n", "PID", "TTY", "CMD");  
69 }
```

Df

- df : 디스크의 전체 용량, 사용 중인 용량, 남은 용량 등을 파일 시스템 단위로 출력합니다.
- -h : 사람이 읽기 쉬운 단위로 출력
- -T : 파일시스템의 종류를 추가로 출력합니다.
- -a : 기본적으로 안 보여주는 내용까지 출력합니다.

```
kim@DESKTOP-OONR3HJ: $ ./df
Filesystem      Size      Used      Avail  Use% Mounted on
drivers          498970620  339206520  159764100  68% /usr/lib/wsl/drivers
/dev/sdc         1055762868  1937656   1000121740  0% /
/dev/sdc         1055762868  1937656   1000121740  0% /mnt/wslg/distro
rootfs           8095924    2372      8093552    0% /init
devpts           0           0           0          0% /dev/pts
binfmt_misc      0           0           0          0% /proc/sys/fs/binfmt_misc
cgrou2           0           0           0          0% /sys/fs/cgroup/unified
```

```
kim@DESKTOP-OONR3HJ: $ ./df -h
Filesystem      Size      Used      Avail  Use% Mounted on
drivers          475.9G  323.5G  152.4G   68% /usr/lib/wsl/drivers
/dev/sdc         1006.9G  1.8G  953.8G    0% /
/dev/sdc         1006.9G  1.8G  953.8G    0% /mnt/wslg/distro
rootfs           7.7G    2.3M   7.7G     0% /init
devpts           0         0         0        0% /dev/pts
binfmt_misc      0         0         0        0% /proc/sys/fs/binfmt_misc
cgrou2           0         0         0        0% /sys/fs/cgroup/unified
cgroup           0         0         0        0% /sys/fs/cgroup/cpuset
```

```
kim@DESKTOP-OONR3HJ:~$ ./df -T
Filesystem      Type      Size      Used      Avail  Use% Mounted on
drivers          9p         498970620  339196300  159774320  68% /usr/lib/wsl/drivers
/dev/sdc         ext4       1055762868  1937652   1000121744  0% /
/dev/sdc         ext4       1055762868  1937652   1000121744  0% /mnt/wslg/distro
rootfs           rootfs     8095924    2372      8093552    0% /init
devpts           devpts     0           0           0          0% /dev/pts
binfmt_misc      binfmt_misc 0           0           0          0% /proc/sys/fs/binfmt_misc
```

```
kim@DESKTOP-OONR3HJ: $ ./df -a
Filesystem      Size      Used      Avail  Use% Mounted on
none            8099416    0      8099416    0% /usr/lib/modules/5.15.167.4-microsoft-standard-WSL2
none            8099416    4      8099412    0% /mnt/wsl
drivers          498970620  339201356  159769264  68% /usr/lib/wsl/drivers
/dev/sdc         1055762868  1937656   1000121740  0% /
none            8099416    148      8099268    0% /mnt/wslg
/dev/sdc         1055762868  1937656   1000121740  0% /mnt/wslg/distro
none            8099416    0      8099416    0% /usr/lib/wsl/lib
rootfs           8095924    2372      8093552    0% /init
none            8095924    0      8095924    0% /dev
sysfs            0           0           0          0% /sys
```

Df, df-h, df-T, df-a

```
114         } else { // 기본: 1K 블록 단위로 출력 (df 기본 동작)
115             printf(" %10lu", total_size / 1024);
116             printf(" %10lu", used_size / 1024);
117             printf(" %10lu", avail_size / 1024);
118         }
```

```
106         if (show_type) {
107             printf(" %-10s", mount_entry->mnt_type);
108         }
```

```
64         if (show_type) printf(" %-10s", "Type");
```

```
70         // --- -a 옵션 필터링 ---
71         // -a 옵션이 없고, 가상/임시 파일 시스템(pseudo-filesystem)이면 건너뛰다.
72         // 보통 장치 이름이 'none'이거나, 타입이 'tmpfs', 'proc' 등인 경우가 해당된다.
73         if (!show_all && (
74             strcmp(mount_entry->mnt_type, "proc") == 0 ||
75             strcmp(mount_entry->mnt_type, "sysfs") == 0 ||
76             strcmp(mount_entry->mnt_type, "devtmpfs") == 0 ||
77             strcmp(mount_entry->mnt_type, "tmpfs") == 0 ||
78             strcmp(mount_entry->mnt_fsname, "none") == 0
79         )) {
80             continue;
81     }
```

```
110         if (human_readable) { // -h 옵션: K, M, G 단위로 출력
111             print_human_readable(total_size);
112             print_human_readable(used_size);
113             print_human_readable(avail_size);
```

```
9     /**
10     * @brief 바이트 단위 크기를 사람이 읽기 쉬운 K, M, G, T 단위로 변환하여 출력
11     * @param bytes 변환할 크기 (바이트 단위)
12     */
13     void print_human_readable(uint64_t bytes) {
14         // 0 바이트는 특별 처리
15         if (bytes == 0) {
16             printf("%5s", "0");
17             return;
18         }
19         const char *units[] = {"B", "K", "M", "G", "T", "P"};
20         int i = 0;
21         double size = bytes;
22
23         // 크기가 1024 이상이면 단위 증가
24         while (size >= 1024 && i < (sizeof(units)/sizeof(char*)-1) ) {
25             size /= 1024;
26             i++;
27         }
28         // 소수점 첫째 자리까지 5칸에 맞춰 우측 정렬하여 출력
29         printf("%5.1f%s", size, units[i]);
30     }
```

Who

- Who :현재 로그인한 사용자들을 표시합니다.
- -a : 일반 사용자 프로세스 외에도 시스템 정보들을 추가로 표시합니다.
- -b : 시스템 부팅 시간만 출력하고 종료합니다.
- -m :현재 터미널과 연결된 사용자 정보만 출력합니다.
- -T :각 사용자의 터미널 쓰기 권한 상태를 표시합니다.
- -q :간단한 형식으로 사용자 목록과 총 인원수만 출력합니다.

```
kim@DESKTOP-00NR3HJ:~$ ./who
kim      pts/1      2025-06-12 12:31
kim@DESKTOP-00NR3HJ:~$ ./who -a
          system boot      Mon Jan  6 20:48:52 3
9794059
          run-level 5      Sun Jun  8 19:17:10
53421417
kim      pts/1      2025-06-12 12:31
kim@DESKTOP-00NR3HJ:~$ ./who -b
          system boot Mon Jan  6 20:48:52 3979405
9
kim@DESKTOP-00NR3HJ:~$ ./who -m
kim@DESKTOP-00NR3HJ:~$ ./who -T
kim      - pts/1      2025-06-12 12:31
kim@DESKTOP-00NR3HJ:~$ ./who -q
kim
# users=1
```

Who, who-a, who-b

```
145     printf("%-12s ", entry->ut_line);
146     printf("%s ", time_buf);
147
148     // 호스트 정보가 있다면 출력
149     if(entry->ut_host[0] != '\0') {
150         printf("(%s)", entry->ut_host);
151     }
152
153     printf("\n");
154 }
155
```

```
116 /**
117  * @brief utmp 레코드 하나를 형식에 맞춰 출력
118  * @param entry 출력할 utmp 구조체 포인터
119  * @param show_term_status 터미널 상태(+/-) 표시 여부
120  */
121 void print_entry(struct utmp *entry, int show_term_status) {
122     char time_buf[30];
123     time_t timestamp = entry->ut_tv.tv_sec;
124
125     // 수정된 부분: localtime(&timestamp)로 올바르게 수정
126     strftime(time_buf, sizeof(time_buf), "%Y-%m-%d %H:%M", localtime(&timestamp));
127
128     // 사용자 이름이 비어있지 않은 경우에만 출력
129     if (entry->ut_user[0] != '\0') {
130         printf("%-8s ", entry->ut_user);
131     }
132 }
```

```
75 // -a 옵션 처리
76 if (show_all) {
77     // -a는 모든 유용한 정보를 출력하므로, 아래 기본 로직도 타야 함
78     if (record_type == RUN_LVL) {
79         printf("run-level %c %s", entry->ut_pid, ctime((time_t *)
80     } else if (record_type == BOOT_TIME) {
81         printf("system boot %s", ctime((time_t *)&entry->ut_tv.tv_sec));
82     } else if (record_type == DEAD_PROCESS) {
83         printf("DEAD_PROCESS\n");
84     }
85 }
86
```

```
87 // -b 옵션 처리
88 if (show_boot_time) {
89     if (record_type == BOOT_TIME) {
90         printf("system boot %s", ctime((time_t *)&entry->ut_tv.tv_sec));
91         endutent(); // 파일을 닫고
92         return 0; // 종료
93     }
94     continue;
95 }
96
```

Who-m, who-T, who-q

```
59 // -m (who am i) 옵션을 위한 현재 터미널 이름 가져오기
60 char *my_tty = NULL;
61 if (show_current_term) {
62     my_tty = ttyname(STDIN_FILENO); // 현재 표준 입력의 터미널 장치 파일 경로
63     if (my_tty) {
64         my_tty += 5; // "/dev/" 부분 건너뛰기
65     }
66 }
67
```

```
97 // -m 옵션 처리
98 if (show_current_term) {
99     if (my_tty && record_type == USER_PROCESS && strcmp(entry->ut_line, my_tty) == 0) {
100         print_entry(entry, show_term_status);
101         break; // 찾았으면 루프 종료
102     }
103     continue;
104 }
105
```

```
133 // 터미널 상태 출력 (+: 쓰기 가능, -: 쓰기 불가, ?: 확인 불가)
134 if (show_term_status) {
135     char term_path[64];
136     struct stat term_stat;
137     snprintf(term_path, sizeof(term_path), "/dev/%s", entry->ut_line);
138     if (stat(term_path, &term_stat) == 0 && (term_stat.st_mode & S_IWGRP)) {
139         printf("+ ");
140     } else {
141         printf("- ");
142     }
143 }
144
```

```
36 // -q (quick) 옵션은 다른 출력 없이 사용자 목록과 수만 보여주고 종료
37 if (show_quick) {
38     struct utmp *entry;
39     char *users[1024]; // 사용자 이름을 저장할 배열
40     int count = 0;
41
42     setutent(); // utmp 파일 열기
43     while ((entry = getutent()) != NULL) {
44         if (entry->ut_type == USER_PROCESS && entry->ut_user[0] != '\0') {
45             users[count++] = strdup(entry->ut_user);
46             if (count >= 1024) break; // 배열 오버플로우 방지
47         }
48     }
49     endutent(); // utmp 파일 닫기
50
51     print_quick_users(users, count); // 퀵 포맷으로 출력
52     // 메모리 해제
53     for (int i = 0; i < count; i++) {
54         free(users[i]);
55     }
56     return 0; // 프로그램 종료
57 }
```

```
162 void print_quick_users(char *users[], int count) {
163     for (int i = 0; i < count; i++) {
164         printf("%s ", users[i]);
165     }
166     printf("\n# users=%d\n", count);
167 }
```

rm

- Rm : 파일을 삭제합니다.
- -f : 파일을 강제로 삭제합니다.
- -i : 삭제전 확인 메시지를 띄웁니다.
- -r : 디렉터리와 그 안의 내용을 재귀적으로 삭제합니다.
- -v : 삭제한 항목을 출력합니다.

```
kim@DESKTOP-00NR3HJ:~$ ls abc.txt
abc.txt
kim@DESKTOP-00NR3HJ:~$ ./rm abc.txt
kim@DESKTOP-00NR3HJ:~$ ls abc.txt
ls: cannot access 'abc.txt': No such file or directory
```

```
kim@DESKTOP-00NR3HJ:~$ ls abc.txt
ls: cannot access 'abc.txt': No such file or directory
kim@DESKTOP-00NR3HJ:~$ ./rm -f abc.txt
kim@DESKTOP-00NR3HJ:~$
```

```
kim@DESKTOP-00NR3HJ:~$ ls abc.txt
abc.txt
kim@DESKTOP-00NR3HJ:~$ ./rm -i abc.txt
rm: remove 'abc.txt'? y
kim@DESKTOP-00NR3HJ:~$ ls abc.txt
ls: cannot access 'abc.txt': No such file or directory
kim@DESKTOP-00NR3HJ:~$
```

```
kim@DESKTOP-00NR3HJ:~$ ls abc.txt
abc.txt
kim@DESKTOP-00NR3HJ:~$ ./rm -r abc.txt
kim@DESKTOP-00NR3HJ:~$ ls abc.txt
ls: cannot access 'abc.txt': No such file or directory
kim@DESKTOP-00NR3HJ:~$
```

```
kim@DESKTOP-00NR3HJ:~$ ls abc.txt
abc.txt
kim@DESKTOP-00NR3HJ:~$ ./rm -v abc.txt
removed 'abc.txt'
kim@DESKTOP-00NR3HJ:~$
```

Rm-f,rm-i,rm-r,rm-v

```
171         case 'f':
172             options.force = 1;
173             options.interactive = 0; // -f는 -i를 무시합니다.
174             break;
175
176 if (confirm_delete(path, opts)) {
177     if (unlink(path) != 0) {
178         // 파일이 존재하지 않는 오류(ENOENT)는 -f 옵션이 있을 때 무시합니다.
179         if (!opts->force || errno != ENOENT) {
180             fprintf(stderr, "rm: cannot remove '%s': %s\n", path, strerror(errno));
181         }
182         // -f 옵션이 있어도 파일이 없는 경우가 아니면 실제로 간주합니다.
183         if (errno != ENOENT) return -1;
184     } else if (opts->verbose) {
185         printf("removed '%s'\n", path);
186     }
187 }
```

```
5     printf("rm: remove '%s'? ", path);
6     // 한 글자만 읽고, 의도치 않은 입력을 방지하기 위해 입력 버퍼를 비웁니다.
7     int response = getchar();
8     // 사용자가 Enter 외에 다른 문자를 입력했을 경우를 대비해 버퍼를 끝까지 비웁니다.
9     int ch;
10     while ((ch = getchar()) != '\n' && ch != EOF);
11
12     return (response == 'y' || response == 'Y');
13 }
14
```

```
52     // -r 또는 -R 옵션이 없으면 디렉터리를 삭제할 수 없습니다.
53     if (!opts->recursive) {
54         fprintf(stderr, "rm: cannot remove '%s': Is a directory\n", path);
55         return -1;
56     }
57
58     DIR *dir = opendir(path);
```

```
70     // 디렉터리 내의 모든 항목을 순회합니다.
71     while ((entry = readdir(dir)) != NULL) {
72         // 현재 디렉터리(.)와 상위 디렉터리(..)는 건너뜁니다.
73         if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0) {
74             continue;
75         }
76
77         // 전체 경로를 안전하게 생성합니다. (고정 크기 대신 PATH_MAX 사용)
78         char subpath[PATH_MAX];
79         int len = snprintf(subpath, sizeof(subpath), "%s/%s", path, entry->d_name);
80         if (len >= sizeof(subpath)) {
81             fprintf(stderr, "rm: path is too long: %s/%s\n", path, entry->d_name);
82             overall_result = -1;
83             continue; // 다음 파일로 진행
84         }
85     }
```

```
101         } else if (opts->verbose) {
102             printf("removed directory '%s'\n", path);
103         }
104
105     } else if (opts->verbose) {
106         printf("removed '%s'\n", path);
107     }
108 }
```


echo

- Echo : 인자로 받은 문자열을 출력으로 내보냅니다.
- -n : 줄바꿈을 생략합니다.
- -e : \n같은 escape 문자를 실제 특수문자로 해석합니다.
- -E : escape문자를 해석하지 않습니다.

```
kim@DESKTOP-00NR3HJ:~$ ./echo hello world
hello world
kim@DESKTOP-00NR3HJ:~$ ./echo -n hello
hellokim@DESKTOP-00NR3HJ:~$ ./echo -e "hello\nworld"
hello
world
kim@DESKTOP-00NR3HJ:~$ ./echo -E "hello\nworld"
hello\nworld
kim@DESKTOP-00NR3HJ:~$
```

Echo-E,echo-e,echo-n

```
22 int process_and_print(const char *str, const EchoOptions *opts) {
23     if (!opts->enable_escapes) {
24         // 이스케이프 비활성화(-E) 모드에서는 문자열을 그대로 출력합니다.
25         fputs(str, stdout);
26         return 0;
27     }
28 }
```

```
29 // 이스케이프 활성화(-e) 모드
30 while (*str) {
31     if [*str == '\\' && *(str + 1) != '\0'] {
32         str++; // 백슬래시 다음 문자로 이동
33         switch (*str) {
34             case 'a': putchar('\a'); break; // 경고음 (bell)
35             case 'b': putchar('\b'); break; // 백스페이스
36             case 'c': return 1;           // 이후 모든 출력을 중단 (개행 포함)
37             case 'e': putchar('\x1B'); break; // ESC (escape) 문자
38             case 'f': putchar('\f'); break; // 폼 피드
39             case 'n': putchar('\n'); break; // 개행
40             case 'r': putchar('\r'); break; // 캐리지 리턴
41             case 't': putchar('\t'); break; // 수평 탭
42             case 'v': putchar('\v'); break; // 수직 탭
43             case '\\': putchar('\\'); break; // 백슬래시 자체
44             // case '': putchar(''); break; // echo는 따옴표를 이스케이프하지 않음 (셸이 처리)
45 }
```

```
112 // -n 옵션이 없고, \c 시퀀스가 등장하지 않았을 때만 마지막에 개행 문자를 출력
113 if (!options.no_newline && !output_halted) {
114     putchar('\n');
115 }
116 }
```

grep

- Grep : 문자열이 포함된 라인을 검색합니다.
- -i : 대소문자를 무시하고 검색합니다.
- -n : 매치된 라인의 번호를 함께 표시합니다
- -v : 문자열이 포함되지 않은 라인을 검색합니다.
- -e : 문자열이 포함된 라인을 검색합니다.
- -c : 포함된 라인의 수를 계산합니다.

```
kim@DESKTOP-00NR3HJ:~$ cat hello.txt
Hello World
hello linux
Linux is a great OS.
This is a test file.
Goodbye world.
kim@DESKTOP-00NR3HJ:~$ ./grep "world" hello.txt
Goodbye world.
kim@DESKTOP-00NR3HJ:~$ ./grep -i "world" hello.txt
Hello World
Goodbye world.
kim@DESKTOP-00NR3HJ:~$ ./grep -n "linux" hello.txt
2:hello linux
kim@DESKTOP-00NR3HJ:~$ ./grep -v "test" hello.txt
Hello World
hello linux
Linux is a great OS.
Goodbye world.
kim@DESKTOP-00NR3HJ:~$ ./grep -e "linux" hello.txt
hello linux
kim@DESKTOP-00NR3HJ:~$ ./grep -c "world" hello.txt
1
```

grep-i, grep-v, grep-n

```
48  /**
49   * @brief 주어진 라인이 패턴과 일치하는지 확인합니다.
50   * @param line 검사할 텍스트 라인
51   * @param opts 프로그램 옵션 구조체
52   * @return 일치하면 1, 그렇지 않으면 0을 반환합니다.
53   */
54  int line_matches(const char *line, const GrepOptions *opts) {
55      const char *match_ptr;
56      if (opts->ignore_case) {
57          match_ptr = my_strcasestr(line, opts->pattern);
58      } else {
59          match_ptr = strstr(line, opts->pattern);
60      }
61
62      // invert_match(-v) 옵션을 고려하여 최종 결과를 반환합니다.
63      // XOR(^) 연산자: (match_ptr != NULL)과 opts->invert_match가 다르면 true(1)
64      // -v 미사용: 매치되면(1) 1, 안되면(0) 0 반환
65      // -v 사용 : 매치되면(1) 0, 안되면(0) 1 반환
66      return (match_ptr != NULL) ^ (opts->invert_match);
67  }
68
```

```
if (!opts->count_only) {
    if (multiple_files) {
        printf("%s:", filename);
    }
    if (opts->show_line_number) {
        printf("%d:", line_number);
    }
    fputs(line, stdout);
}
```

```
27  // --- strcasestr 구현 ---
28  // _GNU_SOURCE가 정의되어 있으면 string.h에 strcasestr이 포함될 가능성이 높습니다.
29  // 여기서는 간단하게 직접 구현한 버전을 사용합니다.
30  const char *my_strcasestr(const char *haystack, const char *needle) {
31      if (!*needle) return haystack; // 빈 패턴은 항상 매치
32
33      for (; *haystack; haystack++) {
34          // 현재 위치에서 needle과 일치하는지 검사
35          const char *h = haystack;
36          const char *n = needle;
37          while (*h && *n && tolower((unsigned char)*h) == tolower((unsigned char)*n)) {
38              h++;
39              n++;
40          }
41          if (*n == '\0') { // needle의 끝까지 도달했다면 매치 성공
42              return haystack;
43          }
44      }
45      return NULL; // 매치 실패
46  }
47
```

Grep-c, grep-e, 파일 처리

```
138 if (optind == argc) {
139     // 처리할 파일 인자가 없으면 표준 입력(stdin)에서 읽어옵니다.
140     process_file(stdin, "(standard input)", &options, 0);
141 } else {
142     // 파일 인자들을 순회하며 처리합니다.
143     for (int i = optind; i < argc; i++) {
144         FILE *fp = fopen(argv[i], "r");
145         if (!fp) {
146             // grep 스타일의 오류 메시지 출력
147             fprintf(stderr, "%s: %s: %s\n", argv[0], argv[i], strerror(2));
148             continue; // 다음 파일로 진행
149         }
150         process_file(fp, argv[i], &options, multiple_files);
151         fclose(fp);
152     }
153 }
154
```

```
97     if (opts->count_only) {
98         if (multiple_files) {
99             printf("%s:", filename);
100         }
101         printf("%d\n", match_count);
102     }
```

```
126 if (!options.pattern) {
127     if (optind >= argc) {
128         fprintf(stderr, "Usage: %s [-ivnc] [-e pattern] [pattern] [file...]\n", argv[0]);
129         return 2;
130     }
131     options.pattern = argv[optind++];
132 }
133
```

sort

- Sort : 각 라인 전체를 알파벳 순서대로 정렬합니다.
- -r : 기본 정렬의 역순으로 결과를 출력합니다.
- -k : -k n(숫자) 옵션을 사용하여 n 번째 필드를 기준으로 정렬합니다.
- -n : 실제 숫자 기준으로 정렬합니다.
- -u : 중복된 라인을 하나만 남기고 제거합니다.

```
kim@DESKTOP-00NR3HJ:~$ cat data.txt
banana 10 apple
apple 30 orange
cherry 5 banana
apple 20 grape
banana 10 apple
kim@DESKTOP-00NR3HJ:~$ ./sort data.txt
apple 20 grape
apple 30 orange
banana 10 apple
banana 10 apple
cherry 5 banana
kim@DESKTOP-00NR3HJ:~$ ./sort -r data.txt
cherry 5 banana
banana 10 apple
banana 10 apple
apple 30 orange
apple 20 grape
kim@DESKTOP-00NR3HJ:~$ ./sort -k 2 data.txt
banana 10 apple
banana 10 apple
apple 20 grape
apple 30 orange
cherry 5 banana
```

```
kim@DESKTOP-00NR3HJ:~$ cat data.txt
banana 10 apple
apple 30 orange
cherry 5 banana
apple 20 grape
banana 10 apple
kim@DESKTOP-00NR3HJ:~$ ./sort -k 2 -n data.txt
cherry 5 banana
banana 10 apple
banana 10 apple
apple 20 grape
apple 30 orange
kim@DESKTOP-00NR3HJ:~$ ./sort -u data.txt
apple 20 grape
apple 30 orange
banana 10 apple
cherry 5 banana
```


Sort-k,sort-r,sort-u

```
34 void get_field_from_line(char *dest, size_t dest_size, const char *line, int key, char delimiter) {
35     const char *start = line;
36     const char *end;
37     int current_key = 1;
38
39     // 원하는 키를 찾을 때까지 구분자를 기준으로 이동
40     while (current_key < key) {
41         start = strchr(start, delimiter);
42         if (!start) { // 구분자를 더 찾을 수 없으면 빈 문자열 처리
43             dest[0] = '\0';
44             return;
45         }
46         start++; // 구분자 다음 문자로 이동
47         current_key++;
48     }
49
50     // 필드의 끝(다음 구분자 또는 문자열의 끝)을 찾음
51     end = strchr(start, delimiter);
52     if (!end) {
53         end = start + strlen(start);
54     }
55
56     // 필드 내용을 버퍼에 안전하게 복사
57     size_t len = end - start;
58     if (len >= dest_size) {
59         len = dest_size - 1;
60     }
61     memcpy(dest, start, len);
62     dest[len] = '\0';
63 }
64
```

```
102     // -r: 역순 정렬
103     return g_opts.reverse ? -cmp_result : cmp_result;
104 }
105
```

```
90     if (g_opts.numeric) {
91         // -n: 숫자 비교
92         double num1 = atof(p1);
93         double num2 = atof(p2);
94         if (num1 < num2) cmp_result = -1;
95         else if (num1 > num2) cmp_result = 1;
96         else cmp_result = 0;
97     }
98 }
```

```
170     // -u 옵션 처리: 이전 라인과 비교하여 다를 때만 출력
171     // 비교 시 qsort에 사용된 것과 동일한 compare_lines 함수를 재사용하여 정확성 보장
172     if (g_opts.unique && i > 0 && compare_lines(&lines[i - 1], &lines[i]) == 0) {
173         continue; // 중복된 라인이면 건너뛰기
174     }
175     printf("%s", lines[i]);
176 }
```



점수:13.5점

- 만들면서 부족한 점을 많이 느꼈습니다.
-