

시스템 프로그래밍 깃허브 , 명령어 정리

<https://github.com/bumsoo1/system>

소프트웨어학과
2021763063
정범수

2주차

system / 0314 /



Add file



bumsoo1 Update readme.md

9568662 · 3 months ago



History

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

Commits

History for system / 0314 on main

All users

All time

Commits on Mar 20, 2025

Update readme.md

bumsoo1 authored on Mar 20

Verified

9568662



Create readme.md

bumsoo1 authored on Mar 20

Verified

e77feaa



End of commit history for this file

3주차

system / 0321 /



bumsoo1 Update readme.md

2130fe2 · 3 months ago

History

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

Commits

History for system / 0321 on main

All users


All time

Commits on Mar 27, 2025

Update readme.md bumsoo1 authored on Mar 27	Verified 2130fe2
Create readme.md bumsoo1 authored on Mar 27	Verified 3c02986


End of commit history for this file




4주차

system / 0328 / 



bumsoo1 Add files via upload


b5c1cbd · 2 months ago  History


Name	Last commit message	Last commit date
 ..		
 Alphabet.c	Add files via upload	2 months ago
 readme.md	Create readme.md	2 months ago

Commits


History for system / 0328 on main

 All users ▾

 All time ▾

 Commits on Apr 10, 2025


Add files via upload

 bumsoo1 authored on Apr 10

Verified


b5c1cbd   <>

Create readme.md

 bumsoo1 authored on Apr 10

Verified

fddd6a8   <>

 End of commit history for this file

5주차

system / 0404 /



bumsoo1

Add files via upload

e5fbf49 · 2 months ago



History

Name	Last commit message	Last commit date
..		
number.c	Add files via upload	2 months ago
readme.md	Create readme.md	2 months ago

Commits

History for system / 0404 on main

All users

All time

Commits on Apr 10, 2025

Add files via upload

bumsoo1 authored on Apr 10

Verified

e5fbf49



Create readme.md

bumsoo1 authored on Apr 10

Verified

6282954



End of commit history for this file

6주차

system / 0411 /



bumsoo1 Update readme.md

56ef252 · 2 months ago

History

Name	Last commit message	Last commit date
..		
readme.md	Update readme.md	2 months ago

Commits

History for system / 0411 on main

All users

All time

Commits on Apr 17, 2025

Update readme.md

bumsoo1 authored on Apr 17

Verified

56ef252



Update readme.md

bumsoo1 authored on Apr 17

Verified

0d33ea5



Commits on Apr 14, 2025

Create readme.md

bumsoo1 authored on Apr 14

Verified

6486fc4



End of commit history for this file

7주차

system / 0418 /



bumsoo1 Add files via upload

06e9fa5 · last month



History

Name	Last commit message	Last commit date
..		
long.c	Add files via upload	last month
readme.md	Create readme.md	last month

Commits

History for system / 0418 on main

All users

All time

Commits on May 6, 2025

Add files via upload

bumsoo1 authored on May 6

Verified

06e9fa5



Create readme.md

bumsoo1 authored on May 6

Verified

f412c15



End of commit history for this file

9주차

system / 0502 /



bumsoo1 Add files via upload

8a328ca · last month History

Name	Last commit message	Last commit date
..		
argument.c	Add files via upload	last month
open1.c	Add files via upload	last month
open2.c	Add files via upload	last month
readme.md	Create readme.md	last month

Commits

History for system / 0502 on [main](#)

All users

All time

Commits on May 6, 2025

Add files via upload

bumsoo1 authored on May 6

Verified

8a328ca

Create readme.md

bumsoo1 authored on May 6

Verified

320c8b7

End of commit history for this file

10주차

system / 0509 /



bumsoo1 Update list2.c

10a6826 · 3 weeks ago History

Name	Last commit message	Last commit date
..		
list2.c	Update list2.c	3 weeks ago
readme.md	Update readme.md	3 weeks ago

Commits

History for system / 0509 on main

All users

All time

Commits on May 19, 2025

Update list2.c

bumsoo1 authored 3 weeks ago

Verified 10a6826

Add files via upload

bumsoo1 authored 3 weeks ago

Verified 013da5d

Update readme.md

bumsoo1 authored 3 weeks ago

Verified dc5ec34

Create readme.md

bumsoo1 authored 3 weeks ago

Verified e217e13

End of commit history for this file

11주차

system / 0516 /



bumsoo1 Update readme.md

64c0c2c · 3 weeks ago

History

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

Commits

History for system / 0516 on main

All users

All time

Commits on May 19, 2025

Update readme.md

bumsoo1 authored 3 weeks ago

Verified 64c0c2c

Create readme.md

bumsoo1 authored 3 weeks ago

Verified 8f9601a

End of commit history for this file

12주차

system / 0523 /



bumsoo1 Update readme.md

2757540 · last week

History

Name	Last commit message	Last commit date
..		
fork1.c	Add files via upload	last week
fork2.c	Add files via upload	last week
fork3.c	Add files via upload	last week
forkwait.c	Add files via upload	last week
readme.md	Update readme.md	last week
waitpid.c	Add files via upload	last week

Commits

History for system / 0523 on main

All users

All time

Commits on Jun 5, 2025

Update readme.md	Verified	2757540			
bumsoo1 authored last week					
Update readme.md	Verified	3fe626d			
bumsoo1 authored last week					
Add files via upload	Verified	c466e9c			
bumsoo1 authored last week					
Create readme.md	Verified	4238429			
bumsoo1 authored last week					

End of commit history for this file

1. pwd

현재 작업 디렉토리 출력

```
C pwd.c x
cli > C pwd.c
1  #include <stdio.h>
2  #include <unistd.h> // getcwd 함수
3  #include <limits.h> // PATH_MAX 상수
4
5  int main() {
6      char cwd[PATH_MAX]; // 현재 작업 디렉토리를 저장할 버퍼
7
8      if (getcwd(cwd, sizeof(cwd)) != NULL) {
9          printf("%s\n", cwd); // 현재 디렉토리 경로 출력
10     } else {
11         perror("getcwd 오류"); // 오류 메시지 출력
12         return 1;
13     }
14
15     return 0;
16 }
17
```

- `getcwd()` 함수로 현재 작업 디렉토리 경로를 얻고,
- 성공 시 경로를 출력, 실패하면 오류 메시지를 출력합니다.

2. mkdir

디렉토리 생성

```
C mkdir.c ×
cli > C mkdir.c
1  #include <stdio.h>
2  #include <sys/stat.h> // mkdir 함수
3  #include <sys/types.h> // mode_t 타입
4  #include <errno.h>
5  #include <string.h>
6
7  int main(int argc, char *argv[]) {
8      if (argc < 2) {
9          printf("사용법: mkdir 디렉토리명\n");
10         return 1;
11     }
12
13     // 디렉토리 생성 (권한 0755)
14     if (mkdir(argv[1], 0755) == -1) {
15         fprintf(stderr, "mkdir 오류: %s\n", strerror(errno));
16         return 1;
17     }
18
19     printf("'%s' 디렉토리를 생성했습니다.\n", argv[1]);
20     return 0;
}
```

- `mkdir()` 함수는 두 번째 인자로 디렉토리 권한을 받습니다 (0755 는 `rwXr-Xr-X`).
- 오류가 발생하면 `strerror(errno)` 를 이용해 자세한 오류 메시지를 출력합니다.
- 인자가 없으면 사용법 안내를 출력합니다.

3. clear

화면 지우기

```
C clear.c X
cli > C clear.c
1  #include <stdio.h>
2
3  int main() {
4      // ANSI escape code로 터미널 화면 지우기
5      printf("\033[2J"); // 화면 전체 지우기
6      printf("\033[H");  // 커서를 좌측 상단(1,1) 위치로 이동
7      fflush(stdout);    // 출력 버퍼 비우기
8      return 0;
9  }
10
```

- `\033[2J` : 화면 전체를 지웁니다.
- `\033[H` : 커서를 화면 좌측 상단으로 이동시킵니다.
- 이 두 개를 출력하면 `clear` 명령어 효과와 동일합니다.

4. whoami

현재 사용자 출력

```
C whoami.c X
cli > C whoami.c
1  #include <stdio.h>
2  #include <unistd.h> // getlogin()
3
4  int main() {
5      char *username = getlogin();
6      if (username == NULL) {
7          perror("getlogin 오류");
8          return 1;
9      }
10     printf("%s\n", username);
11     return 0;
12 }
13
```

- `getlogin()` 함수가 현재 로그인한 사용자 이름을 반환합니다.
- 실패 시 `NULL` 을 반환하고, 오류 메시지를 출력합니다.

5. echo

문자열 출력

```
C echo.c X
cli > C echo.c
1  #include <stdio.h>
2
3  int main(int argc, char *argv[]) {
4      for (int i = 1; i < argc; i++) {
5          printf("%s", argv[i]);
6          if (i < argc - 1) {
7              printf(" ");
8          }
9      }
10     printf("\n");
11     return 0;
12 }
13
```

- 명령어 인자들을 공백으로 구분해 출력합니다.
- 마지막에 줄바꿈을 추가해서 기본 `echo` 동작과 같습니다.

6. date

```
C date.c  X
cli > C date.c
1  #include <stdio.h>
2  #include <time.h>
3
4  int main() {
5      time_t now = time(NULL); // 현재 시간 가져오기
6      if (now == (time_t)(-1)) {
7          perror("time 오류");
8          return 1;
9      }
10
11     struct tm *local = localtime(&now); // 지역 시간으로 변환
12     if (local == NULL) {
13         perror("localtime 오류");
14         return 1;
15     }
16
17     // 날짜 및 시간 출력 형식: YYYY-MM-DD HH:MM:SS
18     printf("%04d-%02d-%02d %02d:%02d:%02d\n",
19           local->tm_year + 1900, local->tm_mon + 1, local->tm_mday,
20           local->tm_hour, local->tm_min, local->tm_sec);
21
22     return 0;
23 }
24
```

현재 시스템 날짜와 시간을 출력

- `time(NULL)` 로 현재 시간을 `time_t` 형식으로 가져오고,
- `localtime()` 으로 현지 시간 정보(`struct tm`)로 변환합니다.
- `printf` 로 년, 월, 일, 시, 분, 초를 포매팅해 출력합니다.

7. uname

```
C uname.c X
cli > C uname.c
1  #include <stdio.h>
2  #include <sys/utsname.h> // uname 구조체 및 함수
3
4  int main() {
5      struct utsname sysinfo;
6
7      if (uname(&sysinfo) == -1) {
8          perror("uname 오류");
9          return 1;
10     }
11
12     printf("시스템 이름: %s\n", sysinfo.sysname);
13     printf("노드 이름: %s\n", sysinfo.nodename);
14     printf("릴리즈: %s\n", sysinfo.release);
15     printf("버전: %s\n", sysinfo.version);
16     printf("머신: %s\n", sysinfo.machine);
17
18     return 0;
19 }
20
```

커널 이름 출력

설명

- `uname()` 함수가 `struct utsname` 구조체에 시스템 정보를 채워줍니다.
- 주요 필드:
 - `sysname`: 운영체제 이름 (예: Linux)
 - `nodename`: 네트워크 노드 이름
 - `release`: 커널 릴리즈 버전
 - `version`: 커널 버전 정보
 - `machine`: 하드웨어 이름 (예: x86_64)

8. exit

C exit.c X

cli > C exit.c

```
1  #include <stdio.h>
2  #include <stdlib.h> // exit 함수
3
4  int main(int argc, char *argv[]) {
5      int status = 0; // 기본 종료 상태
6
7      if (argc > 1) {
8          status = atoi(argv[1]); // 인자가 있으면 종료 상태로 변환
9      }
10
11     printf("프로그램을 종료합니다. 종료 상태: %d\n", status);
12     exit(status); // 프로세스 종료
13 }
14
```

프로그램 종료

- `exit(0)` 은 정상 종료,
- `exit(1)` 이상은 오류 종료 상태를 나타냅니다.
- 인자로 종료 코드를 받아 처리할 수도 있습니다.

9. rm

```
C rm.c ×
cli > C rm.c
1  #include <stdio.h>
2  #include <unistd.h> // unlink 함수
3  #include <errno.h>
4  #include <string.h>
5
6  int main(int argc, char *argv[]) {
7      if (argc < 2) {
8          fprintf(stderr, "사용법: %s <파일 이름>\n", argv[0]);
9          return 1;
10     }
11
12     int ret = unlink(argv[1]); // 파일 삭제
13     if (ret == -1) {
14         fprintf(stderr, "파일 삭제 오류: %s\n", strerror(errno));
15         return 1;
16     }
17
18     printf("파일 '%s'가 삭제되었습니다.\n", argv[1]);
19     return 0;
20 }
21
```

파일 삭제

- `unlink()` 함수가 파일을 삭제합니다.
- 디렉토리는 삭제할 수 없고, 빈 디렉토리는 `rmdir`으로 삭제해야 합니다.

10. rmdir

```
C rmdir.c x
cli > C rmdir.c
1  #include <stdio.h>
2  #include <unistd.h> // rmdir 함수
3  #include <errno.h>
4  #include <string.h>
5
6  int main(int argc, char *argv[]) {
7      if (argc < 2) {
8          fprintf(stderr, "사용법: %s <디렉토리 이름>\n", argv[0]);
9          return 1;
10     }
11
12     int ret = rmdir(argv[1]);
13     if (ret == -1) {
14         fprintf(stderr, "디렉토리 삭제 오류: %s\n", strerror(errno));
15         return 1;
16     }
17
18     printf("디렉토리 '%s'가 삭제되었습니다.\n", argv[1]);
19     return 0;
20 }
21
```

디렉토리 삭제

- `rmdir()` 함수는 빈 디렉토리만 삭제 가능합니다.
- 디렉토리가 비어 있지 않으면 오류가 발생합니다.
- 삭제 성공 시 간단한 메시지를 출력합니다.

11. cat

```
C cat.c x
cli > C cat.c
1  #include <stdio.h>
2  #include <errno.h>
3  #include <string.h>
4
5  int main(int argc, char *argv[]) {
6      if (argc < 2) {
7          fprintf(stderr, "사용법: %s <파일 이름>\n", argv[0]);
8          return 1;
9      }
10
11     FILE *fp = fopen(argv[1], "r");
12     if (fp == NULL) {
13         fprintf(stderr, "파일 열기 오류: %s\n", strerror(errno));
14         return 1;
15     }
16
17     char buffer[1024];
18     while (fgets(buffer, sizeof(buffer), fp) != NULL) {
19         fputs(buffer, stdout);
20     }
21
22     fclose(fp);
23     return 0;
24 }
25
```

파일 내용 출력

- `fopen()` 으로 파일을 읽기 모드로 열고,
- `fgets()` 로 한 줄씩 읽어 표준 출력으로 출력합니다.

12. head

```
C head.c x
cli > C head.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <errno.h>
4  #include <string.h>
5
6  #define DEFAULT_LINES 10
7
8  int main(int argc, char *argv[]) {
9      int lines_to_print = DEFAULT_LINES;
10     const char *filename;
11
12     // 인자 처리: ./head [lines] filename 또는 ./head filename
13     if (argc == 3) {
14         lines_to_print = atoi(argv[1]);
15         if (lines_to_print <= 0) {
16             fprintf(stderr, "유효하지 않은 줄 수: %s\n", argv[1]);
17             return 1;
18         }
19         filename = argv[2];
20     } else if (argc == 2) {
21         filename = argv[1];
22     } else {
23         fprintf(stderr, "사용법: %s [출력할_줄수] <파일 이름>\n", argv[0]);
24         return 1;
25     }
26
27     FILE *fp = fopen(filename, "r");
28     if (!fp) {
29         fprintf(stderr, "파일 열기 오류: %s\n", strerror(errno));
30         return 1;
31     }
32
33     char buffer[1024];
34     int count = 0;
35     while (fgets(buffer, sizeof(buffer), fp) && count < lines_to_print) {
36         fputs(buffer, stdout);
37         count++;
38     }
39
40     fclose(fp);
41     return 0;
42 }
43
```

파일의 처음 N줄(기본 10줄)을 출력

- 기본 10줄을 출력합니다.
- 인자가 2개면 첫 번째를 출력할 줄 수로 해석합니다.
- 파일을 열고, `fgets()` 로 한 줄씩 읽어 출력 후 지정된 줄 수만큼 출력 후 종료합니다.

13. tail

```
C tail.c X
cli> C tail.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <errno.h>
5
6 #define DEFAULT_LINES 10
7 #define MAX_LINES 10000 // 최대 저장 가능한 줄 수 (조절 가능)
8
9 int main(int argc, char *argv[]) {
10     int lines_to_print = DEFAULT_LINES;
11     const char *filename;
12
13     // 인자 처리: ./tail [lines] filename 또는 ./tail filename
14     if (argc == 3) {
15         lines_to_print = atoi(argv[1]);
16         if (lines_to_print <= 0) {
17             fprintf(stderr, "유효하지 않은 줄 수: %s\n", argv[1]);
18             return 1;
19         }
20         filename = argv[2];
21     } else if (argc == 2) {
22         filename = argv[1];
23     } else {
24         fprintf(stderr, "사용법: %s [출력할_줄수] <파일 이름>\n", argv[0]);
25         return 1;
26     }
27 }
```

```
28 FILE *fp = fopen(filename, "r");
29 if (!fp) {
30     fprintf(stderr, "파일 열기 오류: %s\n", strerror(errno));
31     return 1;
32 }
33
34 char *lines[MAX_LINES];
35 int count = 0;
36 size_t len = 0;
37
38 // 파일에서 줄을 읽어 배열에 저장 (최대 MAX_LINES)
39 while (count < MAX_LINES && getline(&lines[count], &len, fp) != -1) {
40     count++;
41     len = 0;
42 }
43
44 fclose(fp);
45
46 // 출력 시작 줄 계산
47 int start = count - lines_to_print;
48 if (start < 0) start = 0;
49
50 for (int i = start; i < count; i++) {
51     printf("%s", lines[i]);
52     free(lines[i]);
53 }
54
55 return 0;
56 }
57
```

파일의 마지막 N줄(기본 10줄)을 출력

- 파일 전체를 줄 단위로 읽어 배열에 저장합니다.
- 기본 10줄, 혹은 지정한 줄 수만큼 마지막부터 출력합니다.
- `getline()` 함수로 동적 할당된 줄을 읽어 메모리 관리 필요.

14. who

```
C who.c x
cli > C who.c
1  #include <stdio.h>
2  #include <utmp.h>
3  #include <fcntl.h>
4  #include <unistd.h>
5  #include <string.h>
6
7  int main() {
8      struct utmp *ut;
9
10     setutent(); // utmp 파일 처음으로 이동
11
12     while ((ut = getutent()) != NULL) {
13         if (ut->ut_type == USER_PROCESS) { // 실제 로그인 사용자만 출력
14             printf("%s\t%s\t%s\n", ut->ut_user, ut->ut_line, ut->ut_host);
15         }
16     }
17
18     endutent(); // utmp 파일 닫기
19     return 0;
20 }
21
```

현재 로그인한 사용자 정보

- `setutent()` : utmp 파일 시작 위치로 이동
- `getutent()` : utmp 파일에서 한 엔트리 읽기
- `ut_type == USER_PROCESS` 인 항목만 로그인 사용자 정보
- 사용자명(`ut_user`), 터미널(`ut_line`), 원격호스트(`ut_host`) 출력

15. basename

C basename.c X

cli > C basename.c

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(int argc, char *argv[]) {
5      if (argc != 2) {
6          fprintf(stderr, "사용법: %s <경로>\n", argv[0]);
7          return 1;
8      }
9
10     char *path = argv[1];
11     char *base = strrchr(path, '/'); // 마지막 '/' 위치 찾기
12
13     if (base != NULL) {
14         base++; // '/' 다음 문자부터가 파일명
15         if (*base == '\0') {
16             // 경로가 '/'로 끝난 경우, 마지막 '/' 이전 문자열 출력
17             // 예: "/usr/local/bin/"
18             // 이런 경우 간단히 '/'를 제거하고 재탐색하는 방식도 있음
19             // 여기서는 그냥 '/' 출력
20             base = "/";
21         }
22     } else {
23         base = path; // '/' 없으면 전체가 파일명
24     }
25
26     printf("%s\n", base);
27     return 0;
28 }
29
```

경로 문자열에서 마지막 파일명만 추출

- `strrchr()` 로 문자열에서 마지막 `/` 위치 찾음
- 마지막 `/` 다음부터가 `basename`이므로 그 부분 출력
- 만약 `/` 가 없으면 원본 전체가 `basename`
- 경로가 `/` 로 끝나는 경우 간단하게 처리

16. dirname

```
C dirname.c X
cli > C dirname.c
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(int argc, char *argv[]) {
5      if (argc != 2) {
6          fprintf(stderr, "사용법: %s <경로>\n", argv[0]);
7          return 1;
8      }
9
10     char path[1024];
11     strncpy(path, argv[1], sizeof(path));
12     path[sizeof(path) - 1] = '\0'; // 널 종료 보장
13
14     char *last_slash = strrchr(path, '/');
15
16     if (last_slash != NULL) {
17         if (last_slash == path) {
18             // 루트 디렉토리(/)만 있는 경우
19             printf("/\n");
20         } else {
21             *last_slash = '\0'; // 마지막 '/' 이후 제거
22             printf("%s\n", path);
23         }
24     } else {
25         // '/'가 없으면 현재 디렉토리
26         printf(".\n");
27     }
28
29     return 0;
30 }
31
```

경로에서 디렉토리만 출력

- `strrchr()` 로 마지막 `/` 위치를 찾음
- 있으면 그 위치를 널 문자 `\0` 로 바꿔서 뒷부분 제거
- 만약 `/` 가 없다면 `.` (현재 디렉토리) 출력
- 경로가 `/` 하나뿐이면 `/` 그대로 출력

17. sleep

```
C sleep.c  X
cli > C sleep.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h> // sleep()
4
5  int main(int argc, char *argv[]) {
6      if (argc != 2) {
7          fprintf(stderr, "사용법: %s <초>\n", argv[0]);
8          return 1;
9      }
10
11     int seconds = atoi(argv[1]);
12     if (seconds < 0) {
13         fprintf(stderr, "양수 시간을 입력하세요.\n");
14         return 1;
15     }
16
17     sleep(seconds);
18
19     return 0;
20 }
21
```

지정한 시간동안 프로그램을 일시 중지

- `atoi()` 로 문자열을 정수로 변환
- `sleep()` 함수로 지정된 초 동안 대기
- 음수 입력 처리 및 인자 개수 확인 포함

18. df

```
C df.c x
cli > C df.c
1  #include <stdio.h>
2  #include <sys/statvfs.h>
3
4  int main() {
5      struct statvfs stat;
6
7      // 현재 디렉토리의 파일 시스템 정보 가져오기
8      if (statvfs(".", &stat) != 0) {
9          perror("statvfs 오류");
10         return 1;
11     }
12
13     unsigned long total = stat.f_blocks * stat.f_frsize;
14     unsigned long free  = stat.f_bfree  * stat.f_frsize;
15     unsigned long avail = stat.f_bavail * stat.f_frsize;
16     unsigned long used  = total - free;
17
18     printf("파일시스템 정보 (현재 디렉토리 기준)\n");
19     printf("전체 용량: %lu bytes\n", total);
20     printf("사용 중: %lu bytes\n", used);
21     printf("사용 가능: %lu bytes\n", avail);
22
23     return 0;
24 }
25
```

디스크 사용량

- `statvfs()` 함수로 파일 시스템 정보 구조체 가져오기
- `f_blocks` × `f_frsize` → 전체 용량
- `f_bfree` × `f_frsize` → 전체 중 사용 가능한 블록
- `f_bavail` × `f_frsize` → 비 root 사용자 기준 사용 가능 블록

19. uptime

```
C uptime.c X
cli > C uptime.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      FILE *fp = fopen("/proc/uptime", "r");
6      if (fp == NULL) {
7          perror("/proc/uptime 열기 실패");
8          return 1;
9      }
10
11     double uptime_seconds;
12     if (fscanf(fp, "%lf", &uptime_seconds) != 1) {
13         fprintf(stderr, "uptime 정보 읽기 실패\n");
14         fclose(fp);
15         return 1;
16     }
17
18     fclose(fp);
19
20     int days = uptime_seconds / (60 * 60 * 24);
21     int hours = ((int)uptime_seconds % (60 * 60 * 24)) / (60 * 60);
22     int minutes = ((int)uptime_seconds % (60 * 60)) / 60;
23     int seconds = (int)uptime_seconds % 60;
24
25     printf("시스템 가동 시간: %d일 %d시간 %d분 %d초\n",
26           days, hours, minutes, seconds);
27
28     return 0;
29 }
30
```

시스템이 부팅된 후 경과 시간

- `/proc/uptime` 파일의 첫 번째 값 (uptime 초)을 읽음
- `fscanf()` 로 double 값 읽기
- 초 → 일/시간/분/초로 변환
- 포매팅해서 가동 시간 출력

20. which

```
C which.c x
cli > C which.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h> // access()
5
6  int main(int argc, char *argv[]) {
7      if (argc != 2) {
8          fprintf(stderr, "사용법: %s <명령어>\n", argv[0]);
9          return 1;
10     }
11
12     char *path_env = getenv("PATH");
13     if (path_env == NULL) {
14         fprintf(stderr, "PATH 환경변수를 찾을 수 없습니다.\n");
15         return 1;
16     }
17
18     char *path_copy = strdup(path_env); // 수정 가능한 복사본
19     if (path_copy == NULL) {
20         perror("메모리 할당 실패");
21         return 1;
22     }
23
24     char *token = strtok(path_copy, ":");
25
26     while (token != NULL) {
27         char fullpath[1024];
28         snprintf(fullpath, sizeof(fullpath), "%s/%s", token, argv[1]);
29
30         if (access(fullpath, X_OK) == 0) { // 실행 가능 여부 확인
31             printf("%s\n", fullpath);
32             free(path_copy);
33             return 0;
34         }
35
36         token = strtok(NULL, ":");
37     }
38
39     printf("%s: 명령어를 찾을 수 없습니다.\n", argv[1]);
40
41     free(path_copy);
42     return 1;
43 }
44
```

주어진 명령어가 어디에 있는지 찾아 경로를 출력

- `getenv("PATH")` 로 환경변수 PATH 가져오기
- `strtok()` 로 : 기준으로 경로 분리
- 각 디렉토리에 `access(fullpath, X_OK)` 로 실행 파일 존재 여부 확인

21. grep

```
C grep.c x
cli > C grep.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX_LINE 1024
6
7  int main(int argc, char *argv[]) {
8      if (argc != 3) {
9          fprintf(stderr, "사용법: %s <검색어> <파일이름>\n", argv[0]);
10         return 1;
11     }
12
13     char *keyword = argv[1];
14     char *filename = argv[2];
15
16     FILE *fp = fopen(filename, "r");
17     if (fp == NULL) {
18         perror("파일 열기 실패");
19         return 1;
20     }
21
22     char line[MAX_LINE];
23     while (fgets(line, sizeof(line), fp)) {
24         if (strstr(line, keyword) != NULL) { // 문자열 포함 여부 검사
25             printf("%s", line);
26         }
27     }
28
29     fclose(fp);
30     return 0;
31 }
32
```

파일에서 특정 문자열이 포함된 줄을 출력

- `fgets()` 로 파일의 한 줄씩 읽음
- `strstr()` 로 해당 줄에 검색어 포함 여부 확인
- 포함되어 있으면 그 줄을 출력
- 파일 끝까지 반복

22. ps

```
C psc x
cli > C psc
1  #include <stdio.h>
2  #include <dirent.h>
3  #include <string.h>
4  #include <ctype.h>
5
6  int main() {
7      DIR *proc_dir;
8      struct dirent *entry;
9
10     proc_dir = opendir("/proc");
11     if (proc_dir == NULL) {
12         perror("/proc 디렉토리 열기 실패");
13         return 1;
14     }
15
16     printf("PID\t프로세스 이름\n");
17     printf("-----\n");
18
19     while ((entry = readdir(proc_dir)) != NULL) {
20         // 디렉토리 이름이 숫자인 경우에만 (프로세스 디렉토리)
21         if (isdigit(entry->d_name[0])) {
22             char stat_path[256];
23             snprintf(stat_path, sizeof(stat_path), "/proc/%s/stat", entry->d_name);
24
25             FILE *stat_file = fopen(stat_path, "r");
26             if (stat_file) {
27                 int pid;
28                 char comm[256];
29                 fscanf(stat_file, "%d (%[^)])", &pid, comm);
30                 printf("%d\t%s\n", pid, comm);
31                 fclose(stat_file);
32             }
33         }
34     }
35
36     closedir(proc_dir);
37     return 0;
38 }
```

현재 실행중인 프로세스 목록을 출력

- `/proc` 디렉토리를 열고 숫자 디렉토리만 탐색
- `/proc/[PID]/stat` 파일의 첫 번째 값이 PID, 두 번째 값이 프로세스 이름
- `fscanf()` 로 PID 와 프로세스 이름 읽어오기

23. touch

```
C touch.c X
cli > C touch.c
1  #include <stdio.h>
2
3  int main(int argc, char *argv[]) {
4      if (argc < 2) {
5          fprintf(stderr, "사용법: %s <파일이름>\n", argv[0]);
6          return 1;
7      }
8
9      for (int i = 1; i < argc; i++) {
10         FILE *fp = fopen(argv[i], "a"); // append 모드: 없으면 생성, 있으면 열기
11         if (fp == NULL) {
12             perror("파일 생성 실패");
13             continue;
14         }
15         fclose(fp);
16     }
17
18     return 0;
19 }
20
```

파일을 생성하거나, 파일의 존재 경우
수정 시간을 갱신

- |인자로 전달된 파일명을 `fopen()` 의 `"a"` 모드로 열면
 - 파일이 없으면 새로 생성
 - 있으면 열기만 하고 수정 시간 갱신됨 (대부분의 파일 시스템에서)
- 파일 열기 실패 시 `perror()` 로 에러 출력
- 여러 개 파일도 동시에 생성 가능

24. history

```
C history.c X
cli > C history.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define HISTORY_FILE "history.txt"
5
6  int main() {
7      FILE *fp = fopen(HISTORY_FILE, "r");
8      if (!fp) {
9          perror("history 파일 열기 실패");
10         return 1;
11     }
12
13     char line[1024];
14     int count = 1;
15     while (fgets(line, sizeof(line), fp)) {
16         printf("%5d %s", count++, line);
17     }
18
19     fclose(fp);
20     return 0;
21 }
22
```

사용자가 입력한 명령어의 기록

history.txt 내용을 번호와 함께 출력

25. cp

```
cli > C cp.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define BUFFER_SIZE 4096
5
6  int main(int argc, char *argv[]) {
7      if (argc != 3) {
8          fprintf(stderr, "사용법: %s <원본파일> <복사할파일>\n", argv[0]);
9          return 1;
10     }
11
12     FILE *src = fopen(argv[1], "rb");
13     if (src == NULL) {
14         perror("원본 파일 열기 실패");
15         return 1;
16     }
17
18     FILE *dst = fopen(argv[2], "wb");
19     if (dst == NULL) {
20         perror("복사할 파일 열기 실패");
21         fclose(src);
22         return 1;
23     }
24
25     char buffer[BUFFER_SIZE];
26     size_t bytes;
27
28     while ((bytes = fread(buffer, 1, BUFFER_SIZE, src)) > 0) {
29         if (fwrite(buffer, 1, bytes, dst) != bytes) {
30             perror("파일 쓰기 실패");
31             fclose(src);
32             fclose(dst);
33             return 1;
34         }
35     }
36
37     fclose(src);
38     fclose(dst);
39
40     return 0;
41 }
```

파일을 복사

- `fopen` 으로 원본을 바이너리 모드(`rb`), 대상 파일을 쓰기 모드(`wb`)로 열기
- `fread` 로 버퍼 단위로 읽고 `fwrite` 로 쓰기 반복
- 에러 발생 시 적절히 처리
- 텍스트 파일이든 바이너리 파일이든 복사 가능

26. mv

파일/디렉토리 이동

```
C mv.c x
cli > C mv.c
1  #include <stdio.h>
2
3  int main(int argc, char *argv[]) {
4      if (argc != 3) {
5          fprintf(stderr, "사용법: %s <원본파일/디렉토리> <목적지>\n", argv[0]);
6          return 1;
7      }
8
9      if (rename(argv[1], argv[2]) != 0) {
10         perror("이동 실패");
11         return 1;
12     }
13
14     return 0;
15 }
16
```

- `rename()` 함수는 파일 또는 디렉토리의 이름을 변경하거나 위치를 이동시킴
- 원본 경로가 존재해야 하며, 목적지 경로가 이미 존재하면 실패할 수 있음
- 실패 시 `perror()` 로 에러 메시지 출력

27. stat

```
C stat.c
cli > C stat.c
1  #include <stdio.h>
2  #include <sys/stat.h>
3  #include <time.h>
4
5  int main(int argc, char *argv[]) {
6      if (argc != 2) {
7          fprintf(stderr, "사용법: %s <파일명>\n", argv[0]);
8          return 1;
9      }
10
11     struct stat st;
12     if (stat(argv[1], &st) == -1) {
13         perror("stat 오류");
14         return 1;
15     }
16
17     printf("파일: %s\n", argv[1]);
18     printf("파일 크기: %ld 바이트\n", st.st_size);
19     printf("블록 크기: %ld 바이트\n", st.st_blksize);
20     printf("블록 수: %ld\n", st.st_blocks);
21     printf("링크 수: %ld\n", st.st_nlink);
22     printf("아이노드 번호: %ld\n", st.st_ino);
23
24     printf("파일 권한: ");
25     printf((S_ISDIR(st.st_mode)) ? "d" : "-");
26     printf((st.st_mode & S_IRUSR) ? "r" : "-");
27     printf((st.st_mode & S_IWUSR) ? "w" : "-");
28     printf((st.st_mode & S_IXUSR) ? "x" : "-");
29     printf((st.st_mode & S_IRGRP) ? "r" : "-");
30     printf((st.st_mode & S_IWGRP) ? "w" : "-");
31     printf((st.st_mode & S_IXGRP) ? "x" : "-");
32     printf((st.st_mode & S_IROTH) ? "r" : "-");
33     printf((st.st_mode & S_IWOTH) ? "w" : "-");
34     printf((st.st_mode & S_IXOTH) ? "x" : "-");
35     printf("\n");
36
37     printf("마지막 접근 시간: %s", ctime(&st.st_atime));
38     printf("마지막 수정 시간: %s", ctime(&st.st_mtime));
39     printf("상태 변경 시간: %s", ctime(&st.st_ctime));
40
41     return 0;
42 }
```

파일/디렉토리 상태 정보 출력

- `stat()` 함수로 파일 상태 정보 읽기
- 파일 크기, 블록 크기, 링크 수, 아이노드 번호 등 출력
- 권한을 `rw-rw-rwx` 형식으로 표시
- 접근(access), 수정(modify), 상태 변경(change) 시간 출력 (`ctime()` 사용)

28. kill

```
C kill.c ×
cli > C kill.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <signal.h>
4
5  int main(int argc, char *argv[]) {
6      if (argc != 2) {
7          fprintf(stderr, "사용법: %s <PID>\n", argv[0]);
8          return 1;
9      }
10
11     int pid = atoi(argv[1]);
12
13     if (kill(pid, SIGKILL) == -1) {
14         perror("프로세스 종료 실패");
15         return 1;
16     }
17
18     printf("PID %d 프로세스가 종료되었습니다.\n", pid);
19     return 0;
20 }
21
```

프로세스 종료

- `kill(pid, SIGKILL)` 은 해당 프로세스를 강제로 즉시 종료합니다.
- `SIGTERM` 을 보내면 프로세스가 종료 요청을 받고 정리 후 종료할 기회를 갖지만, `SIGKILL` 은 무조건 즉시 종료시킵니다.

29. find

```
C find.c x
cli > C find.c
1  #include <stdio.h>
2  #include <dirent.h>
3  #include <sys/stat.h>
4  #include <string.h>
5
6  void find(const char *path, const char *name) {
7      DIR *d = opendir(path);
8      if (!d) return;
9      struct dirent *e;
10     while ((e = readdir(d))) {
11         if (!strcmp(e->d_name, ".") || !strcmp(e->d_name, "..")) continue;
12
13         char p[1024];
14         snprintf(p, sizeof(p), "%s/%s", path, e->d_name);
15
16         if (!strcmp(e->d_name, name))
17             printf("%s\n", p);
18
19         struct stat st;
20         if (!stat(p, &st) && S_ISDIR(st.st_mode))
21             find(p, name);
22     }
23     closedir(d);
24 }
25
26 int main(int c, char **v) {
27     if (c != 3) return 1;
28     find(v[1], v[2]);
29     return 0;
30 }
31
```

파일 검색

- 첫 번째 인자로 탐색 시작 경로, 두 번째 인자로 찾을 파일 이름을 받음
- 디렉터리를 열고(`opendir`), 각 항목을 읽음(`readdir`)
- "." 과 ".." 제외
- 파일 이름이 찾는 이름과 같으면 출력
- 디렉터리면 재귀적으로 `find_files` 호출하여 하위 탐색
- `stat` 으로 파일/디렉터리 구분

30. wc

```
C wc.c
cli > C wc.c
1  #include <stdio.h>
2  #include <ctype.h>
3
4  int main(int argc, char *argv[]) {
5      if (argc != 2) {
6          fprintf(stderr, "사용법: %s <파일이름>\n", argv[0]);
7          return 1;
8      }
9
10     FILE *fp = fopen(argv[1], "r");
11     if (!fp) {
12         perror("파일 열기 실패");
13         return 1;
14     }
15
16     int lines = 0, words = 0, bytes = 0;
17     int in_word = 0;
18     int c;
19
20     while ((c = fgetc(fp)) != EOF) {
21         bytes++;
22
23         if (c == '\n')
24             lines++;
25
26         if (isspace(c))
27             in_word = 0;
28         else if (!in_word) {
29             in_word = 1;
30             words++;
31         }
32     }
33
34     fclose(fp);
35
36     printf("%d %d %d %s\n", lines, words, bytes, argv[1]);
37
38     return 0;
39 }
40
```

행/단어/문자 수 세는 명령어

- 파일을 문자 단위로 읽으면서
- 줄바꿈 `\n` 나오면 줄 수 증가
- 공백 문자를 기준으로 단어 개수 세기 (단어의 시작을 감지)
- 읽은 문자 수를 바이트 수로 사용
- 결과 출력 형식은 `wc` 와 비슷하게 `줄수 단어수 바이트수 파일명` 순서

31. sort

```
C sort.c x
cli > C sort.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX_LINES 10000
6  #define MAX_LEN 1024
7
8  // qsort용 비교 함수 (문자열 사전 순)
9  int cmp(const void *a, const void *b) {
10     const char * const pa = a;
11     const char * const pb = b;
12     return strcmp(pa, pb);
13 }
14
15 int main(int argc, char *argv[]) {
16     if (argc != 2) {
17         fprintf(stderr, "사용법: %s <파일이름>\n", argv[0]);
18         return 1;
19     }
20
21     FILE *fp = fopen(argv[1], "r");
22     if (!fp) {
23         perror("파일 열기 실패");
24         return 1;
25     }
26
27     char *lines[MAX_LINES];
28     int count = 0;
29     char buffer[MAX_LEN];
30
```

```
31     while (fgets(buffer, sizeof(buffer), fp) && count < MAX_LINES) {
32         // 줄 끝 개행 문자 제거
33         size_t len = strlen(buffer);
34         if (len > 0 && buffer[len - 1] == '\n')
35             buffer[len - 1] = '\0';
36
37         lines[count] = malloc(len);
38         if (!lines[count]) {
39             perror("메모리 할당 실패");
40             fclose(fp);
41             return 1;
42         }
43         strcpy(lines[count], buffer);
44         count++;
45     }
46
47     fclose(fp);
48
49     qsort(lines, count, sizeof(char *), cmp);
50
51     for (int i = 0; i < count; i++) {
52         printf("%s\n", lines[i]);
53         free(lines[i]);
54     }
55
56     return 0;
57 }
```

파일 줄을 정렬해서 출력

- 최대 10,000줄, 각 줄 최대 1023자(버퍼 크기)
- fgets() 로 한 줄씩 읽음
- 개행 문자 제거 후 동적 할당하여 저장
- qsort() 로 정렬 (기본 문자열 비교)
- 정렬된 줄 출력 후 메모리 해제

32. ls

```
C ls.c x
cli > C ls.c
1  #include <stdio.h>
2  #include <dirent.h> // 디렉토리 관련 함수
3  #include <errno.h>
4  #include <string.h>
5
6  int main(int argc, char *argv[]) {
7      const char *path = "."; // 기본 경로는 현재 디렉토리
8
9      if (argc > 1) {
10         path = argv[1]; // 인자가 있으면 해당 경로 사용
11     }
12
13     DIR *dir = opendir(path);
14     if (dir == NULL) {
15         fprintf(stderr, "디렉토리 열기 오류: %s\n", strerror(errno));
16         return 1;
17     }
18
19     struct dirent *entry;
20     while ((entry = readdir(dir)) != NULL) {
21         // 현재 디렉토리(.)와 부모 디렉토리(..)는 건너뛰기 원하면 제거 가능
22         if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
23             continue;
24
25         printf("%s\n", entry->d_name);
26     }
27
28     closedir(dir);
29     return 0;
30 }
31
```

현재 디렉토리 파일 , 디렉토리 이름 출력

- `opendir()` 로 디렉토리 핸들 얻기
- `readdir()` 로 한 항목씩 읽기
- 항목 이름을 출력 (`.` 와 `..` 은 생략)
- `closedir()` 로 디렉토리 닫기

33. ls -l

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <dirent.h>
4  #include <sys/stat.h>
5  #include <pwd.h>
6  #include <grp.h>
7  #include <time.h>
8  #include <unistd.h>
9
10 void print_permissions(mode_t mode) {
11     char perms[11] = "-----";
12
13     if (S_ISDIR(mode)) perms[0] = 'd';
14     else if (S_ISLNK(mode)) perms[0] = 'l';
15     else if (S_ISCHR(mode)) perms[0] = 'c';
16     else if (S_ISBLK(mode)) perms[0] = 'b';
17
18     if (mode & S_IRUSR) perms[1] = 'r';
19     if (mode & S_IWUSR) perms[2] = 'w';
20     if (mode & S_IXUSR) perms[3] = 'x';
21
22     if (mode & S_IRGRP) perms[4] = 'r';
23     if (mode & S_IWGRP) perms[5] = 'w';
24     if (mode & S_IXGRP) perms[6] = 'x';
25
26     if (mode & S_IROTH) perms[7] = 'r';
27     if (mode & S_IWOTH) perms[8] = 'w';
28     if (mode & S_IXOTH) perms[9] = 'x';
29
30     printf("%s ", perms);
31 }
32
33 int main(int argc, char *argv[]) {
34     const char *dir_path = "."; // 기본 현재 디렉토리
35     if (argc > 1) dir_path = argv[1];
36
37     DIR *dir = opendir(dir_path);
38     if (!dir) {
39         perror("디렉토리 열기 실패");
40         return 1;
41     }
42
43     struct dirent *entry;
44     while ((entry = readdir(dir)) != NULL) {
45         char fullpath[1024];
46         snprintf(fullpath, sizeof(fullpath), "%s/%s", dir_path, entry->d_name);
47
48         struct stat st;
49         if (lstat(fullpath, &st) == -1) {
50             perror("stat 실패");
51             continue;
52         }
53
54         print_permissions(st.st_mode);
55         printf("%2lu ", (unsigned long)st.st_nlink);
56
57         struct passwd *pw = getpwuid(st.st_uid);
58         struct group *gr = getgrgid(st.st_gid);
59
60         printf("%-8s %-8s ", pw ? pw->pw_name : "?", gr ? gr->gr_name : "?");
61         printf("%8lld ", (long long)st.st_size);
62
63         char timebuf[64];
64         struct tm *tm = localtime(&st.st_mtime);
65         strftime(timebuf, sizeof(timebuf), "%b %d %H:%M", tm);
66         printf("%s ", timebuf);
67
68         printf("%s", entry->d_name);
69
70         // 심볼릭 링크면 -> 링크 타겟도 출력
71         if (S_ISLNK(st.st_mode)) {
72             char linktarget[1024];
73             ssize_t len = readlink(fullpath, linktarget, sizeof(linktarget)-1);
74             if (len != -1) {
75                 linktarget[len] = '\0';
76                 printf(" -> %s", linktarget);
77             }
78         }
79
80         printf("\n");
81     }
82
83     closedir(dir);
84     return 0;
85 }
```

현재 디렉토리 파일 디렉토리 이름 상세 출력

- 디렉토리 내 모든 파일에 대해 `lstat()` 호출
- 권한, 링크 수, 소유자명, 그룹명, 크기, 수정 시간, 파일 이름 출력
- 심볼릭 링크면 링크 대상 경로도 표시
- `argv[1]` 에 디렉토리 경로 지정 가능 (없으면 현재 디렉토리)

34. ls -a

```
cli > C ls-a.c
1  #include <stdio.h>
2  #include <dirent.h>
3
4  int main(int argc, char *argv[]) {
5      const char *dir_path = ".";
6      if (argc > 1) dir_path = argv[1];
7
8      DIR *dir = opendir(dir_path);
9      if (!dir) {
10         perror("디렉토리 열기 실패");
11         return 1;
12     }
13
14     struct dirent *entry;
15     while ((entry = readdir(dir)) != NULL) {
16         printf("%s\n", entry->d_name); // 숨김파일 포함해서 모두 출력
17     }
18
19     closedir(dir);
20     return 0;
21 }
```

현재 디렉토리 파일 , 디렉토리 이름 , 숨김파일 출력

- `readdir` 는 기본적으로 숨김파일 포함 모든 항목 반환
- `ls -a` 는 별도의 필터링 없이 모든 항목 보여줌
- `argv[1]` 로 경로 지정 가능 (없으면 현재 디렉토리)

35.chmod

```
cli > C chmod.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/stat.h>
4
5  int main(int argc, char *argv[]) {
6      if (argc != 3) {
7          fprintf(stderr, "사용법: %s <파일이름> <권한(8진수)>\n", argv[0]);
8          return 1;
9      }
10
11     const char *filename = argv[1];
12     // 문자열 8진수 -> 정수 변환
13     mode_t mode = strtol(argv[2], NULL, 8);
14
15     if (chmod(filename, mode) == -1) {
16         perror("chmod 실패");
17         return 1;
18     }
19
20     return 0;
21 }
22
```

파일 , 디렉토리의 권한을 변경

- 두 번째 인자를 `strtol` 로 8진수 변환(`chmod` 는 `mode_t` 타입)
- `chmod()` 호출로 권한 변경
- 실패 시 `perror` 출력

36. id

```
cli > C id.c
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <sys/types.h>
4  #include <pwd.h>
5  #include <grp.h>
6
7  int main() {
8      uid_t uid = getuid();
9      gid_t gid = getgid();
10     struct passwd *pw = getpwuid(uid);
11     struct group *gr = getgrgid(gid);
12
13     if (!pw || !gr) {
14         perror("사용자/그룹 정보 조회 실패");
15         return 1;
16     }
17
18     printf("uid=%d(%s) gid=%d(%s)\n", uid, pw->pw_name, gid, gr->gr_name);
19
20     // 보조 그룹 조회
21     int ngroups = 0;
22     getgrouplist(pw->pw_name, gid, NULL, &ngroups);
23
24     gid_t groups[ngroups];
25     getgrouplist(pw->pw_name, gid, groups, &ngroups);
26
27     printf("groups=");
28     for (int i = 0; i < ngroups; i++) {
29         struct group *g = getgrgid(groups[i]);
30         if (g) {
31             printf("%d(%s)", groups[i], g->gr_name);
32         } else {
33             printf("%d", groups[i]);
34         }
35         if (i != ngroups - 1)
36             printf(",");
37     }
38     printf("\n");
39
40     return 0;
41 }
```

사용자의 정보

- `getuid()`, `getgid()` 로 현재 사용자 및 그룹 ID 가져오기
- `getpwuid()`, `getgrgid()` 로 이름 조회
- `getgrouplist()` 로 보조 그룹 목록 조회
- 보조 그룹 이름과 ID 출력

37. hostname

```
cli > C id.c
1  #include <stdio.h>
2  #include <unistd.h>
3  #include <sys/types.h>
4  #include <pwd.h>
5  #include <grp.h>
6
7  int main() {
8      uid_t uid = getuid();
9      gid_t gid = getgid();
10     struct passwd *pw = getpwuid(uid);
11     struct group *gr = getgrgid(gid);
12
13     if (!pw || !gr) {
14         perror("사용자/그룹 정보 조회 실패");
15         return 1;
16     }
17
18     printf("uid=%d(%s) gid=%d(%s)\n", uid, pw->pw_name, gid, gr->gr_name);
19
20     // 보조 그룹 조회
21     int ngroups = 0;
22     getgrouplist(pw->pw_name, gid, NULL, &ngroups);
23
24     gid_t groups[ngroups];
25     getgrouplist(pw->pw_name, gid, groups, &ngroups);
26
27     printf("groups=");
28     for (int i = 0; i < ngroups; i++) {
29         struct group *g = getgrgid(groups[i]);
30         if (g) {
31             printf("%d(%s)", groups[i], g->gr_name);
32         } else {
33             printf("%d", groups[i]);
34         }
35         if (i != ngroups - 1)
36             printf(",");
37     }
38     printf("\n");
39
40     return 0;
41 }
```

현재 시스템 호스트의 이름

- `getuid()`, `getgid()` 로 현재 사용자 및 그룹 ID 가져오기
- `getpwuid()`, `getgrgid()` 로 이름 조회
- `getgrouplist()` 로 보조 그룹 목록 조회
- 보조 그룹 이름과 ID 출력

38. diff

```
cli > C diff.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX_LINE 1024
6
7  int main(int argc, char *argv[]) {
8      if (argc != 3) {
9          fprintf(stderr, "사용법: %s <파일1> <파일2>\n", argv[0]);
10         return 1;
11     }
12
13     FILE *fp1 = fopen(argv[1], "r");
14     FILE *fp2 = fopen(argv[2], "r");
15
16     if (!fp1 || !fp2) {
17         perror("파일 열기 실패");
18         return 1;
19     }
20
21     char line1[MAX_LINE], line2[MAX_LINE];
22     int lineno = 1;
23
24     while (1) {
25         char *res1 = fgets(line1, sizeof(line1), fp1);
26         char *res2 = fgets(line2, sizeof(line2), fp2);
27
28         if (!res1 && !res2) break; // 둘 다 끝났으면 종료
29
30         // 줄바꿈 문자 제거
31         if (res1) line1[strcspn(line1, "\n")] = '\0';
32         if (res2) line2[strcspn(line2, "\n")] = '\0';
33     }
```

```
34     if (!res1) {
35         printf("%s: EOF\n", argv[1]);
36         printf("%d: %s\n", lineno, line2);
37     } else if (!res2) {
38         printf("%s: EOF\n", argv[2]);
39         printf("%d: %s\n", lineno, line1);
40     } else if (strcmp(line1, line2) != 0) {
41         printf("%d:\n", lineno);
42         printf("< %s\n", line1);
43         printf("> %s\n", line2);
44     }
45
46     lineno++;
47 }
48
49 fclose(fp1);
50 fclose(fp2);
51 return 0;
52 }
53
```

두 파일을 줄 단위로 비교

- 두 파일을 동시에 한 줄씩 읽음
- 줄바꿈 문자 `\n` 제거
- 두 줄을 `strcmp()` 로 비교
- 다르면 몇 번째 줄인지, 각 파일의 내용을 출력
- 어느 한 쪽이 먼저 끝나도 알려줌

39. du

```
cli > C du.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/stat.h>
4
5  void usage(const char *prog) {
6      fprintf(stderr, "사용법: %s <파일 또는 디렉토리>\n", prog);
7      exit(1);
8  }
9
10 unsigned long filesize(const char *path) {
11     struct stat st;
12     if (stat(path, &st) != 0) {
13         perror("stat 실패");
14         return 0;
15     }
16     return st.st_size;
17 }
18
19 int main(int argc, char *argv[]) {
20     if (argc != 2) usage(argv[0]);
21     unsigned long sz = filesize(argv[1]);
22     printf("%lu\t%s\n", sz, argv[1]);
23     return 0;
24 }
```

디렉토리/파일의 디스크의 사용량

1. `stat()` 함수로 파일이나 디렉토리의 크기(`st_size`)를 가져온다.
2. 디렉토리 내부 파일 크기는 합산하지 않고, 디렉토리 엔트리 자체의 크기만 출력한다.
3. 인자 오류 시 사용법을 출력하고 종료하며, 크기와 경로를 바이트 단위로 출력한다.

40. sum

```
cli > C sum.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char *argv[]) {
5      if (argc != 2) {
6          fprintf(stderr, "사용법: %s <파일명>\n", argv[0]);
7          return 1;
8      }
9
10     FILE *fp = fopen(argv[1], "rb");
11     if (!fp) {
12         perror("파일 열기 실패");
13         return 1;
14     }
15
16     unsigned long checksum = 0;
17     unsigned long size = 0;
18     int ch;
19
20     while ((ch = fgetc(fp)) != EOF) {
21         checksum += (unsigned char)ch;
22         size++;
23     }
24
25     fclose(fp);
26
27     unsigned long blocks = (size + 1023) / 1024; // 1024바이트 블록 수 계산
28
29     printf("%lu %lu %s\n", checksum, blocks, argv[1]);
30
31     return 0;
32 }
33
```

체크섬과 블록 수 계산하는 명령어

- `fgetc()` 로 파일의 모든 바이트를 읽으며 checksum 계산
- 파일 크기(size)를 카운트
- 1024로 나눠서 블록 수 계산
- checksum, 블록 수, 파일명을 출력

41. ln

```
cli > C ln.c
1  #include <stdio.h>
2  #include <unistd.h>
3
4  int main(int argc, char *argv[]) {
5      if (argc != 3) {
6          fprintf(stderr, "사용법: %s <기존파일> <새링크>\n", argv[0]);
7          return 1;
8      }
9
10     if (link(argv[1], argv[2]) != 0) {
11         perror("링크 생성 실패");
12         return 1;
13     }
14
15     printf("하드링크 %s → %s 생성 완료\n", argv[2], argv[1]);
16
17     return 0;
18 }
19
```

하드링크 / 심볼릭링크 만드는 명령어

- `link(기존파일, 새링크)` 함수로 하드링크 생성
- 인자 2개 (원본파일, 새링크명) 확인
- 실패 시 `perror` 로 오류 메시지 출력
- 성공 시 안내 메시지 출력

42. rm -r

디렉토리와 그 안의 파일을 재귀적으로 삭제

```
cli> C rm -r.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <dirent.h>
6  #include <sys/stat.h>
7
8  void remove_recursive(const char *path) {
9      struct stat st;
10     if (lstat(path, &st) != 0) {
11         perror("stat 실패");
12         return;
13     }
14
15     if (S_ISDIR(st.st_mode)) {
16         // 디렉토리면 내부 항목 먼저 삭제
17         DIR *dir = opendir(path);
18         if (!dir) {
19             perror("디렉토리 열기 실패");
20             return;
21         }
22
23         struct dirent *entry;
24         char fullpath[1024];
25
26         while ((entry = readdir(dir)) != NULL) {
27             // . , .. 는 건너뛰기
28             if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
29                 continue;
30
31             snprintf(fullpath, sizeof(fullpath), "%s/%s", path, entry->d_name);
32             remove_recursive(fullpath);
33         }
34
35         closedir(dir);
36
37         // 디렉토리 비우고 난 뒤 삭제
38         if (rmdir(path) != 0)
39             perror("디렉토리 삭제 실패");
40     } else {
41         // 일반 파일이나 링크 삭제
42         if (unlink(path) != 0)
43             perror("파일 삭제 실패");
44     }
45 }
46
47 int main(int argc, char *argv[]) {
48     if (argc != 2) {
49         fprintf(stderr, "사용법: %s <삭제할 경로>\n", argv[0]);
50         return 1;
51     }
52
53     remove_recursive(argv[1]);
54     return 0;
55 }
56
```

- `lstat()` 으로 경로의 타입 확인
 - 디렉토리: `opendir()` → `readdir()` 로 내부 파일 탐색
 - 파일/링크: `unlink()` 로 삭제
- 디렉토리는 내부 항목 전부 삭제 후 `rmdir()` 로 삭제
- 재귀 호출 방식

43. rm -f

```
cli > C rm -f.c
1  #include <stdio.h>
2  #include <unistd.h>
3
4  int main(int argc, char *argv[]) {
5      if (argc < 2) {
6          fprintf(stderr, "사용법: %s <파일1> [파일2 ...]\n", argv[0]);
7          return 1;
8      }
9
10     for (int i = 1; i < argc; i++) {
11         // unlink() 호출, 실패해도 에러 출력 안 함
12         unlink(argv[i]);
13     }
14
15     return 0;
16 }
17
```

파일이 없어도 에러 출력 없는 삭제 명령어

- `unlink()` 호출
- 삭제 실패해도 `perror()` 같은 에러 메시지 출력 없음
- 인자 여러 개 받아서 순차적으로 삭제

44. rm -rf

```
cli > C rm -rf.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <unistd.h>
5  #include <dirent.h>
6  #include <sys/stat.h>
7
8  void remove_recursive_force(const char *path) {
9      struct stat st;
10     if (lstat(path, &st) != 0) {
11         // 파일/디렉토리가 없거나 권한 문제라도 무시 (-f 옵션)
12         return;
13     }
14
15     if (S_ISDIR(st.st_mode)) {
16         DIR *dir = opendir(path);
17         if (!dir) {
18             // 열기 실패해도 무시
19             return;
20         }
21
22         struct dirent *entry;
23         char fullpath[1024];
24
25         while ((entry = readdir(dir)) != NULL) {
26             if (strcmp(entry->d_name, ".") == 0 || strcmp(entry->d_name, "..") == 0)
27                 continue;
28
29             snprintf(fullpath, sizeof(fullpath), "%s/%s", path, entry->d_name);
30             remove_recursive_force(fullpath);
31         }
32
33         closedir(dir);
34         rmdir(path); // 실패해도 무시
35     } else {
36         unlink(path); // 실패해도 무시
37     }
38 }
39
```

파일이 없어도 에러 출력 없이 강제로 재귀 삭제
하는 명령어

```
40 int main(int argc, char *argv[]) {
41     if (argc < 2) {
42         fprintf(stderr, "사용법: %s <삭제할 경로> [추가 경로...]\n", argv[0]);
43         return 1;
44     }
45
46     for (int i = 1; i < argc; i++) {
47         remove_recursive_force(argv[i]);
48     }
49
50     return 0;
51 }
52
```

- `lstat()` 실패해도 무시 → `-f` 옵션 효과
- 디렉토리는 재귀적으로 내부 삭제
- 삭제 실패해도 출력 없이 조용히 넘어감

45. env

```
cli > C env.c
1  #include <stdio.h>
2
3  extern char **environ;
4
5  int main() {
6      for (char **env = environ; *env != NULL; env++) {
7          printf("%s\n", *env);
8      }
9      return 0;
10 }
11
```

현재 환경변수 목록을 출력

1. `extern char **environ;`
 - 환경변수들이 저장된 전역 변수 `environ` 을 참조합니다.
 - `environ` 은 문자열 배열로, 각 문자열은 "키=값" 형태의 환경변수를 가리킵니다.
2. `for (char **env = environ; *env != NULL; env++)`
 - `environ` 배열을 처음부터 끝(NULL)까지 순회합니다.
 - 각 `*env` 는 환경변수 문자열입니다.
3. `printf("%s\n", *env);`
 - 각 환경변수를 한 줄씩 출력합니다.

46. split

```
cli > C split.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define CHUNK_SIZE 1024 // 1KB씩 자르기
5
6  int main(int argc, char *argv[]) {
7      if (argc != 2) {
8          fprintf(stderr, "사용법: %s <파일명>\n", argv[0]);
9          return 1;
10     }
11
12     FILE *src = fopen(argv[1], "rb");
13     if (!src) {
14         perror("파일 열기 실패");
15         return 1;
16     }
17
18     char buffer[CHUNK_SIZE];
19     int part_num = 0;
20     size_t nread;
21
22     while ((nread = fread(buffer, 1, CHUNK_SIZE, src)) > 0) {
23         char filename[256];
24         snprintf(filename, sizeof(filename), "part_%03d", part_num++);
25
26         FILE *dst = fopen(filename, "wb");
27         if (!dst) {
28             perror("출력 파일 생성 실패");
29             fclose(src);
30             return 1;
31         }
32
33         fwrite(buffer, 1, nread, dst);
34         fclose(dst);
35     }
36
37     fclose(src);
38     return 0;
39 }
```

큰 파일을 일정 크기로 나누는 명령어

- 입력 파일을 1024바이트씩 읽음
- `part_000`, `part_001`, ... 같은 이름으로 조각 파일 생성
- 마지막 조각은 1024바이트 미만일 수 있음

47. free

```
cli > C free.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      FILE *fp = fopen("/proc/meminfo", "r");
6      if (!fp) {
7          perror("meminfo 열기 실패");
8          return 1;
9      }
10
11     char line[256];
12     while (fgets(line, sizeof(line), fp)) {
13         // MemTotal, MemFree, MemAvailable, Buffers, Cached 등만 출력
14         if (strncmp(line, "MemTotal:", 9) == 0 ||
15             strncmp(line, "MemFree:", 8) == 0 ||
16             strncmp(line, "MemAvailable:", 13) == 0 ||
17             strncmp(line, "Buffers:", 8) == 0 ||
18             strncmp(line, "Cached:", 7) == 0) {
19             printf("%s", line);
20         }
21     }
22
23     fclose(fp);
24     return 0;
25 }
26
```

메모리의 사용 현황

`/proc/meminfo` 에서 메모리 관련 주요 항목을 읽어 출력

48. tee

```
cli > C tee.c
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char *argv[]) {
5      if (argc != 2) {
6          fprintf(stderr, "사용법: %s <출력파일>\n", argv[0]);
7          return 1;
8      }
9
10     FILE *fp = fopen(argv[1], "w");
11     if (!fp) {
12         perror("파일 열기 실패");
13         return 1;
14     }
15
16     int c;
17     while ((c = getchar()) != EOF) {
18         putchar(c);    // 표준 출력으로 출력
19         fputc(c, fp);   // 파일에도 쓰기
20     }
21
22     fclose(fp);
23     return 0;
24 }
25
```

출력 결과를 파일에 저장하면서 터미널에도 출력

- 인자로 받은 파일을 쓰기 모드로 열고
- 표준 입력을 한 문자씩 읽어서
- 표준 출력과 파일 모두에 씁니다.
- EOF 입력 시 종료됩니다.

49. time

```
cli > C time.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <sys/time.h>
4  #include <sys/resource.h>
5  #include <sys/wait.h>
6  #include <unistd.h>
7
8  int main(int argc, char *argv[]) {
9      if (argc < 2) {
10         fprintf(stderr, "사용법: %s <명령어> [인자...]\n", argv[0]);
11         return 1;
12     }
13
14     struct timeval start, end;
15     gettimeofday(&start, NULL);
16
17     pid_t pid = fork();
18     if (pid < 0) {
19         perror("fork 실패");
20         return 1;
21     }
22
23     if (pid == 0) {
24         // 자식 프로세스: 명령어 실행
25         execvp(argv[1], &argv[1]);
26         perror("exec 실패");
27         exit(1);
28     } else {
29         // 부모 프로세스: 자식 대기
30         int status;
31         waitpid(pid, &status, 0);
32
33         gettimeofday(&end, NULL);
34
35         double elapsed = (end.tv_sec - start.tv_sec) + (end.tv_usec - start.tv_usec) / 1e6;
36
37         printf("실행 시간: %.6f초\n", elapsed);
38
39         return WIFEXITED(status) ? WEXITSTATUS(status) : 1;
40     }
41 }
```

명령어 실행 시간 측정

- `fork()` 로 자식 프로세스 생성
- 자식 프로세스는 `execvp()` 로 인자로 받은 명령어 실행
- 부모 프로세스는 `waitpid()` 로 자식 종료 대기
- `gettimeofday()` 로 시작과 끝 시간을 측정해서 실행 시간 계산
- 실행 시간이 초 단위로 출력됨

50. yes

```
cli > C yes.c
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(int argc, char *argv[]) {
5      const char *text = "y";
6
7      if (argc > 1) {
8          text = argv[1];
9      }
10
11     while (1) {
12         printf("%s\n", text);
13     }
14
15     return 0;
16 }
17
```

문자열 반복 출력

- 인자가 없으면 기본값 `y` 출력
- 인자가 있으면 그 문자열을 출력
- `while (1)` 무한루프로 계속 출력

점수

깃허브 12점 + 명령어 15점 = 27점

레포트 1주차 미제출과 늦게 제출한 레포트들 때문에 12점을 생각했고
명령어는 50개의 명령어를 다 구현해서 15점이라 생각했습니다