

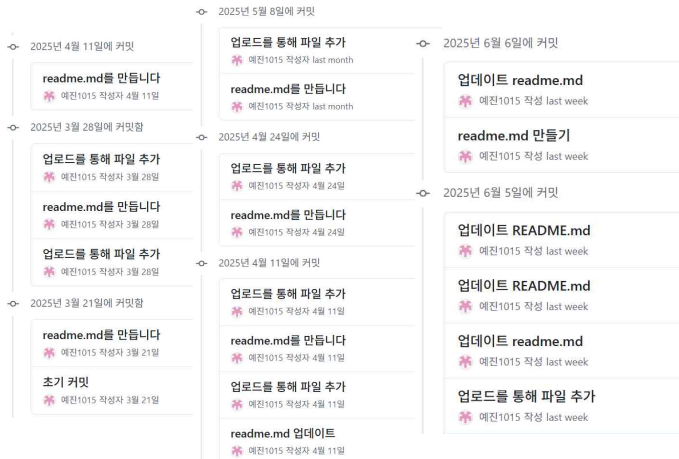
# 시스템 프로그래밍

50개 구현 & 깃허브 정리

<https://github.com/yejin1015/Systemprogram>

2023864032 양예진















# 깃허브 히스토리



Apr 11, 2025, 12:09 AM GMT+9

0404파일을 올려야할 날짜가  
히스토리에  
보이지 않는 이유는  
0404 파일이 4월11일 오전 12시9분  
에 업로드 됐기때문이다  
시간에 맞춰 제대로 업로드 되었다

# 업로드된 파일들

|   |                      |   |   |
|---|----------------------|---|---|
| yejin1015 Update readme.md  |                      | 6cfe06d · now   |  |
|  0321      | Update readme.md     | last week   |   |
|  0328      | Update readme.md     | 8 hours ago   |   |
|  0404      | Update readme.md     | 8 hours ago   |   |
|  0411      | Update readme.md     | 7 hours ago   |   |
|  0418      | Update readme.md     | 7 hours ago   |   |
|  0502      | Update readme.md     | 7 hours ago   |   |
|  0509      | Update readme.md     | 7 hours ago   |   |
|  0516      | Add files via upload | 7 hours ago   |   |
|  0523      | Update readme.md     | 7 hours ago   |   |
|  0530      | Update readme.md     | now   |   |
|  README.md | Update README.md     | 7 hours ago   |   |
|   |                      |   |   |
|  README    |                      |  |   |



시스템 프로그래밍 핵심 요약

0516 파일을 업로드하는 것을 깜빡  
하여 정해진 날짜 보다 늦어  
지각하였다

모든 폴더에 readme.md가 꾸며져  
있다

# readme.md 작성

Systemprogram / 0411 /  
readme.md

## 🔥 Vi 편집기 핵심 요약

Vi는 명령 모드와 입력 모드를 구분하여 사용하는 CU 기반 텍스트 편집기입니다.

### ★ 기본 커서 이동

- h, j, k, l : ←, ↓, ↑, →
- w, b : 단어 앞 / 뒤로
- e, E : 마지막 줄 / n번째 줄
- Ctrl+f/B : 화면 단위 이동

### 🔍 검색

- /문자 : 아래 방향 검색
- :문자 : 위 방향 검색

### 🔧 입력 모드 진입

- i, a : 현재 위치 앞 / 뒤 입력
- I, A : 줄의 앞 / 뒤 입력
- o, O : 아래 / 위에 새 줄 입력

### 🔧 수정 & 삭제

- x, dd, dw : 문자 / 줄 / 단어 삭제
- cc, cw : 줄 / 단어 수정

Systemprogram / 0528 /  
readme.md

## 🐧 Linux 파일 관리 명령어 정리

리눅스에서 자주 사용하는 파일 관리 명령어인 cp, mv, rm, ln 을 예시와 함께 정리한 문서입니다.

### 📁 cp — 파일 복사 (copy)

파일이나 디렉터리를 복사할 때 사용하는 명령어입니다.

#### ✅ 사용법

cp [옵션] 원본 대상

#### 🔧 주요 옵션

- i : 덮어쓰기 전에 확인하는 대화형 모드

#### ★ 예시

cp a.txt b.txt # a.txt를 b.txt로 복사  
cp -i a.txt b.txt # 덮어쓰기 전에 사용자에게 확인 요청

### 📁 mv — 파일 이동 또는 이름 변경 (move)

파일을 다른 위치로 옮기거나 파일명을 바꿀 때 사용하는 명령어입니다.

#### ✅ 사용법

mv [옵션] 원본 대상

#### 🔧 주요 옵션

- f : 기존 파일의 이름 필요 없이 덮어쓰기 저 사용자에게 확인

그날 배운 이론, 명령어 그리고 파일을 업로드 하는 방법 등을 작성하여  
보기 좋게 꾸며봄

# 구현 - 50개

- C basename.c
- C cat.c
- C cd.c
- C chmod.c
- C clear.c
- C cp.c
- C date.c
- C dirname.c
- C du.c
- C echo.c
- C env.c
- C export.c
- C false.c
- C find.c
- C grep.c
- C head.c
- C history.c
- C hostname.c
- C id.c
- C ls -a.c
- C ls -d.c
- C ls -r.c
- C ls -s.c
- C ls.c
- C make.c
- C man.c
- C mkdir.c
- C mv.c
- C printf.c
- C pwd.c
- C rm.c
- C set.c

- C sleep.c
- C stat.c
- C tail.c
- C test.c
- C time.c
- C touch.c
- C true.c
- C tty.c
- C uname.c
- C uptime.c
- C vi.c
- C wait.c
- C wc.c
- C whatis.c
- C which.c
- C who.c
- C whoami.c
- C yes.c

수업시간에 사용한 명령어를 구현하고  
ls 옵션 구현하고  
나머지는 검색해서 나온 명령어를  
c언어로 50개 모두 구현하였다

# ls 코드 -파일/디렉토리 목록 출력

## 주요 기능 설명

```
#include <stdio.h> // 표준 입출력 함수 사용을 위한 헤더 파일
#include <dirent.h> // 디렉토리 관련 함수(opendir, readdir 등) 사용을 위한 헤더 파일

int main() {
    // "." 은 현재 디렉토리를 의미한다.
    // 현재 디렉토리를 열고 DIR 포인터로 반환받는다.
    DIR *dir = opendir(".");

    // 디렉토리에서 항목(파일, 디렉토리 등)을 하나씩 읽기 위한 구조체 포인터 선언
    struct dirent *entry;

    // 디렉토리 열기에 실패한 경우 (예: 권한 없음, 디렉토리 존재하지 않음 등)
    if (dir == NULL) {
        printf("디렉토리를 열 수 없습니다.\n"); // 오류 메시지 출력
        return 1; // 비정상 종료 코드 반환
    }

    // 디렉토리 안의 모든 항목을 하나씩 읽음 (NULL이 반환되면 끝)
    while ((entry = readdir(dir)) != NULL) {
        // entry->d_name은 현재 항목의 이름 (파일명 또는 디렉토리명)을 나타냄
        printf("%s\n", entry->d_name); // 항목 이름 출력
    }

    // 디렉토리 스트림을 닫아 자원 해제
    closedir(dir);

    return 0; // 정상 종료
}
```

→ 현재 디렉토리(.)를 열고 디렉토리 스트림을 반환합니다

→ 디렉토리 안에 있는 항목들을 하나씩 읽어옵니다

→ 현재 읽은 항목의 이름(파일명 또는 디렉토리명)을 문자열로 가져옵니다

→ 항목 이름을 한 줄씩 출력합니다

→ 열었던 디렉토리를 닫습니다

# cat 코드 -파일 내용 출력

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    // FILE 포인터: 파일과 연결된 스트림을 나타내는 구조체 포인터
    FILE *fp;

    // 한 줄씩 읽기 위한 문자 배열 (버퍼) 선언
    char line[1024]; // 최대 1023자 + 널 문자('\0')

    // argc: 인자의 개수
    // argv: 문자열 배열로, 사용자가 입력한 인자들이 저장됨
    if (argc != 2) {
        printf("사용법: %s 파일이름\n", argv[0]); // 사용법 안내 출력
        return 1; // 비정상 종료
    }

    const char *filename = argv[1]; // argv[1]에 입력된 파일 이름을 변수에 저장

    // fopen: 파일을 읽기 모드("r")로 연다
    fp = fopen(filename, "r");
    if (fp == NULL) {
        perror("파일 열기 실패");
        return 1;
    }

    // fgets: 파일에서 한 줄씩 읽음
    while (fgets(line, sizeof(line), fp)) {
        printf("%s", line);
    }

    // 파일 스트림을 닫고 자원 해제
    fclose(fp);

    return 0;
}
```

## 주요 기능 설명

argc  
→ 실행 시 입력된 인자의 개수를 의미합니다 (예:  
프로그램명 + 파일명 → 2)

argv[1]  
→ 사용자가 입력한 파일 이름을 문자열로 받아옵니다

FILE \*fp = fopen(filename, "r")  
→ 입력받은 파일을 읽기 모드("r")로 엽니다

fgets(line, sizeof(line), fp)  
→ 파일에서 한 줄씩 읽어 line 버퍼에 저장합니다

fclose(fp);  
→ 파일 스트림을 닫고 자원을 해제합니다

# 총점

0516 파일 지각하였기 때문에

-1점 으로 14점

50개의 명령어를 c언어로

구현해냈으므로 15점

총 29점이라고 생각합니다



감사합니다

참고 : ChatGPT