LAB10
시스템 프로그래밍_류은경**Prof**

2021115744_권구태

1. 파일 data를 "표준 입력"처럼 받아 정렬하고, 결과를 표준 출력으로 출력하는 프로그램 작성

- $ sort < f1.txt 와 동일한 기능 수행

```
$ cat f1.txt
banana
apple
cherry
date
$ ./ex2 f1.txt
apple
banana
cherry
date
$
```

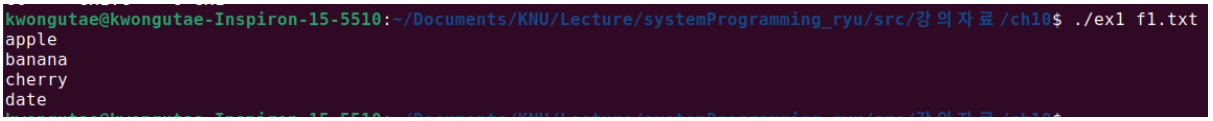
[코드 리뷰]

```c
/*
lab10_ex1
*/
#include <stdio.h>
#include <fcntl.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

int main(int argc, char *argv[])
{
   char line[100];
   char matrix[100][100];
   char temp[100];

   FILE *fp;

   fp = fopen(argv[1], "r");

   if(fp == NULL){
       fprintf(stderr, "Could not duplicate fd to 0\n");
       exit(1);
```

```
    } else {
        int i = 0;
        while (fgets(line, 100, fp) != NULL) {
            strcpy(matrix[i], line);
            i++;
        }

        // 문자열 정렬
        for (int j = 0; j < i; j++) {
            for (int k = j; k < i; k++) {
                if (strcmp(matrix[k], matrix[j]) < 0) {
                    strcpy(temp, matrix[k]);
                    strcpy(matrix[k], matrix[j]);
                    strcpy(matrix[j], temp);
                }
            }
        }

        for (int m = 0; m < i; m++) {
            printf("%s", matrix[m]);
        }
    }

    return 0;
}
```

argv[1] 파일을 읽기 모드로 열고, 해당 파일의 문자열들을 fgets로 한줄씩 읽어 matrix라는 2차원배열에 저장한 후, sorting 하고 출력.

[실행결과]



2. pipe.c를 확장하여 아래 명령어가 수행되도록 프로그램 작성

- $ cat data.txt | sort | head와 동일한 기능 수행
- 3개의 명령어 인자를 받아 순차적으로 파이프 연결
- fork, pipe, dup2, execvp 등의 사용
- 각 프로세스에서 사용하지 않는 fd는 close

$ ./ex3 cat f2.txt sort head apple banana blueberry cat dog grape kiwi lemon mango orange $	/* f2.txt banana apple zebra dog cat grape orange peach kiwi strawberry mango blueberry lemon pineapple watermelon */

[코드 리뷰]

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

#define oops(m, x) { perror(m); exit(x); }

int main(int argc, char *argv[]) {
    if (argc != 5) {
        fprintf(stderr, "usage: %s cmd1 arg1 cmd2 cmd3\n", argv[0]);
        exit(1);
    }

    int pipe1[2], pipe2[2];
    pid_t pid1, pid2;

    // pipe1: cat f2.txt -> sort
    if (pipe(pipe1) == -1)
        oops("pipe1 failed", 1);

    if ((pid1 = fork()) == -1)
        oops("fork1 failed", 2);

    if (pid1 == 0) {
        // child 1: run "cat f2.txt"
```

```
        dup2(pipe1[1], STDOUT_FILENO);
        close(pipe1[0]);
        close(pipe1[1]);

        char *cmd1_args[] = { argv[1], argv[2], NULL };  // cat f2.txt
        execvp(cmd1_args[0], cmd1_args);
        oops("execvp cmd1 failed", 3);
    }

    // pipe2: sort -> head
    if (pipe(pipe2) == -1)
        oops("pipe2 failed", 4);

    if ((pid2 = fork()) == -1)
        oops("fork2 failed", 5);

    if (pid2 == 0) {
        // child 2: run "sort"
        dup2(pipe1[0], STDIN_FILENO);
        dup2(pipe2[1], STDOUT_FILENO);
        close(pipe1[0]); close(pipe1[1]);
        close(pipe2[0]); close(pipe2[1]);

        char *cmd2_args[] = { argv[3], NULL };  // sort
        execvp(cmd2_args[0], cmd2_args);
        oops("execvp cmd2 failed", 6);
    }

    // parent: run "head"
    dup2(pipe2[0], STDIN_FILENO);
    close(pipe1[0]); close(pipe1[1]);
    close(pipe2[0]); close(pipe2[1]);

    char *cmd3_args[] = { argv[4], NULL };  // head
    execvp(cmd3_args[0], cmd3_args);
    oops("execvp cmd3 failed", 7);
}
```

파이프 2개 생성, sort와 head를 위한.
1번째 자식 프로세스 (cat f2.txt) 표준 출력을 pip1[1]으로 리다이렉트 -> 결과를 sort에게 전달]

2번째 자식 프로세스(sort)
부모 프로세스(head)

[실행 결과]