# 과제3 보고서

소프트웨어학과 32200588 김민지

〈시스템프로그래밍 1분반〉

## 0. 문제

- ‘mycat’, ‘mycp’ command 구현하기
- argvc, argv[] 사용
- 현재 디렉토리에 같은 이름의 파일이 이미 존재한다면 파일 생성하지 않음.
- 내용과 attribute까지 복사

## 1. mycat 구현

```c
/**
 * mycat.c  : cat command program
 * @author  : Minji Kim
 * @email   : 32200588@dankook.ac.kr
 * @version : 1.0
 * @data    : 2022.11.03
**/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64

int main(int argc, char *argv[]){
    int fd, read_size, write_size;
    char buf[MAX_BUF];

    if(argc != 2) {
        printf("잘못된 사용법입니다. (사용법 : %s 파일이름)\n", argv[0]);
        exit(-1);
    }
    fd = open(argv[1], O_RDONLY);
    if(fd<0) {
        printf("해당 파일 %s가 존재하지 않습니다. (ERRNO : %d)\n", argv[1], errno);
        exit(-1);
    }
    while(1) {
        read_size = read(fd, buf, MAX_BUF);
        if(read_size == 0)
            break;
        write_size = write(STDOUT_FILENO, buf, read_size);
    }
    close(fd);
    return 0;
}
```

### 1.1 cat command란?

cat은 인자로 파일 이름을 받아서, 파일의 전체 내용을 보여주는 명령이다.

사용법은 다음과 같다.

```
$ cat [파일명]
```

### 1.2 cat 구현을 위해

(1) argc, argv[] 사용

- argc는 main 함수에 전달된 인자의 개수이다. cat은 내용을 보기 위한 파일명 인자가 있어야 하므로, agrc가 2여야 한다.
- argv[0]은 ./cat이고, argv[1]은 파일명이다.

(2) 기본 패턴

- open -> read -> write -> close의 패턴으로 구현된다.

(3) while 루프

- 파일의 전체 내용을 읽기 위해 while 루프 사용이 필요하다.

### 1.3 코드 리뷰

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64
```

- 여러 기능을 사용하기 위한 헤더 파일 선언
- 파일의 내용을 읽을 최대 크기를 상수 MAX_BUF로 선언

```
int main(int argc, char *argv[]){
        int fd, read_size, write_size;
        char buf[MAX_BUF];
```

- 변수 fd (file descriptor), 파일을 read할 크기 read_size, 파일을 write할 크기 write_size와 read한 문자열을 잠시 저장할 배열 buf 선언

```
        if(argc !=2) {
                printf("잘못된 사용법입니다. (사용법 : %s 파일이름)₩n", argv[0]);
                exit(-1);
        }
```

- cat command는 argc가 2여야 한다. 따라서 2가 아니면 사용자에게 잘못된 사용법

이라는 사실과 사용법을 알려주고 종료한다.

```
        fd = open(argv[1], O_RDONLY);
        if(fd <0) {
                printf("해당 파일 %s가 존재하지 않습니다. (ERRNO : %d)₩n",
argv[1], errno);
                exit(-1);
        }
```

- argv[1] 즉, 사용자가 입력한 파일명을 읽기 전용으로 open한다.
- 정상적으로 파일이 open 되었다면, file descripter가 return되고, 실패하면, -1이 return된다.
- fd의 값이 음수면, 즉 open에 실패했다면 해당 파일이 존재하지 않는다는 사실과 에러 코드를 출력하고 종료한다.

```
        while(1) {
                read_size = read(fd, buf, MAX_BUF);
                if(read_size ==0)
                        break;
                write_size = write(STDOUT_FILENO, buf, read_size);
        }
        close(fd);
        return 0;
}
```

- 파일의 전체 내용을 읽기 위해 while 루프를 사용한다.
- fd의 내용을 MAX_BUF만큼 읽어 buf 문자열에 저장하고, 읽은 문자열의 크기는 리턴되어 read_size에 저장된다.
- buf 문자열을 read_size만큼 읽어 터미널에 출력한다.
- 만약, 문자열의 모든 내용을 읽어 offset이 문자열의 마지막에 있어 read_size가 0이라면 while문을 종료한다.
- 마지막으로 fd를 close하고 종료한다.

### 1.4 결과

```
root@goorm:/workspace/sys32200588# ./mycat
잘못된 사용법입니다. (사용법 : ./mycat 파일이름)
root@goorm:/workspace/sys32200588# ./mycat a.txt
abcdefghijklmnopqrstuvwxyz
```

- ./mycat 코드를 실행하면, argc가 1이기 때문에, 즉 읽을 파일명이 없는 사용법이 잘못된 코드기 때문에, 잘못된 사용법이라 알려주고 종료한다.
- ./mycat a.txt 코드를 실행하면 a.txt의 내용이 전부 정상적으로 터미널에 출력된다.

## 2. mycp 구현

```c
/**
 * mycp.c   : copy command programm
 * @author  : Minji Kim
 * @email   : 32200588@dankook.ac.kr
 * @version : 1.0
 * @date    : 2022.11.04
**/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#define MAX_BUF 64

int main(int argc, char* argv[]) {
    int fd, fd1, read_size, write_size;
    char buf[MAX_BUF];
    int attribute = 0;
    struct stat fd_attribute;

    stat(argv[1], &fd_attribute);
    attribute = fd_attribute.st_mode;

    // 입력 형식이 잘못된 경우, 사용법 출력 후 종료
    if(argc != 3) {
        printf("잘못된 사용법입니다. (사용법 : %s 원본파일명 새로운파일명)\n", argv[0]);
        exit(-1);
    }

    fd = open(argv[1], O_RDONLY);
    // 복사하려는 파일이 없는 경우, 에러 출력 후 종료
    if(fd < 0) {
        printf("해당 파일 %s가 존재하지 않습니다. (ERRNO : %d)\n", argv[1], errno);
        exit(-1);
    }

    fd1 = open(argv[2], O_RDWR | O_CREAT | O_EXCL, attribute);
    // 만드려는 파일 이름이 중복일 경우, 에러 출력 후 종료
    if(fd1 < 0) {
        printf("파일 %s은 현재 디렉토리에 이미 존재하므로 생성할 수 없습니다.(ERRNO : %d)\n", argv[2], errno);
        exit(-1);
    }

    while(1) {
        read_size = read(fd, buf, MAX_BUF);
        if(read_size == 0)
            break;
        write_size = write(fd1, buf, read_size);
    }

    close(fd);
    close(fd1);
    return 0;
}

```

### 2.1 cp command란?

cp는 해당 파일의 내용을 복사하여 새로운 파일을 생성하는 명령이다.

사용법은 다음과 같다.

```
$ cp [원본파일명] [새로운파일명]
```

### 2.2 cat 구현을 위해

(1) argc, argv[] 사용

- cp의 argc는 3이다.
- argv[0]은 ./cp이고, argv[1]은 원본파일명, argv[2[는 새로운파일명이다.

(2) 기본 패턴

- open -> read -> write -> close의 패턴으로 구현된다.

(3) while 루프

- 파일의 전체 내용을 읽어 복사하기 위해 while 루프 사용이 필요하다.

(4) O_EXCL 옵션

- 현재 디렉토리에 같은 이름의 파일이 이미 존재한다면 파일 생성하지 않아야 하고, 이를 위해 새로운 파일 open 시에 O_EXCL 옵션을 사용한다.

(5) stat()

- attribute를 복사하기 위해 파일의 상태 정보 얻어오는 stat() 함수를 사용한다.
- stat() 함수 사용법은 인터넷 검색을 참고하였다.(https://hyeonbell.tistory.com/110, https://bodamnury.tistory.com/37)

### 2.3 코드 리뷰

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/stat.h>
#define MAX_BUF 64
```

- stat() 사용을 위한 헤더 파일 <sys/types.h>, <sys/stat.h>을 추가하였다.

```
int main(int argc, char * argv[]) {
        int fd, fd1, read_size, write_size;
        char buf[MAX_BUF];
        int attribute =0;
        struct stat fd_attribute;
        stat(argv[1], &fd_attribute);
        attribute = fd_attribute.st_mode;
```

- 새로운 파일을 위한 file descriptor 변수 fd1을 추가하였다.
- 원본 파일의 속성(권한)값을 넣기 위한 변수 attribute와 파일의 정보를 복사해올 stat 구조체 변수 fd_attribute를 선언한다.
- stat 함수를 사용해 argv[1] 즉, 원본 파일의 정보를 fd_attribute에 복사한다.
- 파일 정보 fd_attribute에서 st_mode 즉, 권한 정보를 변수 attribute에 저장한다.

```
        if(argc !=3) {
                printf("잘못된 사용법입니다. (사용법 : %s 원본파일명 새로운파일명)₩n", argv[0]);
                exit(-1);
        }
```

- cp command는 argc가 3이어야 한다. 따라서 3이 아니면 사용자에게 잘못된 사용법이라는 사실과 사용법을 알려주고 종료한다.

```
        fd = open(argv[1], O_RDONLY);
        if(fd <0) {
                printf("해당 파일 %s가 존재하지 않습니다. (ERRNO : %d)₩n", argv[1], errno);
                exit(-1);
        }
```

- 원본 파일(argv[1])을 읽기 전용으로 open한다. (fd)
- 원본 파일 open에 실패했다면 해당 파일이 존재하지 않는다는 사실과 에러 코드를 출력하고 종료한다.

```
        fd1 = open(argv[2], O_RDWR | O_CREAT | O_EXCL, attribute);
        if(fd1 <0) {
                printf("파일 %s은 현재 디렉토리에 이미 존재하므로 생성할 수 없습니다.(ERRNO : %d)₩n", argv[2], errno);
                exit(-1);
        }
```

- 새로운 파일(argv[2])을 읽기 쓰기 모두 가능하게 open한다. (fd1)
- O_CREAT 옵션을 사용해 해당 이름의 파일이 없다면 새로 만들고,
  O_EXCL 옵션을 함께 사용해 현재 디렉토리에 같은 이름의 파일이 이미 존재한다면 파일 생성하지 않도록 한다.
- open의 3번째 인자에는 파일에 대한 권한 옵션을 넣는다. 위에서 원본 파일의 권한을 복사한 attribute 변수를 넣어준다.

```
        while(1) {
                read_size = read(fd, buf, MAX_BUF);
                if(read_size ==0)
                        break;
                write_size = write(fd1, buf, read_size);
        }
        close(fd);
        close(fd1);
        return 0;
}
```

- 원본 파일의 전체 내용을 읽어오기 위해 while 루프를 사용한다.
- buf 문자열을 read_size만큼 읽어 fd1에 write한다.
- 마지막을 fd와 fd1을 close하고 종료한다.

### 2.4 결과

```
root@goorm:/workspace/sys32200588# ls
README.md  a.txt  bin  gcc  goorm.manifest  mycat  mycat.c  mycp  mycp.c  src
root@goorm:/workspace/sys32200588# ./mycp
잘못된 사용법입니다. (사용법 : ./mycp 원본파일명 새로운파일명)
root@goorm:/workspace/sys32200588# ./mycp b.txt b_new.txt
해당 파일 b.txt가 존재하지 않습니다. (ERRNO : 2)
root@goorm:/workspace/sys32200588# ./mycp a.txt a_new.txt
root@goorm:/workspace/sys32200588# ./mycat a.txt
abcdefghijklmnopqrstuvwxyz
root@goorm:/workspace/sys32200588# ./mycat a_new.txt
abcdefghijklmnopqrstuvwxyz
root@goorm:/workspace/sys32200588# ls -l
합계 60
-rw-r--r-- 1 root root 2000  1월  6  2021 README.md
-rw-rw-r-- 1 root root   27 11월  4 02:04 a.txt
-rw-rw-r-- 1 root root   27 11월  4 06:13 a_new.txt
drwxrwxr-x 2 root root 4096 11월  4 04:22 bin
drwxrwxr-x 2 root root 4096 10월 21 04:05 gcc
-rw-rw-r-- 1 root root  682 11월  4 06:13 goorm.manifest
-rwxrwxr-x 1 root root 8768 11월  4 02:42 mycat
-rw-rw-r-- 1 root root  771 11월  4 02:42 mycat.c
-rwxrwxr-x 1 root root 8880 11월  4 06:01 mycp
-rw-rw-r-- 1 root root 1408 11월  4 06:01 mycp.c
drwxr-xr-x 2 root root 4096 11월 13  2020 src
root@goorm:/workspace/sys32200588# vi b.txt
root@goorm:/workspace/sys32200588# ./mycp b.txt a_new.txt
파일 a_new.txt은 현재 디렉토리에 이미 존재하므로 생성할 수 없습니다.(ERRNO : 17)
root@goorm:/workspace/sys32200588# date
2022. 11. 04. (금) 06:15:13 UTC
```

- ./mycp 코드를 실행하면, argc가 1이기 때문에, 즉 원본 파일명과 새로운 파일명이 없는 사용법이 잘못된 코드기 때문에, 잘못된 사용법이라 알려주고 종료한다.
- ./mycp b.txt b_new.txt 코드를 실행하면 원본파일에 해당하는 b.txt 파일이 현재 디렉토리에 없는 파일이기 때문에, 파일이 존재하지 않음과 에러 코드를 알려주고 종료한다.
- ./mycp a.txt a_new.txt 코드를 실행하고, a.txt와 a_new.txt의 내용을 각각 확인해보면, 내용이 잘 복사된 것을 확인할 수 있다.
- ls -l 명령으로 각 파일의 권한을 확인해보면 a.txt와 a_new.txt의 권한이 동일한 것도 확인할 수 있다.
- ./mycp b.txt a_new.txt 코드를 실행하고, 파일 a_new.txt가 이미 현재 디렉토리에 존재하고 있는 파일이기 때문에 파일을 생성하지 않고, 에러 코드를 알려주고 종료한다.