시스템 프로그래밍 cp 명령어 구현

컴퓨터공학과 32181827 박종기

1. 소개

리눅스의 명령어 중 cp라는 명령어를 직접 구현하는 프로그램을 작성한다.

2. 본문

1) 프로그램 설계 개요

프로그램 구현을 하기에 앞서 첫줄 주석문으로 파일의 이름 : 간략히 프로그램의 기능을 설명하는 문구 만든사람 이메일주소 날짜를 명시하였다.

리눅스의 cp 명령어를 구현하기 위해서는 단순히 파일 내용을 복사하여 새로운 파일에 저장하는 프로그램을 작성한 후 기존파일의 속성까지 복사해보고 프로그램의 완성도를 높히기 위해 코드를 수정하고 정리하는 3가지 과정을 거쳐서 프로그램 설계를 진행하였다.

단순히 파일의 내용을 복사하는 과정은 다음과 같다. 사용할 명령어의 형식은 "mycp <기존파일> <새로 생성하려는 파일> "으로 정했다. 메인함수로 전달되는 인자의 수를 argc를 이용하여 사용자의 입력형태를 확인한다. 그 후 기존 파일을 오픈하고 새로운 파일을 생성한 후 내용을 복사하는 과정이 필요하다. 내용을 하나의 파일에서 다른 파일로 복사하는 과정은 cat명령어를 구현할 때 STDOUT_FILENO 의 터미널에 복사하는 것 대신 새로 생성할 파일의 open() 반환값을 주기만 하면 되었다. 새로운 파일을 생성할 때는 배타적으로 생성하도록 하기위해서 flag에 O_EXCL 을 추가적으로 주었다.

파일의 속성을 복사할 때는 파일의 세부사항들 중 접근권한만을 어떻게 추출해 낼것인가 라는 것이 문제점이였다. 이 문제를 해결하기 위해 fstat함수의 원형을 살펴보기로 했다. 전처리기로는 <sys/types.h>, <sys/stat.h>, <unistd.h>의 3개를 가지며 인자로는 fd(file descripter) 와 stat 구조체의 주소 *buf 포인터를 인자로 갖는다. 설명을 보면 이 함수는 buf 가 가리키는 버퍼에서 파일에 대한 정보 즉 stat 구조체의 필드들을 반환한다고 한다. 이 필드들 중에 mode_t 의 st_mode가 필요한 정보였다.

stat 구조체는 <sys/stat.h> 파일에 정의되어있다. 그리고 이 구조체에서 st_mode의 값을 추출했을 때 어떻게 나오는지 살펴봐야 했다. 8진수 형태로 출력을 해서 살펴보았다.

```
sys32181827@embedded:~$ ./mycp poem.txt new.txt
mode : 100664
```

<사진1>

<사진1>을 보면 파일의 접근권한 정보와 더불어 파일이 regular file인지 디렉토리인지에 대한 정보까지 함께 출력이 되는 것을 확인할 수 있었다. 그래서 파일을 생성할때 3번째 인자인 mode부분에 초기에 입력했던 0664 대신 100664 를 입력해보았더니 프로그램이 정상적으로 작동하는 것을 확인할 수 있었다. 이제 속성을 복사할때 fstat의 st_mode의 값을 그대로 파일 생성시 open의 mode 인자에 주기만 하면 된다는 것을 알 수 있었다.

마지막으로 프로그램의 완성도를 높이기 위해서 기존의 error number로 error의 형태를 표기하는 것을 사용자가 바로 알아차릴 수 있는 문구로 바꾸기 위해서 <string.h>헤더파일을 추가적으로 삽입하였다. <string.h>의 strerror()함수에 errno 를 인자로 주면 텍스트로 바꾸어준다.

2) 프로그램 설명 및 실행

프로그램 설명에 앞서 코드를 볼때는 가독성 및 설명에 도움이 되도록 cat명령어에 -n 옵션을 주어 행 번호를 붙었다.

<사진 2>의 코드 1번행 부터 22행까지는 주석문, 헤더파일들과 함께 main함수의 argc인자를 이용하여 새로 만든 mycp명령어의 사용형태를 규정하는 부분이다. 후에 파일을 복사할때 어떤 크기만큼씩 만큼 복사할지를 정하기 위해서 MAX_BUF 상수를 정의하였다. 이 프로그램에서는 MAX_BUF를 64로 정의했다.

```
sys32181827@embedded:~$ cat -n mycp.c
     1  /*mycp.c : copy data from one file to another by 32181827 park jong ki jkipark1@naver.com
2021.10.8*/
     2  #include <stdio.h>
     3  #include <stdlib.h>
     4  #include <unistd.h>
     5  #include <fcntl.h>
     6  #include <errno.h>
     7  #define MAX_BUF 64
     8  #include <sys/types.h>
     9  #include <sys/stat.h>
    10  #include <string.h>
    11
    12  int main(int argc, char *argv[])
    13  {
    14          int fd, fdn, read_size, write_size;
    15          char buf[MAX_BUF];
    16
    17          if (argc !=3){
    18                  printf("Wrong useage\n");
    19                  printf("Usage : %s file_name target__file_name\n",argv[0]);
    20                  exit(-1);
    21
    22          }
```

<사진2>

<사진3>의 코드 23번째 행부터 47번째 행은 복사할 파일을 여는 부분, 복사할 파일의 속성을 추출하여 mode변수에 저장하는 부분, 새로운 파일을 생성하는 부분, 새로운 파일에 기존 파일의 내용을 복사해넣는 부분으로 나누어져있다. argv의 1번째인자로 받은 파일을 open 하고 리턴값을 fd로 받는다. 마찬가지로 기존에 같은 이름의 파일이 없을 경우 새로운 파일을 생성해 open 하여 리턴값을 fdn 으로 받는다. 그 후 while 루프를 이용하여 fd에서 64개씩 읽어서 읽을 문자가 없을때까지 64개씩 fdn에 쓴다. 각각의 상황에서 에러값을 strerror함수를 이용해 사용자가 읽을 수 있는 문자로 변환해준다.

```
23          //open current file
24          fd = open(argv[1],O_RDONLY);
25          if(fd<0){
26                  printf("Can't open %s file because %s\n",argv[1],strerror(errno));
27                  exit(-1);
28          }
29
30          //extract mode from stat structure
31          struct stat statbuf;
32          fstat(fd,&statbuf);
33          int mode = statbuf.st_mode;
34
35          //create new file
36          fdn = open(argv[2],O_RDWR | O_CREAT | O_EXCL, mode);
37          if(fdn<0){
38                  printf("Can't create %s file with because  %s\n",argv[2],strerror(errno));
39                  exit(-1);
40          }
41
42          //copy file data
43          while(1){
44                  read_size = read(fd,buf,MAX_BUF);
45                  if(read_size ==0) break;
46                  write_size = write(fdn,buf,read_size);
47          }
```

<사진3>

<사진4>의 49행에서 52행까지는 열었던 파일 두개를 close하며 메인함수를 마무리한다.

```
48
49          close(fdn);
50          close(fd);
51
52  }
```

<사진4>

프로그램 실행

<사진5>에서는 현재 working directory에 존재하는 모든 파일을 ls -l로 보였다. 목록의 8번째에 만든 mycp.c라는 c파일이 존재하고 후에 파일 복사에 사용될 poem.txt 이 9번째 목록에 존재하는 것을 확인할 수 있다. 또한 alpha.txt라는 파일은 존재하지 않는다는것을 확인할 수 있다.

```
sys32181827@embedded:~$ ls -l
total 72
-rwxrwxr-x 1 sys32181827 sys32181827  4684 10월   1 16:29 a.out
-rw-rw-r-- 1 sys32181827 sys32181827  1983 10월  10 16:25 cat_man.txt
-rw-r--r-- 1 sys32181827 sys32181827  8980  4월  20  2016 examples.desktop
-rw-rw-r-- 1 sys32181827 sys32181827 25138 10월  10 14:42 fstat_man.txt
-rw-rw-r-- 1 sys32181827 sys32181827    83 10월   1 16:29 hello.c
-rw-rw-r-- 1 sys32181827 sys32181827   872 10월   1 14:33 hello.o
-rw-rw-r-- 1 sys32181827 sys32181827   432 10월   1 14:32 hello.s
-rw-rw-r-- 1 sys32181827 sys32181827  1105 10월  10 16:37 mycp.c
-rw-rw-r-- 1 sys32181827 sys32181827   334  9월  30 23:24 poem.txt
sys32181827@embedded:~$ gcc -o mycp mycp.c
sys32181827@embedded:~$ ./mycp
Wrong useage
Usage : ./mycp file_name target__file_name
sys32181827@embedded:~$ ./mycp alpha.txt alpha_new.txt
Can't open alpha.txt file because No such file or directory
```

<사진5>

후에 gcc -o 로 mycp.c파일을 컴파일하였고 목적파일이름을 mycp로 생성하였다. ./mycp를 입력하였더니 입력 오류메세지와 함께 명령어의 사용법을 명시하고 있는것을 볼 수 있다. 다음은 존재하지 않는 파일 alpha.txt파일을 복사하려고 하자 파일을 열 수 없다는 에러문구와 함께 "No such file or directory" 라는 에러 원인 문구가 정상적으로 뜨는 것을 확인할 수 있다.

<그림6,7>에서는 존재하는 poem.txt라는 파일에 대한 복사를 수행하였다. 복사될 파일의 이름은 copied_poem.txt로 하였다. 다시 ls -l명령어를 입력하면 목록의 3번째 줄에 복사된 파일이 존재하는 것을 확인할 수 있다. 이제 cat명령어로 원래 파일의 내용과 복사된 파일의 내용을 확인한다.

```
sys32181827@embedded:~$ ./mycp poem.txt copied_poem.txt
sys32181827@embedded:~$ ls -l
total 84
-rwxrwxr-x 1 sys32181827 sys32181827  4684 10월   1 16:29 a.out
-rw-rw-r-- 1 sys32181827 sys32181827  1983 10월  10 16:25 cat_man.txt
-rw-rw-r-- 1 sys32181827 sys32181827   334 10월  10 18:16 copied_poem.txt
-rw-r--r-- 1 sys32181827 sys32181827  8980  4월  20  2016 examples.desktop
-rw-rw-r-- 1 sys32181827 sys32181827 25138 10월  10 14:42 fstat_man.txt
-rw-rw-r-- 1 sys32181827 sys32181827    83 10월   1 16:29 hello.c
-rw-rw-r-- 1 sys32181827 sys32181827   872 10월   1 14:33 hello.o
-rw-rw-r-- 1 sys32181827 sys32181827   432 10월   1 14:32 hello.s
-rwxrwxr-x 1 sys32181827 sys32181827  6032 10월  10 18:12 mycp
-rw-rw-r-- 1 sys32181827 sys32181827  1105 10월  10 16:37 mycp.c
-rw-rw-r-- 1 sys32181827 sys32181827   334  9월  30 23:24 poem.txt
sys32181827@embedded:~$ cat poem.txt
<서  시>
죽는  날까지  하늘을  우러러
한  점  부끄럼이  없기를,
잎새에  이는  바람에도
나는  괴로워했다.

별을  노래하는  마음으로
모든  죽어가는  것을  사랑해야지
그리고  나한테  주어진  길을
걸어가야겠다.

오늘  밤에도  별이  바람에  스치운다.

-  윤  동  주  -
```

<그림6>

```
sys32181827@embedded:~$ cat copied_poem.txt
<서  시 >
죽 는  날 까 지  하 늘 을  우 러 러
한  점  부 끄 럼 이  없 기 를 ,
잎 새 에  이 는  바 람 에 도
나 는  괴 로 워 했 다 .

별 을  노 래 하 는  마 음 으 로
모 든  죽 어 가 는  것 을  사 랑 해 야 지
그 리 고  나 한 테  주 어 진  길 을
걸 어 가 야 겠 다 .

오 늘  밤 에 도  별 이  바 람 에  스 치 운 다 .

- 윤  동  주 -
```

<그림7>

<그림8> 을 보면 같은 이름으로 복사 파일을 생성하려 했을 때 파일이 이미 존재한다는 에러문구 역시 잘 뜨는 것을 확인할 수 있다. 속성 복사가 잘 수행되고 있나 확인하기 위해 접근권한이 특이한 a.out파일을 new_a.out이라는 파일로 복사해보았다. a.out파일은 hello.c를 컴파일한 수행가능한 목적파일이다. 목록 첫줄의 a.out파일이 목록의 11번째 줄 new_a.out파일로 파일 종류 및 접근권한 속성까지 복사가 잘 된것을 확인해볼 수 있다.

```
sys32181827@embedded:~$ ./mycp poem.txt copied_poem.txt
Can't create copied_poem.txt file with because  File exists
sys32181827@embedded:~$ ./mycp poem.txt recopied_poem.txt
sys32181827@embedded:~$ ./mycp a.out new_a.out
sys32181827@embedded:~$ ls -l
total 96
-rwxrwxr-x 1 sys32181827 sys32181827  4684 10월   1 16:29 a.out
-rw-rw-r-- 1 sys32181827 sys32181827  1983 10월  10 16:25 cat_man.txt
-rw-rw-r-- 1 sys32181827 sys32181827   334 10월  10 18:16 copied_poem.txt
-rw-r--r-- 1 sys32181827 sys32181827  8980  4월  20  2016 examples.desktop
-rw-rw-r-- 1 sys32181827 sys32181827 25138 10월  10 14:42 fstat_man.txt
-rw-rw-r-- 1 sys32181827 sys32181827    83 10월   1 16:29 hello.c
-rw-rw-r-- 1 sys32181827 sys32181827   872 10월   1 14:33 hello.o
-rw-rw-r-- 1 sys32181827 sys32181827   432 10월   1 14:32 hello.s
-rwxrwxr-x 1 sys32181827 sys32181827  6032 10월  10 18:12 mycp
-rw-rw-r-- 1 sys32181827 sys32181827  1105 10월  10 16:37 mycp.c
-rwxrwxr-x 1 sys32181827 sys32181827  4684 10월  10 18:40 new_a.out
-rw-rw-r-- 1 sys32181827 sys32181827   334  9월  30 23:24 poem.txt
-rw-rw-r-- 1 sys32181827 sys32181827   334 10월  10 18:39 recopied_poem.txt
sys32181827@embedded:~$ whoami
sys32181827
sys32181827@embedded:~$ date
2021. 10. 10. (일) 18:41:11 KST
sys32181827@embedded:~$
```

<그림8>

마지막으로 프로그램을 만들고 수행한 본인을 확인할 수 있도록 아이디와 시간을 명시하였다.

3. 결론

리눅스의 cp명령어를 기본적인 file system call 인 open, read, write, close만으로도 만들 수 있다는 사실을 알 수 있었다. 한가지 흥미로웠던 점은 새로운 파일을 생성하려고 할때 open에 flag로 O_CREAT 를 주게되면 다음 인자로 mode를 명시해야 하는데 이때 파일의 종류에 대한 정보까지 속성으로 주어도 상관 없다는 점이였다. 물론 파일의 종류를 명시한다고 디렉토리가 복사가 되지는 않는다. 읽는 방식이 단순히 텍스트를 읽어 복사하는 방식이 아니기 때문이다.