시스템 프로그래밍 과제3. mycp 작성

32181854 박준영

1. 수행 결과

```
sys32181854@embedded:~/sys_assignment$ vi mycp.c
sys32181854@embedded:~/sys_assignment$ gcc -o mycp_t mycp.c
sys32181854@embedded:~/sys_assignment$ mycp_t original.txt new.txt
mycp_t: command not found
sys32181854@embedded:~/sys_assignment$ ./mycp_t original.txt new.txt
sys32181854@embedded:~/sys_assignment$ ls -l
total 80
-rwxrwxr-x 1 sys32181854 sys32181854 5836 10월   9 12:40 a.out
-rwxrwxr-x 1 sys32181854 sys32181854 6144  9월  22 15:06 gdb_test.out
-rwxrwxr-x 1 sys32181854 sys32181854 5836 10월   9 12:37 mycp
-rw-rw-r-- 1 sys32181854 sys32181854 1163 10월   9 12:48 mycp.c
-rwxrwxr-x 1 sys32181854 sys32181854 5592 10월   9 12:48 mycp_t
---------- 1 sys32181854 sys32181854    0 10월   9 12:46 new.c
---------x 1 sys32181854 sys32181854   16 10월   9 12:48 new.txt
-rw-rw-r-- 1 sys32181854 sys32181854   16 10월   9 11:29 original.txt
-rwxrwxr-x 1 sys32181854 sys32181854 4980  9월  22 15:27 sys_2_1.out
-rw-rw-r-- 1 sys32181854 sys32181854  269  9월  22 15:35 sys_2.c
-rw-rw-r-- 1 sys32181854 sys32181854 1284  9월  22 14:54 sys_2.o
-rwxrwxr-x 1 sys32181854 sys32181854 4980  9월  22 15:31 sys_2.out
-rw-rw-r-- 1 sys32181854 sys32181854 1377  9월  22 14:48 sys_2.s
-rwxrwxr-x 1 sys32181854 sys32181854 4864  9월  22 15:37 test.out
sys32181854@embedded:~/sys_assignment$
```

접근 권한 까지 복사하지 않았을 때

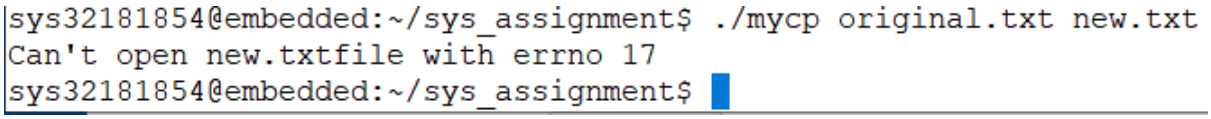
```
sys32181854@embedded:~/sys_assignment$ ./mycp original.txt new.txt
sys32181854@embedded:~/sys_assignment$ ls
a.out          mycp     new.txt       sys_2_1.out  sys_2.o    sys_2.s
gdb_test.out   mycp.c   original.txt  sys_2.c      sys_2.out  test.out
sys32181854@embedded:~/sys_assignment$ ls -l
total 72
-rwxrwxr-x 1 sys32181854 sys32181854 5832 10월   9 12:17 a.out
-rwxrwxr-x 1 sys32181854 sys32181854 6144  9월  22 15:06 gdb_test.out
-rwxrwxr-x 1 sys32181854 sys32181854 5832 10월   9 12:26 mycp
-rw-rw-r-- 1 sys32181854 sys32181854 1093 10월   9 12:26 mycp.c
-rw-rw-r-- 1 sys32181854 sys32181854   16 10월   9 12:26 new.txt
-rw-rw-r-- 1 sys32181854 sys32181854   16 10월   9 11:29 original.txt
-rwxrwxr-x 1 sys32181854 sys32181854 4980  9월  22 15:27 sys_2_1.out
-rw-rw-r-- 1 sys32181854 sys32181854  269  9월  22 15:35 sys_2.c
-rw-rw-r-- 1 sys32181854 sys32181854 1284  9월  22 14:54 sys_2.o
-rwxrwxr-x 1 sys32181854 sys32181854 4980  9월  22 15:31 sys_2.out
-rw-rw-r-- 1 sys32181854 sys32181854 1377  9월  22 14:48 sys_2.s
-rwxrwxr-x 1 sys32181854 sys32181854 4864  9월  22 15:37 test.out
```

접근 권한을 복사하는 코드를 삽입 후 mycp로 재빌드

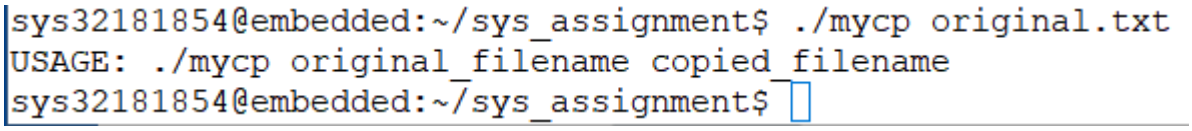
new.txt를 삭제하고 mycp를 실행하면 접근권한도 복사됐음을 확인

(작성자나 작성 시간 등은 예제 snapshot에서 언급되지 않아서 접근권한 attribute만 복사했습니다.)

(실제 작성시에는 mycp를 먼저 생성하고 snapshot을 찍은 후 attributes를 복사하는 코드만 주석처리 후 mycp_t를 빌드 했습니다.)



기존에 있던 파일을 오픈하려고 하면 오류 출력 (O_EXCL)



인자 개수를 잘못 입력 시 사용방법 지시

2. 구현 설명 및 소스 코드

sys32181854@embedded: ~/sys_assignment

```
sys     0m0.000s
sys32181854@embedded:~/sys_assignment$ cat mycp.c
/*mycp.c: copy files with attributes  and display the copied file. by park junyeong
wndwls1024@naver.com*/
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#define MAX_BUF 64


int main(int argc, char* argv[]){
        int fd_ori,fd_new;
        int readsize;
        int buf[MAX_BUF];
        struct stat buf2; //파일 attributes를 담을 구조체

        if(argc != 3){
                printf("USAGE: %s original_filename copied_filename\n",argv[0]);
                exit(1);
        } //인자수 확인 후 사용법 알리며 프로세스 종료

        fd_ori = open(argv[1],O_RDONLY);
        if(fd_ori < 0) {
                printf("Can't open %sfile with errno %d\n", argv[1],errno);
                exit(1);
        } //open 검사

        fstat(fd_ori,&buf2); //원본 파일의 attributes 복사

        fd_new = open(argv[2],O_RDWR|O_CREAT|O_EXCL,buf2.st_mode);
        //원본 파일과 동일한 접근권한 부여

        if(fd_new < 0) {
                printf("Can't open %sfile with errno %d\n", argv[2],errno);
                exit(1);
        }

        while(1){
                readsize = read(fd_ori,buf,MAX_BUF);
                if(readsize == 0) break;
                write(fd_new,buf,readsize);
        } //내용복사

        close(fd_ori);

          close(fd_new);
}



sys32181854@embedded:~/sys_assignment$ whoami
sys32181854
sys32181854@embedded:~/sys_assignment$ date
2020. 10. 09. (금) 12:56:56 KST
sys32181854@embedded:~/sys_assignment$
```

1) 변수

fd_ori : 원본 파일의 file descriptor

fd_new : copy된 파일의 file descriptor

readsize : 원본파일에서 읽은 size

buf[] : 원본 파일에서 읽은 contents를 저장하는 buf

stat buf2 : 원본 파일의 attributes를 저장할 stat 구조체

2) 동작

-인자가 3개가 아니면 프로세스 종료

-argv[1](원본파일) open

-fstat()으로 fd_ori의 attributes들을 buf2 구조체에 불러옴

-argv[2](복사된 파일) open

-파일 내용 복사 (redirection과 동일한 과정)

-복사가 끝난 후 fd_ori, fd_new를 close함

3. discussion

이번 과제에서 다소 어려웠던 점은 stat의 사용이다. stat system call을 사용하면 파일의 속성을 출력해주는 기능은 쉽게 사용할 수 있었지만 그 내용을 가져와서 활용하는 것은 생각해보지 않았기 때문이다. stat의 활용에 대해 검색하자 많은 내용들이 나왔는데 이 때 symbolic link라는 개념을 발견해 이에 대해 간단히 정리해보고싶다.

코드에서는 open된 파일의 fd를 이용했기 때문에 fstat()를 사용했지만 동일한 기능을 하는 system call 중 stat(2)는 첫 번째 인자로 symbolic link인 파일의 path를 받는다. 그렇다면 symbolic link란 무엇일까? 수업에서 link라는 개념을 배웠다. 교수님께서는 hard link를 그림을 통해 설명해주셨다. hard link는 두 개의 다른 이름을 가진 파일이 하나의 inode와 같은 데이터 블록을 가리킨다. 그러나 symbolic link (or soft link)는 이와는 조금 다르다. 심볼릭 링크에서 두 링크는 별개의 파일이다. 하나는 실제 데이터를 저장하고, 나머지는 첫번째 파일의 이름을 포함하는 '포인터'이다. 이 포인터를 링크라고 부르는데 심볼릭 링크는 자신에 대한 inode는 별도로 가지고 있고 그 만큼의 파일 시스템 공간을 차지한다는 것이 hard link와의 차이점이다. 또한 포인터로 첫번째 파일과 연결되어 있기 때문에 symbolic link에서 파일을 수정하면 원래 파일의 내용 또한 수정된다.

비록 이 개념에 대해 아직은 추상적으로 알고있어서 활용하지는 못했지만 학습했던 link개념을 상기시키면서 학습할 수 있었다. 또한 fstat()을 사용하여 stat구조체 멤버변수들을 쉽게 불러올 수 있다는 것도 알 수 있었다. 이전에 fread()를 이용하여 PE헤더의 구조체 내용을 읽어오는 과제를 한 적이 있었는데 fstat()도 이 때 작성했던 코드들과 비슷한 동작을 하는 코드라고 생각했다.