과제 3: "mycp 작성"

시스템 프로그래밍 7분반
소프트웨어학과 32170578 김산

<보고서>

1. 구현 설명

리눅스 커널의 cp 명령과 비슷하게 프로그램 시작시 인자 2개를 받아 파일의 내용과 속성을 복사하는 프로그램을 작성하였습니다. 프로그램의 main함수는 source에 해당하는 인자와 destination 에 해당하는 인자 2개를 받습니다.

- argument check
인자가 3개인지 확인하고 3개가 아니면 에러 메시지를 출력하고 종료합니다.

- source open
argv[1]에 해당하는 파일을 open()함수를 통해 읽기 전용으로 엽니다. 만약 open의 반환 값이 -1이면 오류가 발생한 것이므로 에러 메시지를 출력하고 종료합니다.

- destination open
argv[2]에 해당하는 파일을 open() 함수를 통해 쓰기 전용으로 엽니다. 파일이 존재하지 않으면 생성해야 하므로 O_CREAT 옵션을 적용하였고 만약 생성하려는 파일이 이미 존재하면 생성하지 말고 실패로 처리해야 하므로 O_EXCL옵션을 적용합니다. open의 반환 값이 -1이면 오류가 발생한 것이므로 에러 메시지를 출력하고 종료합니다.

- file content copy
read(), write() 함수를 이용하여 source파일의 내용을 MAX_BUF크기만큼 읽어들이고 destination에 쓰기를 파일의 내용이 끝날 때까지 반복합니다.

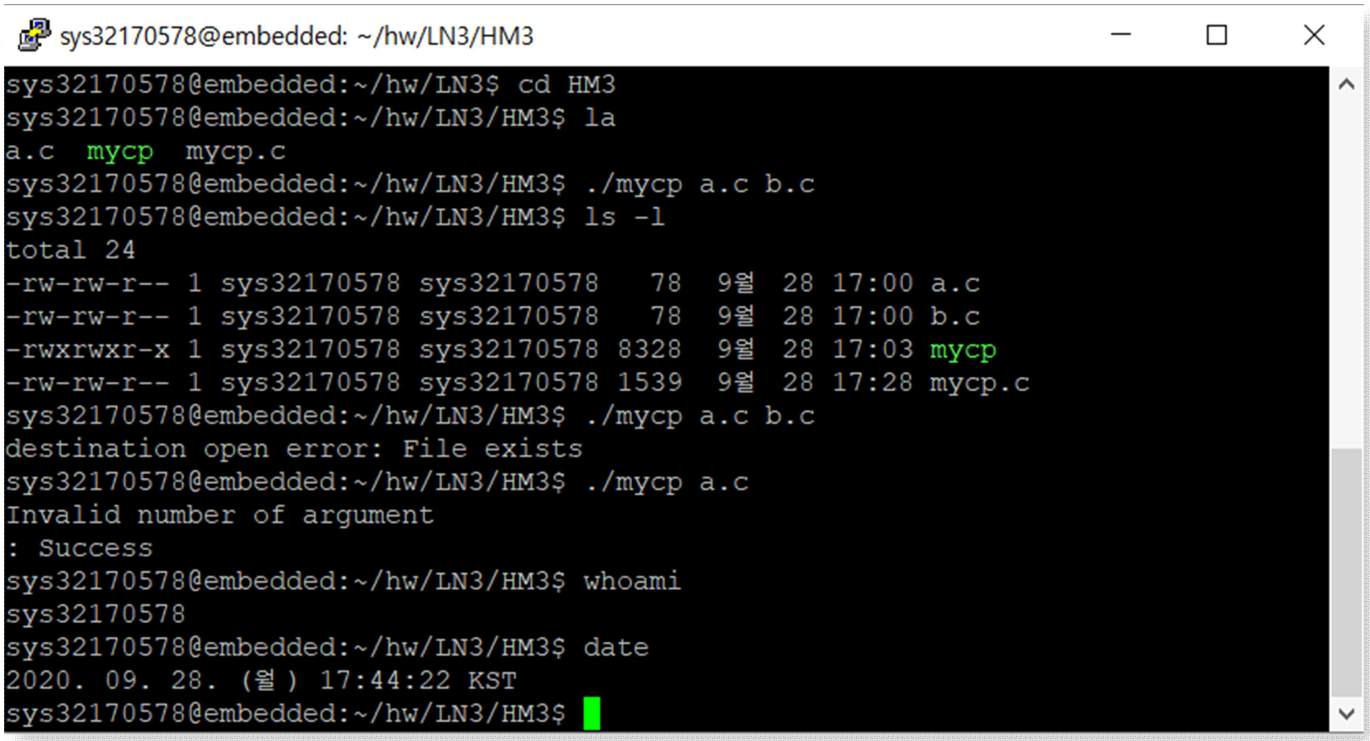
- assign attribute to destination file

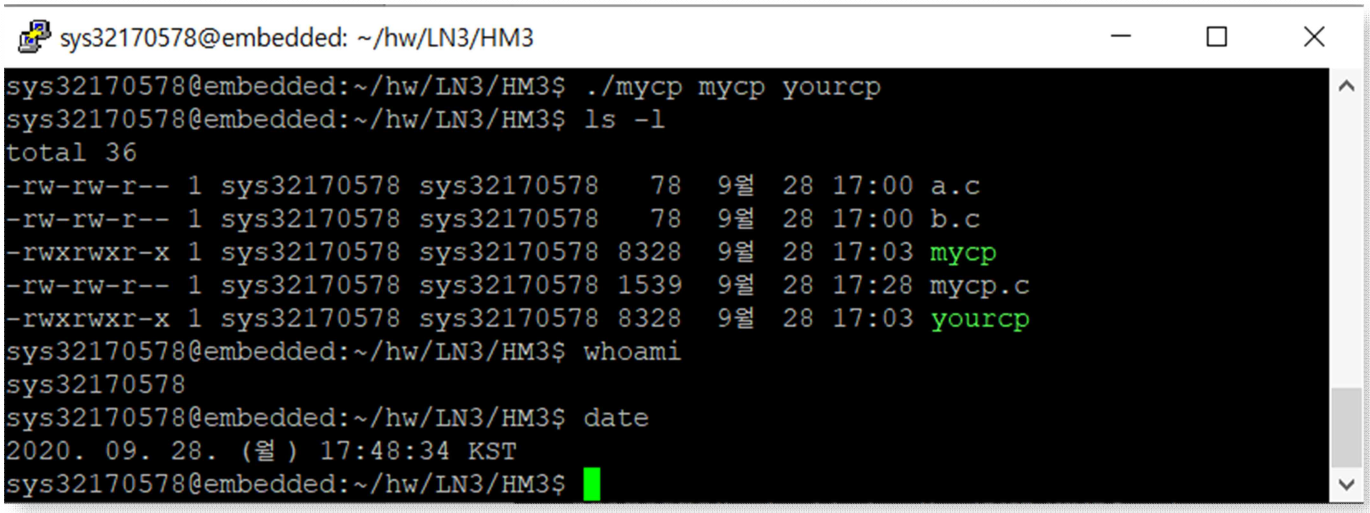
fstat()함수를 이용하여 source파일의 속성을 얻습니다.

- assign attribute to destination file
chmod(), chown() 함수를 이용하여 파일에 대한 접근권한과 user id, group id를 복사합니다.
또, utime() 함수를 이용하여 파일이 마지막으로 access된 시간, 마지막으로 modify된 시간을 복사합니다.

- close file desriptor
파일 디스크립터를 닫고 힙 메모리를 확보합니다.

2. 수행결과 스냅샷





먼저 복사할 파일 'a.c'를 준비하고 './mycp a.c b.c'를 입력하여 파일 'a.c'의 복사본인 'b.c'를 생성합니다. 'ls -l' 명령어를 통해 파일의 속성까지 복사되었는지 확인합니다. 한번더 './mycp a.c b.c'를 입력하여 생성하려는 파일이 이미 존재할 때 오류가 발생하는지 확인하였습니다. File exit 에러가 나면서 종료되었습니다.

인자가 잘못 입력되었을 때 (1개만 입력되었을 경우) Invalid number of argument 라는 에러를 출력하고 종료되었습니다. (mycp파일을 복사할 때에도 파일 속성까지 복사해내었습니다.)

3. Discussion

먼저 과제 외적으로, VI편집기를 사용하면서 생소한 개발환경이라 그런지 불편하다고 생각을 했습니다. 처음 사용할 때는 자동으로 인덴트도 안되고 문법강조도 없을뿐더러 줄번호도 없어 메모장이랑 다를바 없다 생각했지만 편집기를 좀더 편하게 사용하기위해 vi ~/.vimrc를 수정하여 불편하다고 생각했던것들을 해결하면서 vim의 다양한 플러그인들을 접하게 되었습니다. 각종 플러그인을 설치하면서 저에게 맞는 환경을 만들어가면서 재미를 느꼈고 제가 평소에 자주쓰던 vscode못지않은 개발환경을 리눅스에서 구축할 수 있게 되었습니다. 사실 아직도 vscode의 ssh나 wsl 통해 리눅스를 접하는 것이 편하지만 vi의 명령어를 사용하면서 vi편집기에 익숙해지고 숙달된다면 아주 편하고 강력한 편집기가 될 것 같습니다.

mycp 파일을 만들면서 컴퓨터 시스템에 대한 이해도가 높아진 것 같습니다. 평상시에 아무렇지 않게 자주 사용하던 복사 붙여넣기 과정이 open(), read(), write(), close()라는 여러 시스템 콜을 거쳐 결과를 만들어 내는 원리에 흥미를 느꼈습니다. 또 기존에 파일의 속성까지 복사하기 위해서는 cp -a 와 같이 옵션을 붙여야 했지만 mycp는 옵션없이 파일의 속성까지 복사하는 것을 보면서 사용자의 목적에 맞는 명령어들을 가지고있는 OS를 만들면 편리하겠다 라는 생각이 들었습니다. 실제로 찾아보니 리눅스 마다 명령어에 다소 차이가 있다고 합니다.

<코드>

```
/* mycp.c file copy program using system call, by san kim, waterfog9580@gmail.com*/
#include <unistd.h >// SEEK_CUR, SEEK_END, SEEK_SET
#include <stdio.h >//standard IO
#include <errno.h >//error code
#include <fcntl.h >// O_RDONLY, O_CREAT, O_EXCL
#include <sys /stat.h >//file stat structure
#include <sys /types.h >
#include <utime.h >//last access, modify time modify
#define MAX_BUF 64 //Buffer size
int main(int argc,char *argv[]){
    int fd_src, fd_dest;
    int read_size, write_size;
    char buf[MAX_BUF];
    struct stat st_source;
    struct utimbuf ust_source;
    /* argument check */
    if(argc !=3){
            perror("Invalid number of argument₩n");
            exit(-1);
        }
    /* source open */
        if((fd_src = open(argv[1], O_RDONLY)) ==-1){
            perror("source open error");
            exit(-1);
        }
    /* destination open */
    if((fd_dest = open(argv[2], O_WRONLY | O_CREAT | O_EXCL)) ==-1){
            perror("destination open error");
            exit(-1);
        }
    /* file content copy */
    while(1){
        read_size = read(fd_src,buf,MAX_BUF);
        if(read_size ==0) break;
        write_size = write(fd_dest, buf, read_size);
    }
    /* get attribute from source file */
    fstat(fd_src, &st_source);
    ust_source.actime = st_source.st_atime;
    ust_source.modtime = st_source.st_mtime;
    /* assign attribute to destination file */
    chmod(argv[2], st_source.st_mode);
    chown(argv[2], st_source.st_uid, st_source.st_gid);
    utime(argv[2], &ust_source);
    /* close file desriptor */
    close(fd_src);
    close(fd_dest);
    return 0;
}
```