2022년 컴퓨터망 프로그래밍 중간고사-수요일 (30점 만점)

☞ 주의 사항

아래의 조건(UDP 소켓, 구조체 등)을 따르지 않고 구현한 경우, 0점 처리함
각 소스 파일에 학번, 이름(영문 가능) 주석이 없는 경우, 파일당 -1점 감점
각 기능에 대한 소스 코드만 있고, 동작이 되지 않는 경우에는 아래의 점수를 받을 수 없음
화면 출력 과정에서 쓰레기 값이 출력되거나, 출력 내용이 맞지 않으면 항목당 -5점 감점함

1. UDP 통신을 이용한 원격 파일 뷰어 프로그램 (총 30점)

- 제출파일: **udp_server.c, udp_client.c**

- 주어진 파일(udp,txt, tcp.txt) 파일을 이용하여 구현된 기능을 검증하세요.
  - 공통 사항
    - ✓ 클라이언트, 서버의 파일 송수신 **버퍼 크기는 1024 바이트로 고정**
    - ✓ 파일 입출력 함수는 저수준(read, write)함수 또는 고수준(fread, fwrite) 함수 모두 사용 가능함
    - ✓ 서버의 전송 바이트 수와 클라이언트에서 수신된 바이트 수는 동일해야 되며, 클라이언트에서 화면에 출력하는 내용은 실제 파일 내용과 동일해야 됨
    - ✓ 클라이언트와 서버는 서로 다른 폴더에서 동작을 시킴
  - 클라이언트, 서버 공용 데이터 구조

```c
#define BUF_SIZE 1024
// cmd type
#define FILE_REQ 0
#define FILE_RES 1
#define FILE_END 2
#define FILE_END_ACK 3
#define FILE_NOT_FOUND 4

typedef struct {
    int cmd;
    int buf_len; // 실제 전송되는 파일의 크기 저장
    char buf[BUF_SIZE];
}PACKET;
```

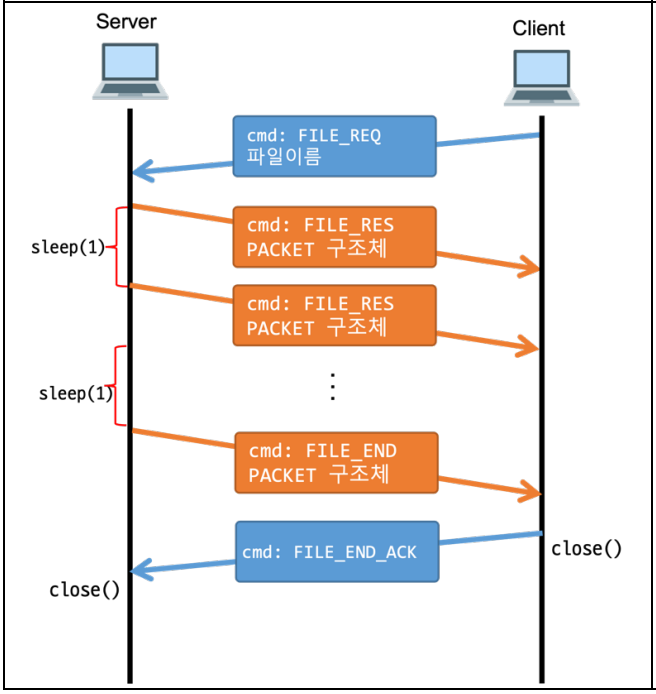
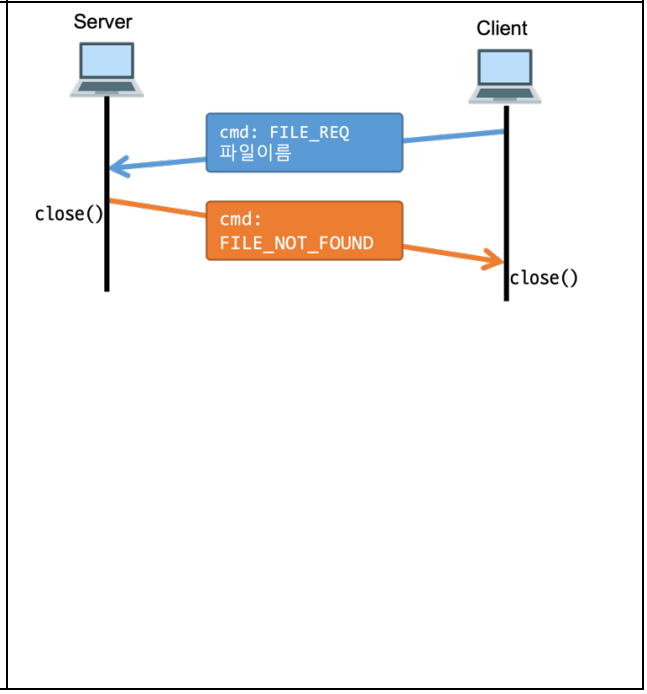
  - **클라이언트 기능 구현** (12점)
    - ✓ 클라이언트는 사용자로부터 전송 받을 파일의 이름을 입력 받고, 서버에 해당 파일 이름을 전송함 (FILE_REQ) (2점)
    - ✓ 서버로 부터 FILE_RES 메시지를 수신하면, 수신된 PACKET 구조체의 buf 내용을 화면에 출력 (2점)
    - ✓ 클라이언트는 서버에서 FILE_END 메시지를 수신하면 FILE_END_ACK를 전송하고 소켓을 닫고 프로그램을 종료함 (2점)

- ✓ 서버에서 전송한 횟수 및 전송된 바이트 수는 클라이언트에서 수신된 횟수 및 수신된 바이트 수와 일치해야 됨 (6점)

■ **서버 기능 구현** (18점)

- ✓ 서버는 클라이언트에게 전송 받은 파일이름을 확인해서 파일이 있는 경우, 파일을 버퍼의 크기(`BUF_SIZE`) 만큼 읽고 `buf_len` 필드에는 실제 읽은 바이트 수를 저장하여 1초 간격으로 전송함 (6점)
- ✓ 파일이 서버에 없는 경우, `FILE_NOT_FOUND`를 클라이언트에게 전송하고 소켓 종료 (2점)
- ✓ 메시지 송수신 과정을 화면에 출력함(실행 결과 참조) (4점)
- ✓ 파일의 마지막 부분은 `FILE_END` 명령을 사용하여 남은 파일 내용을 전송 (4점)
- ✓ 클라이언트로 부터 `FILE_END_ACK(3)`를 수신하면 소켓을 닫고, 프로그램 종료 (2점)

동작 과정

클라이언트가 요청한 파일이 있는 경우	클라이언트가 요청한 파일이 없는 경우
	

실행 결과 #1: 해당 파일이 없는 경우

- **cmd=4(FILE_NOT_FOUND)** 전송 후 서버와 클라이언트 종료

서버	클라이언트
$ ./server 9190 [Rx] cmd: 0, file_name: abc.txt [Tx] cmd: 4, abc.txt: File Not Found ---------- Total Tx count: 0, bytes: 0 UDP Server Socket Close! ----------	$ ./client 127.0.0.1 9190 Input file name: abc.txt [Tx] cmd: 0, file name: abc.txt [Rx] cmd: 4, abc.txt: File Not Found ---------- Total Rx count: 0, bytes: 0 UDP Client Socket Close! ----------

실행 결과 #2: 파일의 크기가 1024 바이트 보다 작은 경우(low_open.txt)

- 클라이언트는 수신된 파일의 내용을 화면에 출력

서버 동작 과정

```
$ ./server 9190
[Rx] cmd: 0, file_name: low_open.txt
[Tx] cmd: 2, len: 641, total_tx_cnt: 1, total_tx_bytes: 641
[Rx] cmd: 3, FILE_END_ACK
----------
Total Tx count: 1, bytes: 641
UDP Server Socket Close!
----------
```

클라이언트 동작 과정 (서버에 udp.txt 파일을 요청한 경우)

```
$ ./client 127.0.0.1 9190
Input file name: low_open.txt
[Tx] cmd: 0, file name: low_open.txt
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <unistd.h>

void error_handling(char* message);

int main(void)
{
        int fd;
        int size;
        char buf[]="Let's go!\n";

        fd=open("data.txt", O_CREAT|O_WRONLY|O_TRUNC, 0644);
        if(fd==-1)
                error_handling("open() error!");
        printf("file descriptor: %d \n", fd);

        size = write(fd, buf, sizeof(buf));
        printf("write size: %d\n", size);
        //if(write(fd, buf, sizeof(buf))==-1)
        if(size ==-1)
                error_handling("write() error!");
```

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

void error_handling(char* message)
{
        fputs(message, stderr);
        fputc('\n', stderr);
        exit(1);
}
[Tx] cmd: 3, FILE_END_ACK
-----------------------------------
Total Rx count: 1, bytes: 641
UDP Client Socket Close!
-----------------------------------
```

실행 결과 #3: 파일의 크기가 1024 바이트 보다 큰 경우(udp.txt, tcp.txt)

– 클라이언트는 수신된 파일의 내용을 화면에 출력

서버 동작 과정

```
$ ./server 9190
[Rx] cmd: 0, file_name: udp.txt
[Tx] cmd: 1, len: 1024, total_tx_cnt: 1, total_tx_bytes: 1024
[Tx] cmd: 1, len: 1024, total_tx_cnt: 2, total_tx_bytes: 2048
[Tx] cmd: 1, len: 1024, total_tx_cnt: 3, total_tx_bytes: 3072
[Tx] cmd: 1, len: 1024, total_tx_cnt: 4, total_tx_bytes: 4096
[Tx] cmd: 1, len: 1024, total_tx_cnt: 5, total_tx_bytes: 5120
[Tx] cmd: 2, len: 776, total_tx_cnt: 6, total_tx_bytes: 5896
[Rx] cmd: 3, FILE_END_ACK
----------------------------------------
Total Tx count: 6, bytes: 5896
UDP Server Socket Close!
----------------------------------------
$
```

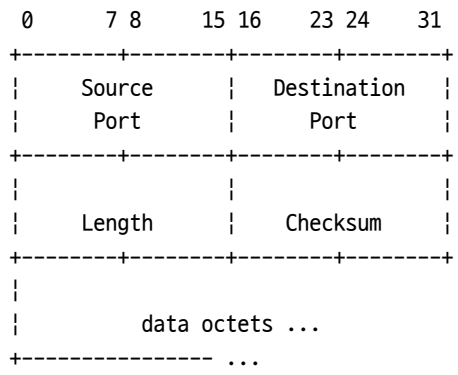
클라이언트 동작 과정 (서버에 udp.txt 파일을 요청한 경우)

```
$ ./client 127.0.0.1 9190
Input file name: udp.txt
[Tx] cmd: 0, file name: udp.txt


RFC 768                                                    J. Postel
                                                               ISI
                                                    28 August 1980



                        User Datagram Protocol
                        ----------------------

Introduction
------------

This User Datagram  Protocol  (UDP)  is  defined  to  make  available  a
```

datagram mode of packet-switched computer communication in the environment of an interconnected set of computer networks. This protocol assumes that the Internet Protocol (IP) [1] is used as the underlying protocol.

This protocol provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. Applications requiring ordered reliable delivery of streams of data should use the Transmission Control Protocol (TCP) [2].

Format



User Datagram Header Format

. . .

References
----

[1] Postel, J., "Internet Protocol," RFC 760, USC/Information Sciences Institute, January 1980.

[2] Postel, J., "Transmission Control Protocol," RFC 761, USC/Information Sciences Institute, January 1980.

[3] Postel, J., "Internet Name Server," USC/Information Sciences Institute, IEN 116, August 1979.

[4] Sollins, K., "The TFTP Protocol," Massachusetts Institute of Technology, IEN 133, January 1980.

[5] Postel, J., "Assigned Numbers," USC/Information Sciences Institute, RFC 762, January 1980.

[Tx] cmd: 3, FILE_END_ACK

Total Rx count: 6, bytes: 5896
UDP Client Socket Close!

```
$
```