네트워크 프로그래밍 과제 #01

☞ 주의 사항

아래의 조건(TCP 소켓, 구조체 및 cmd type)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. TCP 통신을 이용한 원격 파일 전송 및 뷰어 프로그램 (총 20점)

- 제출파일: **view_server.c, view_client.c**

- 주어진 파일(udp,txt, tcp.txt, road.txt 등) 파일을 이용하여 구현된 기능을 검증하세요.
  - ■ 공통 사항
    - ✓ 클라이언트, 서버의 파일 송수신 버퍼 크기는 100 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
  - ■ 클라이언트, 서버 공용 데이터 구조

```c
#define BUF_SIZE 100
// cmd type
#define FILE_REQ 1
#define FILE_RES 2
#define FILE_END 3
#define FILE_END_ACK 4
#define FILE_NOT_FOUND 5

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

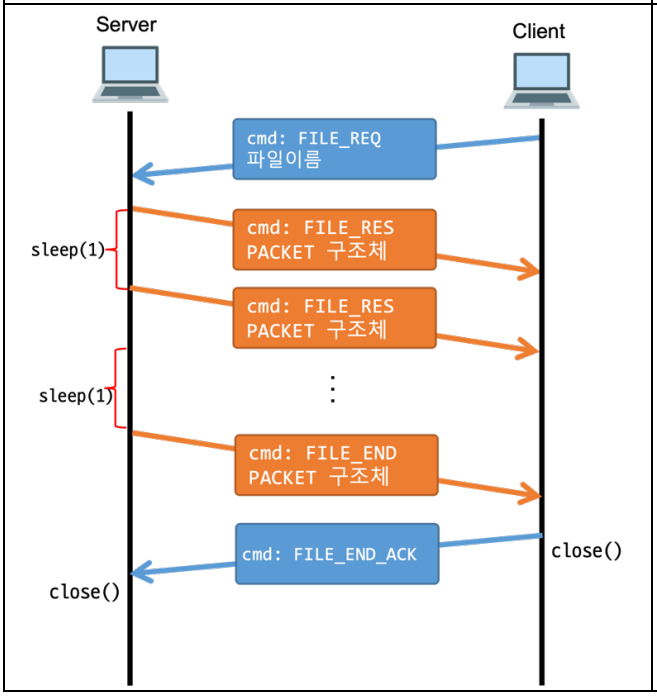
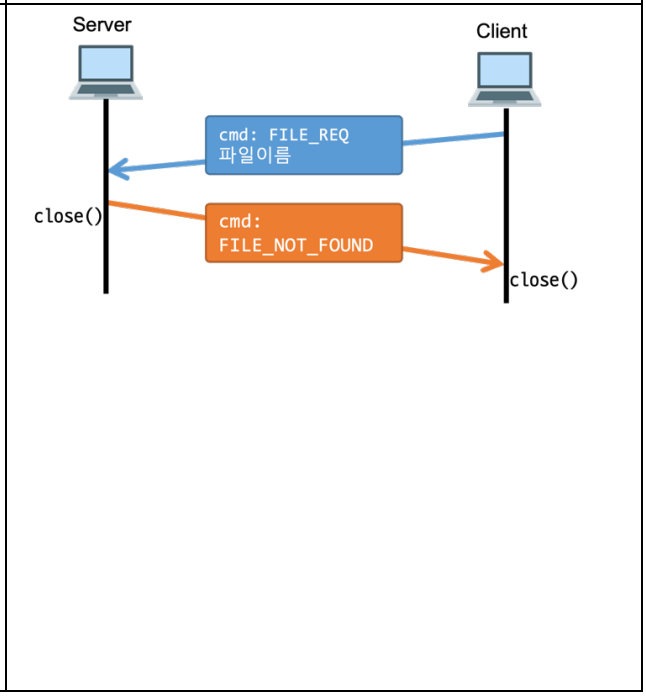
  - ■ 클라이언트 기능 구현 (8점)
    - ✓ 클라이언트는 사용자로부터 전송 받을 파일의 이름을 입력 받고, 서버에 해당 파일 이름을 전송함 (FILE_REQ) (2점)
    - ✓ 서버로 부터 FILE_RES 메시지를 수신하면, 수신된 PACKET 구조체의 buf 내용을 화면에 출력 (2점)
    - ✓ 클라이언트는 서버에서 FILE_END 메시지를 수신하면 FILE_END_ACK를 전송 후 클라이언트의 소켓을 닫고 프로그램을 종료함 (2점)
    - ✓ 서버에서 전송한 회수 및 전송된 바이트 수는 클라이언트에서 수신된 회수 및 수신된

바이트 수와 일치해야 됨 (2점)

■ **서버 기능 구현** (12점)

- ✓ 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 (4점)
- ✓ 파일이 서버에 없는 경우, FILE_NOT_FOUND를 클라이언트에게 전송하고 소켓 종료 (2점)
- ✓ 메시지 송수신 과정을 화면에 출력함(실행 결과 참조) (2점)
- ✓ 파일의 마지막 부분은 FILE_END 명령을 사용하여 남은 파일 내용을 전송 (2점)
- ✓ 클라이언트로 부터 FILE_END_ACK를 수신하면 소켓을 닫고, 프로그램 종료 (2점)
- ✓ 반드시 읽은 파일은 close() 시켜야 됨

동작 과정

클라이언트가 요청한 파일이 있는 경우	클라이언트가 요청한 파일이 없는 경우
	

실행 결과 #1: 해당 파일이 없는 경우

- cmd=5(FILE_NOT_FOUND) 전송 후 서버와 클라이언트 종료

서버	클라이언트
$ ./server 9190 ------------------------------ TCP Remote File View Server ------------------------------ [Rx] cmd: 1, file_name: aaa.txt [Tx] cmd: 5, aaa.txt: File Not Found -- Total Tx count: 0, bytes: 0 TCP Server Socket Close! --	$ ./client 127.0.0.1 9190 Input file name: aaa.txt [Tx] cmd: 1, file name: aaa.txt [Rx] cmd: 5, aaa.txt: File Not Found ---------------------------------- Total Rx count: 0, bytes: 0 TCP Client Socket Close! ----------------------------------

실행 결과 #2: 파일의 크기가 작은 경우(road.txt)

- 클라이언트는 수신된 파일의 내용을 화면에 출력

서버 동작 과정

```
$ ./server 9190
------------------------------
TCP Remote File View Server
------------------------------
[Rx] cmd: 1, file_name: road.txt
[Tx] cmd: 2, len: 100, total_tx_cnt:   1, total_tx_bytes: 100
[Tx] cmd: 2, len: 100, total_tx_cnt:   2, total_tx_bytes: 200
[Tx] cmd: 2, len: 100, total_tx_cnt:   3, total_tx_bytes: 300
[Tx] cmd: 2, len: 100, total_tx_cnt:   4, total_tx_bytes: 400
[Tx] cmd: 2, len: 100, total_tx_cnt:   5, total_tx_bytes: 500
[Tx] cmd: 2, len: 100, total_tx_cnt:   6, total_tx_bytes: 600
[Tx] cmd: 2, len: 100, total_tx_cnt:   7, total_tx_bytes: 700
[Tx] cmd: 2, len: 100, total_tx_cnt:   8, total_tx_bytes: 800
[Tx] cmd: 3, len:   7, total_tx_cnt:   9, total_tx_bytes: 807
[Rx] cmd: 4, FILE_END_ACK
----------------------------------------
Total Tx count: 9, bytes: 807
TCP Server Socket Close!
----------------------------------------
```

클라이언트 동작 과정 (서버에 road.txt 파일을 요청한 경우)

```
$ ./client 127.0.0.1 9190
Input file name: road.txt
[Tx] cmd: 1, file name: road.txt
The Road Not Taken

                    Robert Lee Frost

Two roads diverged in a yellow wood
And sorry I could not travel both
And be on traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth;

Then took the other, as just as fair
```

```
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that, the passing there
Had worn them really about the same,

And both that morning equally lay
In leaves no step had trodden black.
Oh, I kept the first for another day!
Yet knowing how way leads on to way,
I doubted if I should ever come back,

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood and I-
I took the one less traveled by,
And that has made all the difference.
---------------------------
[Rx] cmd: 3, FILE_END
[Tx] cmd: 4, FILE_END_ACK
-------------------------------------
Total Rx count: 9, bytes: 807
TCP Client Socket Close!
-------------------------------------
```

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

- 클라이언트는 수신된 파일의 내용을 화면에 출력

서버 동작 과정

```
$ ./server 9191
------------------------------
TCP Remote File View Server
------------------------------
[Rx] cmd: 1, file_name: udp.txt
[Tx] cmd: 2, len: 100, total_tx_cnt:   1, total_tx_bytes: 100
[Tx] cmd: 2, len: 100, total_tx_cnt:   2, total_tx_bytes: 200
[Tx] cmd: 2, len: 100, total_tx_cnt:   3, total_tx_bytes: 300
[Tx] cmd: 2, len: 100, total_tx_cnt:   4, total_tx_bytes: 400

                . . . (중간 생략)

[Tx] cmd: 2, len: 100, total_tx_cnt:  58, total_tx_bytes: 5800
[Tx] cmd: 3, len:  96, total_tx_cnt:  59, total_tx_bytes: 5896
[Rx] cmd: 4, FILE_END_ACK
---------------------------------------
Total Tx count: 59, bytes: 5896
TCP 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

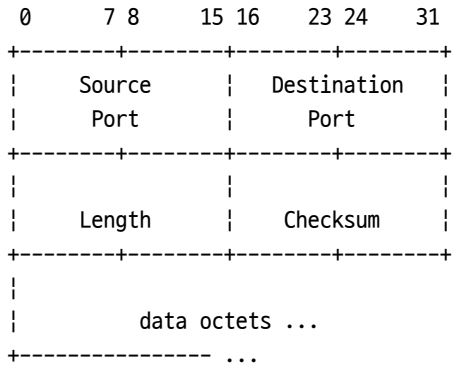
User Datagram Protocol

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

. . . (중간 생략)

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.



Postel                                                          [page 3]


---------------------------
[Rx] cmd: 3, FILE_END
[Tx] cmd: 4, FILE_END_ACK
-----------------------------------
Total Rx count: 59, bytes: 5896
TCP Client Socket Close!
-----------------------------------
$
```