Network Programming 과제 #06

☞ 주의 사항

아래의 조건(TCP 소켓, 구조체)을 따르지 않고 구현한 경우, 0 점 처리함 각 소스 파일에 학번, 이름(영문 가능) 주석이 없는 경우, 파일당 -1 점 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 -5 점 감점함

1. TCP 통신을 이용한 다중 접속 Ping-Pong 메시지 전송 프로그램 구현 (20점)

- 파일제출: hw6_server.c, hw6_client.c

■ 공통 사항

✓ 10장 echo_mpserver.c echo_client.c 를 최대한 활용

✓ 클라이언트, 서버의 구조체는 동일함

■ 클라이언트, 서버 공용 데이터 구조

```c
#define BUF_SIZE 100
#define PING_MSG 1 // Client -> Server
#define PONG_MSG 2 // Server -> Client
#define TERMINATE_MSG 3 // Server -> Client

// Packet structure for communication
typedef struct
{
    int cmd; // PING_MSG, PONG_MSG, TERMINATE_MSG
    char time_msg[BUF_SIZE];
} PACKET;
```

■ 현재 시간 정보를 가져오는 샘플 코드

```c
#include <stdlib.h>
#include <time.h>
. . .
// example code
time_t now = time(NULL); // 1970 년 1 월 1 일부터 0 시부터 현재까지 경과된 시간

struct tm *p;
p = localtime(&now);
char res[50];
// Year-Month-Day Hour:Minute:Second: 모든 구조체 멤버는 int 형
sprintf(res, "%d-%d-%d  %d:%d:%d",
       1900 + p->tm_year, 1 + p->tm_mon, p->tm_mday,
       p->tm_hour, p->tm_min, p->tm_sec);
```

■ 시간 차이 계산 함수

double difftime(time_t time_end, time_t time_start); 초 단위 시간 리턴

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

int main()
{
    time_t start, end;
    double elapsed_time = 0;

    start = time(NULL);
    for (int i = 0; i < 3; i++)
    {
        sleep(1);
    }
    end = time(NULL);

    elapsed_time = difftime(end, start);
    printf("elapsed_time: %.0f sec\n", elapsed_time);

    return 0;
}
```

■ 클라이언트 (8점)

✓ rand() 함수를 이용하여 1~5 사이의 숫자(rand_sec)를 랜덤하게 생성하고 sleep(rand_sec)함수를 호출하여 임의의 지연을 생성함 (2점)

✓ 현재 시간 정보를 계산하여 PACKET 구조체의 time_msg에 저장 후 서버로 전송 (4점)

- cmd=PING_MSG, time_msg: 현재 시간 정보(시:분:초) 저장
- 정상적인 경우, 총 10회 전송 후 클라이언트 종료
- 반드시 화면에 전송 정보 출력 (실행 결과 참조)

✓ 서버로 부터 TERMINATE_MSG를 수신하면 클라이언트는 연결을 종료함 (2점)

■ 서버 (무한 반복) (12점)

✓ 서버는 반드시 여러 클라이언트의 접속 요청을 수용해야 됨

- 1:1 통신만 되는 경우: -10점

✓ 클라이언트의 PING_MSG를 수신하면 첫 번째 수신 메시지는 시간 계산을 하지 않고, 두 번째 PING_MSG부터 이전 메시지와 시간차이(초 단위)가 4초 이내 이면 정상적인 PONG_MSG를 각 클라이언트로 전송 (8점)

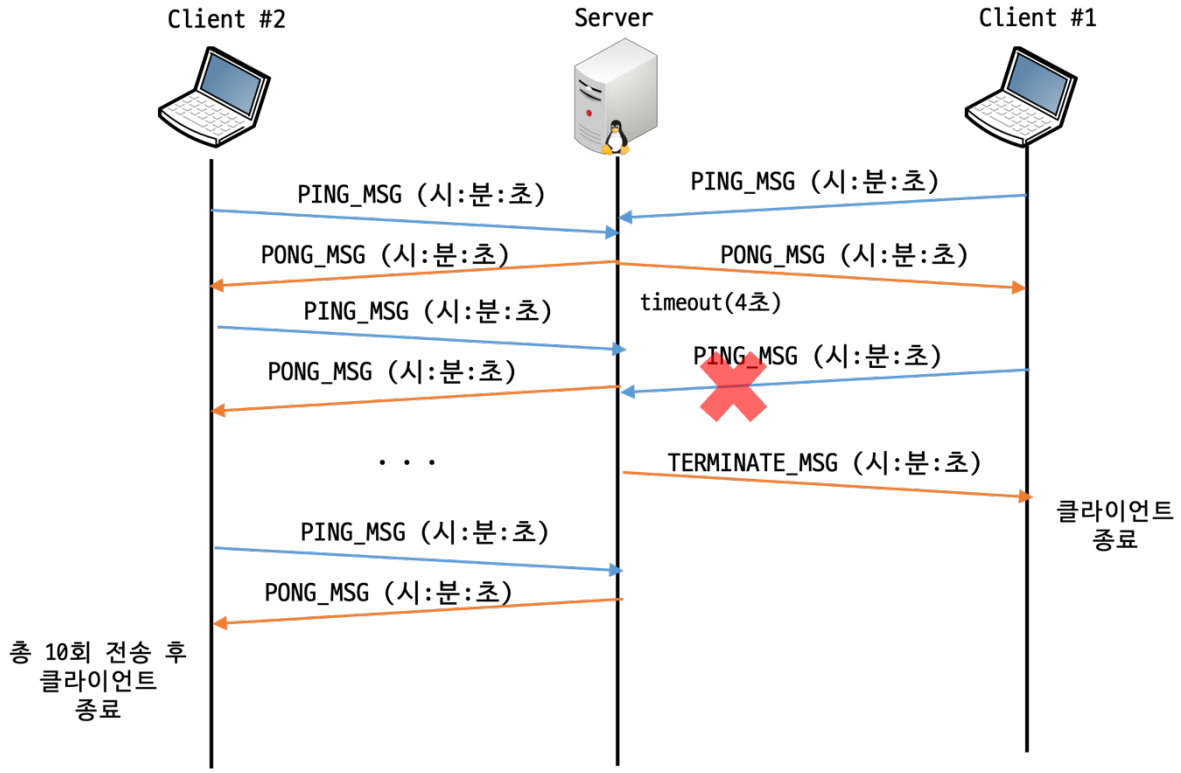
- 클라이언트를 구분하기 위해 port 번호 출력: 2점
- PONG_MSG 전송 기능: 현재 시간 정보 추가함: 6점

✓ 이전 메시지와의 시간 간격(time_diff)이 4초가 넘으면 서버는 TERMINATE_MSG 를 해당 클라이언트에게 전송함 (4점, 각 2점)

- 이전 PING 메시지와의 시간 차이 계산
- 4초가 넘으면 TERMINATE_MSG 전송

■ 동작 과정

- 클라이언트는 정상 상태인 경우, 총 10 회 PING_MSG 를 전송함



[실행 결과]

서버

```
$ ./server 9190
new client connected: 4
new client connected: 4
[Rx] PING(1) time: 13:4:40 from Port(32810)=> [Tx] PONG(2) 13:4:40 to Port(32810)
[Rx] PING(1) time: 13:4:40 from Port(32816)=> [Tx] PONG(2) 13:4:40 to Port(32816)
[Rx] PING(1) time: 13:4:42 from Port(32810)=> [Tx] PONG(2) 13:4:42 to Port(32810)
[Rx] PING(1) time: 13:4:42 from Port(32816)=> [Tx] PONG(2) 13:4:42 to Port(32816)
[Rx] PING(1) time: 13:4:47 from Port(32810)=> [Tx] TERMINATE(3): 13:4:47 to Port(32810)
client disconnected...
removed proc id: 734707
[Rx] PING(1) time: 13:4:47 from Port(32816)=> [Tx] TERMINATE(3): 13:4:47 to Port(32816)
client disconnected...
```

```
removed proc id: 734709
```

클라이언트 #1

```
$ ./client 127.0.0.1 9190
Connected..........
[Tx] PING(1), sleep(2), [ 1]: 13:4:40 => [Rx] PONG(2), time: 13:4:40
[Tx] PING(1), sleep(2), [ 2]: 13:4:42 => [Rx] PONG(2), time: 13:4:42
[Tx] PING(1), sleep(5), [ 3]: 13:4:47 => [Rx] TERMINATE(3), Connection close
Client close
```

클라이언트 #2

```
$ ./client 127.0.0.1 9190
Connected..........
[Tx] PING(1), sleep(2), [ 1]: 13:4:40 => [Rx] PONG(2), time: 13:4:40
[Tx] PING(1), sleep(2), [ 2]: 13:4:42 => [Rx] PONG(2), time: 13:4:42
[Tx] PING(1), sleep(5), [ 3]: 13:4:47 => [Rx] TERMINATE(3), Connection close
Client close
```