리눅스 시스템 term project

제출자: 2017312249 이정민

*첫번째 코드 구하기 과제에서 참고한 블로그 :

https://m.blog.naver.com/PostView.nhn?blogId=clown7942&logNo=110108516505&proxyReferer=https:%2F%2Fwww.google.com%2F

1. 소스코드

<server.c>

```
/* server.c */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <time.h>

#define SOCKSIZE 1460

void error_handling(char *message);

int main(int argc, char **argv)
{
        int serv_sock;
        int clnt_sock;
        int fd;
        struct sockaddr_in serv_addr;

        struct sockaddr_in clnt_addr;
                int clnt_addr_size;
                char clnt_message[30];
                   int str_len;
       char buffer[SOCKSIZE]={0};



       char message[] = "Hello Client! \n";


        if(argc != 3) {
               printf("Usage : &s <port>\n", argv[0]);
               exit(1);
        }

        serv_sock = socket(PF_INET, SOCK_STREAM, 0); /*
서버 소켓 생성 */
        if(serv_sock == -1)
               error_handling("socket() error");

        memset(&serv_addr, 0, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
        serv_addr.sin_port = htons(atoi(argv[1]));
```

```
        // 소켓에 주소 할당
        if(bind(serv_sock, (struct sockaddr*) &serv_addr,
 sizeof(serv_addr)) == -1)
                error_handling("bind() error");

        if(listen(serv_sock, 5) == -
1) // 연결 요청 대기 상태로 진입
                error_handling("listen() error");

        clnt_addr_size = sizeof(clnt_addr);

       // 연결 요청 수락
        clnt_sock = accept(serv_sock, (struct sockaddr*)&
clnt_addr, &clnt_addr_size);
        if(clnt_sock == -1)
                error_handling("accept() error");

       //hello server 받기
       str_len = read(clnt_sock, clnt_message, sizeof(cln
t_message)-1);

       if(str_len == -1)
                error_handling("read() error");

       message[str_len]=0;
       printf("Message from client: %s \n", clnt_message)
;

      // hello client 전송
       write(clnt_sock, message, sizeof(message));
```

```
       //전송할 파일 열기
       FILE *myfile = fopen(argv[2], "rb");
       if(myfile==NULL) error_handling("fopen() error");
       else printf("open success\n");

       //파일 전송
       clock_t start= clock();
       while(1){
       int tmp=fread(buffer, sizeof(char), SOCKSIZE, myfi
le);
       write(clnt_sock, buffer, tmp);
       sleep(5);
       if(feof(myfile)) break;
       }
       clock_t end= clock();
       printf("end time - start time : %lf\n", (double)(e
nd-start)/CLOCKS_PER_SEC);
       // 연결 종료
        close(clnt_sock);
        close(serv_sock);

        return 0;
}

void error_handling(char *message)
{
        fputs(message, stderr);
        fputc('\n', stderr);
        exit(1);
}
```

<client.c>

```c
/* client.c */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>

#define SOCKSIZE 1460
void error_handling(char *message);

int main(int argc, char **argv)
{
        int sock;
        struct sockaddr_in serv_addr;
        char message[30];
        char buffer[SOCKSIZE]={0};
        int str_len_tocli;
        char new_message[30]="Hello Server!!";
        char file_name[1020]="";



        if(argc != 3) {
               printf("Usage : &s <IP>\n", argv[0]);
               exit(1);
        }

        sock = socket(PF_INET, SOCK_STREAM, 0); /* 서버 접속을 위한 소켓 생성 */
        if(sock == -1)
            error_handling("socket() error");

        memset(&serv_addr, 0, sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
        serv_addr.sin_port = htons(atoi(argv[2]));

    if( connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) == -1)
            error_handling("connect() error");

      //hello server 전송
        write(sock, new_message, sizeof(new_message));

       //hello client 수신
        str_len_tocli = read(sock, message, sizeof(message)-1);

        if(str_len_tocli == -1)
               error_handling("read() error");
        message[str_len_tocli]=0;
        printf("Message from server: %s \n", message);

      //영상수신
```

```
        FILE *recvfile= fopen("/mnt/c/SKKU/2020 1학기/Lin
ux/termpj/client/copyfile.mp4","wb");

        int tmp=read(sock, buffer, SOCKSIZE);
        fwrite(buffer, sizeof(char), tmp, recvfile);

        while(tmp!=0){
        tmp=read(sock, buffer, SOCKSIZE);
        fwrite(buffer, sizeof(char), tmp, recvfile);
        }



        close(sock);      // 연결 종료

        return 0;
}

void error_handling(char *message)
{
        fputs(message, stderr);
        fputc('\n', stderr);
        exit(1);
}
```

2. 실행화면 캡쳐

1.49GB 파일을 서버에서 클라이언트로 전송한 결과

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/startfile2.m
p4"
Message from client: Hello Server!
open success
end time - start time : 22.671875
```

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ gcc -o client client.c
client.c: In function ‘main’:
client.c:27:23: warning: too many arguments for format [-Wformat-extra-args]
   27 |                 printf("Usage : &s <IP>\n", argv[0]);
      |                       ^~~~~~~~~~~~~~~~~~~~
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./client 127.0.0.1 10000
Message from server: Hello Client!
```

리눅스에서 두 파일이 같은지 여부 검사

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj$ diff ./no1fin/startfile2.mp4 ./client/copyfile.mp4
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj$ diff ./no1fin/startfile.mp4 ./client/copyfile.mp4
Binary files ./no1fin/startfile.mp4 and ./client/copyfile.mp4 differ
```

->첫번째줄이 원본파일(startfile2)과 복사파일(copyfile)이며 두번째 줄이 다름의 예시를 보기 위한 다른 파일(startfile)과 복사파일(copyfile)

->확인결과 일치함을 확인

3. 구현/개발 내용

1) 처음시도

```
//전송할 파일 열기
	FILE *myfile = fopen(argv[2], "rb");
	if(myfile==NULL) error_handling("fopen() error");
	else {
		printf("open success\n");
		fseek(myfile, 0, SEEK_END);	// 파일 포인터를 파일의 끝으로 이동시킴
		int size = ftell(myfile);
		printf("%d\n", size);
		fseek(myfile,0, SEEK_SET);
	}
```

처음에는 파일을 열어 파일 포인터를 끝으로 이동시키고 ftell()함수를 이용해서 마지막 16진수의 버퍼를 보고 직접 while문을 돌면서 1460개씩 읽으며 직접 계산했지만 마지막 처리과정에서 쓰레기값을 넘기는 것을 확인해 지금과 같은 방식으로 방법을 바꿨습니다.

2) 현재 구현 결과

서버에서 소켓을 생성하고 주소를 할당한 뒤 클라이언트의 연결 요청을 기다린다. 클라이언트도 서버접속을 위한 소켓을 생성하고 주소를 할당한다. 클라이언트가 서버에 연결 요청을 보내면 서버는 수락하고 연결에 성공하면 클라이언트는 서버에게 "Hello Server"이라는 메시지를 전송하고 서버는 이를 출력한다. 그 후 서버는 클라이언트에게 "Hello Client"를 전송하고 클라이언트는 이를 출력한다. 그 후 서버는 argv[3]의 데이터를 1460씩 나누어 클라이언트에 전송하고 클라이언트는 copyfile이라는 파일을 만든다. 수신이 완료되면 연결을 각각 종료한다. 이때 전송 시간을 측정하기 위해 clock()함수를 이용한다. 전송이 계속되면 데이터가 밀릴 수 있기 때문에 중간에 sleep()함수를 넣었다.

3) 실험내용

너무 큰 사이즈로 실험을 하기에는 시간의 지나치게 많이 소요되기 때문에 542MB의 테스트 파일로 진행하였습니다. 시도마다 소요시간이 다르게 나오기 때문에 두번의 평균값을 이용했습니다.

<전송패킷 단위 변화>

-100000

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 1.453125
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 1.140625
```

-5000

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 6.515625
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 5.953125
```

-2000

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 9.375000
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 8.921875
```

-1600

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 10.375000
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 10.843750
```

-1460

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 11.437500
```

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 11.109375
```

-1200

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 12.984375
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 12.937500
```

-1000

```
^[[Aljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile
.mp4"
Message from client: Hello Server!
open success
end time - start time : 14.609375
```

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin$ ./server 10000 "/mnt/c/SKKU/2020 1학기/Linux/termpj/no1fin/testfile.mp4
"
Message from client: Hello Server!
open success
end time - start time : 13.843750
```

-정리결과

전송 단위	소요시간(초)
100000	1.296875
5000	6.234375
2000	9.1484375
1600	10.609375
1460	11.2734375
1200	12.9609375
1000	14.2265625

복사가 잘 되었는지 검사

```
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj$ diff ./no1fin/testfile.mp4 ./client/copyfile.mp4
ljm9748@LAPTOP-99TN79NB:/mnt/c/SKKU/2020 1학기/Linux/termpj$
```

패킷 단위를 다양하게 설정해보았는데 무조건 패킷크기가 클수록 초가 짧아지게 나왔습니다. 짧아지다가 어느순간 커지거나 중간에 깨질 수 있다고 생각했지만 복사가 잘 이루어진 것을 확인했습니다. 시간이 부족해 끝까지 확인하지 못했지만 이렇게 된 이유는 sleep() 혹은 파일의 사이즈가 작아서이기 때문이라고 생각합니다. 다음에 시간이 더 있다면 ①파일의 사이즈를 늘려서 실험을 하고 그 상황에서도 이런 결과가 나온다면 ②sleep()을

조정하며 diff를 이용해 파일의 손상이 없는 상태에서 확인하고 싶습니다.