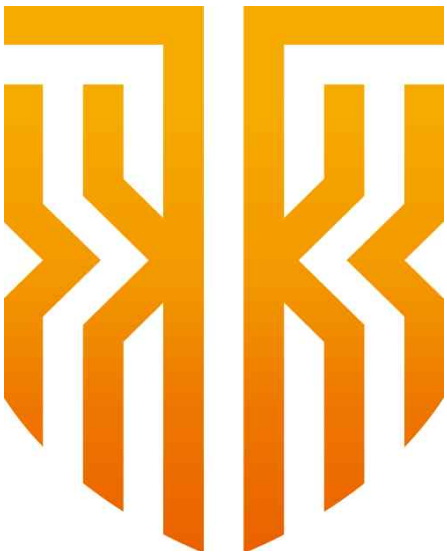
과제 4
소캣의 우아한 연결종료



목차

1. 과제 설명

가. 과제 설명

과제 4

파일 전송 프로그램을 다음과 같은 형태로 작성하고 레포트로 제출하시오.
레포트 포함 내용 (표지, 코드, 중요코드 설명, 실행화면 캡쳐, 전송된 파일 확인 캡쳐)

클라인언트 명령 : file_client 127.0.0.1 9190
서버 명령 : file_server 9190 send.txt

나. 명령 설명

1) 클라이언트 : argv[2]을 전달받는다.
   가) argv[0] = file_client // 파일 이름
   나) argv[1] = 127.0.0.1 // 로컬 주소
   다) argv[2] = 9190 // 포트 번호
2) 서버 : argv[2]을 전달한다.
   가) avgv[0] = file_server // 파일 이름
   나) argv[1] = 9190 // 로컬 주소
   다) argv[2] = send.txt // 전달할 파일 이름

2. 코드 설명

가. 전체 코드

1) file_server.c

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

#define BUF_SIZE 1024
void error_handling(char *message);

int main(int argc, char *argv[]){
        int serv_sd, clnt_sd;
        FILE *fp;//, file_name;
        char buf[BUF_SIZE];
        char message[BUF_SIZE];
        int read_cnt;

        size_t fsize, nsize = 0;
        size_t fsize2;

        struct sockaddr_in serv_adr, clnt_adr;
        socklen_t clnt_adr_sz;

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

        serv_sd=socket(PF_INET, SOCK_STREAM, 0);
        if(serv_sd==-1)
                error_handling("socket() error");

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

        if(bind(serv_sd, (struct sockaddr*)&serv_adr, sizeof(serv_adr))==-1)
                error_handling("bind() error");
        if(listen(serv_sd, 5) == -1)
                error_handling("listen() error");

        clnt_adr_sz = sizeof(clnt_adr);
```

```
	clnt_sd = accept(serv_sd, (struct sockaddr*)&clnt_adr, &clnt_adr_sz);
	if(clnt_sd == -1)
		error_handling("accept() error");


	write(clnt_sd, argv[2], BUF_SIZE);

	// print message
	read(clnt_sd, message, BUF_SIZE);
	printf("Message from client : %s ₩n", message);

	// send file ( server to client) is argv[2]
	fp=fopen(argv[2], "rb");
	// move file pointer to file end
	fseek(fp, 0, SEEK_END);
	// calculate file size
	fsize = ftell(fp);
	// move file pointer to file start
	fseek(fp, 0, SEEK_SET);

	// send file contents
	while ( nsize != fsize){
		// read from file to buf

		int fpsize = fread(buf, 1, BUF_SIZE, fp);
		nsize += fpsize;
		send(clnt_sd, buf, fpsize, 0);
	}

	shutdown(clnt_sd, SHUT_WR);

	fclose(fp); //fclose(file_name);
	close(clnt_sd); close(serv_sd);
	return 0;
}
void error_handling(char *message){
	fputs(message, stderr);
	fputc('₩n', stderr);
	exit(1);
}
```

2) file_client.c

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

#define BUF_SIZE 1024
void error_handling(char *message);

int main(int argc, char *argv[])
{
        int sd, i = 0; //소켓 디스크립터
        int nbyte = BUF_SIZE;
        size_t buf_size = BUF_SIZE;
        FILE *fp; //파일 포인터
        char buf[BUF_SIZE]; //버퍼 크기
        int read_cnt; //읽은 개수
        struct sockaddr_in serv_adr; //서버 주소 소켓
        if(argc!=3) { //사용방법 표시  ex) ./fclient 127.0.0.1 9190
                printf("Usage: %s <IP> <port>₩n", argv[0]);
                exit(1);
        }
        char file_name[BUF_SIZE];
        char your_message[BUF_SIZE];

        sd=socket(PF_INET, SOCK_STREAM, 0);      //소켓 디스크립터 생성

        memset(&serv_adr, 0, sizeof(serv_adr)); //서버의 주소 정보 메모리 할당
        serv_adr.sin_family=AF_INET;
        serv_adr.sin_addr.s_addr=inet_addr(argv[1]);
        serv_adr.sin_port=htons(atoi(argv[2]));

        connect(sd, (struct sockaddr*)&serv_adr, sizeof(serv_adr)); //클라이언트 소켓을 서버 소켓에 연
결
        read( sd, file_name, BUF_SIZE );
        printf("Received File Name : %s₩n", file_name);
        fp=fopen(file_name, "wb");


        printf("your message?");
        fgets(your_message, BUF_SIZE, stdin);
        while(your_message[i++]){} //read_cnt가 0이 아니면 계속 읽기
        write(sd, your_message, sizeof(your_message)+1);  //sd를 통해 서버쪽에 메세지 전달.
```

```
	while(nbyte != 0){
		nbyte = recv(sd, buf, buf_size, 0);
		fwrite(buf, sizeof(char), nbyte, fp);
	}

	puts("Received file data₩n");
	fclose(fp); //파일 닫기
	close(sd);  //클라이언트 소켓 닫기
	return 0;
}

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

나. file_server 중요코드 설명

```
#define BUF_SIZE 1024
. . .
int serv_sd, clnt_sd;
FILE *fp;//, file_name;
char buf[BUF_SIZE];
char message[BUF_SIZE];
int read_cnt;

size_t fsize, nsize = 0;
size_t fsize2;
struct sockaddr_in serv_adr, clnt_adr;
socklen_t clnt_adr_sz;
```

1) send.txt의 이름을 client로 보내는 부분

```
write(clnt_sd, argv[2], BUF_SIZE);
```

: argv[0] argv[1] argv[2] == ./fserver 9190 send.txt
client와 연결된 이후에 write를 통해 client로 보내면
client는 read를 통해 server로부터 파일명을 받는다.

2) server 메시지 수신 부분

```
char message[BUF_SIZE];
 . . .
// print message
read(clnt_sd, message, BUF_SIZE);
printf("Message from client : %s ₩n", message);
```

: client에서 보낸 메시지를 수신하는 부분이다.

3) 파일 송신 부분

```
// send file ( server to client) is argv[2]
fp=fopen(argv[2], "rb");
// move file pointer to file end
fseek(fp, 0, SEEK_END);
// calculate file size
fsize = ftell(fp);
// move file pointer to file start
fseek(fp, 0, SEEK_SET);

// send file contents
while ( nsize != fsize){
        // read from file to buf

        int fpsize = fread(buf, 1, BUF_SIZE, fp);
        nsize += fpsize;
        send(clnt_sd, buf, fpsize, 0);
}
```

: 파일 포인터를 이용해 argv[2]에 해당하는 파일에 저장된 내용을 읽고
그 내용을 send를 통해 client로 보낸다.

다. file_client 중요코드 설명

```
int sd, i = 0; //소켓 디스크립터
int nbyte = BUF_SIZE;
size_t buf_size = BUF_SIZE;
FILE *fp; //파일 포인터
char buf[BUF_SIZE]; //버퍼 크기
int read_cnt; //읽은 개수
struct sockaddr_in serv_adr; //서버 주소 소켓
if(argc!=3) { //사용방법 표시  ex) ./fclient 127.0.0.1 9190
        printf("Usage: %s <IP> <port>₩n", argv[0]);
        exit(1);
}
char file_name[BUF_SIZE];
char your_message[BUF_SIZE];
```

1) send.txt의 이름을 server로부터 받는 부분

```
char file_name[BUF_SIZE];
. . .
read( sd, file_name, BUF_SIZE );
printf("Received File Name : %s₩n", file_name);
fp=fopen(file_name, "wb");
```

: file_name을 server로부터 읽어와서 그 이름을 출력시키고
fp라는 파일 변수에 server의 argv[2]에 해당하는 파일명으로
client 폴더에 새 파일을 만들어준다.

2) client 메시지 송신 부분

```
char your_message[BUF_SIZE];
. . .
printf("your message?");
fgets(your_message, BUF_SIZE, stdin);
while(your_message[i++]){} //read_cnt가 0이 아니면 계속 읽기
write(sd, your_message, sizeof(your_message)+1);  //sd를 통해 서버쪽에 메세지 전달.
```

: fgets를 통해 your_message에 server로 보낼 문구를 입력하고
while을 통해 your_message 문자열의 끝까지 읽고 난 후
write를 통해 client로 your_message를 전달한다.

3) 파일 수신 부분

```
while(nbyte != 0){
        nbyte = recv(sd, buf, buf_size, 0);
        fwrite(buf, sizeof(char), nbyte, fp);
}

puts("Received file data₩n");
```

: server로부터 send된 내용을 recv로받아 fp에 저장시킨다.
즉, 1)에서 입력받아진 send.txt라는 빈 파일에
3) 에서는 server의 3)에서 보내진 내용물들을 받는다고 이해할 수 있다.

라. 실행한 화면 및 전송된 파일 확인 화면 캡쳐