

Term Project: Linux System(Socket programming)

-다중 파일 전송 방식에 따른 속도 비교

목차

1. 실시간 쌍방향 채팅 프로그램

- 1.1 소스코드
- 1.2 실행 화면
- 1.3 설명

2. 파일 전송

2.1 다중 스레드를 이용한 다중 파일 전송

- 2.1.1 소스코드
- 2.1.2 실행 화면
- 2.1.3 설명

2.2 파일 압축을 이용한 다중 파일 전송

- 2.2.1 소스코드
- 2.2.2 실행 화면
- 2.2.3 설명

2.3 순차적 다중 파일 전송

- 2.3.1 소스코드
- 2.3.2 실행화면
- 2.3.3 설명

2.4 전송 시간 비교

1. 실시간 쌍방향 채팅 프로그램

1.1 소스코드

채팅 프로그램 서버 소스코드

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <pthread.h>
void error_handling(char* message);

typedef struct {
    int* clnt_sock;
    char* message;
}recieve_Data;

void* send_function(void* clnt_sock){
    int* cs=(int*)clnt_sock;
    while(1){
        char message[30];
        printf("\nserver -> : ");
        fgets(message,30,stdin);
        write(*cs,message,sizeof(message));
    }
}

void* recieve_function(void* rcvdt){
    recieve_Data* data =(recieve_Data*)rcvdt;
    while(1){
        int
str_len=read(*(data->clnt_sock),data->message,sizeof(char)*30);

        if(str_len!=-1){
            printf("\n <- client :
%s\n",data->message);
        }
    }
}

int main(int argc, char *argv[]){
    int serv_sock;
    int clnt_sock;
    char message[30];
    struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
    socklen_t clnt_addr_size;

    printf("read port....\n");
    if(argc!=2){
        printf("Usage : %s <port>\n",argv[0]);
        exit(1);
    }

    printf("set server socket\n");
    serv_sock=socket(PF_INET, SOCK_STREAM,0);
```

```
    if(serv_sock==-1){
        error_handling("socket() error");
    }

    printf("set server addr...\n");
    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]));

    printf("binding...\n");
    if(bind(serv_sock,(struct sockaddr*)
&serv_addr,sizeof(serv_addr))==-1){
        error_handling("bind() error");
    }

    if(listen(serv_sock,5)==-1){
        error_handling("listen() error");
    }
    printf("waiting...\n");

    clnt_addr_size =sizeof(clnt_addr);
    clnt_sock=accept(serv_sock,(struct
sockaddr*)&clnt_addr,&clnt_addr_size);
    if(clnt_sock==-1) error_handling("accept() error");
    printf("accept!\n");

    recieve_Data rcvdt;
    rcvdt.clnt_sock=&clnt_sock;
    rcvdt.message=message;

    pthread_t p_thread[2];
    int t;
    int status;
    for(t=0;t<2;t++){
        if(t==0)
pthread_create(&p_thread[t],NULL,send_function,(void*)&clnt_sock);
        else
if(t==1)pthread_create(&p_thread[t],NULL,recieve_function,(void*)
&rcvdt);
    }

    pthread_join(p_thread[0],(void **)&status);
    pthread_join(p_thread[1],(void **)&status);

    close(clnt_sock);
    close(serv_sock);
    return 0;
}

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

채팅 프로그램 클라이언트 소스코드

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

typedef struct recieve_data{
    char* message;
    int* serv_sock;
}recieve_data;

void* send_function(void* serv_sock){
    int* cs=(int*)serv_sock;
    while(1){
        char message[30];
        printf("\nclient -> : ");
        scanf("%s",message);
        write(*cs,message,sizeof(message));
    }
}

void* recieve_function(void* rcvDt){
    recieve_data* data=(recieve_data*)rcvDt;
    char* msg=data->message;
    while(1){
        int
str_len=read(*(data->serv_sock),msg,sizeof(char)*30
);
        if(str_len!=-1){
            printf("\n <- server :
%s\n",msg);
        }
    }
}

int main(int argc, char *argv[]){

    int serv_sock;
    struct sockaddr_in serv_addr;
```

```
char message[30];
int str_len;

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

serv_sock=socket(PF_INET,SOCK_STREAM,0);
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]));

connect(serv_sock,(struct sockaddr
*)&serv_addr,sizeof(serv_addr));

pthread_t p_thread[2];
int t;
int status;
recieve_data rcvDt;
rcvDt.message=message;
rcvDt.serv_sock=&serv_sock;
for(t=0;t<2;t++){
    if(t==0)
pthread_create(&p_thread[t],NULL,send_function,(vo
id*)&serv_sock);
    else
if(t==1)pthread_create(&p_thread[t],NULL,recieve_fu
nction,(void*)&rcvDt);
}
pthread_join(p_thread[0],(void **)&status);
pthread_join(p_thread[1],(void **)&status);

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

1.2 채팅 프로그램 실행화면

서버

```
junho@junho-VirtualBox:~/Desktop/termProject/chatting$ ./chat_server 9080
read port....
set server socket
set server addr...
binding...
waiting...
accept!

server -> :
<- client : Hi I'm client

Hi I'm Server
```

```
server -> : I think everything is fine

server -> :
<- client : I think so. It works very wel

<- client : l

yeah bye

server -> :
<- client : see ya
```

클라이언트

```
junho@junho-VirtualBox:~/Desktop/termProject/chat$ ./chat_client 192.168.219.101
9080

client -> : Hi I'm client

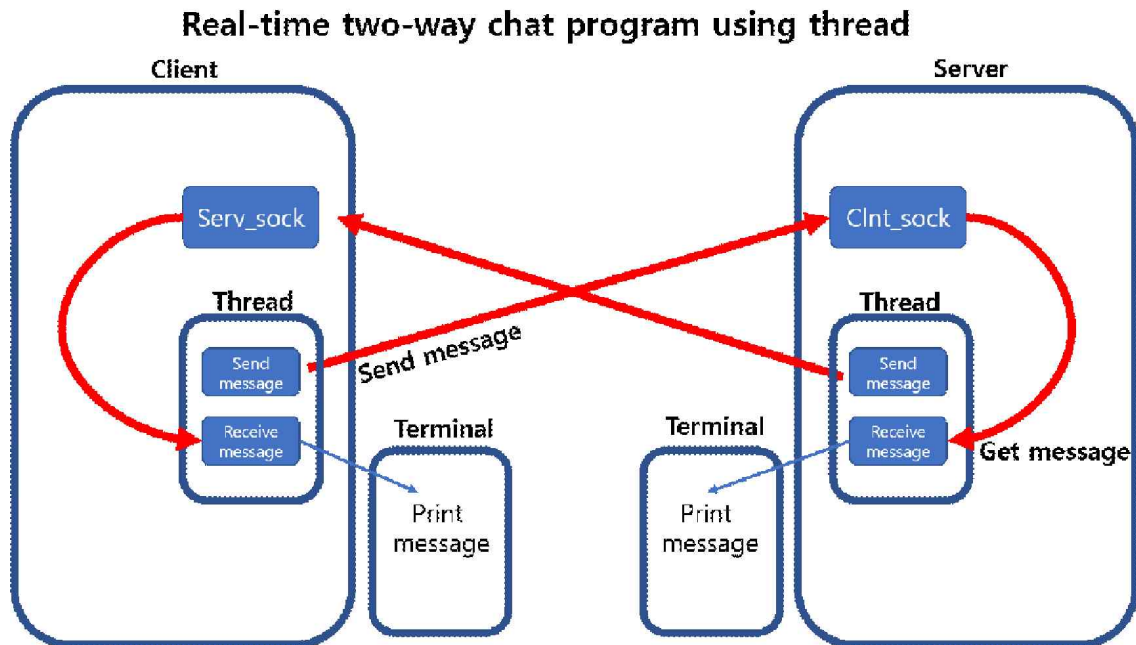
client -> :
<- server : Hi I'm Server

<- server : I think everything is fine
I think so. It works very well

client -> :
client -> :
<- server : yeah bye

see ya
```

1.3 설명



스레드를 활용해 구현하였다. 스레드를 2개 실행시키는데 각각, 메시지 전송 함수, 메시지 수신 함수이다. 모두 while문이 돌며 메시지 전송함수에는 입력값이 들어오면 메시지를 전송하고 수신 함수는 버퍼에 데이터가 쌓이면 바로 읽어 와서 출력한다. 작동방법은 서버와 클라이언트가 동일하고, 다음과 같다.

1. send_message 함수, receive_message 함수 각각 스레드 실행.
2. send_message 함수에는 while문이 돌며 키보드 입력 값을 바로 write.
3. receive_message 함수에는 while문이 돌며 커널 버퍼에 쌓인 값을 바로 read해서 출력.

2. 파일 전송

2.1 다중 스레드를 이용한 다중 파일 전송

2.1.1 소스코드

다중 스레드 파일 전송 서버 소스코드

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <sys/sendfile.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <pthread.h>
#include <time.h>

void error_handling(char* message);
void* send_file(void* file_name);

typedef struct {
    unsigned short port;
    int order;
    int clnt_sock;
    char* file_name;
}socket_data;

int main(int argc, char *argv[]){
    struct timespec ts;
    int clnt_sock;
    int serv_sock;
    char buf[30];

    struct stat obj;
    struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
    socklen_t clnt_addr_size;

    if(argc!=2){
        printf("Usage : %s <port>\n",argv[0]);
        exit(1);
    }

    serv_sock=socket(PF_INET, SOCK_STREAM,0);
    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]));

    if(bind(serv_sock,(struct sockaddr*)&serv_addr,sizeof(serv_addr))==-1){
        error_handling("bind() error");
    }

    printf("waiting...\n");
    if(listen(serv_sock,5)==-1){
        error_handling("listen() error");
    }

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

    // 0.get "Hello_Server"
    char message[20];
    recv(clnt_sock, message, 20, 0);
    printf("%s\n",message);

    // 1. send ls information

    //1-1. struct send info

    struct stat send_info;
    system("ls >ls.txt");
    stat("ls.txt",&send_info);
    int send_info_size=send_info.st_size;

    //1-2. send info

    send(clnt_sock,&send_info_size,sizeof(int),0);

    //1-3. send txt file

    int ls_FD=open("ls.txt",O_RDONLY);
    sendfile(clnt_sock,ls_FD,NULL,send_info_size);

    // 2. get send file information

    char* file_list[10];
```

```
int file_num=0;
while(1){
    char* fileName=(char*)malloc(sizeof(char)*30);
    recv(clnt_sock,fileName,sizeof(char)*30,0);
    if(fileName[0]=='x') break;
    file_list[file_num++]=fileName;
}
for(int i=0;i<file_num;i++){
    printf("%d : %s\n",i,file_list[i]);
}

// 3. open thread

clock_gettime(CLOCK_REALTIME,&ts);
int start=ts.tv_sec;
printf("open Thread : 0 sec \n");
pthread_t p_thread[10];
int t;
for(t=0;t<file_num;t++){
    socket_data* sd=(socket_data*)malloc(sizeof(socket_data));
    sd->port=serv_addr.sin_port;
    sd->clnt_sock=clnt_sock;
    sd->order=t;
    sd->file_name=file_list[t];
    pthread_create(&p_thread[t],NULL,send_file,(void*)sd);
}
for(t=0;t<file_num;t++){
    pthread_join(p_thread[t],NULL);
}

clock_gettime(CLOCK_REALTIME,&ts);
printf("inish : %d sec\n",ts.tv_sec-start);
close(clnt_sock);
close(serv_sock);
return 0;
}

void* send_file(void* socketdata){

    //1. connect server-client
    socket_data* sd=(socket_data*)socketdata;

    char buf[30];
    int ct_sock;
    int sv_sock;
    struct sockaddr_in sv_addr;
    struct sockaddr_in ct_addr;
    socklen_t ct_addr_size;

    unsigned short port=sd->port+sd->order*10-1;
    while(1){
        sv_sock=socket(PF_INET, SOCK_STREAM,0);
        if(sv_sock==-1)error_handling("socket() error");
        memset(&sv_addr,0,sizeof(sv_addr));
        sv_addr.sin_family=AF_INET;
        sv_addr.sin_addr.s_addr=htonl(INADDR_ANY);
        sv_addr.sin_port=++port;
        if(bind(sv_sock,(struct sockaddr*)&sv_addr,sizeof(sv_addr))==-1){
            continue;
        }
        break;
    }

    if(listen(sv_sock,5)==-1){
        error_handling("listen() error");
    }

    write(sd->clnt_sock,&port,sizeof(unsigned short));

    ct_addr_size =sizeof(ct_addr);
    ct_sock=accept(sv_sock,(struct sockaddr*)&ct_addr,&ct_addr_size);
    if(ct_sock==-1) error_handling("accept() error");

    //2. start send file

    struct stat file;
    char* filename=sd->file_name;

    stat(filename, &file);
    int fd = open(filename, O_RDONLY);
    int size = file.st_size;
    printf("fd:%d / size:%d /filename:%s\n",fd,size,filename);
    if (fd == -1) return NULL;

    send(ct_sock, &size, sizeof(int), 0);
    sleep(1);
    send(ct_sock, filename, sizeof(char)*30,0);

    if (size) {
        printf("sending %s file\n",filename);
        printf("fd : %d , size = %d \n",fd,size);
        sendfile(ct_sock, fd, NULL, size);
        printf("Finish\n");
    }
    close(ct_sock);
    close(sv_sock);
}

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

다중 스레드 파일전송 클라이언트 소스코드

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

#include <sys/stat.h>
#include <sys/sendfile.h>
#include <netinet/in.h>
#include <fcntl.h>

#include <pthread.h>

#define MAX_LINE 511

typedef struct {
    char* file_name;
    int serv_sock;
    unsigned short port;
    struct sockaddr_in* serv_addr;
}val;

void* recieve_file(void* data);

int main(int argc, char *argv[]){

    int serv_sock;
    struct sockaddr_in serv_addr;
    char buf[100];
    int str_len;

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

    serv_sock=socket(PF_INET,SOCK_STREAM,0);
    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]));

    connect(serv_sock,(struct sockaddr *)&serv_addr,sizeof(serv_addr));

    //0. send "Hello_Server"
    char message[20]="Hello_Server!";
    send(serv_sock,message,20,0);

    //1. recieve ls info
    int ls_size;
    //1-1 get info
    recv(serv_sock,&ls_size,sizeof(int),0);
    char* ls_file=malloc(ls_size);

    //1-1 get file
    recv(serv_sock,ls_file,ls_size,0);

    int ls_FD;
    char ls_name[20]="temp.txt";
    while(1){
        ls_FD=open(ls_name,O_CREAT|O_EXCL|O_WRONLY,0666);
        if(ls_FD!=-1){
            sprintf(ls_name+strlen(ls_name),"_1");
        }else break;
    }
    write(ls_FD,ls_file,ls_size);
    char command[20]="cat ";
    strcat(command,ls_name);
    printf("\n-----list of server file-----\n");
    system(command);
    printf("-----\n");

    printf("\n type a file name and press 'Enter' \n Type 'x' when you want
to start transferring files.\n");
    int file_num=0;
    char* file_list[10];
    while(1){
        char* menu=(char*)malloc(sizeof(char)*30);
        printf("input : ");
        scanf("%s",menu);
        if(menu[0]!='x') {
            send(serv_sock,menu,sizeof(char)*30,0);

            break;
        }else{
            send(serv_sock,menu,sizeof(char)*30,0);
            file_list[file_num++]=menu;
        }
    }
}
```

```
pthread_t p_thread[10];
for(int t=0;t<file_num;t++){
    val* data=(val*)malloc(sizeof(data));
    data->file_name=file_list[t];
    data->serv_sock=serv_sock;
    data->serv_addr=&serv_addr;
    recv(serv_sock,&(data->port),sizeof(unsigned short),0);
    pthread_create(&p_thread[t],NULL,recieve_file,(void*)data);
}
for(int t=0;t<file_num;t++){
    pthread_join(p_thread[t],NULL);
}
close(serv_sock);
return 0;
}

void* recieve_file(void* data){
    val* file_data=(val*)data;
    int sv_sock;
    struct sockaddr_in* serv_addr=file_data->serv_addr;
    struct sockaddr_in sv_addr;
    char buf[100];
    int str_len;

    //0. recieve port
    unsigned short port=file_data->port;
    sv_sock=socket(PF_INET,SOCK_STREAM,0);
    memset(&sv_addr,0,sizeof(sv_addr));
    sv_addr.sin_family=AF_INET;
    sv_addr.sin_addr.s_addr=serv_addr->sin_addr.s_addr;
    sv_addr.sin_port=port;
    connect(sv_sock,(struct sockaddr *)&sv_addr,sizeof(sv_addr));

    //1. recieve file data
    char filename[30];
    int size;
    recv(sv_sock,&size,sizeof(int),0);
    recv(sv_sock,filename,sizeof(char)*30,0);
    printf("filename : %s / file size : %d byte\n",filename,size);

    //2. set size
    char* f=malloc(size);
    char* ptr=f;

    //3. get movie!!
    int remain=size;
    int download=0;
    int percent=0;
    while(0<remain){
        int recvsize=0;
        if(remain<1460){
            recvsize=recv(sv_sock,ptr,remain,0);
        }else{
            recvsize=recv(sv_sock,ptr,1460,0);
        }
        if(percent<download/(size/100)){
            printf("%s : download : %d\n",filename,++percent);
        }

        ptr+=recvsize;
        download+=recvsize;
        remain-=recvsize;
    }

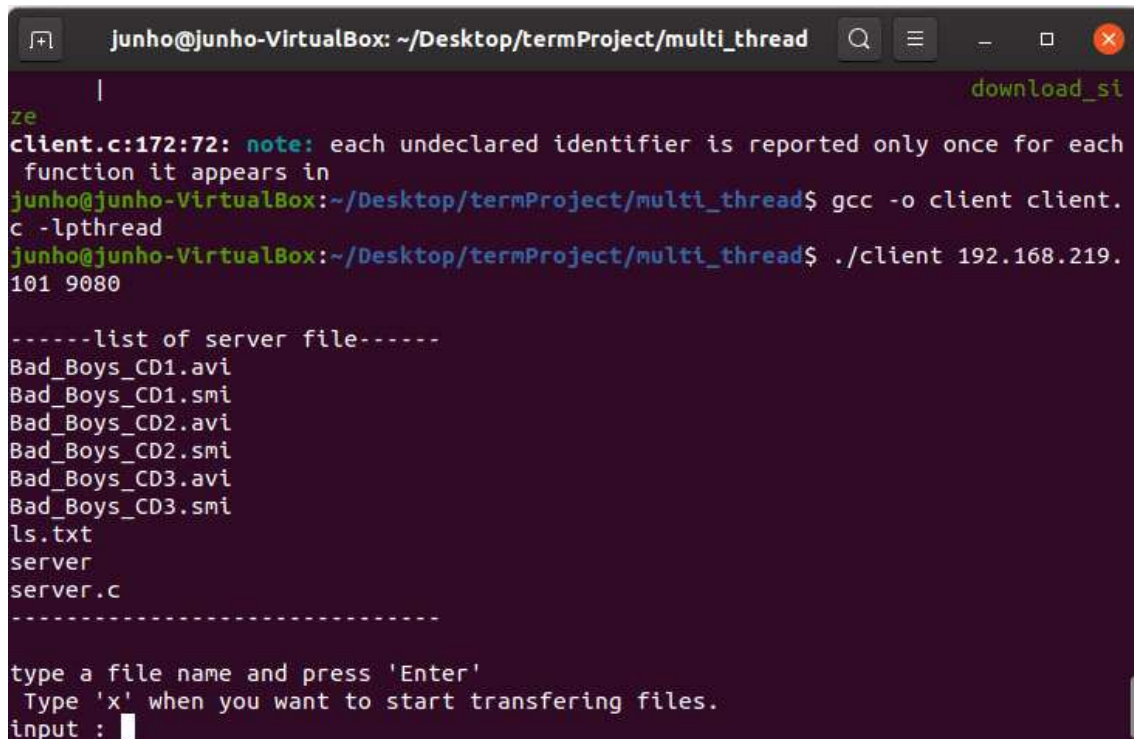
    //4. make file dir
    int fd;
    while(1){
        fd=open(filename,O_CREAT|O_EXCL|O_WRONLY,0666);
        if(fd!=-1){
            sprintf(filename+strlen(filename),"_1");
        }else break;
    }

    //5. save file
    printf("fd : %d / size = %d\n",fd,size);
    write(fd,f,size);
    close(fd);

    struct stat downloadfile;
    stat(filename,&downloadfile);
    int download_size=downloadfile.st_size;
    if(download_size==size){
        printf("download_size : %dbyte \nfile size : %dbyte
\n--perfect--\n",download_size,size);
    }else{
        printf("download_size : %dbyte \nfile size : %dbyte
\n--%dbyte is not downloaded--\n",download_size,size,size-download_size);
    }
    close(sv_sock);
}
```


2.1.2 실행 화면

사진1. 클라이언트 - 받을 파일 입력 대기

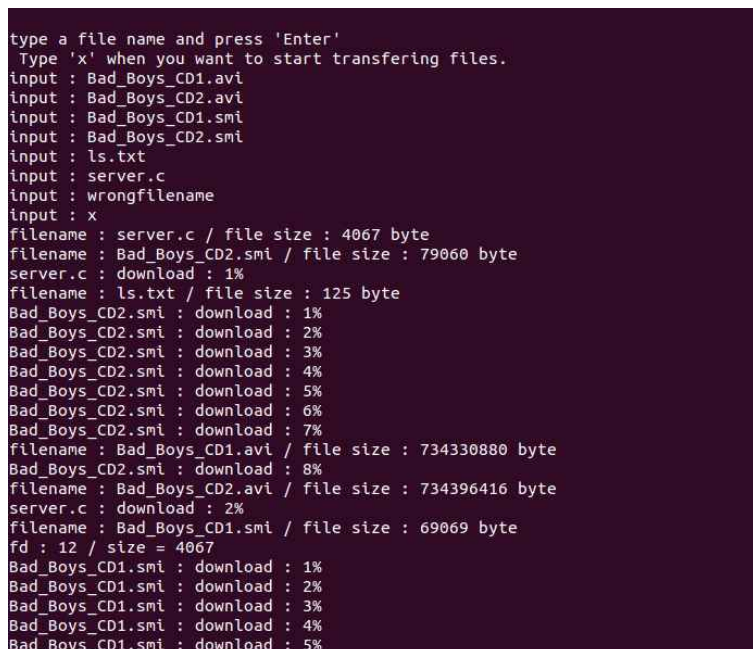


```
junho@junho-VirtualBox: ~/Desktop/termProject/multi_thread
|
download_si
ze
client.c:172:72: note: each undeclared identifier is reported only once for each
function it appears in
junho@junho-VirtualBox:~/Desktop/termProject/multi_thread$ gcc -o client client.
c -lpthread
junho@junho-VirtualBox:~/Desktop/termProject/multi_thread$ ./client 192.168.219.
101 9080

-----list of server file-----
Bad_Boys_CD1.avi
Bad_Boys_CD1.smi
Bad_Boys_CD2.avi
Bad_Boys_CD2.smi
Bad_Boys_CD3.avi
Bad_Boys_CD3.smi
ls.txt
server
server.c
-----

type a file name and press 'Enter'
Type 'x' when you want to start transferring files.
input : 
```

사진2. 클라이언트 - 받을 파일 이름 입력 후, 파일 전송 시작(Bad_Boys_CD 1,2편 동영상 파일과 자막파일, 그리고 ls.txt, server.c 파일 그리고 일부러 잘못된 이름)



```
type a file name and press 'Enter'
Type 'x' when you want to start transferring files.
input : Bad_Boys_CD1.avi
input : Bad_Boys_CD2.avi
input : Bad_Boys_CD1.smi
input : Bad_Boys_CD2.smi
input : ls.txt
input : server.c
input : wrongfilename
input : x
filename : server.c / file size : 4067 byte
filename : Bad_Boys_CD2.smi / file size : 79060 byte
server.c : download : 1%
filename : ls.txt / file size : 125 byte
Bad_Boys_CD2.smi : download : 1%
Bad_Boys_CD2.smi : download : 2%
Bad_Boys_CD2.smi : download : 3%
Bad_Boys_CD2.smi : download : 4%
Bad_Boys_CD2.smi : download : 5%
Bad_Boys_CD2.smi : download : 6%
Bad_Boys_CD2.smi : download : 7%
filename : Bad_Boys_CD1.avi / file size : 734330880 byte
Bad_Boys_CD2.smi : download : 8%
filename : Bad_Boys_CD2.avi / file size : 734396416 byte
server.c : download : 2%
filename : Bad_Boys_CD1.smi / file size : 69069 byte
fd : 12 / size = 4067
Bad_Boys_CD1.smi : download : 1%
Bad_Boys_CD1.smi : download : 2%
Bad_Boys_CD1.smi : download : 3%
Bad_Boys_CD1.smi : download : 4%
Bad_Boys_CD1.smi : download : 5%
```

사진3. 클라이언트 - 동영상 파일 다운로드가 완료가 되고 파일을 쓰고 크기를 비교하는 모습

```
Bad_Boys_CD2.avi : download : 97%
Bad_Boys_CD1.avi : download : 95%
Bad_Boys_CD2.avi : download : 98%
Bad_Boys_CD1.avi : download : 96%
Bad_Boys_CD1.avi : download : 97%
Bad_Boys_CD2.avi : download : 99%
Bad_Boys_CD1.avi : download : 98%
fd : 5 / size = 734396416
Bad_Boys_CD1.avi : download : 99%
fd : 7 / size = 734330880
download_size : 734396416byte
file size : 734396416byte
--perfect--
download_size : 734330880byte
file size : 734330880byte
--perfect--
junho@junho-VirtualBox:~/Desktop/termPro
```

사진4. 클라이언트 - 서버에게 요청한 영화 2편 동영상파일과, 자막파일 , ls.txt, server.c
파일이 디렉토리에 저장된 모습

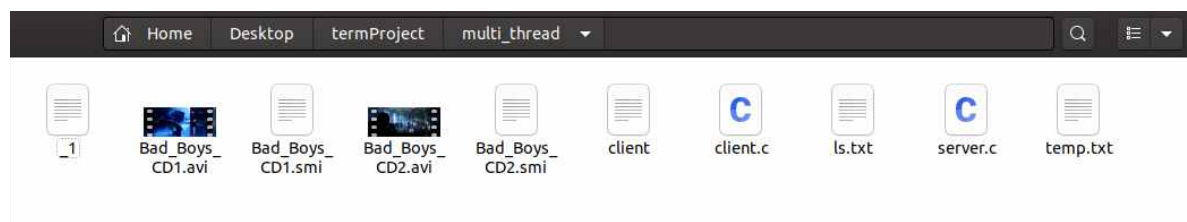
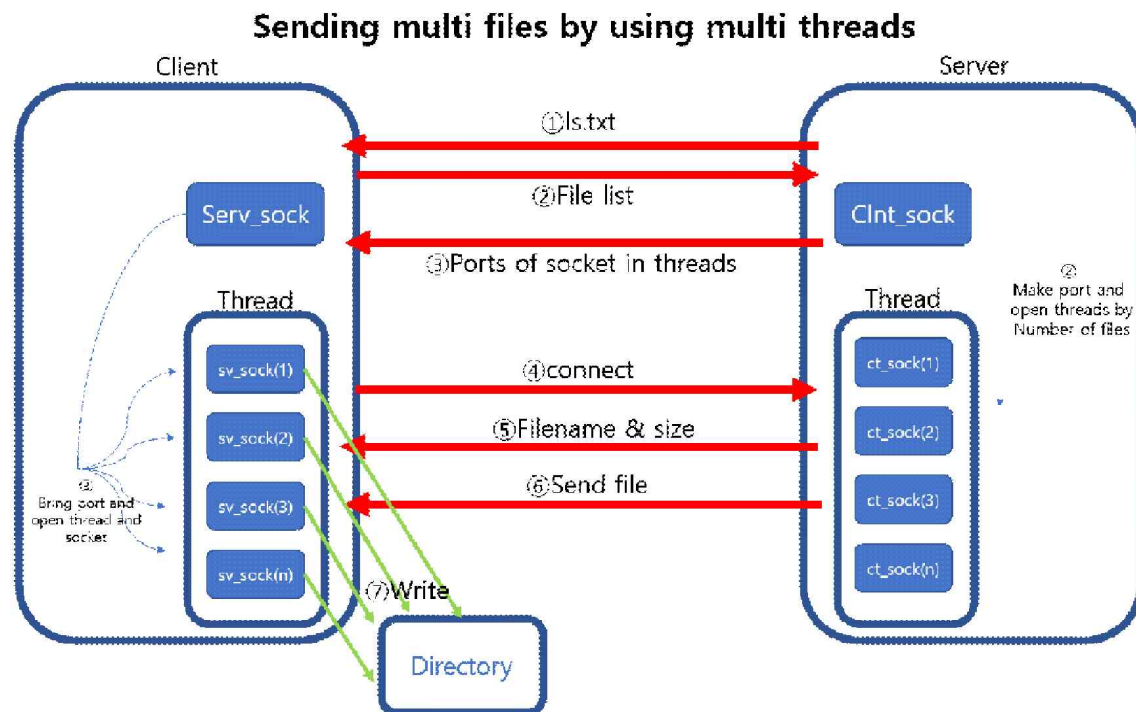


사진 5. 서버 - 파일들을 모두 전송을 한 모습

```
junho@junho-VirtualBox:~/Desktop/termProject/multi_thread$ ./server 9080
waiting...
Hello_Server!
0 : Bad_Boys_CD1.avi
1 : Bad_Boys_CD2.avi
2 : Bad_Boys_CD1.smi
3 : Bad_Boys_CD2.smi
4 : ls.txt
5 : server.c
6 : wrongfilename
fd:-1 / size:0 /filename:wrongfilename
fd:21 / size:4067 /filename:server.c
fd:22 / size:125 /filename:ls.txt
fd:23 / size:79060 /filename:Bad_Boys_CD2.smi
fd:24 / size:69069 /filename:Bad_Boys_CD1.smi
fd:25 / size:734396416 /filename:Bad_Boys_CD2.avi
fd:26 / size:734330880 /filename:Bad_Boys_CD1.avi
sending server.c file
fd : 21 , size = 4067
Finish
sending ls.txt file
fd : 22 , size = 125
sending Bad_Boys_CD2.smi file
fd : 23 , size = 79060
Finish
sending Bad_Boys_CD1.smi file
fd : 24 , size = 69069
Finish
Finish
sending Bad_Boys_CD2.avi file
fd : 25 , size = 734396416
sending Bad_Boys_CD1.avi file
fd : 26 , size = 734330880
Finish
Finish
```

2.1.3 설명



n개의 파일을 스레드를 n개 열고 거기에 소켓을 한 개씩 열어 동시에 다중 전송하는 프로그램이다.

1. 서버 : `ls.txt`를 전송
 - 1-1. `ls` 명령어를 `system("ls >ls.txt")` 을 이용해 `ls.txt`로 저장함.
 - 1-2. `ls.txt`파일을 불러와서 전송
2. 클라이언트 : 전송받기 원하는 파일의 목록을 전송
 - 2-1. 수신한 `ls.txt` 파일을 `temp.txt`로 저장
 - 2-2. `system("cat temp.txt")`을 이용해 출력
 - 2-3. 전송받기 원하는 파일명을 입력 받고 파일명 전송, x를 누를 때 까지 반복
3. 서버 : 파일 수 만큼 스레드를 열고, 해당 스레드 소켓의 포트들을 전송
 - 3-1. 파일 수 만큼 스레드를 열고, 스레드의 순서 매개변수로 전달
 - 3-2. 스레드 소켓의 포트 번호는 기존포트+순서*10, 오류가 날 때마다 1씩 증가 (10씩 차이를 둔 이유는 bind오류가 발생하여 포트가 겹칠 가능성 때문에)
 - 3-3. 소켓을 열었으면, 포트번호 전송
4. 클라이언트 : 받은 포트 번호로 스레드 열기 및 연결
 - 4-1. 포트 번호 수신
 - 4-2. 스레드를 열고 해당 포트번호로 소켓 생성 후 연결
 - 4-3. 파일 수 만큼 반복

5. 서버 스레드 : 파일의 이름과 사이즈 전송
 - 5-1. 클라이언트 : 파일 이름과 사이즈 수신
 - 5-2. 사이즈만큼의 메모리 동적 할당
6. 서버 스레드 : 파일 전송
 - 6-1. 서버 스레드 : 계속 전송
 - 6-2. 클라이언트 스레드 : 1460바이트만큼 계속 수신
 - 6-3. 클라이언트 스레드 : 수신데이터 양과 사이즈 비교하며 다운로드 퍼센트 출력
7. 클라이언트 스레드 : 파일 저장 및 확인
 - 7-1. 파일을 저장할 공간을 만들고 write함수를 사용해 메모리에 저장된 값 저장
 - 7-2. 저장을 한 파일을 불러와 사이즈를 불러오고 기존의 사이즈와 비교
8. 모든 스레드가 파일 전송을 마쳤다면 종료

2.2 파일 압축을 이용한 다중 파일 전송

2.2.1 소스코드

파일 압축 전송 서버 소스코드

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <sys/sendfile.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <pthread.h>
#include <time.h>
void error_handling(char* message);
void* send_file(void* file_name);

typedef struct{
    unsigned short port;
    int order;
    int clnt_sock;
    char* file_name;
}socket_data;

int main(int argc, char *argv[]){
    struct timespec ts;
    int clnt_sock;
    int serv_sock;
    char buf[30];

    struct stat obj;
    struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
    socklen_t clnt_addr_size;

    if(argc!=2){
        printf("Usage : %s <port>\n",argv[0]);
        exit(1);
    }

    serv_sock=socket(PF_INET, SOCK_STREAM,0);
    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]));

    if(bind(serv_sock,(struct sockaddr*)&serv_addr,sizeof(serv_addr))==-1){
        error_handling("bind() error");
    }

    printf("waiting...\n");
    if(listen(serv_sock,5)==-1){
        error_handling("listen() error");
    }

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

    // 0.get "Hello_Server"
    char message[20];
    recv(clnt_sock, message, 20, 0);
    printf("%s\n",message);

    // 1. send ls information

    //1-1. struct send info
```

```
struct stat send_info;
system("ls >ls.txt");
stat("ls.txt",&send_info);
int send_info_size=send_info.st_size;

//1-2. send info

send(clnt_sock,&send_info_size,sizeof(int),0);

//1-3. send txt file

int ls_FD=open("ls.txt",O_RDONLY);
sendfile(clnt_sock,ls_FD,NULL,send_info_size);

// 2. get send file information

char* file_list[10];
int file_num=0;
while(1){
    char* fileName=(char*)malloc(sizeof(char)*30);
    recv(clnt_sock,fileName,sizeof(char)*30,0);
    if(fileName[0]!='x') break;
    file_list[file_num++]=fileName;
}
for(int i=0;i<file_num;i++){
    printf("%d : %s\n",i,file_list[i]);
}

struct stat file;
char filename[30]="zipData.tar.gz";
char command[120]="tar cvzf ";
strcat(command,filename);
for(int i=0;i<file_num;i++){
    strcat(command," ");
    strcat(command,file_list[i]);
}
printf("%s\n",command);
clock_gettime(CLOCK_REALTIME,&ts);
int zipStart=ts.tv_sec;
printf("zip start : 0 \n");
system(command);
clock_gettime(CLOCK_REALTIME,&ts);
printf("zip end : %ld \n",ts.tv_sec-zipStart);
stat(filename, &file);
int fd = open(filename, O_RDONLY);
int size = file.st_size;
printf("fd:%d / size:%d /filename:%s\n",fd,size,filename);
if (fd == -1) return 0;

send(clnt_sock, &size, sizeof(int), 0);
sleep(1);
send(clnt_sock, filename, sizeof(char)*30,0);

if (size) {
    printf("sending %s file\n",filename);
    printf("fd : %d , size = %d \n",fd,size);
    clock_gettime(CLOCK_REALTIME,&ts);
    printf("send file : %ld \n",ts.tv_sec-zipStart);
    sendfile(clnt_sock, fd, NULL, size);
    clock_gettime(CLOCK_REALTIME,&ts);
    printf("Finish : %ld \n",ts.tv_sec-zipStart);
}

close(clnt_sock);
close(serv_sock);
return 0;
}

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

파일 압축 전송 클라이언트 소스코드

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

#include <sys/stat.h>
#include <sys/sendfile.h>
#include <netinet/in.h>
#include <fcntl.h>

#include <pthread.h>

#define MAX_LINE 511

typedef struct {
    char* file_name;
    int serv_sock;
    unsigned short port;
    struct sockaddr_in* serv_addr;
}val;

void* recieve_file(void* data);

int main(int argc, char *argv[]){

    int serv_sock;
    struct sockaddr_in serv_addr;
    char buff[100];
    int str_len;

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

    serv_sock=socket(PF_INET,SOCK_STREAM,0);
    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]));

    connect(serv_sock,(struct sockaddr *)&serv_addr,sizeof(serv_addr));

//0. send "Hello_Server"
    char message[20]="Hello_Server!";
    send(serv_sock,message,20,0);

//1. recieve ls info
    int ls_size;
    //1-1 get info
    recv(serv_sock,&ls_size,sizeof(int),0);
    char* ls_file=malloc(ls_size);

    //1-1 get file
    recv(serv_sock,ls_file,ls_size,0);

    int ls_FD;
    char ls_name[20]="temp.txt";
    while(1){
        ls_FD=open(ls_name,O_CREAT|O_EXCL|O_WRONLY,0666);
        if(ls_FD==1){
            sprintf(ls_name+strlen(ls_name),"_1");
        }else break;
    }
    write(ls_FD,ls_file,ls_size);
    char command[20]="cat ";
    strcat(command,ls_name);
    printf("\n-----list of server file-----\n");
    system(command);
    printf("-----\n");

    printf("\n\ntype a file name and press 'Enter' \n Type 'x' when you want
to start transferring files.\n");
    int file_num=0;
```

```
char* file_list[10];
while(1){

    char* menu=(char*)malloc(sizeof(char)*30);
    printf("input : ");
    scanf("%s",menu);
    if(menu[0]!='x') {
        send(serv_sock,menu,sizeof(char)*30,0);

        break;
    }else{
        send(serv_sock,menu,sizeof(char)*30,0);
        file_list[file_num++]=menu;
    }
}

char filename[30];
int size;
recv(serv_sock,&size,sizeof(int),0);
recv(serv_sock,filename,sizeof(char)*30,0);
printf("filename : %s / file size : %d byte\n",filename,size);

//2. set size
char* f=malloc(size);
char* ptr=f;

//3. get movie!!
int remain=size;
int download=0;
int percent=0;
while(0<remain){
    int recvsz=0;
    if(remain<1460){
        recvsz=recv(serv_sock,ptr,remain,0);
    }else{
        recvsz=recv(serv_sock,ptr,1460,0);
    }
    if(percent<download/(size/100)){
        printf("%s : download :
%d%%\n",filename,++percent);
    };

    ptr+=recvsz;
    download+=recvsz;
    remain-=recvsz;
}

//4. make file dir
int fd;
while(1){
    fd=open(filename,O_CREAT|O_EXCL|O_WRONLY,0666);
    if(fd==1){
        sprintf(filename+strlen(filename),"_1");
    }else break;
}

//5. save file
printf("fd : %d / size = %d\n",fd,size);
write(fd,f,size);
close(fd);

struct stat downloadfile;
stat(filename,&downloadfile);
int download_size=downloadfile.st_size;
if(download_size==size){
    printf("download_size : %dbyte \nfile size : %dbyte
\n--perfect--\n",download_size,size);
}else{
    printf("download_size : %dbyte \nfile size : %dbyte
\n--%dbyte is not downloaded--\n",download_size,size,size-download_size);
}

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


2.2.2 실행 화면

사진1. 클라이언트 : 전송받을 파일 이름을 입력하고, 전송을 받기 시작한 모습

```
junho@junho-VirtualBox:~/Desktop/termProject/zip$ ./zip_clint 192.168.219.101 9090
-----list of server file-----
Bad_Boys_CD1.avi
Bad_Boys_CD1.smi
Bad_Boys_CD2.avi
Bad_Boys_CD2.smi
Bad_Boys_CD3.avi
Bad_Boys_CD3.smi
ls.txt
zipData.tar.gz
zip_server
zip_server.c
-----
type a file name and press 'Enter'
Type 'x' when you want to start transferring files.
input : Bad_Boys_CD1.avi
input : Bad_Boys_CD2.avi
input : Bad_Boys_CD1.smi
input : Bad_Boys_CD2.smi
input : x
filename : zipData.tar.gz / file size : 1417028338 byte
zipData.tar.gz : download : 1%
zipData.tar.gz : download : 2%
zipData.tar.gz : download : 3%
zipData.tar.gz : download : 4%
zipData.tar.gz : download : 5%
```

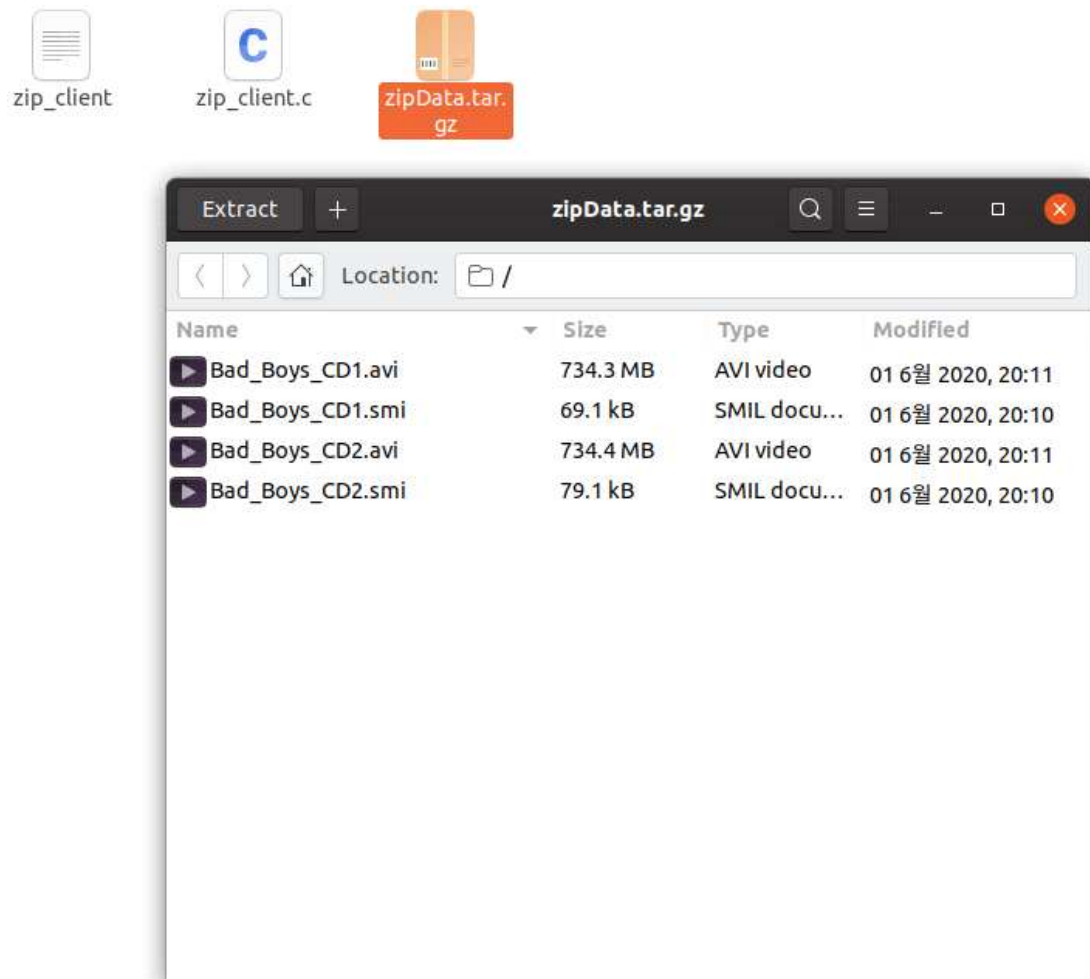
사진 2. 클라이언트 : 다운로드가 완료된 모습

```
zipData.tar.gz : download : 93%
zipData.tar.gz : download : 94%
zipData.tar.gz : download : 95%
zipData.tar.gz : download : 96%
zipData.tar.gz : download : 97%
zipData.tar.gz : download : 98%
zipData.tar.gz : download : 99%
fd : 5 / size = 1417028338
download_size : 1417028338byte
file size : 1417028338byte
--perfect--
junho@junho-VirtualBox:~/Desktop/termProject/zip$
```

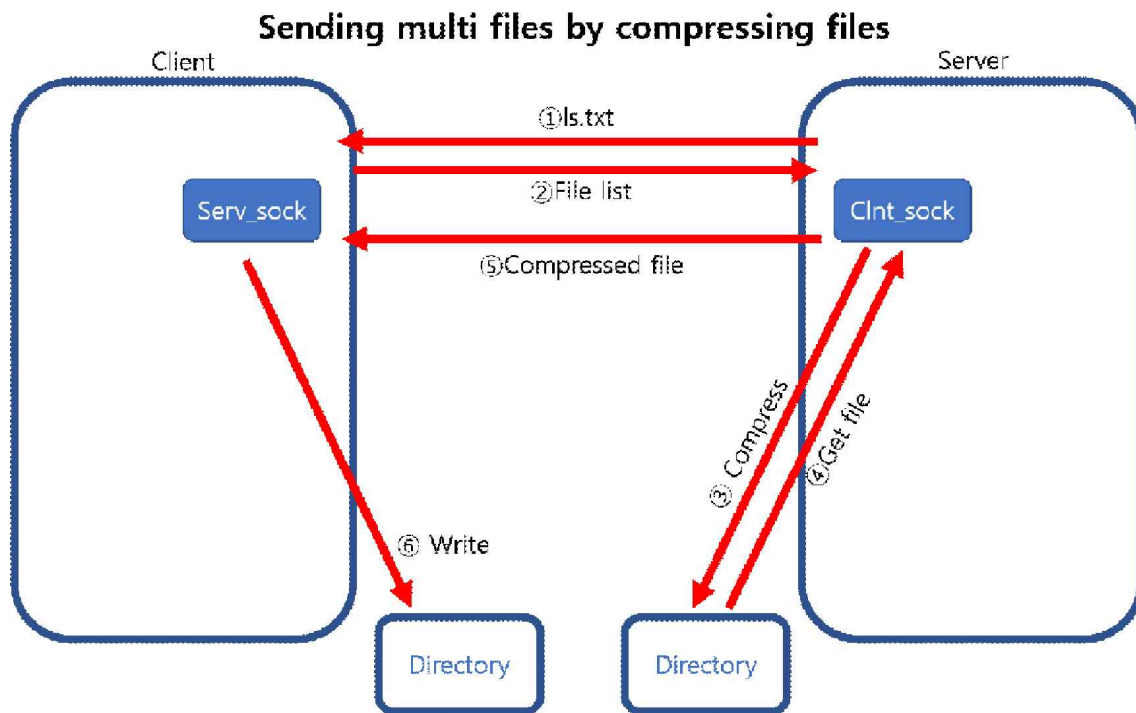
사진3. 서버 : 전송이 시작하고 완료된 모습

```
junho@junho-VirtualBox:~/Desktop/termProject/zip$ ./zip_server 9090
waiting...
Hello_Server!
0 : Bad_Boys_CD1.avi
1 : Bad_Boys_CD2.avi
2 : Bad_Boys_CD1.smi
3 : Bad_Boys_CD2.smi
tar cvzf zipData.tar.gz Bad_Boys_CD1.avi Bad_Boys_CD2.avi Bad_Boys_CD1.smi Bad_Boys_CD2.smi
zip start : 0
Bad_Boys_CD1.avi
Bad_Boys_CD2.avi
Bad_Boys_CD1.smi
Bad_Boys_CD2.smi
zip end : 50
fd:6 / size:1417028338 /filename:zipData.tar.gz
sending zipData.tar.gz file
fd : 6 , size = 1417028338
send file : 51
Finish : 157
```


사진 4. 클라이언트 : 전송 받은 압축된 파일의 목록



2.2.3 설명



ls.txt를 보내고, 전송받을 파일 이름을 입력받는 부분까지 위에 설명했던 것과 동일하다. 과정은 다음과 같다.

1. 클라이언트 : ls.txt 파일을 수신
2. 서버 : 전송할 파일의 목록을 수신
3. 서버 : 파일 압축
 - 3-1. strcat 함수를 이용하여 command문자열 생성
(ex. command[50] = "tar cvzf data.tar.gz file1 file2 file3")
 - 3-2. system(command)을 이용하여 파일 압축
4. 서버 : 파일 불러오기
 - 4-1. open함수를 이용하여 파일의 fd를 리턴 받는다.
5. 서버 : 파일 전송
 - 5-1. 먼저 파일의 사이즈를 보냄
 - 5-1. sendfile 함수를 이용해 계속 클라이언트로 전송.
6. 클라이언트 : 수신 후 저장 및 비교
 - 6-1. 받을 사이즈만큼 메모리 동적할당
 - 6-1. 1460byte로 나누어 받고 다운로드 진척도 출력
 - 6-2. 수신이 완료되면 write를 이용, 저장 및 불러와서 크기 비교

2.3 순차적 다중 파일 전송

2.3.1 소스코드

순차적 파일 전송 서버 소스코드

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <sys/sendfile.h>
#include <netinet/in.h>
#include <fcntl.h>
#include <pthread.h>
#include <time.h>
void error_handling(char* message);
void* send_file(void* file_name);

typedef struct{
    unsigned short port;
    int order;
    int clnt_sock;
    char* file_name;
}socket_data;

int main(int argc, char *argv[]){
    struct timespec ts;
    int clnt_sock;
    int serv_sock;
    char buf[30];

    struct stat obj;
    struct sockaddr_in serv_addr;
    struct sockaddr_in clnt_addr;
    socklen_t clnt_addr_size;

    if(argc!=2){
        printf("Usage : %s <port>\n",argv[0]);
        exit(1);
    }

    serv_sock=socket(PF_INET, SOCK_STREAM,0);
    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]));

    if(bind(serv_sock,(struct sockaddr*)&serv_addr,sizeof(serv_addr))!=-1){
        error_handling("bind() error");
    }

    printf("waiting...\n");
    if(listen(serv_sock,5)==-1){
        error_handling("listen() error");
    }

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

    // 0.get "Hello_Server"
    char message[20];
    recv(clnt_sock, message, 20, 0);
    printf("%s\n",message);
```

```
// 1. send ls information

//1-1. struct send info

struct stat send_info;
system("ls >ls.txt");
stat("ls.txt",&send_info);
int send_info_size=send_info.st_size;

//1-2. send info

send(clnt_sock,&send_info_size,sizeof(int),0);

//1-3. send txt file

int ls_FD=open("ls.txt",O_RDONLY);
sendfile(clnt_sock,ls_FD,NULL,send_info_size);

// 2. get send file information

char* file_list[10];
int file_num=0;
while(1){
    char* fileName=(char*)malloc(sizeof(char)*30);
    recv(clnt_sock,fileName,sizeof(char)*30,0);
    if(fileName[0]!='x') break;
    file_list[file_num++]=fileName;
}
for(int i=0;i<file_num;i++){
    printf("%d : %s\n",i,file_list[i]);
}

clock_gettime(CLOCK_REALTIME,&ts);
int start=ts.tv_sec;
printf("start : 0 sec\n");
for(int i=0;i<file_num;i++){
    struct stat file;
    char* filename=file_list[i];

    stat(filename, &file);
    int fd = open(filename, O_RDONLY);
    int size = file.st_size;
    printf("fd:%d / size:%d

filename:%s\n",fd,size,filename);
    if (fd == -1) return 0;

    send(clnt_sock, &size, sizeof(int), 0);
    sleep(1);
    send(clnt_sock, filename, sizeof(char)*30,0);

    if (size) {
        printf("sending %s

\n",filename);
        printf("fd : %d , size = %d

\n",fd,size);

        clock_gettime(CLOCK_REALTIME,&ts);
        sendfile(clnt_sock, fd, NULL,
size);

        clock_gettime(CLOCK_REALTIME,&ts);
        printf("Finish %d file: %ld
sec\n",i,ts.tv_sec-start);

        clock_gettime(CLOCK_REALTIME,&ts);
    }
    clock_gettime(CLOCK_REALTIME,&ts);
    printf("All Finish : %ld sec\n",ts.tv_sec-start);
    close(clnt_sock);
    close(serv_sock);
    return 0;
}

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

순차적 파일 전송 클라이언트 소스코드

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

#include <sys/stat.h>
#include <sys/sendfile.h>
#include <netinet/in.h>
#include <fcntl.h>

#include <pthread.h>

#define MAX_LINE 511

typedef struct {
    char* file_name;
    int serv_sock;
    unsigned short port;
    struct sockaddr_in* serv_addr;
}val;

void* recieve_file(void* data);

int main(int argc, char *argv[]){

    int serv_sock;
    struct sockaddr_in serv_addr;
    char buf[100];
    int str_len;

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

    serv_sock=socket(PF_INET,SOCK_STREAM,0);
    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]));

    connect(serv_sock,(struct sockaddr
*&serv_addr,sizeof(serv_addr));
//0. send "Hello_Server"
    char message[20]="Hello_Server!";
    send(serv_sock,message,20,0);

//1. recieve ls info
    int ls_size;
    //1-1 get info
    recv(serv_sock,&ls_size,sizeof(int),0);
    char* ls_file=malloc(ls_size);

    //1-1 get file
    recv(serv_sock,ls_file,ls_size,0);

    int ls_FD;
    char ls_name[20]="temp.txt";
    while(1){
        ls_FD=open(ls_name,O_CREAT|O_EXCL|O_WRONLY,0666);
        if(ls_FD!=-1){
            sprintf(ls_name+strlen(ls_name),"_1");
        }
        else break;
        write(ls_FD,ls_file,ls_size);
        char command[20]="cat ";
        strcat(command,ls_name);
        printf("\n-----list of server file-----\n");
        system(command);
        printf("-----\n");

        printf("\ntype a file name and press 'Enter' \n Type 'x' when
you want to start transferring files.\n");
        int file_num=0;
        char* file_list[10];
```

```
while(1){
    char* menu=(char*)malloc(sizeof(char)*30);
    printf("input : ");
    scanf("%s",menu);
    if(menu[0]!='x') {

        send(serv_sock,menu,sizeof(char)*30,0);

        break;
    }else{

        send(serv_sock,menu,sizeof(char)*30,0);

        file_list[file_num++]=menu;
    }
}

for(int i=0;i<file_num;i++){
    char filename[30];
    int size;
    recv(serv_sock,&size,sizeof(int),0);
    recv(serv_sock,filename,sizeof(char)*30,0);
    printf("filename : %s / file size : %d
byte\n",filename,size);
    //2. set size
    char* f=malloc(size);
    char* ptr=f;

    //3. get movie!!
    int remain=size;
    int download=0;
    int percent=0;
    while(0<remain){
        int rcvsize=0;
        if(remain<1460){
            rcvsize=recv(serv_sock,ptr,remain,0);

        }else{
            rcvsize=recv(serv_sock,ptr,1460,0);
        }
        if(percent<download/(size/100)){
            printf("%s :
download : %d%%\n",filename,++percent);
        };

        ptr+=rcvsize;
        download+=rcvsize;
        remain-=rcvsize;
    }

    //4. make file dir
    int fd;
    while(1){
        fd=open(filename,O_CREAT|O_EXCL|O_WRONLY,0666);
        if(fd!=-1){
            sprintf(filename+strlen(filename),"_1");
        }
        else break;

    }

    //5. save file
    printf("fd : %d / size = %d\n",fd,size);
    write(fd,f,size);
    close(fd);

    struct stat downloadfile;
    stat(filename,&downloadfile);
    int download_size=downloadfile.st_size;
    if(download_size==size){
        printf("download_size : %dbyte
\nfile size : %dbyte \n--perfect--\n",download_size,size);
    }else{
        printf("download_size : %dbyte
\nfile size : %dbyte \n--%dbyte is not
downloaded--\n",download_size,size,size-download_size);
    }
}

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

2.3.2 실행 화면

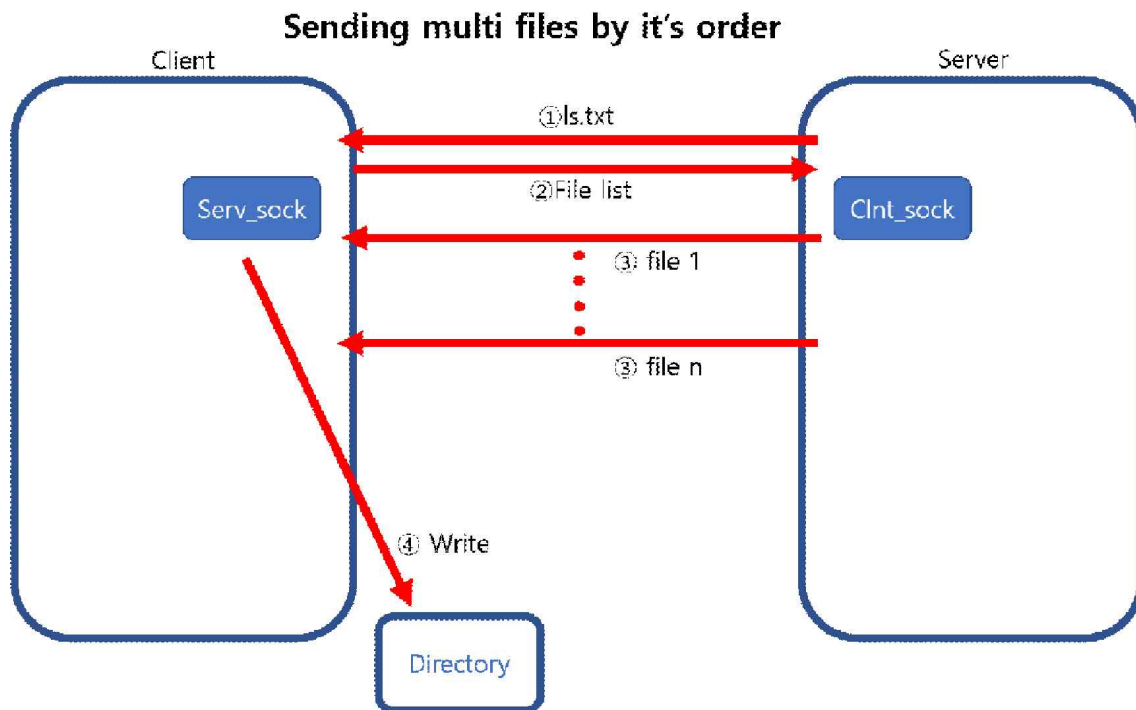
서버

```
junho@junho-VirtualBox:~/Desktop/termProject/linear$ gcc -o linear_server linear_server.c
junho@junho-VirtualBox:~/Desktop/termProject/linear$ ./linear_server 9080
waiting...
Hello_Server!
0 : Bad_Boys_CD1.avi
1 : Bad_Boys_CD1.smi
2 : Bad_Boys_CD2.avi
3 : Bad_Boys_CD2.smi
start : 0 sec
fd:6 / size:734330880 /filename:Bad_Boys_CD1.avi
sending Bad_Boys_CD1.avi file
fd : 6 , size = 734330880
Finish 0 file: 65 sec
fd:7 / size:69069 /filename:Bad_Boys_CD1.smi
sending Bad_Boys_CD1.smi file
fd : 7 , size = 69069
Finish 1 file: 66 sec
fd:8 / size:734396416 /filename:Bad_Boys_CD2.avi
sending Bad_Boys_CD2.avi file
fd : 8 , size = 734396416
Finish 2 file: 138 sec
fd:9 / size:79060 /filename:Bad_Boys_CD2.smi
sending Bad_Boys_CD2.smi file
fd : 9 , size = 79060
Finish 3 file: 139 sec
All Finish : 139 sec
```

클라이언트

```
Junho@junho-VirtualBox: ~/Desktop/termProject/linear
Input : Bad_BoyC
junho@junho-VirtualBox:~/Desktop/termProject/linear$ ./linear_client 192.168.219.101 9080
-----list of server file-----
Bad_Boys_CD1.avi
Bad_Boys_CD1.smi
Bad_Boys_CD2.avi
Bad_Boys_CD2.smi
linear_server
linear_server.c
ls.txt
zip_client.c
-----
type a file name and press 'Enter'
Type 'x' when you want to start transferring files.
Input : Bad_Boys_CD1.avi
Input : Bad_Boys_CD1.smi
Input : Bad_Boys_CD2.avi
Input : Bad_Boys_CD2.smi
Input : x
filename : Bad_Boys_CD1.avi / file size : 734330880 byte
Bad_Boys_CD1.avi : download : 1%
Bad_Boys_CD1.avi : download : 2%
Bad_Boys_CD1.avi : download : 3%
Bad_Boys_CD1.avi : download : 4%
Bad_Boys_CD1.avi : download : 5%
Bad_Boys_CD1.avi : download : 6%
Bad_Boys_CD1.avi : download : 7%
Bad_Boys_CD1.avi : download : 8%
Bad_Boys_CD1.avi : download : 9%
Bad_Boys_CD1.avi : download : 10%
Bad_Boys_CD1.avi : download : 11%
Bad_Boys_CD1.avi : download : 12%
Bad_Boys_CD1.avi : download : 13%
Bad_Boys_CD1.avi : download : 14%
Bad_Boys_CD1.avi : download : 15%
Bad_Boys_CD1.avi : download : 16%
Bad_Boys_CD1.avi : download : 17%
Bad_Boys_CD1.avi : download : 18%
Bad_Boys_CD1.avi : download : 19%
Bad_Boys_CD1.avi : download : 20%
Bad_Boys_CD1.avi : download : 21%
Bad_Boys_CD1.avi : download : 22%
Bad_Boys_CD1.avi : download : 23%
Bad_Boys_CD1.avi : download : 24%
```

2.3.3 설명



반복문을 활용한 순차적 파일전송이다. 위에 것들과 크게 다른 점은 없다. 순서는 다음과 같다.

1. 클라이언트 : ls.txt 파일을 수신
2. 서버 : 전송할 파일의 목록을 수신
3. 서버 : 4~5번 과정 0부터 파일 개수까지 반복
 4. 서버 : n번 파일 전송
 - 5-1. 파일의 사이즈를 전송
 - 5-2. sendfile 함수를 이용해 파일을 계속 클라이언트로 전송
5. 클라이언트 : 수신 후 저장 및 비교
 - 6-1. 받을 사이즈만큼 메모리 동적할당
 - 6-1. 1460byte로 나누어 받고 다운로드 진척도 출력
 - 6-2. 수신이 완료되면 write를 이용, 저장 및 불러와서 크기 비교

2.4 전송 속도 비교

영화 1편,2편,3편을 기준으로 세 전송방식의 속도 차이를 구해보려고 한다. <time.h> 헤더파일을 이용해 구했으며 단위는 '초'이다.

먼저 다중 스레드를 활용한 방식이다. 서버에서 캡처 했으며, 스레드를 열기 시작한 순간부터 시간을 세었다. 압축을 하지 않은 데이터의 양은 1.5G이다. 모든 파일을 모두 전송하는데 걸린 시간은 137초이다.

```
junho@junho-VirtualBox:~/Desktop/termProject/multi_thread$ gcc -o thread_server thread_server.c -lpthread
junho@junho-VirtualBox:~/Desktop/termProject/multi_thread$ ./thread_server 9090
waiting...
Hello_Server!
0 : Bad_Boys_CD1.avi
1 : Bad_Boys_CD1.smi
2 : Bad_Boys_CD2.avi
3 : Bad_Boys_CD2.smi
open Thread : 0 sec
fd:15 / size:79060 /filename:Bad_Boys_CD2.smi
fd:16 / size:734396416 /filename:Bad_Boys_CD2.avi
fd:17 / size:69069 /filename:Bad_Boys_CD1.smi
fd:18 / size:734330880 /filename:Bad_Boys_CD1.avi
sending Bad_Boys_CD2.smi file
fd : 15 , size = 79060
Finish
sending Bad_Boys_CD2.avi file
fd : 16 , size = 734396416
sending Bad_Boys_CD1.smi file
fd : 17 , size = 69069
Finish
sending Bad_Boys_CD1.avi file
fd : 18 , size = 734330880
Finish
Finish
finish : 137 sec
```

다음은 파일 압축을 활용한 방식이다. 서버에서 캡처 했으며, 파일을 압축하기 시작한 순간부터 시간을 세었다. 압축을 한 데이터의 크기는 1.4G였으며, 파일을 모두 압축하는데 걸린 시간은 50초 그리고 압축된 파일을 전송하는데 걸린 시간은 107초였다. 파일을 압축하고 전송하는데 걸린 시간의 총합은 157초였다.

```
junho@junho-VirtualBox:~/Desktop/termProject/zip$ ./zip_server 9090
waiting...
Hello_Server!
0 : Bad_Boys_CD1.avi
1 : Bad_Boys_CD2.avi
2 : Bad_Boys_CD1.smi
3 : Bad_Boys_CD2.smi
tar cvzf zipData.tar.gz Bad_Boys_CD1.avi Bad_Boys_CD2.avi Bad_Boys_CD1.smi Bad_Boys_CD2.smi
zip start : 0
Bad_Boys_CD1.avi
Bad_Boys_CD2.avi
Bad_Boys_CD1.smi
Bad_Boys_CD2.smi
zip end : 50
fd:6 / size:1417028338 /filename:zipData.tar.gz
sending zipData.tar.gz file
fd : 6 , size = 1417028338
send file : 51
Finish : 157
```

다음은 순차적 파일 전송 방식이다. 서버에서 캡처했으며, 파일을 전송하기 시작하는 순간부터 시간을 세었다. 전체 데이터의 크기는 1.5GB였으며, 1번째 파일은 65초, 2번째 파일은 1초, 세 번째 파일은 62초, 네 번째 파일은 1초가 걸려 총 139초가 걸렸다.

```

junho@junho-VirtualBox:~/Desktop/termProject/linear$ gcc -o linear_server linear_server.c
junho@junho-VirtualBox:~/Desktop/termProject/linear$ ./linear_server 9080
waiting...
Hello_Server!
0 : Bad_Boys_CD1.avi
1 : Bad_Boys_CD1.smi
2 : Bad_Boys_CD2.avi
3 : Bad_Boys_CD2.smi
start : 0 sec
fd:6 / size:734330880 /filename:Bad_Boys_CD1.avi
sending Bad_Boys_CD1.avi file
fd : 6 , size = 734330880
Finish 0 file: 65 sec
fd:7 / size:69069 /filename:Bad_Boys_CD1.smi
sending Bad_Boys_CD1.smi file
fd : 7 , size = 69069
Finish 1 file: 66 sec
fd:8 / size:734396416 /filename:Bad_Boys_CD2.avi
sending Bad_Boys_CD2.avi file
fd : 8 , size = 734396416
Finish 2 file: 138 sec
fd:9 / size:79060 /filename:Bad_Boys_CD2.smi
sending Bad_Boys_CD2.smi file
fd : 9 , size = 79060
Finish 3 file: 139 sec
All Finish : 139 sec

```

이러한 과정을 전송하는 영화의 개수를 1개, 2개, 3개로 늘리며 총 9번 진행하였다. 이 결과를 표로 정리하면 다음과 같다.

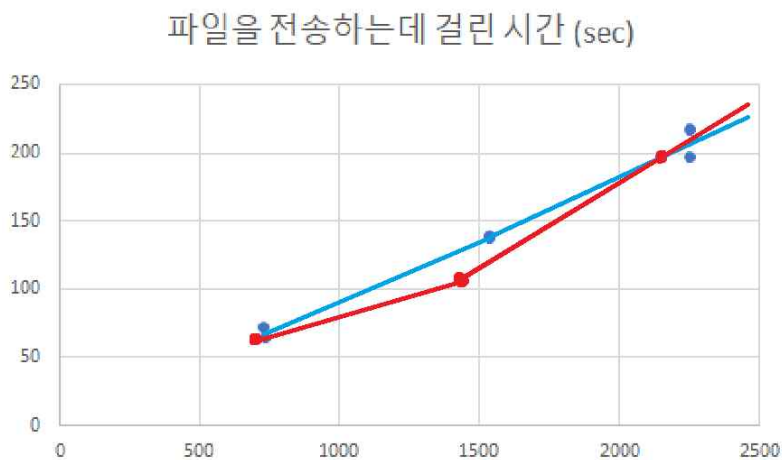
	다중 스레드 동시 전송			파일 압축 후 전송			순차적 전송		
데이터의 크기(MB)	732	1536	2252	701	1433	2150	730	1536	2252
파일을 압축하는데 걸린 시간 (sec)				24	50	73			
파일을 전송하는데 걸린 시간 (sec)	65	137	197	63	107	197	72	139	217
총 걸린 시간 (sec)	65	137	197	87	157	270	72	139	219

이를 그래프화 하여 도식화를 했다. 빨간색이 압축전송방식, 파란색이 다중 스레드와 순차적 전송이다. 그 중 걸린 시간이 적은 쪽이 다중 스레드이다.

사진 1. 총 걸린 시간



사진 2. 파일을 전송하는데 걸린 시간



먼저 총 걸린 시간 먼저 보면, 압축 방식은 나머지 방식보다 느린 뿐더러 크기가 커질수록 격차가 커짐을 알 수 있다. 즉, 보내는 속도보다 압축하는 속도가 더 느림을 알 수 있다. 이는 압축파일의 전송방식에 있어서 기하급수적인 전송속도 변화를 낼 것이라 예상된다. 또한 다중 스레드 방식이 항상 걸리는 시간이 제일 적음을 알 수 있다. 그리고 파일을 전송하는데 걸린 시간을 보면, 압축 방식이 대체로 시간이 적게 걸리는 것을 알 수 있다. 하지만, 이는 2GB가 넘어가니 서로 속도가 비슷해졌고, 추월하지는 않아도 데이터 양이 커지면 서로 비슷할 것으로 예상된다. 또한 여기서도 다중 스레드 방식이 간소하게나마 더 빠름을 볼 수 있다.

이로부터 도출 할 수 있는 결론은, 2GB 아래의 파일 중, 압축이 되어있으면 압축 전송 방식이 제일 빠르고, 압축이 되어있지 않으면 항상 다중 스레드 방식이 빠르다는 것이다.