`23-1 리눅스 프로그래밍 기말 프로젝트 결과보고서

학번/이름: 202011714 / 이현우

0. 요약

현재 개발을 완성한 단계는?	3단계
개발을 완료하지 못한 단계는?	
개발한 프로그램을 실행한 시스템의 환경은?	xshell

1. 실행 스크린샷

```
root@linux-hw:~/final3# ls
client  example.txt  gnu.tar.gz  sample3.txt  server  server.c
root@linux-hw:~/final3# ./server &
[1] 6700
root@linux-hw:~/final3# Server running...

root@linux-hw:~/final3# cd client
root@linux-hw:~/final3/client# ./client gnu.tar.gz & ./client example.txt & ./client sample3.
txt &
[2] 6726
[3] 6727
[4] 6728
root@linux-hw:~/final3/client# [Server]File sent: sample3.txt
[Client]File size: 3541 bytes
[Client]File received: sample3.txt
[Server]File sent: example.txt
[Client]File size: 2574 bytes
[Client]File received: example.txt
[Server]File sent: gnu.tar.gz
[Client]File size: 3565643 bytes
[Client]File received: gnu.tar.gz

[2]   Done                    ./client gnu.tar.gz
[3]-  Done                    ./client example.txt
[4]+  Done                    ./client sample3.txt
```

Ls로 fianl3 디렉토리에 안에 있는 파일과 디렉토리를 확인하고

./server로 Final3 디렉토리 안에 있는 서버(후면처리)를 실행시킨다

Client 폴더로 이동 후에 클라이언트들을 동시 실행한다.

각 파일을 서버에서 보냈다는 메시지와 클라이언트에서 받았다는 메시지를 출력하고 추가로 파일의 용량도 출력해서 서버에 있는 파일의 크기와 비교해보고 서버에 있는 파일이 클라이언트가 있는 폴더에 정상적으로 다운로드 되었는지 확인할 수 있다.

```
root@linux-hw:~/final3/client# ls -al
total 3520
drwxr-xr-x 2 root root    4096 Jun 18 04:23 .
drwxr-xr-x 3 root root    4096 Jun 18 04:13 ..
-rwxr-xr-x 1 root root   13192 Jun 18 04:06 client
-rw-r--r-- 1 root root    2595 Jun 18 04:14 client.c
-rw-r--r-- 1 root root    2574 Jun 18 04:23 example.txt
-rw-r--r-- 1 root root 3565643 Jun 18 04:23 gnu.tar.gz
-rw-r--r-- 1 root root    3541 Jun 18 04:23 sample3.txt
root@linux-hw:~/final3/client# cd
root@linux-hw:~# cd final3
root@linux-hw:~/final3# ls -al
total 3524
drwxr-xr-x  3 root root    4096 Jun 18 04:13 .
drwx------ 30 root root    4096 Jun 18 04:23 ..
drwxr-xr-x  2 root root    4096 Jun 18 04:23 client
-rw-r--r--  1 root root    2574 Jan 25  2022 example.txt
-rw-r--r--  1 root root 3565643 Sep 12  2021 gnu.tar.gz
-rw-r--r--  1 root root    3541 Dec 10  2019 sample3.txt
-rwxr-xr-x  1 root root   13480 Jun 18 04:02 server
-rw-r--r--  1 root root    3278 Jun 18 04:13 server.c
```

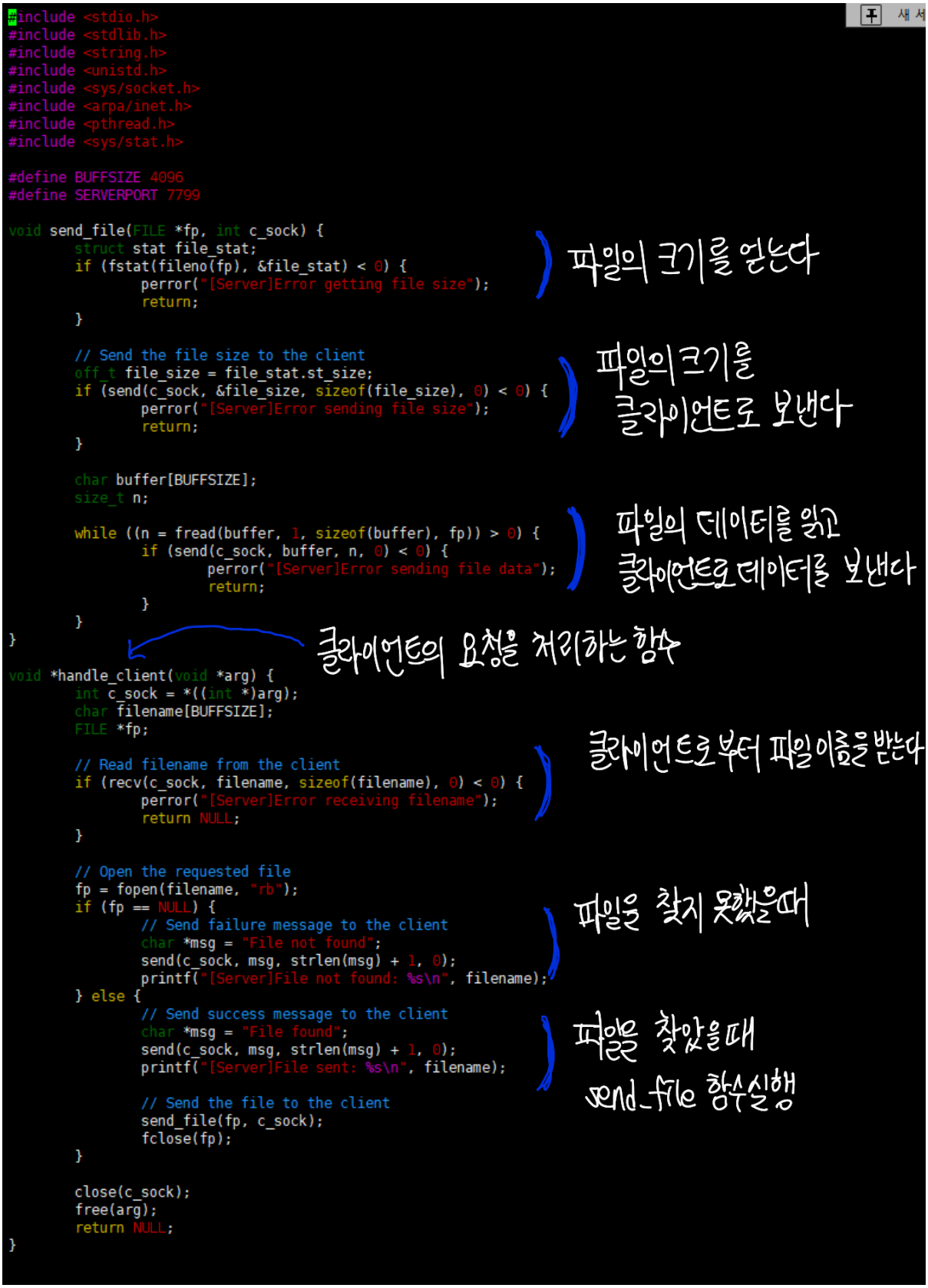
파일을 전송 받고 ls –al로 다운로드 받은 파일 상태 및 용량을 확인한다. 추가로 파일이 있는 final3 디렉토리에서 ls –al을 통해 클라이언트와 용량이 같은지 확인한다.

```
root@linux-hw:~/final3# cd client
root@linux-hw:~/final3/client# ./client aaa.bbb
[Client]File not found: aaa.bbb
root@linux-hw:~/final3/client# [Server]File not found: aaa.bbb

root@linux-hw:~/final3/client# ls -al
total 3520
drwxr-xr-x 2 root root    4096 Jun 18 04:23 .
drwxr-xr-x 3 root root    4096 Jun 18 04:13 ..
-rwxr-xr-x 1 root root   13192 Jun 18 04:06 client
-rw-r--r-- 1 root root    2595 Jun 18 04:14 client.c
-rw-r--r-- 1 root root    2574 Jun 18 04:23 example.txt
-rw-r--r-- 1 root root 3565643 Jun 18 04:23 gnu.tar.gz
-rw-r--r-- 1 root root    3541 Jun 18 04:23 sample3.txt
root@linux-hw:~/final3/client#
```

존재하지 않는 파일 (aaa.bbb)에 대해서도 실행을 해보고 파일이 없다면 서버에서 파일이 없다는 메시지를 출력한다. 클라이언트 디렉토리에서 ls –al을 통해 서버에 존재하지 않는 파일은 저장하지 않는 것을 볼 수 있다.

2. 개발한 코드 스크린샷 및 설명

Server.c 소스코드




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

#define BUFFSIZE 4096
#define SERVERPORT 7799

void send_file(FILE *fp, int c_sock) {
        struct stat file_stat;
        if (fstat(fileno(fp), &file_stat) < 0) {
                perror("[Server]Error getting file size");
                return;
        }

        // Send the file size to the client
        off_t file_size = file_stat.st_size;
        if (send(c_sock, &file_size, sizeof(file_size), 0) < 0) {
                perror("[Server]Error sending file size");
                return;
        }

        char buffer[BUFFSIZE];
        size_t n;

        while ((n = fread(buffer, 1, sizeof(buffer), fp)) > 0) {
                if (send(c_sock, buffer, n, 0) < 0) {
                        perror("[Server]Error sending file data");
                        return;
                }
        }
}

void *handle_client(void *arg) {
        int c_sock = *((int *)arg);
        char filename[BUFFSIZE];
        FILE *fp;

        // Read filename from the client
        if (recv(c_sock, filename, sizeof(filename), 0) < 0) {
                perror("[Server]Error receiving filename");
                return NULL;
        }

        // Open the requested file
        fp = fopen(filename, "rb");
        if (fp == NULL) {
                // Send failure message to the client
                char *msg = "File not found";
                send(c_sock, msg, strlen(msg) + 1, 0);
                printf("[Server]File not found: %s\n", filename);
        } else {
                // Send success message to the client
                char *msg = "File found";
                send(c_sock, msg, strlen(msg) + 1, 0);
                printf("[Server]File sent: %s\n", filename);

                // Send the file to the client
                send_file(fp, c_sock);
                fclose(fp);
        }

        close(c_sock);
        free(arg);
        return NULL;
}
```

```c
        return NULL;
}

int main() {
        int s_sock, c_sock;
        struct sockaddr_in server_addr, client_addr;
        socklen_t client_addr_len;
        pthread_t tid;

        // Create socket
        s_sock = socket(AF_INET, SOCK_STREAM, 0);
        if (s_sock < 0) {
                perror("[Server]Error creating socket");
                exit(1);
        }

        bzero(&server_addr, sizeof(server_addr));

        // Set server address
        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(SERVERPORT);
        server_addr.sin_addr.s_addr = INADDR_ANY;

        // Bind socket to address
        if (bind(s_sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
                perror("[Server]Error binding socket");
                exit(1);
        }

        // Listen for connections
        if (listen(s_sock, 1) < 0) {
                perror("[Server]Error listening for connections");
                exit(1);
        }

        printf("Server running...\n");

        while (1) {
                // Accept a new connection
                client_addr_len = sizeof(struct sockaddr);
                c_sock = accept(s_sock, (struct sockaddr *)&client_addr, &client_addr_len);
                if (c_sock < 0) {
                        perror("[Server]Error accepting connection");
                        exit(1);
                }

                // Create a new thread to handle the client request
                int *new_sock = malloc(sizeof(int));
                *new_sock = c_sock;

                if (pthread_create(&tid, NULL, handle_client, (void *)new_sock) != 0) {
                        perror("[Server]Error creating thread");
                        exit(1);
                }
        }

        close(s_sock);
        return 0;
}

~
~
~
```

소켓, 주소, 스레드변수초기화

서버 소켓 생성

서버주소 설정

주소와결합

클라이언트 접속요청대기

스레드 생성 클라이언트 소켓전달

Client.c 소스코드

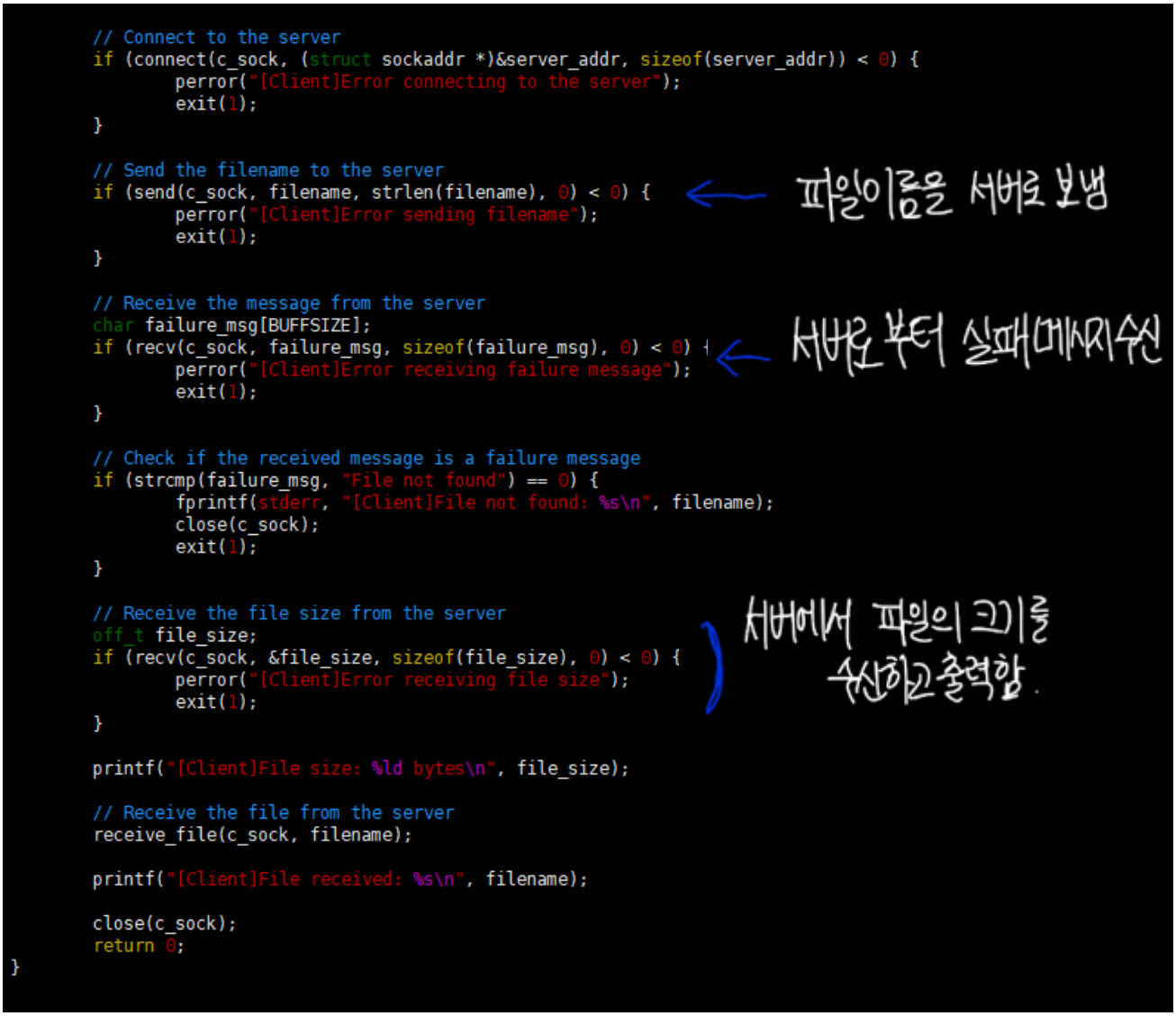
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <sys/socket.h>

#define BUFFSIZE 4096
#define SERVERPORT 7799

void receive_file(int c_sock, const char *filename) {
        FILE *fp = fopen(filename, "wb");
        if (fp == NULL) {
                perror("[Client]Error opening file");
                exit(1);
        }

        char buffer[BUFFSIZE];
        int n;

        while ((n = recv(c_sock, buffer, sizeof(buffer), 0)) > 0) {
                if (fwrite(buffer, 1, n, fp) != n) {
                        perror("[Client]Error writing to file");
                        exit(1);
                }
        }

        if (n < 0) {
                perror("[Client]Error receiving file");
                exit(1);
        }

        fclose(fp);
}

int main(int argc, char *argv[]) {
        if (argc != 2) {
                fprintf(stderr, "Usage: %s <filename>\n", argv[0]);
                exit(1);
        }

        char *filename = argv[1];
        int c_sock;
        struct sockaddr_in server_addr;
        socklen_t c_addr_size;

        // Create socket
        c_sock = socket(AF_INET, SOCK_STREAM, 0);
        if (c_sock < 0) {
                perror("[Client]Error creating socket");
                exit(1);
        }

        bzero(&server_addr, sizeof(server_addr));

        // Set server address
        server_addr.sin_family = AF_INET;
        server_addr.sin_port = htons(SERVERPORT);
        server_addr.sin_addr.s_addr = inet_addr("127.0.0.1");

        // Connect to the server
        if (connect(c_sock, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
                perror("[Client]Error connecting to the server");
                exit(1);
        }
```

서버로부터 데이터수신후 받은데이터를 파일에작성

소켓생성

서버주소 초기화

서버연결



전체 로직과 주요 기능

서버를 실행하고 클라이언트를 실행한다. (서버는 멀티스레드를 기반으로 클라이언트의 요청이 들어올 때마다 새로운 스레드가 생성되어 여러 클라이언트 요청을 동시에 처리할 수 있다.) 클라이언트는 명령 행 인자로 파일 이름을 서버에 전송하고 서버는 파일을 찾아서 전송할 준비를 한다. 클라이언트가 서버로부터 응답을 받고 파일이 존재하지 않으면 오류 메시지를 출력 후 종료하고, 파일이 존재하면 클라이언트는 파일 크기를 받고 파일을 저장한다. 파일 전송이 완료되면 클라이언트는 성공 메시지를 출력하고 종료한다. 서버는 별도의 종료가 있기 전까지 다른 클라이언트의 요청을 대기한다.

3. 최종 개발을 실패한 내용 (있다면)