Report

System Programing

Assignment #3

수 업	시스템프로그래밍실습
수업시간	월 1, 2
O S	Ubuntu 12.04 64bit
과제번호	Assignment #3
제출날짜	2014-06-13
전 공	컴퓨터공학과
학 번	2009720151
이 름	이재해

<Program Overview>

-프로그램소개

이번 Assignment는 지난 Assignment를 이용하여 User authentication, Access control, Transmission mode(binary/ASCII), Split connection(data/control) 그리고 log file format의 기능을 추가하고 몇 가지의 명령어가 추가된 프로그램을 제작하는 것입니다.

간단히 설명하면 Login기능, 특정 IP만을 허용하는 기능, 데이터를 binary or ascii로 전달하는 기능, 여러 socket을 동시에 연결해서 control과 data의 전송을 나누는 기능 그리고 프로그램이 실행되는 중 log를 기록하는 기능입니다.

-문제분석

이번 Assignment를 제작하는 중에는 어려운 점이 많았습니다. 결국 프로그램을 완성하지 못하고 제출하게 되기도 했습니다.

일단 하나하나 집고 넘어가면 User authentioncation을 구현하는 것은 어렵지 않았습니다. 강의 자료에 나와 있는 소스들을 잘 이용하면 가능한 기능으로 client에서 server로 접속하고 client는 ID와 Password를 입력하고 그 데이터를 이용해서 server는 이전에 미리 입력해둔 자료를 이용해서 client가 login의 가능 여부를 확인합니다.

다음으로 Access control이 것 또한 이전에 미리 입력해둔 자료를 이용해서 현재 접속한 client의 ip를 확인하여 해당 client가 접속이 가능한지의 여부를 확인합니다.

위의 기능들은 Assignment 3의 첫 번째 내용으로 구현하는 데에 별다른 문제가 없었습니다.

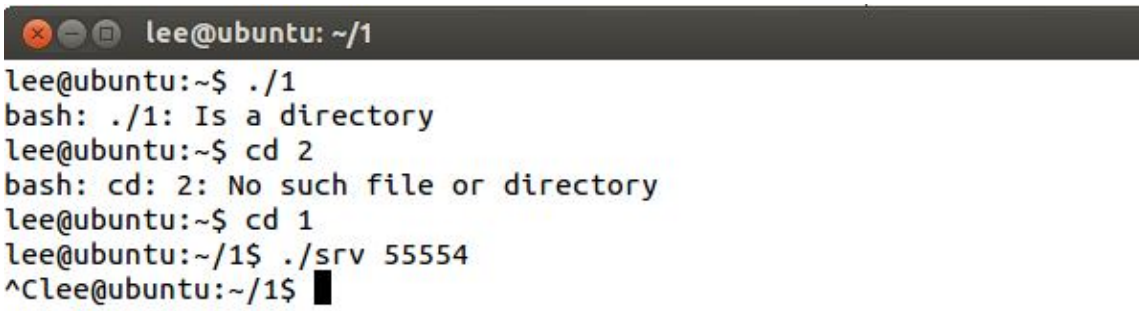
이 다음 부터는 Transmission mode(binary/ASCII), Split connection(data,control)이 가장 큰 문제가 되었습니다. 먼저 데이터를 binary와 ASCII 중에서 어떤 것으로 처리하는지를 확인하고 각각의 방식으로 처리를 해야 하는데 원래 파일의 입출력과는 방식과는 조금 다르게 사용해야 합니다. 이 점에 대한 공부가 부족하여 제대로 구현하지 못하고 생략하였습니다. 그래서 제가 제작한 프로그램은 Transmission mode(binary/ASCII)가 구현돼지 않았습니다. 다음으로는 Split connection(data, control)의 구현이었습니다. 이 부분은 practice 3-2에서 이미 구현해 보았기 때문에 크게 제작하는 과정에서 문제는 없었지만, 이전에 구현하는 중에는 여러 가지로 구현에 문제가 있었습니다. 현재 제작한 프로그램은 프로그램을 시작하고 server에 client가 접속해서 3-1차의 내용을 수행하고 난 뒤에 명령어를 입력하게 됩니다. 이 명령어를 server에 전달하는 부분이 Split connection에서 control부분입니다. 그리고 명령어가 입력되고 입력된 명령어가 client와 server 간에 데이터 전송이 필요한 명령어라면 Split connection에서 data를 전송하는 새로운 연결을 만듭니다. 이 만들어지는 연결은 data 연결면에서는 client가 server로 이용되고 server는 client로 사용됩니다. 이러한 구성은 client가 server로 현재 자신의 ip와 port 정보를 전달해서 server가 그를 이용해서 client에 접속되게 되는 것입니다. 이 외에도 강의자료 3-2에서 제안하고 있는 기능들이 있지만 모두 구현하지는 못하였습니다.

마지막으로 3-2의 내용으로 Log file format입니다. 이는 현재 server에서 일어나는 몇 가지 경우의 행동들을 기록하는 기능입니다. 별다른 어려움 없이 해결 가능한 문제였습니다.

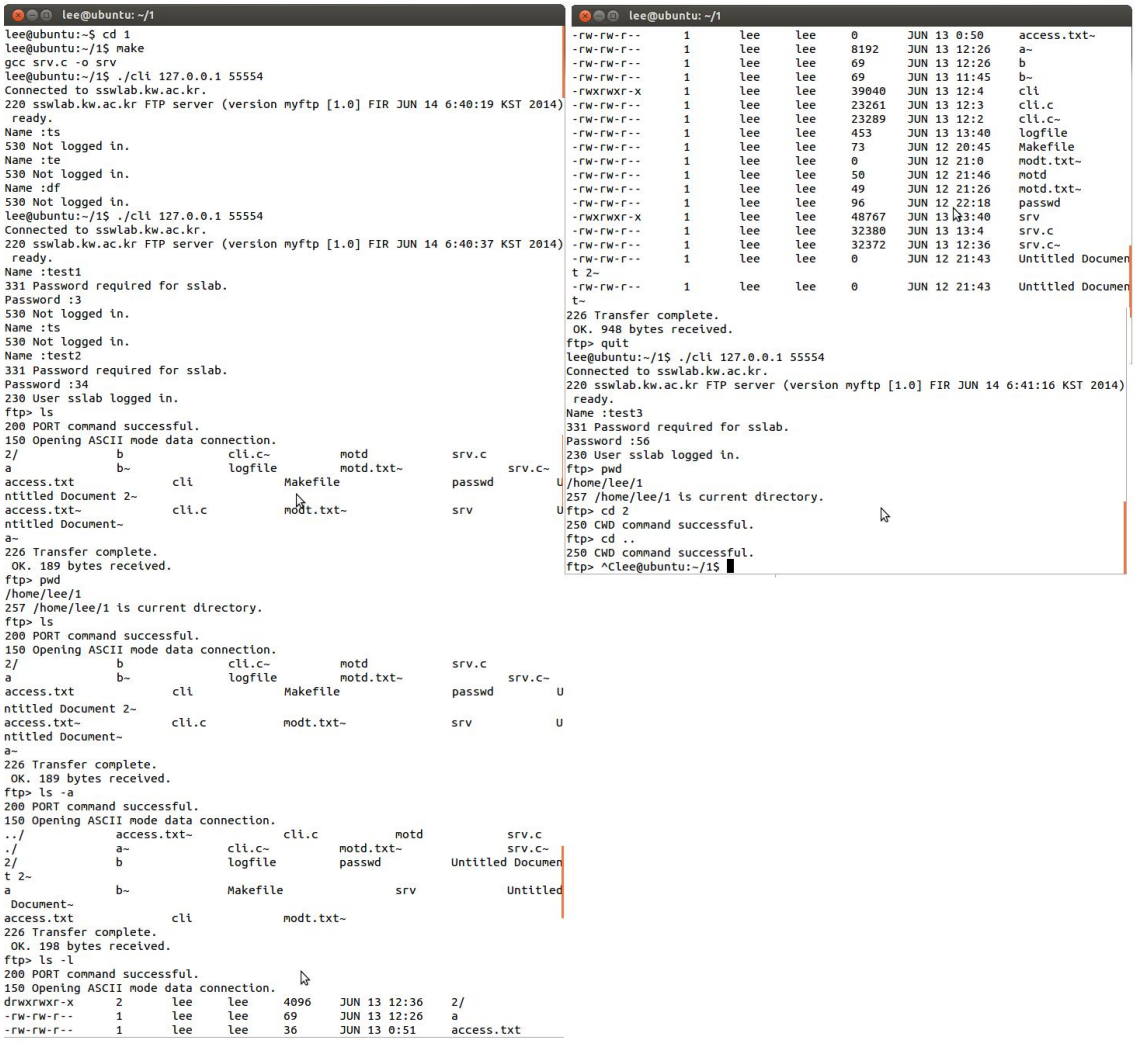
큰 문제점들은 설명했지만 이 외에도 여러 가지 작은 문제점들이 많이있었습니다.

-실행 결과 캡쳐 화면

<server화면>



<client 화면>



<logfile>

```
FIR JUN 14 6:40:10 2014Server is started
FIR JUN 14 6:40:32 2014 [127.0.0.1:43654] ts FOG_FAIL
FIR JUN 14 6:40:34 2014 [127.0.0.1:43654] te FOG_FAIL
FIR JUN 14 6:40:34 2014 [127.0.0.1:43654] df FOG_FAIL
FIR JUN 14 6:40:34 2014 [127.0.0.1:43654] df  LOG_OUT
[total service time : 24 sec]
FIR JUN 14 6:40:41 2014 [127.0.0.1:43655] test1 FOG_FAIL
FIR JUN 14 6:40:43 2014 [127.0.0.1:43655] ts FOG_FAIL
FIR JUN 14 6:40:47 2014 [127.0.0.1:43655] test2 FOG_IN
FIR JUN 14 6:41:15 2014 [127.0.0.1:43655] test2  LOG_OUT
[total service time : 65 sec]
FIR JUN 14 6:41:21 2014 [127.0.0.1:43660] test3 FOG_IN
FIR JUN 14 6:41:30 2014 [127.0.0.1:43660] test3  LOG_OUT
[total service time : 80 sec]
FIR JUN 14 6:41:33 2014 Server is terminated
```

<Algorithm>

<Pseudo Code>

각 코드의 일부분을 가져와서 설명하겠습니다.

-client

이미 커넥트가 된 이후입니다.

```
i=0;
        while(i<3)//login하기위한 절차 3번실패시 종료
        {
                write(STDOUT_FILENO, "Name :", 6);
                memset(buff, 0, sizeof(buff));
                len=read(STDIN_FILENO, buff, BUF_MAX_SIZE);
                buff[len-1]='₩0';
                write(sockfd, buff, BUF_MAX_SIZE);
                read(sockfd, buff, BUF_MAX_SIZE);
                write(STDOUT_FILENO, buff, BUF_MAX_SIZE);
                write(STDOUT_FILENO, "₩n", 1);
                if(strncmp(buff, "331", 3)==0)
                {
                        write(STDOUT_FILENO, "Password :", 10);
                        memset(buff, 0, sizeof(buff));
                        len=read(STDIN_FILENO, buff, BUF_MAX_SIZE);
                        buff[len-1]='₩0';
                        write(sockfd, buff, BUF_MAX_SIZE);
                        read(sockfd, buff, BUF_MAX_SIZE);
                        write(STDOUT_FILENO, buff, BUF_MAX_SIZE);
```

```
                write(STDOUT_FILENO, "₩n", 1);

                if(strncmp(buff, "230", 3)==0)
                        break;
        }

        i++;
    }

    while(i<3) //로그인에 성공
    {
        memset(buff, 0, sizeof(buff));
        write(STDOUT_FILENO, "ftp> ", 5);
        read(STDIN_FILENO, buff, BUF_MAX_SIZE); //명령어 입력

        m_argc = make_argv(buff, m_argv);//명령어 정리

        memset(buff, 0, sizeof(buff));
        convert(m_argc, m_argv, buff);//명령어 변환

        if(write(sockfd, buff, BUF_MAX_SIZE) > 0)
        {
            if(strstr(buff, "NLST") || strstr(buff, "RETR") || strstr(buff,
"STOR"))//Split connection Data가 필요할 때
            {//PORT
            n=rand()%20000;
            n+=10001;//포트 생성

//새로이 data connection에 사용될 socket을 생성
            tempfd = socket(PF_INET, SOCK_STREAM,0);
            memset(&add_temp, 0, sizeof(add_temp));
            add_temp.sin_family=AF_INET;
            add_temp.sin_addr.s_addr=inet_addr(argv[1]);
            add_temp.sin_port=htons(n);
            bind(tempfd, (struct sockaddr *)&add_temp,
sizeof(add_temp));
            listen(tempfd, 5);//client는 Split connection data에서는 server
측이다.
            /* your own codes */
            memset(&hostport, 0, sizeof(hostport));
            convert_addr_to_str(add_temp.sin_addr.s_addr, n, hostport);//
```

클라이언트의 데이터를 전송하기 위해 정리

```
                        /* make control connection and data connection */
                        /* your own codes */
                        len = sizeof(cli_addr);
                        n=write(sockfd, hostport, 25);//클라이언트의 주소 전송

                        server_fd = accept(tempfd, (struct sockaddr*)&cli_addr,
&len);//서버에서 클라이언트로 connect를 확인

                        read(server_fd, buff, BUF_MAX_SIZE);
                        write(STDOUT_FILENO, buff, BUF_MAX_SIZE);

                        close(server_fd);
                        }
                        else if(read(sockfd, buff, BUF_MAX_SIZE) > 0)
                                if(strncmp(buff, "QUIT", 4)==0)
                                        sh_quit(0);
                                else
                                        write(STDOUT_FILENO,                    buff,
BUF_MAX_SIZE);
                        else
                                break;
                }
                else
                        break;
        }
```

-server
accept가 일어날 때

```
while(1)
        {
                pid_t pid;
                len = sizeof(client_addr);
                client_fd = accept(server_fd, (struct sockaddr*)&client_addr, &len);
                //client가 접속
                if((PID = fork()) == 0) //child process
                {//접속 가능한 client인지 확인/////////
                        fp_checkIP = open("access.txt", O_RDONLY);
                        memset(buff, 0, sizeof(buff));
                        len=read(fp_checkIP, buff, BUF_MAX_SIZE);
```

```
IP = inet_ntoa(client_addr.sin_addr);//printf("%s₩n",IP);
temp = strtok(IP, ".");
strcpy(_IP[0], temp);
temp = strtok(NULL, ".");
strcpy(_IP[1], temp);
temp = strtok(NULL, ".");
strcpy(_IP[2], temp);
temp = strtok(NULL, "₩n");
strcpy(_IP[3], temp);

while(access == 0)
{
        memset(_checkIP[0], 0, sizeof(_checkIP[0]));
        memset(_checkIP[1], 0, sizeof(_checkIP[1]));
        memset(_checkIP[2], 0, sizeof(_checkIP[2]));
        memset(_checkIP[3], 0, sizeof(_checkIP[3]));
        if(che == 0)
        {
                temp = strtok(buff, ".");
                che++;
        }
        else
                temp = strtok(NULL, ".");

        if(!temp)
                break;

        strcpy(_checkIP[0], temp);
        temp = strtok(NULL, ".");
        strcpy(_checkIP[1], temp);
        temp = strtok(NULL, ".");
        strcpy(_checkIP[2], temp);
        temp = strtok(NULL, "₩n");
        strcpy(_checkIP[3], temp);

        for(i=0; i<4; i++)
        {
                if(strcmp(_IP[i], _checkIP[i])!=0)
                        break;
        }
```

```
                        if(i==4)
                                i=3;

                        if(strcmp(_checkIP[i],    "*")==0    ||    strcmp(_IP[i],
_checkIP[i])==0)
                                access=1;
                }

                close(fp_checkIP);

                if(access==1)
                        write(client_fd, "ok", 2);
                else
                        write(client_fd, "no", 2);
////////////////////////////////////////////////////////////////////////////여기까지 access control

                memset(buff, 0, sizeof(buff));
                strcpy(buff, "220 ");

                timer=time(NULL);
                t=localtime(&timer);//접속시간을 위해서
                //welcome msg를 위해 부분
                fp=open("motd", O_RDONLY);
                len=read(fp, a, BUF_MAX_SIZE);

                i=0;
                while(a[len-i]!=')')
                {
                        a[len-i]='₩0';
                        i++;
                }
                a[len-i]='₩0';
                strcat(buff, a);
                //현재시간을 단위별로 정리
                switch(t->tm_wday)
                {
                        case 0: strcat(buff, " SUN "); break;
                        case 1: strcat(buff, " MON "); break;
                        case 2: strcat(buff, " TUE "); break;
                        case 3: strcat(buff, " WED "); break;
```

```
                case 4: strcat(buff, " THU "); break;
                case 5: strcat(buff, " FIR "); break;
                case 6: strcat(buff, " SAT "); break;
        }
        switch(t->tm_mon)
        {
                case 0: strcat(buff, "JAN "); break;
                case 1: strcat(buff, "FEB "); break;
                case 2: strcat(buff, "MAR "); break;
                case 3: strcat(buff, "APR "); break;
                case 4: strcat(buff, "MAY "); break;
                case 5: strcat(buff, "JUN "); break;
                case 6: strcat(buff, "JUL "); break;
                case 7: strcat(buff, "AUG "); break;
                case 8: strcat(buff, "SEP "); break;
                case 9: strcat(buff, "OCT "); break;
                case 10: strcat(buff, "NOV"); break;
                case 11: strcat(buff, "DEC"); break;
        }
        memset(t_t, 0, 16);
        uitoa((t->tm_mday)+1, t_t);
        strcat(buff, t_t);
        strcat(buff," ");
        memset(t_t, 0, 16);
        uitoa(t->tm_hour, t_t);
        strcat(buff, t_t);
        strcat(buff,":");
        memset(t_t, 0, 16);
        uitoa(t->tm_min, t_t);
        strcat(buff, t_t);
        strcat(buff,":");
        memset(t_t, 0, 16);
        uitoa(t->tm_sec, t_t);
        strcat(buff, t_t);
        strcat(buff," KST ");
        memset(t_t, 0, 16);
        uitoa((t->tm_year)+1900, t_t);
        strcat(buff, t_t);
        strcat(buff,") ready.₩n");
//welcome메세지를 보내준다
        write(client_fd, buff, BUF_MAX_SIZE);
```

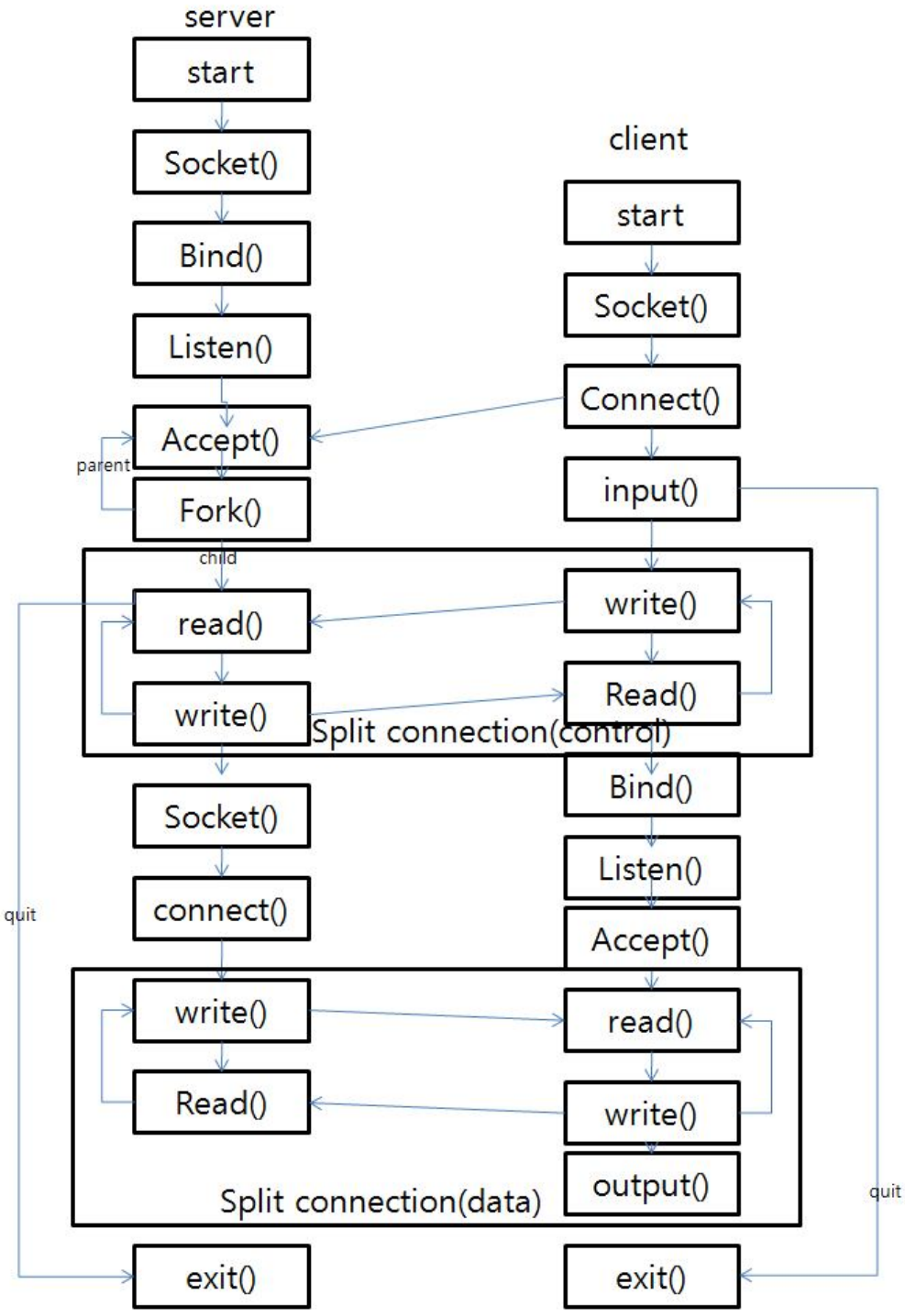
```
                        close(fp);

                        i=0;//로그인 가능함수인지 확인
                        while(access&&i<3)//기회는 3번
                        {
                                memset(buff, 0, sizeof(buff));
                                n=read(client_fd, buff, BUF_MAX_SIZE);
                                memset(id, 0, sizeof(id));
                                strcpy(id, buff);

                                memset(buff, 0, sizeof(buff));
                                if(id_match(id)==1)//입력된 id 확인
                                {//id 성공
                                        strcpy(buff, "331 Password required for
sslab.");
                                        write(client_fd, buff, BUF_MAX_SIZE);

                                        memset(buff, 0, sizeof(buff));
                                        n=read(client_fd, buff, BUF_MAX_SIZE);
                                        memset(pass, 0, sizeof(pass));
                                        strcpy(pass, buff);

                                        memset(buff, 0, sizeof(buff));
                                        if(pass_match(id, pass)==1)//입력된 passwod
확인
                                        {//password 성공
                                                strcpy(buff, "230 User sslab logged
in.");
                                                write(client_fd,                buff,
BUF_MAX_SIZE);

log_infor(inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port), id);
                                                write(fd_log, "FOG_IN₩n", 7);

                                                break;
                                        }
                                        else
                                        {//password 실패
                                                strcpy(buff, "530 Not logged in.");
                                                write(client_fd,                buff,
```

```
BUF_MAX_SIZE);

log_infor(inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port), id);
                                        write(fd_log, "FOG_FAIL₩n", 9);i++;
                                }
                        }
                        else
                        {//id 실패
                                strcpy(buff, "530 Not logged in.");
                                write(client_fd, buff, BUF_MAX_SIZE);
                                log_infor(inet_ntoa(client_addr.sin_addr),
ntohs(client_addr.sin_port), id);
                                        write(fd_log, "FOG_FAIL₩n", 9);i++;
                        }
                }
//기회는 3번 실패면 종료
                if(i==3)
                        kill(getpid(), SIGINT);

                while(access)
                {
                        memset(buff, 0, sizeof(buff));
                        n=read(client_fd, buff, BUF_MAX_SIZE);

                        if(strstr(buff, "NLST") || strstr(buff, "RETR") ||
strstr(buff, "STOR"))
                        {//split connection이 필요한 명령어면 사용
                                read(client_fd, add_temp, 25);

                                convert_str_to_addr(add_temp,          host_ip,
(unsigned int *) &port_num);
//command connection을 통해서 받은 정보를 주소 추출

                                sockfd          =          socket(AF_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(host_ip);
                                serv_addr.sin_port=htons(port_num);
//split connection data에서는 server가 client측이 된다.
//data connection
```

```
                                        connect(sockfd,(struct sockaddr*)&serv_addr,sizeof(serv_addr));

                                        memset(reply, 0, sizeof(reply));
                                        strcpy(reply, "200 PORT command successful.₩n");

                                        if(strstr(buff, "NLST"))
                                                strcat(reply, "150 Opening ASCII mode data connection.₩n");
                                        if(strstr(buff, "RETR"))
                                                strcat(reply, "150 Opening BINARY mode data connection for exam1 (24644 bytes)₩n");
                                        if(strstr(buff, "STOR"))
                                                strcat(reply, "150 150 Opening BINARY mode data connection for vhdl.₩n");

                                        command(buff);

                                        n=strlen(buff);

                                        memset(a, 0, sizeof(a));
                                        strcpy(a, reply);
                                        strcat(a, buff);
                                        strcat(a, "226 Transfer complete.₩n OK. ");
                                        memset(add_temp, 0, sizeof(add_temp));
                                        uitoa(n, add_temp);
                                        strcat(a, add_temp);
                                        strcat(a, " bytes received.₩n");
                                        memset(buff, 0, sizeof(buff));
                                        strcpy(buff, a);

                                        write(sockfd, buff, BUF_MAX_SIZE);

                                        close(sockfd);
                                }
                                else
                                {
                                        command(buff);

                                if(strncmp(buff, "QUIT", 4) == 0)
```

```
                                kill(getpid(), SIGINT);

                        n=strlen(buff);
                        write(client_fd, buff, BUF_MAX_SIZE);
                        }

                }
        }
        else //parent process
        {

        }

        close(client_fd);
}
```

-flow chart


server
start
Socket()
Bind()
Listen()
Accept()
parent
Fork()
child
read()
write()
Socket()
connect()
quit
write()
Read()
exit()
client
start
Socket()
Connect()
input()
write()
Read()
Split connection(control)
Bind()
Listen()
Accept()
read()
write()
output()
Split connection(data)
quit
exit()

<Conslusion and discussion>

이번 과제는 지난 1차와 2차보다 어렵게 느껴졌습니다. 예전에 1차 과제가 가장 어렵다고 설명하셨었는데 저는 지난 1차보다 이번 과제가 더 어려웠습니다. 일단 지난 1차에서는 일단 제안서에서 설명하는 기능들을 수행하도록 제작은 했었지만 이번 과제에서는 프로그램이 제대로 구현하지 못하였습니다. 가장 큰 문제점은 Transmission mode(binary/ASCII)의 이해가 부족하여 구현하지 못하였다는 점입니다. 예전부터 파일의 입출력은 여러 과제에서 수행해 보았기 때문에 별다른 문제점을 느끼지 못하고 있었는데 이번 과제를 통해서 파일의 입출력 기능에 대한 이해가 많이 부족하다는 것을 알게 되었습니다. 제가 알고 있는 기능들은 가장 간단한 파일의 입출력 기능으로 이번 과제에서 제안하고 있는 기능들을 구현하기 위해서 어떻게 제작해야 하는지에 대해 생각해보고 나름대로 구현해보려고 노력 해보았지만 구현하면 구현할수록 원래의 코드에서 이상이 발생하고 여기저기 건드리면 건드릴수록 프로그램의 작동이 제대로 되지 않게 되었습니다. 그래서 문제가 크게 발생해서 원래의 상태로 돌아가는 것도 큰 문제였었습니다. 또한 이번 과제를 통해서 이전에 제작했던 과제의 내용을 사용해서 제작한다는 것에 큰 어려움을 느끼게 됐습니다. 이전에 과목에서는 프로젝트여도 이전의 내용을 이용하는게 아니라 처음부터 시작하는 것이기 때문에 하나하나 쌓아 올리면 됐지만, 이번처럼 이전의 내용을 이용해서 제작하는 것은 이전에 제작할 때부터 소스에 유연성을 줘서 나중에 수정을 가하거나 프로그램에 기능을 추가할 때 편하게 만들어 줘야 했었습니다. 일단 이전까지의 습관인지 일단 프로그램을 구현하고 과제에서 바라는 내용들을 하나하나 맞춰갔지만, 이번 과제를 통해서 프로그램을 제작하기에 앞서서 프로그램을 어떻게 구현할 것인지에 대해서 전체적인 그림을 그려보고 제작해보도록 하겠습니다. 게다가 소스에 주석을 달아두는 습관이 없어서 나중에 프로그램을 수정하고 싶어도 이게 어디에 어떻게 연결되고 어떻게 사용되었었는지에 대해서 기억하지 못할 때가 있어서 그 점에 대해서도 다시 생각하게 되었습니다. 이전 과제까지는 그래도 구현이라도 됐는데 이번 과제는 아에 제대로 구현이 되지 않아서 조금 충격이 컸습니다. 일단 지금 현재 부족한 부분에 대해서 공부를 해서 zero에서부터 시작해서 프로그램을 다시 한 번 구현해보고 싶습니다.