시스템 프로그래밍 실습

[Assignment3-3]

Class : **[A]**

Professor : **[김태석 교수님]**

Student ID : **[2019202032]**

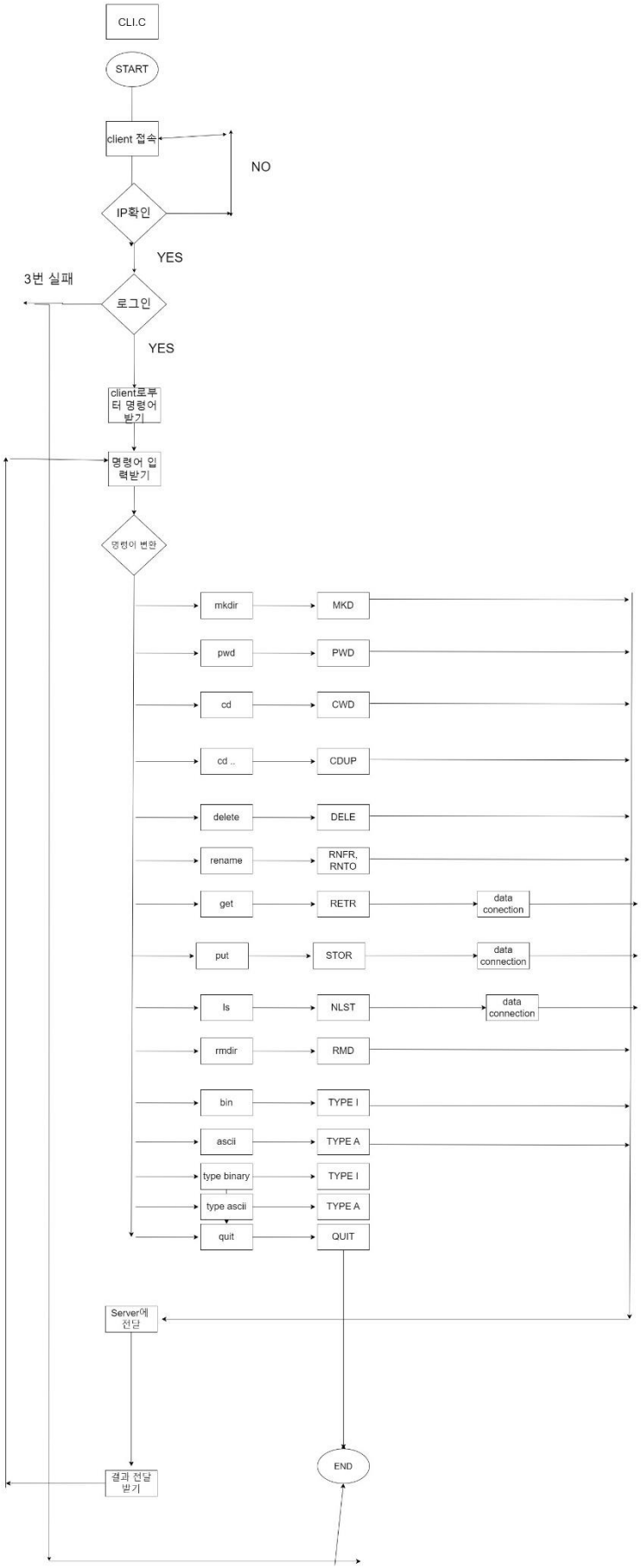
Name : **[이상현]**

Introduction

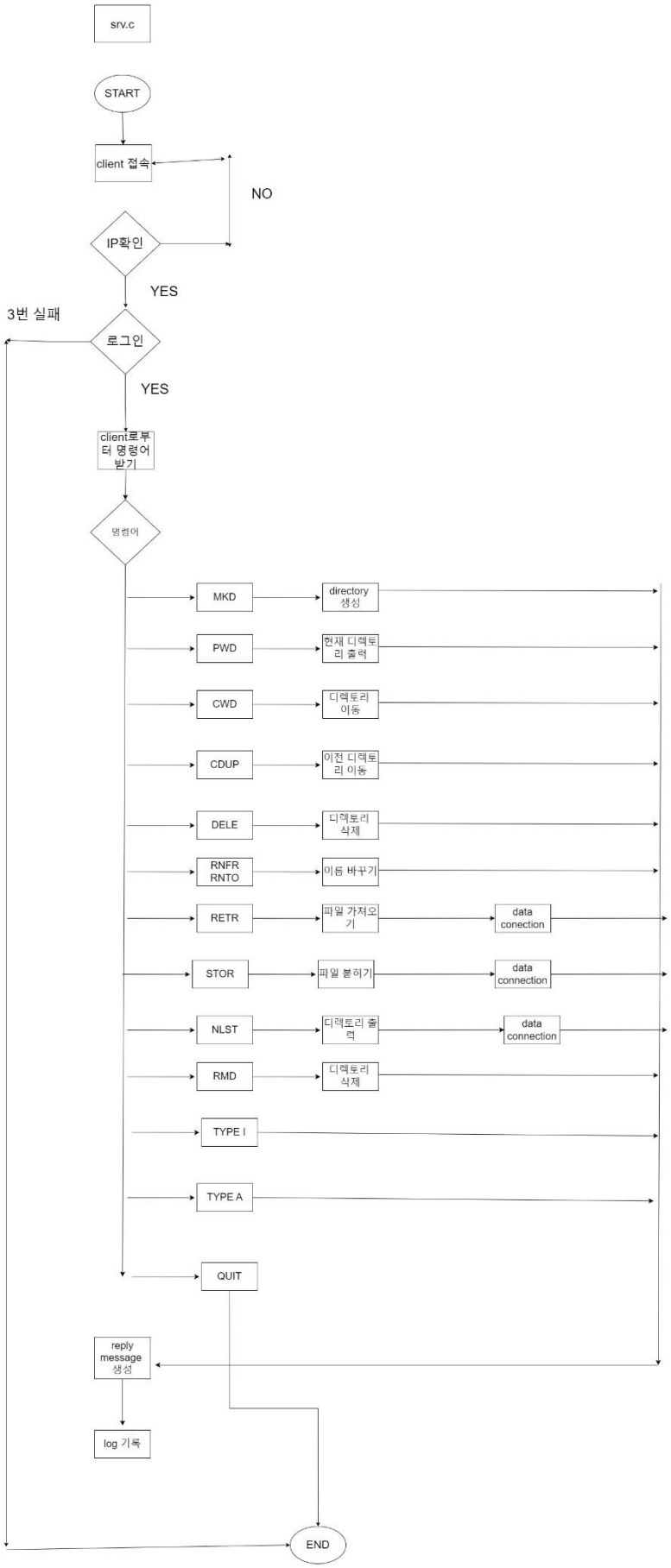
해당 과제는 client 로부터 명령어를 받고 ftp 명령어로 변환한 뒤, 서버에 접속하여 명령어를 수행하는 과제이다. Client 는 server 에 접속한 후 접속가능한 IP 인지 확인받은 후 로그인과정을 거친 후, 로그인이 되면 명령어를 수행한다. 이때 ls, get, put 명령어를 처리할 때는 port 명령어를 사용하여 data connection 을 연결하고, data connection 을 통해 결과를 주고받는다. 또한 각 명령어에 대해 client 의 ip, port 번호, 시간 등의 정보와 함께 reply message & reply code 를 log 에 기록으로 남긴다.

Flow chart

[cli.c]


CLI.C
START
client 접속
NO
IP확인
YES
3번 실패
로그인
YES
client로부터 명령어 받기
명령어 입력받기
명령어 변환
mkdir
MKD
pwd
PWD
cd
CWD
cd ..
CDUP
delete
DELE
rename
RNFR, RNTO
get
RETR
data conection
put
STOR
data connection
ls
NLST
data connection
rmdir
RMD
bin
TYPE I
ascii
TYPE A
type binary
TYPE I
type ascii
TYPE A
quit
QUIT
Server에 전달
결과 전달 받기
END

[srv.c]


srv.c
START
client 접속
NO
IP확인
YES
3번 실패
로그인
YES
client로부터 명령어 받기
명령어
MKD
directory 생성
PWD
현재 디렉토리 출력
CWD
디렉토리 이동
CDUP
이전 디렉토리 이동
DELE
디렉토리 삭제
RNFR RNTO
이름 바꾸기
RETR
파일 가져오기
data conection
STOR
파일 붙히기
data connection
NLST
디렉토리 출력
data connection
RMD
디렉토리 삭제
TYPE I
TYPE A
QUIT
reply message 생성
log 기록
END

Pseudo code

<srv.c>

```
int main(int argc, char **argv)
{
logFile = fopen("logfile", "w");
	if(logFile == NULL){
		write(1, "failed to open", 15);
	}

	FILE *fp_checkIP;	//FILE stream to check client's IP
	fp_checkIP = fopen("access.txt", "r");
	if(fp_checkIP == NULL) {
		printf("Error: cannot open access file₩n");
		//close(connfd);
	}

	int listenfd, connfd;
	struct sockaddr_in servaddr, cliaddr;
	listenfd = socket(PF_INET, SOCK_STREAM, 0);
	int opt = 1;
	setsockopt(listenfd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));
```

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

        if(bind(listenfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) < 0){   //bind socket
                printf("Server: Can't bind local address₩n");
                return 0;
        }

        listen(listenfd, 5);     //listen from client

        int clilen= sizeof(cliaddr);
        connfd = accept(listenfd, (struct sockaddr *) &cliaddr, &clilen);

get_time(cur_time);
        fprintf(logFile, "%s Server is started₩n", cur_time);
        fflush(logFile);
char client_ip[MAX_BUF];
        int n = read(connfd, client_ip, MAX_BUF);
        client_ip[n] = '₩0';

        //print information of client//
        //client_info(cliaddr, client_ip);
```

```
char IPs[MAX_BUF];
	int find = 0;
	while((fgets(IPs, MAX_BUF, fp_checkIP) != NULL))
	{

		/*
		*		Slicing IP address by dot(.)
		*/
		Ip_Slicing(IPs, access_IP);
		Ip_Slicing(client_ip, input_IP);

		/*
		*		Check string
		*/
		for(int i = 0; i<4; i++){
			if(!strcmp(input_IP[i], "*"))	//wildcard
				continue;
			if(!strcmp(access_IP[i], "*"))	//wildcard
				continue;
			if(strcmp(access_IP[i], input_IP[i]))	//not equal string
				find = 0;
		}
```

```
/*
*           Free string array
*/
for(int i = 0; i<5; i++){
      free(access_IP[i]);
}
free(access_IP);

for(int i = 0; i<5; i++){
      free(input_IP[i]);
}
free(input_IP);

/*
*           Client connected
*/
if(find) {
      write(1, "** Client is connected **₩n", 27);
      write(connfd, "ACCEPTED", 9);
      char welcome_msg[200];
      get_time(cur_time);

      read(connfd, buf, MAX_BUF);//NOTHING
```

```
                sprintf(welcome_msg, "sswlab.kw.ac.kr FTP server (version myftp [1.0] %s) ready.₩n", cur_time);
                write(1, welcome_msg, strlen(welcome_msg));
                fprintf(logFile, "%s", welcome_msg);
                fflush(logFile);
                write(connfd, welcome_msg, strlen(welcome_msg));
                break;
            }
        }

        /*
        *     Failed to connect
        */


        /*
        *             log-in failed
        */
        char user[100];
        if(log_auth(connfd, cliaddr, IPs, user) == 0) {         //log-in failed
            write(1, "** Fail to log-in **₩n", 22);
            get_time(cur_time);
            fprintf(logFile, "%s Server is terminated₩n", cur_time);
            exit(0);
```

```
        close(connfd);
//      continue;
}
else {
        /******log-in success*******/
        write(1, "** Success to log-in **₩n", 25);
        //close(connfd);
        /************************/
}



/**
 *
 * Get command from control connection
 *
 */
for(;;){
        memset(buf, 0, MAX_BUF);
        read(connfd, buf, MAX_BUF);
        buf[strlen(buf)] = '₩0';
        char result_buff[MAX_BUF];
        memset(result_buff, 0, MAX_BUF);
```

```
        char logstr[500];



        int stor = 0;
        if(buf[0] == 'S' && buf[1] == 'T' && buf[2] == 'O' && buf[3] == 'R')
            stor = 1;
        if(buf[0] == 'N' && buf[1] == 'L' && buf[2] == 'S' && buf[3] == 'T' || buf[0] == 'R' && buf[1] == 'E' && buf[2] == 'T' && buf[3] == 'R' || stor == 1) {
        /*****************  receive random port number by control connection ***************/
        char temp[MAX_BUF];
        char *host_ip = (char*)malloc(sizeof(char) * MAX_BUF);
        unsigned int port_num = 0;

        write(connfd, "dummy", MAX_BUF);

        n = read(connfd, temp, MAX_BUF);      //receive port command
        temp[strlen(temp)] = '₩0';
        write(1, temp, MAX_BUF);
        write(1, "₩n", 2);
        host_ip = convert_str_to_addr(temp, (unsigned int *) &port_num);      //convert port number and ip address
        /*******************************************************************************/

        sleep(0.5);
```

```
        /********************* connect data connection ******************************/
        struct sockaddr_in data_temp;
        int data_sockfd;

        data_sockfd = socket(AF_INET, SOCK_STREAM, 0);
        memset(&data_temp, 0, sizeof(data_temp));
        data_temp.sin_family = AF_INET;
        inet_pton(AF_INET, host_ip, &data_temp.sin_addr);
        data_temp.sin_port = htons(port_num);
        if(connect(data_sockfd, (struct sockaddr*) &data_temp, sizeof(data_temp)) < 0){
            write(1, "550 Failed to access₩n", 2);
            write(connfd, "550 Failed to access₩n", MAX_BUF);
}
        /*****************************************************************************/

        /************* write message and send message ************/
        sleep(0.5);
        write(1, "200 Port command preformed successfully₩n", 41);
        write(connfd, "200 Port command preformed successfully₩n", MAX_BUF);
/*********************************************************/
```

```
        /************* write message and send message ************/
        sleep(0.5);
        if(buf[0] == 'N' && buf[1] == 'L' && buf[2] == 'S' && buf[3] == 'T') {
            write(1, "150 Opening data connection for directory list₩n", 48);
            write(connfd, "150 Opening data connection for directory list₩n", 48);
        }
        if(buf[0] == 'R' && buf[1] == 'E' && buf[2] == 'T' && buf[3] == 'R') {
            int cpn = cmd_process(buf, result_buff, logstr, cliaddr, IPs, user, connfd);
//get file
            write(data_sockfd, result_buff, MAX_BUF);     //send to client

            if(cpn == 7)       //faild to open
                continue;
            memset(buf, 0, MAX_BUF);
            read(connfd, buf, MAX_BUF);

            //send message to client
//write log file
            /*****************************************************************************/

            close(data_sockfd);
            continue;
        }
        /*******************************************************/
```

```
if(stor) {          //case of stor

        //get argumnet
        char arg[100];
        int fnn = 0;
        for(fnn = 5; fnn < strlen(buf); fnn++){
              arg[fnn-5] = buf[fnn];
        }
        arg[fnn] = '₩0';
        sleep(0.5);
        memset(result_buff, 0, MAX_BUF);
        sprintf(result_buff, "150 Opening binary/ascii mode data connection for %s₩n", arg);

        /*write log file*/

        memset(logstr, 0, 500);
        log_info(logstr, cliaddr, IPs, user);
        fprintf(logFile, "%s %s", logstr, result_buff);
        fflush(logFile);
        write(1, result_buff, strlen(result_buff));
        write(connfd, result_buff, MAX_BUF);
        memset(logstr, 0, 500);
```

```
            int cpn = cmd_process(buf, result_buff, logstr, cliaddr, IPs, user, data_sockfd);
//get file
            if(cpn == 10)
                exit(0);
            memset(buf, 0, MAX_BUF);
            read(connfd, buf, MAX_BUF);

            //send message to client
            sleep(0.5);
            write(1, "226 Complete transmission₩n", 27);
            write(connfd, "226 Complete transmission₩n", 27);

            //write log file
            memset(logstr, 0, 500);
            log_info(logstr, cliaddr, IPs, user);
            fprintf(logFile, "%s 226 Complete transmission₩n", logstr);
            fflush(logFile);
            memset(logstr, 0, 500);
            /*****************************************************************************/

            close(data_sockfd);
            continue;

        }
```

```
        /*write log file*/

        int cp = cmd_process(buf, result_buff, logstr, cliaddr, IPs, user, connfd);
//command processing
        int wn = write(data_sockfd, result_buff, MAX_BUF);     //send result
        if(wn < 0)          //if failed to send result
        {
            write(1, "550 Failed transmission₩n", 25);
            write(connfd, "550 Failed transmission₩n", 25);

            memset(logstr, 0, 500);
            log_info(logstr, cliaddr, IPs, user);
            fprintf(logFile, "%s 550 Failed transmission₩n", logstr);
            fflush(logFile);
            memset(logstr, 0, 500);
        }

        /********** write message and send message ("226 result successful") **********/
        memset(buf, 0, MAX_BUF);
        read(connfd, buf, MAX_BUF);
/*****************************************************************************/

        close(data_sockfd);
        continue;
```

```
    }
    int cp = cmd_process(buf, result_buff, logstr, cliaddr, IPs, user, connfd);   //command processing
    if(cp == 3) {
        memset(logstr, 0, 500);
        continue;
    }
    if(cp == 5) {     //case of QUIT
        close(connfd);          //close socket
        memset(logstr, 0, 500);
        log_info(logstr, cliaddr, IPs, user);
        fprintf(logFile, "%s %s", logstr, result_buff);
        fflush(logFile);
        memset(logstr, 0, 500);

        exit(0);
    }

    sleep(0.5);
    write(connfd, result_buff, MAX_BUF);       //write to socket
    //write log file
    memset(logstr, 0, 500);
    log_info(logstr, cliaddr, IPs, user);
    fprintf(logFile, "%s %s", logstr, result_buff);
```

```
        fflush(logFile);
        memset(logstr, 0, 500);
      }

      return 0;

}
```

<cli.c>

```
int main(int argc, char **argv)
{
	srand(time(NULL));

	char buf[MAX_BUF];

	char *hostport;
	struct sockaddr_in temp;
	int control_sockfd;

	/********************* prepare control connection **********************/
	control_sockfd = socket(AF_INET, SOCK_STREAM, 0);
	memset(&temp, 0, sizeof(temp));
	temp.sin_family = AF_INET;
	inet_pton(AF_INET, argv[1], &temp.sin_addr);
	//pointer to dotted-decimal string//
	strcpy(ip_str,inet_ntoa(temp.sin_addr));
	temp.sin_port = htons(atoi(argv[2]));
	connect(control_sockfd, (struct sockaddr*) &temp, sizeof(temp));	//connect with server
	/*************************************************************/
```

```
	log_in(control_sockfd); //log_in process


	/*
	*
	*		get Command from user and convert command
	*
	*/
	for(;;){
if(conv_cmd(buf, cmd_buf) < 0){
			write(1, "NO command₩n", 12);
		};

		//send FTP command

		if(!strcmp(buf, "quit"))		//type of quit
		{
			write(control_sockfd, cmd_buf, MAX_BUF);
			exit(0);
		}

		int get = 0;
		if(buf[0] == 'g' && buf[1] == 'e' && buf[2] == 't')	//type of get
			get = 1;
		int put = 0;
```

```
        if(buf[0] == 'p' && buf[1] == 'u' && buf[2] == 't')        //type of put
            put = 1;

        if(buf[0] == 'l' && buf[1] == 's' || buf[0] == 'g' && buf[1] == 'e' && buf[2] == 't'
|| put == 1) {
            sleep(0.5);
            write(control_sockfd, cmd_buf, MAX_BUF);

            /****************** make random port number *******************/
            int data_port = rand() %50001 + 10000;
            hostport = convert_addr_to_str(temp.sin_addr.s_addr, data_port);       //make
port command
            /**************************************************************/

            memset(buf, 0, MAX_BUF);
            read(control_sockfd, buf, MAX_BUF);


            /*
            *
            *                Connect data connection
            *
            */
            int data_listenfd, data_connfd;
            struct sockaddr_in data_servaddr, data_cliaddr;
```

```
data_listenfd = socket(PF_INET, SOCK_STREAM, 0);
int opt = 1;
setsockopt(data_listenfd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));

memset(&data_servaddr, 0, sizeof(data_servaddr));
data_servaddr.sin_family = AF_INET;
data_servaddr.sin_addr.s_addr = htonl(INADDR_ANY);//set address
data_servaddr.sin_port = htons(data_port);//set port

if(bind(data_listenfd, (struct sockaddr *)&data_servaddr, sizeof(data_servaddr))
< 0){   //bind socket
    printf("Server: Can't bind local address₩n");
    return 0;
}

listen(data_listenfd, 5);     //listen from client

int clilen= sizeof(data_cliaddr);

data_connfd = accept(data_listenfd, (struct sockaddr *) &data_cliaddr, &clilen);


/********get message from control connection("200 PORT successful")********/
```

```
memset(buf, 0, MAX_BUF);
read(control_sockfd, buf, MAX_BUF);
buf[strlen(buf)] = '₩0';
write(1, buf, strlen(buf));
/*************************************************************************/

sleep(0.01);

/********get message from control connection("150 opening successful")********/

int total_bytes = strlen(buf);
/****************** get result from data_connfd *********************/

/*
*          case of PUT
*/
if(put) {
      char filename[50];
      int fi = 5;
      for(fi; fi<strlen(cmd_buf); fi++){

            filename[fi-5] = cmd_buf[fi];
```

```
                }
                filename[fi-1] = '₩0';
                memset(buf, 0, MAX_BUF);

                //get filename
                char arg[MAX_BUF];
                snprintf(arg, sizeof(arg), "%s/%s", getcwd(NULL, 0), filename);

                //open file
                FILE*temp_fp;
                temp_fp = fopen(arg, "r");


                if(temp_fp == NULL){      //failed to open
}
                else{                  //succeeded to open
}
                }

                sleep(0.5);
                write(data_connfd, buf, MAX_BUF);
```

```
                }
                else if(get) {          //case of get
}
                        filename[fi-1] = '₩0';
                        memset(buf, 0, MAX_BUF);
                        int bytes_read;
                        bytes_read = read(data_connfd, buf, MAX_BUF);
                        buf[strlen(buf)] = '₩0';
                        if(buf[0] == 'N'&& buf[1] == 'o' && buf[2] == 'F'&& buf[3] == '#'){ //failed to open file
                                write(1, "Failed to open₩n", 16);
                                memset(buf, 0, MAX_BUF);
                                memset(cmd_buf, 0, 100);
                                memset(filename, 0, 50);
                                exit(0);
                        }
                        FILE*temp_fp = fopen(filename, "w");
                        fprintf(temp_fp, "%s", buf);
                        fflush(temp_fp);
                }
                else{           //succeeded to open file
                        memset(buf, 0, MAX_BUF);
                        read(data_connfd, buf, MAX_BUF);
                        buf[strlen(buf)] = '₩0';
```

```
                write(1, buf, strlen(buf));
            }
            /*********************************************************************/



            /************ receive message from control connection("226 Result successful")
**************/
            write(control_sockfd, "dummy₩n", MAX_BUF);
/*************************************************************************************************/

            printf("OK. %d bytes is received₩n", total_bytes);
}



        //normal case of command -> get reply message
if(cmd_buf[0] == 'R' && cmd_buf[1] == 'N' && cmd_buf[2] == 'F' && cmd_buf[3] ==
'R'){     //case of rnfr, it should get more buf
            memset(buf, 0, MAX_BUF);
            read(control_sockfd, buf, MAX_BUF);



            buf[strlen(buf)] = '₩0';
            write(1, buf, strlen(buf));
        }
```

```
            memset(buf, 0, MAX_BUF);
            memset(cmd_buf, 0, 100);

        }

        return 0;

}
```

결과화면

[최초접속]

- srv

```
kw2019202032@ubuntu:~/Assignment3_3$ ./srv 10000
** Client is connected **
sswlab.kw.ac.kr FTP server (version myftp [1.0] Thu Jun  6 03:50:29 2024) ready.
** User is trying to log-in (1/3) **
```

-cli

```
kw2019202032@ubuntu:~/temp$ ./cli 127.0.0.1 10000
Connected to sswlab.kw.ac.kr
sswlab.kw.ac.kr FTP server (version myftp [1.0] Thu Jun  6 03:50:29 2024) ready
.
Input Name :
```

- Log file

```
Thu Jun  6 03:50:29 2024 Server is started
sswlab.kw.ac.kr FTP server (version myftp [1.0] Thu Jun  6 03:50:29 2024) ready.
```

[로그인]

틀린 Id-> 틀린 passwd -> 로그인 성공 순으로 진행

- srv

```
kw2019202032@ubuntu:~/Assignment3_3$ ./srv 10000
** Client is connected **
sswlab.kw.ac.kr FTP server (version myftp [1.0] Thu Jun  6 03:50:29 2024) ready.
** User is trying to log-in (1/3) **
430 Invalid username of password
** User is trying to log-in (2/3) **
430 Invalid username of password
** User is trying to log-in (3/3) **
331 Password is required for test1
230 User test1 logged in
** Success to log-in **
```

- cli

```
kw2019202032@ubuntu:~/temp$ ./cli 127.0.0.1 10000
Connected to sswlab.kw.ac.kr
sswlab.kw.ac.kr FTP server (version myftp [1.0] Thu Jun  6 03:50:29 2024) ready
.
Input Name : noid
430 Invalid username of password
Input Name : trying to fail
430 Invalid username of password
Input Name : test1
331 Password is required for test1
Input Password :
** User 'test1' logged in **
230 User test1 logged in
```

-logfile

```
1   Thu Jun  6 03:50:29 2024 Server is started
2   sswlab.kw.ac.kr FTP server (version myftp [1.0] Thu Jun  6 03:50:29 2024) ready.
3   Thu Jun  6 03:52:35 2024 [127:24721] noid 430 Invalid username of password
4   Thu Jun  6 03:52:49 2024 [127:24721] trying to fail 430 Invalid username of password
5   Thu Jun  6 03:52:51 2024 [127:24721] test1 331 Password is required for test1
6   Thu Jun  6 03:52:53 2024 [127:24721] test1 230 User test1 logged in
7
```

[cdup] + [pwd] + [ls]

-cli

```
230 User test1 logged in
cd ..
250 CWD command succeeds
pwd
257 "/home/kw2019202032" is current directory
ls
200 Port command preformed successfully
150 Opening data connection for directory list
Assignment1_1/  Assignment1_2/  Assignment1_2_A_2019202032_이상현.pdf   Assignm
ent1_2_A_2019202032_이상현.tar.gz       Assignment1_3/
Assignment1_3_A_2019202032_이상현.pdf   Assignment1_3_A_2019202032_이상현.tar.g
z       Assignment2_1/  Assignment2_2/  Assignment2_3/
Assignment3_1/  Assignment3_2/  Assignment3_3/  Assignments/    cli
cli.c   cli.out Desktop/        Documents/      Downloads/
kw2019202032_ls kw2019202032_ls.c       Makefile        Music/  Pictures/
practice/       Public/ snap/   srv     srv.c
system_programming_test/        temp/   Templates/      Videos/
226 Complete transmission
OK. 47 bytes is received
```

-srv

```
CDUP
250 CWD command succeeds
PWD
257 "/home/kw2019202032" is current directory
PORT 127,0,0,1,151,179
200 Port command preformed successfully
150 Opening data connection for directory list
226 Complete transmission
```

-logfile

```
1   Thu Jun  6 03:50:29 2024 Server is started
2   sswlab.kw.ac.kr FTP server (version myftp [1.0] Thu Jun  6 03:50:29 2024) ready.
3   Thu Jun  6 03:52:35 2024 [127:24721] noid 430 Invalid username of password
4   Thu Jun  6 03:52:49 2024 [127:24721] trying to fail 430 Invalid username of password
5   Thu Jun  6 03:52:51 2024 [127:24721] test1 331 Password is required for test1
6   Thu Jun  6 03:52:53 2024 [127:24721] test1 230 User test1 logged in
7   Thu Jun  6 03:55:20 2024 [127:24721] test1 250 CWD command succeeds
8   Thu Jun  6 03:55:33 2024 [127:24721] test1 257 "/home/kw2019202032" is current directory
9   Thu Jun  6 03:55:38 2024 [127:24721] test1 200 Port command preformed successfully
10  Thu Jun  6 03:55:38 2024 [127:24721] test1 150 Opening data connection for directory list
11  Thu Jun  6 03:55:38 2024 [127:24721] test1 226 Complete transmission
```

이전 directory 로 이동 후 pwd 하고 ls 출력

[cd Assignment3_3] + [pwd] + [ls -al] + 이후 cd temp 로 이동

- cli

```
cd Assignment3_3
250 CWD command succeeds
pwd
257 "/home/kw2019202032/Assignment3_3" is current directory
ls -al
200 Port command preformed successfully
150 Opening data connection for directory list
drwxrwxr-x 3 kw2019202032 kw2019202032 4096 Jun 6 03:4903:49 ./
drwxr-xr-x 33 kw2019202032 kw2019202032 4096 Jun 6 03:4903:49 ../
drwxrwxr-x 2 kw2019202032 kw2019202032 4096 Jun 3 07:5307:53 .vscode/
-rw-rw-r-- 1 kw2019202032 kw2019202032 11 Jun 5 13:5513:55 access.txt
-rwxrwxr-x 1 kw2019202032 kw2019202032 26552 Jun 6 03:4903:49 cli
-rw-rw-r-- 1 kw2019202032 kw2019202032 20082 Jun 6 03:4703:47 cli.c
-rw-rw-r-- 1 kw2019202032 kw2019202032 1173 Jun 6 03:5503:55 logfile
-rwxrw-rw- 1 kw2019202032 kw2019202032 162 Apr 17 07:2107:21 Makefile
-rw-rw-r-- 1 kw2019202032 kw2019202032 95 May 16 22:4422:44 passwd
-rwxrwxr-x 1 kw2019202032 kw2019202032 48368 Jun 6 03:4903:49 srv
-rw-rw-r-- 1 kw2019202032 kw2019202032 61116 Jun 6 03:4803:48 srv.c
-rw-rw-r-- 1 kw2019202032 kw2019202032 11 Jun 6 02:5902:59 temp.txt
226 Complete transmission
OK. 47 bytes is received
cd /home/kw2019202032/temp
250 CWD command succeeds
pwd
257 "/home/kw2019202032/temp" is current directory
```

-srv

```
250 CWD command succeeds
PWD
257 "/home/kw2019202032/Assignment3_3" is current directory
PORT 127,0,0,1,234,35
200 Port command preformed successfully
150 Opening data connection for directory list
226 Complete transmission
CWD
250 CWD command succeeds
PWD
257 "/home/kw2019202032/temp" is current directory
```

-logfile

```
Thu Jun  6 03:50:29 2024 Server is started
sswlab.kw.ac.kr FTP server (version myftp [1.0] Thu Jun  6 03:50:29 2024) ready.
Thu Jun  6 03:52:35 2024 [127:24721] noid 430 Invalid username of password
Thu Jun  6 03:52:49 2024 [127:24721] trying to fail 430 Invalid username of password
Thu Jun  6 03:52:51 2024 [127:24721] test1 331 Password is required for test1
Thu Jun  6 03:52:53 2024 [127:24721] test1 230 User test1 logged in
Thu Jun  6 03:55:20 2024 [127:24721] test1 250 CWD command succeeds
Thu Jun  6 03:55:33 2024 [127:24721] test1 257 "/home/kw2019202032" is current directory
Thu Jun  6 03:55:38 2024 [127:24721] test1 200 Port command preformed successfully
Thu Jun  6 03:55:38 2024 [127:24721] test1 150 Opening data connection for directory list
Thu Jun  6 03:55:38 2024 [127:24721] test1 226 Complete transmission
Thu Jun  6 03:55:44 2024 [127:24721] test1 250 CWD command succeeds
Thu Jun  6 03:55:45 2024 [127:24721] test1 257 "/home/kw2019202032/Assignment3_3" is current directory
Thu Jun  6 03:55:48 2024 [127:24721] test1 200 Port command preformed successfully
Thu Jun  6 03:55:48 2024 [127:24721] test1 150 Opening data connection for directory list
Thu Jun  6 03:55:48 2024 [127:24721] test1 226 Complete transmission
Thu Jun  6 03:59:13 2024 [127:24721] test1 250 CWD command succeeds
Thu Jun  6 03:59:15 2024 [127:24721] test1 257 "/home/kw2019202032/temp" is current directory
```

[bin] + [ascii] + [type binary] + [type ascii]

- srv

```
201 Type set to I
201 Type set to A
201 Type set to I
201 Type set to A
```

-cli

```
bin
201 Type set to I
ascii
201 Type set to A
type binary
201 Type set to I
type ascii
201 Type set to A
```

-log file

```
Thu Jun  6 03:50:29 2024 Server is started
sswlab.kw.ac.kr FTP server (version myftp [1.0] Thu Jun  6 03:50:29 2024) ready.
Thu Jun  6 03:52:35 2024 [127:24721] noid 430 Invalid username of password
Thu Jun  6 03:52:49 2024 [127:24721] trying to fail 430 Invalid username of password
Thu Jun  6 03:52:51 2024 [127:24721] test1 331 Password is required for test1
Thu Jun  6 03:52:53 2024 [127:24721] test1 230 User test1 logged in
Thu Jun  6 03:55:20 2024 [127:24721] test1 250 CWD command succeeds
Thu Jun  6 03:55:33 2024 [127:24721] test1 257 "/home/kw2019202032" is current directory
Thu Jun  6 03:55:38 2024 [127:24721] test1 200 Port command preformed successfully
Thu Jun  6 03:55:38 2024 [127:24721] test1 150 Opening data connection for directory list
Thu Jun  6 03:55:38 2024 [127:24721] test1 226 Complete transmission
Thu Jun  6 03:55:44 2024 [127:24721] test1 250 CWD command succeeds
Thu Jun  6 03:55:45 2024 [127:24721] test1 257 "/home/kw2019202032/Assignment3_3" is current directory
Thu Jun  6 03:55:48 2024 [127:24721] test1 200 Port command preformed successfully
Thu Jun  6 03:55:48 2024 [127:24721] test1 150 Opening data connection for directory list
Thu Jun  6 03:55:48 2024 [127:24721] test1 226 Complete transmission
Thu Jun  6 03:59:13 2024 [127:24721] test1 250 CWD command succeeds
Thu Jun  6 03:59:15 2024 [127:24721] test1 257 "/home/kw2019202032/temp" is current directory
Thu Jun  6 04:01:09 2024 [127:24721] test1 200 Port command preformed successfully
Thu Jun  6 04:01:09 2024 [127:24721] test1 150 Opening data connection for directory list
Thu Jun  6 04:01:10 2024 [127:24721] test1 226 Complete transmission
Thu Jun  6 04:01:18 2024 [127:24721] test1 201 Type set to I
Thu Jun  6 04:01:22 2024 [127:24721] test1 201 Type set to A
Thu Jun  6 04:01:30 2024 [127:24721] test1 201 Type set to I
Thu Jun  6 04:01:34 2024 [127:24721] test1 201 Type set to A

```

[delete]

a.c 파일을 삭제하고 ls 를 통해 확인하여 a.c 파일이 삭제된 것을 확인한다.

-cli

```
ls
200 Port command preformed successfully
150 Opening data connection for directory list
a.c     a.out   cli     cli.c   kw2019202032_ls
kw2019202032_ls.c       sdf
226 Complete transmission
OK. 47 bytes is received
delete a.c
250 DELE command performed successfully.
ls
200 Port command preformed successfully
150 Opening data connection for directory list
a.out   cli     cli.c   kw2019202032_ls kw2019202032_ls.c
sdf
```

-srv

```
PORT 127,0,0,1,67,105
200 Port command preformed successfully
150 Opening data connection for directory list
226 Complete transmission
DELE
250 DELE command performed successfully.
PORT 127,0,0,1,160,189
200 Port command preformed successfully
150 Opening data connection for directory list
226 Complete transmission
```

-logfile

```
Thu Jun  6 04:01:10 2024 [127:24721] test1 226 Complete transmission
Thu Jun  6 04:01:18 2024 [127:24721] test1 201 Type set to I
Thu Jun  6 04:01:22 2024 [127:24721] test1 201 Type set to A
Thu Jun  6 04:01:30 2024 [127:24721] test1 201 Type set to I
Thu Jun  6 04:01:34 2024 [127:24721] test1 201 Type set to A
Thu Jun  6 04:03:31 2024 [127:24721] test1 200 Port command preformed successfully
Thu Jun  6 04:03:31 2024 [127:24721] test1 150 Opening data connection for directory list
Thu Jun  6 04:03:32 2024 [127:24721] test1 226 Complete transmission
Thu Jun  6 04:03:37 2024 [127:24721] test1 250 DELE command performed successfully.
Thu Jun  6 04:03:40 2024 [127:24721] test1 200 Port command preformed successfully
Thu Jun  6 04:03:40 2024 [127:24721] test1 150 Opening data connection for directory list
Thu Jun  6 04:03:40 2024 [127:24721] test1 226 Complete transmission
```

[mkdir] + [rmdir]

Mkdir 을 통해 temp 라는 이름의 directory 를 만들고 해당 디렉토리를 삭제한다.

-cli

```
mkdir temp
250 MKD command performed successfully.
ls
200 Port command preformed successfully
150 Opening data connection for directory list
a.out   cli     cli.c   kw2019202032_ls kw2019202032_ls.c
sdf     temp/
```

```
rmdir temp
250 RMD command performed successfully.
ls
200 Port command preformed successfully
150 Opening data connection for directory list
a.out   cli     cli.c   kw2019202032_ls kw2019202032_ls.c
sdf
226 Complete transmission
```

-srv

```
MKD
250 MKD command performed successfully.
PORT 127,0,0,1,179,170
200 Port command preformed successfully
150 Opening data connection for directory list
226 Complete transmission
```

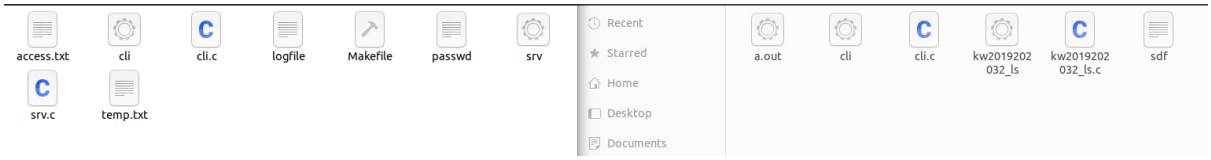
```
RMD
250 RMD command performed successfully.
PORT 127,0,0,1,97,247
200 Port command preformed successfully
150 Opening data connection for directory list
226 Complete transmission
```

-logfile

```
33  Thu Jun  6 04:06:11 2024 [127:24721] test1 250 MKD command performed successfully.
34  Thu Jun  6 04:06:13 2024 [127:24721] test1 200 Port command preformed successfully
35  Thu Jun  6 04:06:13 2024 [127:24721] test1 150 Opening data connection for directory list
36  Thu Jun  6 04:06:14 2024 [127:24721] test1 226 Complete transmission
37  Thu Jun  6 04:06:19 2024 [127:24721] test1 250 RMD command performed successfully.
38  Thu Jun  6 04:06:21 2024 [127:24721] test1 200 Port command preformed successfully
39  Thu Jun  6 04:06:21 2024 [127:24721] test1 150 Opening data connection for directory list
40  Thu Jun  6 04:06:21 2024 [127:24721] test1 226 Complete transmission
```

[get]

Assignment3_3 디렉토리에 있는 temp.txt 를 temp 디렉토리로 이동시킨다



현재 temp 디렉토리에 temp.txt 파일이 없는 것을 확인할 수 있다.

-cli

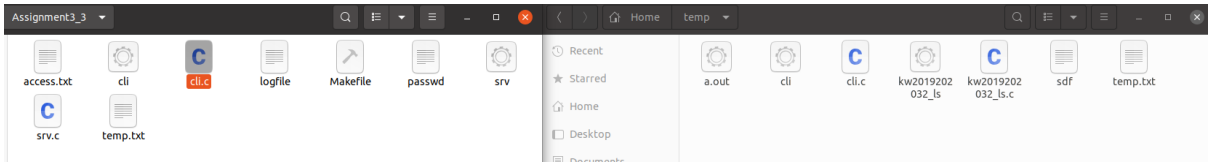
```
get temp.txt
200 Port command preformed successfully
150 Opening binary/ascii mode data connection for temp.txt
226 Complete transmission
OK. 59 bytes is received
```

-srv

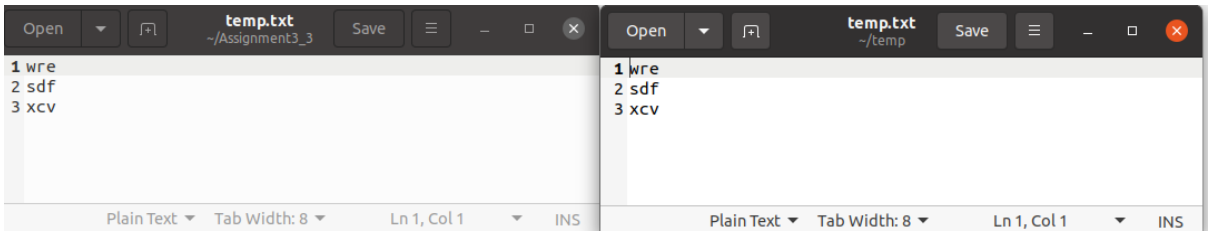
```
PORT 127,0,0,1,191,151
200 Port command preformed successfully
150 Opening binary/ascii mode data connection for temp.txt
226 Complete transmission
```

-logfile

```
1  Thu Jun  6 05:22:51 2024 Server is started
2  sswlab.kw.ac.kr FTP server (version myftp [1.0] Thu Jun  6 05:22:51 2024) ready.
3  Thu Jun  6 05:22:52 2024 [127:39647] test1 331 Password is required for test1
4  Thu Jun  6 05:22:57 2024 [127:39647] test1 230 User test1 logged in
5  Thu Jun  6 05:23:03 2024 [127:39647] test1 200 Port command preformed successfully
6  Thu Jun  6 05:23:03 2024 [127:39647] test1 150 Opening binary/ascii mode data connection for temp.txt
7  Thu Jun  6 05:23:03 2024 [127:39647] test1 226 Complete transmission
8
```



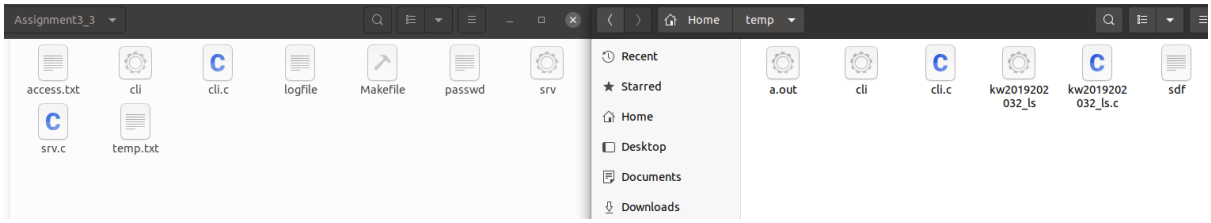
Temp directory 에 temp.txt 파일이 생긴 것을 확인할 수 있다.



두 개의 파일의 내용이 같은 것을 확인할 수 있다.

[put]

Temp 디렉토리에 있는 sdf 파일을 Assignment3_3 으로 이동시킨다.



현재 Assignment3_3 directory 에 sdf 파일이 없는 것을 확인할 수 있다.

-cli

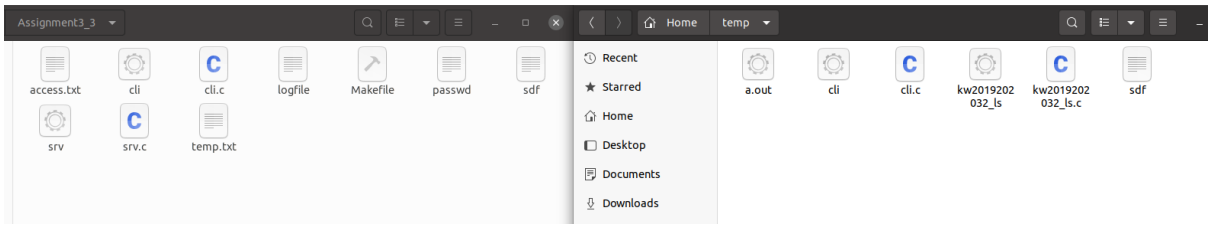
```
put sdf
200 Port command preformed successfully
150 Opening binary/ascii mode data connection for sdf
226 Complete transmission
OK. 54 bytes is received
```

-srv

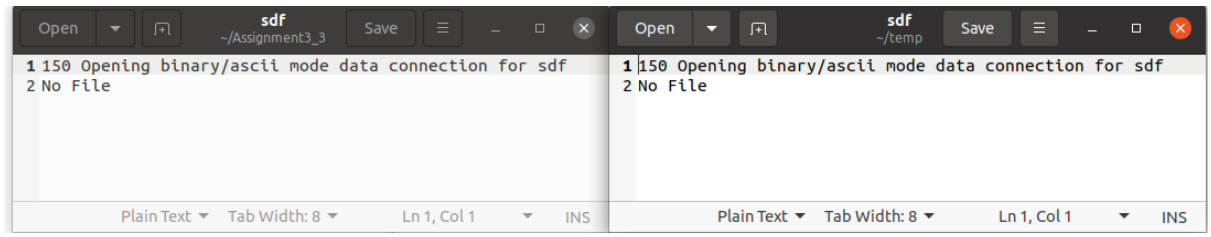
```
PORT 127,0,0,1,73,172
200 Port command preformed successfully
150 Opening binary/ascii mode data connection for sdf
226 Complete transmission
```

-logfile

```
1  Thu Jun  6 05:04:41 2024 Server is started
2  sswlab.kw.ac.kr FTP server (version myftp [1.0] Thu Jun  6 05:04:41 2024) ready.
3  Thu Jun  6 05:04:44 2024 [127:56530] test1 331 Password is required for test1
4  Thu Jun  6 05:04:45 2024 [127:56530] test1 230 User test1 logged in
5  Thu Jun  6 05:04:49 2024 [127:56530] test1 200 Port command preformed successfully
6  Thu Jun  6 05:04:49 2024 [127:56530] test1 150 Opening binary/ascii mode data connection for sdf
7  Thu Jun  6 05:04:49 2024 [127:56530] test1 226 Complete transmission
8
```



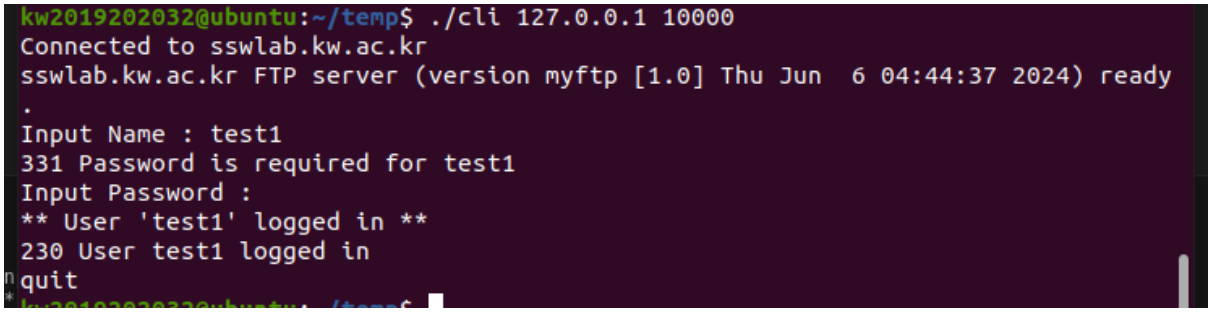
이후 파일이 생성된 것을 확인할 수 있다.



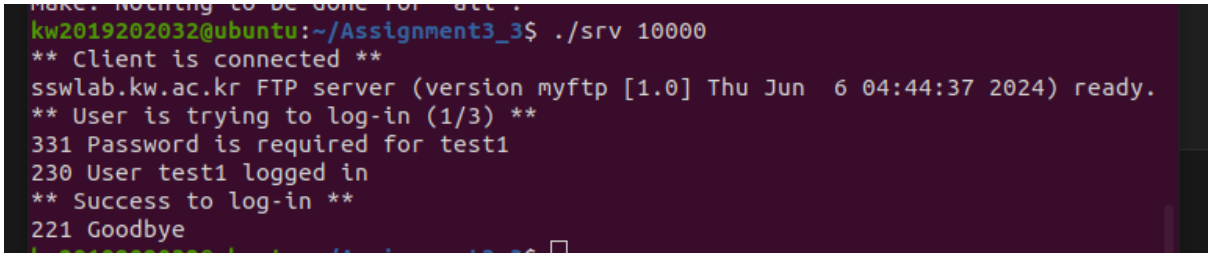
두 파일을 비교했을 때 내용이 같은 것을 확인할 수 있다.

[quit]

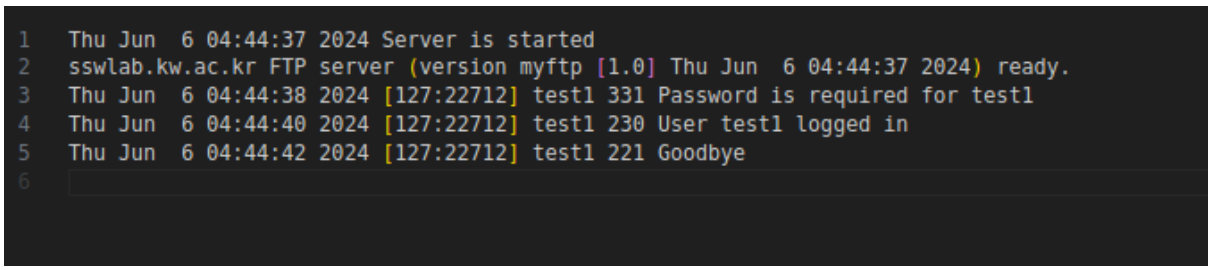
-cli



-srv



-logfile



고찰

해당 과제를 진행하는 과정에서 가장 어려웠던 부분은 put 과 get 명령어였다. 처음에 파일을 어떤 식으로 전달해 줄 수 있는지 몰라서 어떻게 구현해야하는지 잘 몰랐는데, 전달하고자 하는 파일의 이름을 전달해 반대쪽 directory 에서 해당 파일을 만들고, 다시 반대쪽에서는 파일을 열어서 내용을 buiffer 에 저장한 뒤, buffer 를 반대쪽으로 보내고, 수신한 socket 에서 해당 내용을 파일에 작성하는 식으로 해당 명령어를 구현하였다. 이 과정에서 파일이 열리지 않거나 전달하고자 하는 파일이 존재하지 않는 경우에는 파일 포인터가 null 인지를 확인하여 예외처리되도록 하였다. 이 과제를 통해 소켓을 단방향이

아니라 양방향으로 통신하는 방법에 대해 배울 수 있었고, 또한 파일의 정보 뿐만 아니라 파일 자체를 전달하는 방법에 대해서도 배울 수 있었다.

Reference

강의자료만 참고하였습니다.