

시스템 프로그래밍 실습

[Assignment3-2]

Class : [D]

Professor : [최상호교수님]

Student ID : [2020202060]

Name : [홍왕기]

Introduction

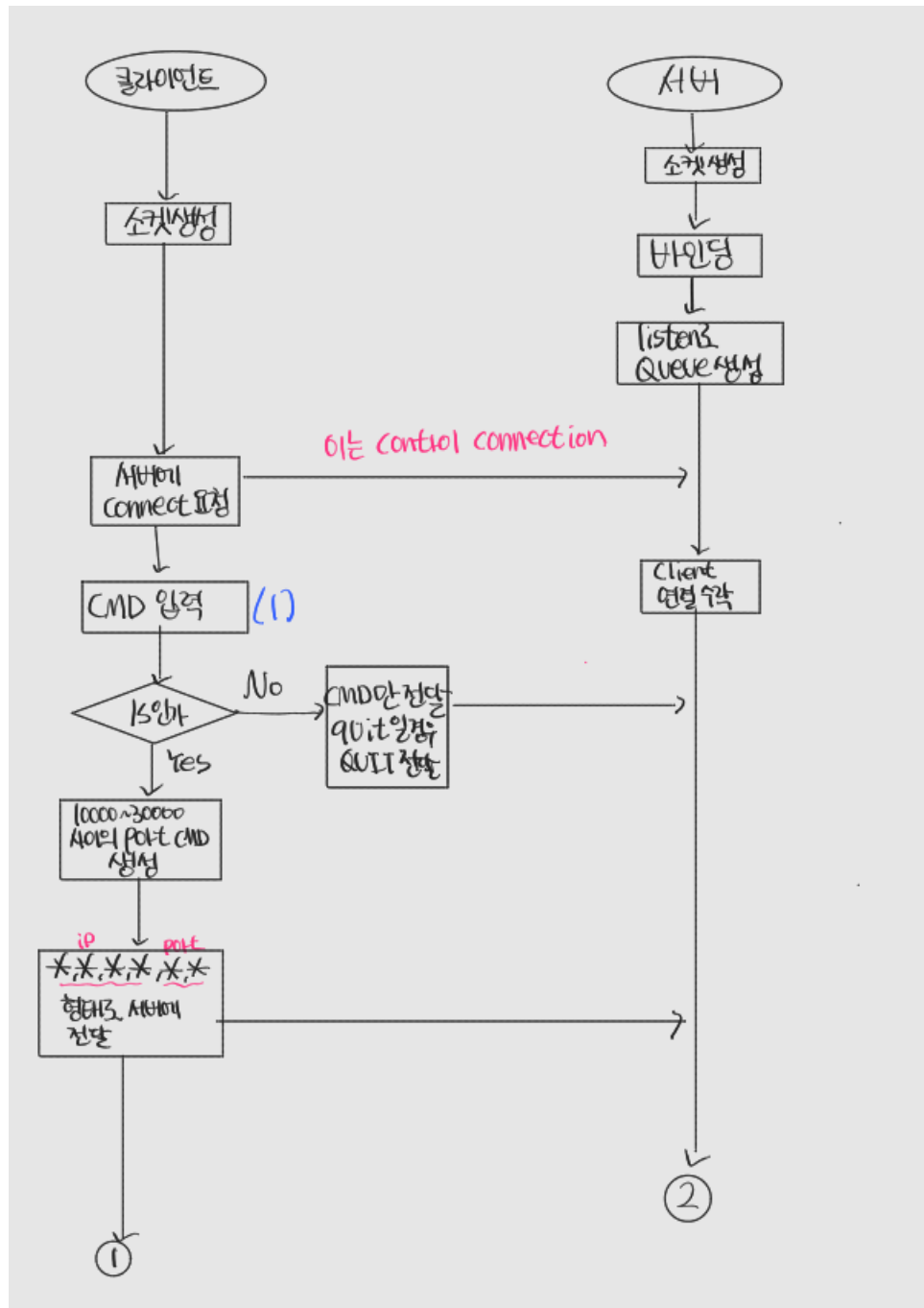
이 프로젝트는 데이터 통신, 소켓 프로그래밍, 그리고 Port command 의 통합을 통해 최종적으로 안정적이고 신뢰할 수 있는 FTP(파일 전송 프로토콜) 서버를 구축하는 것을 목표로 하고 있다.

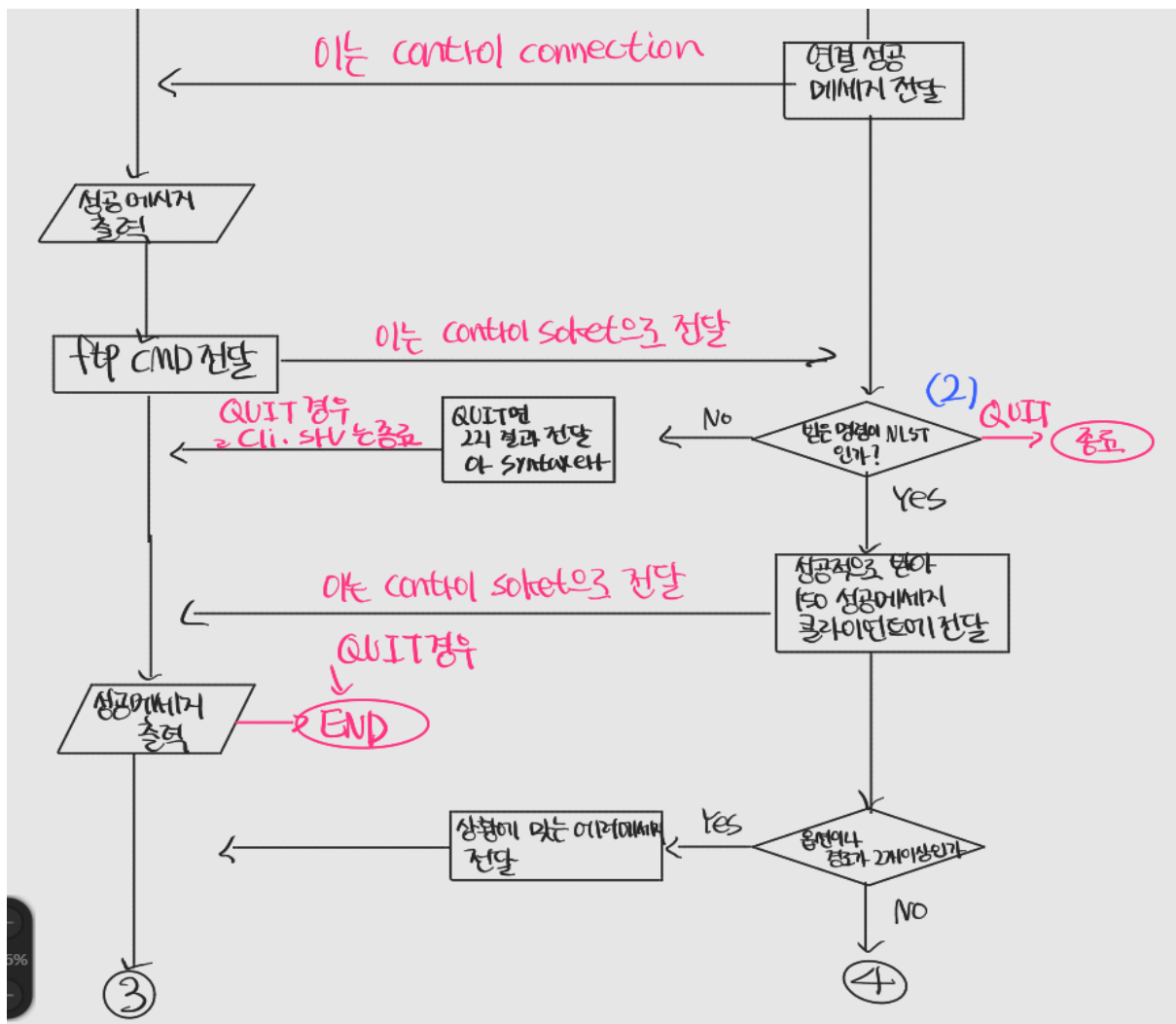
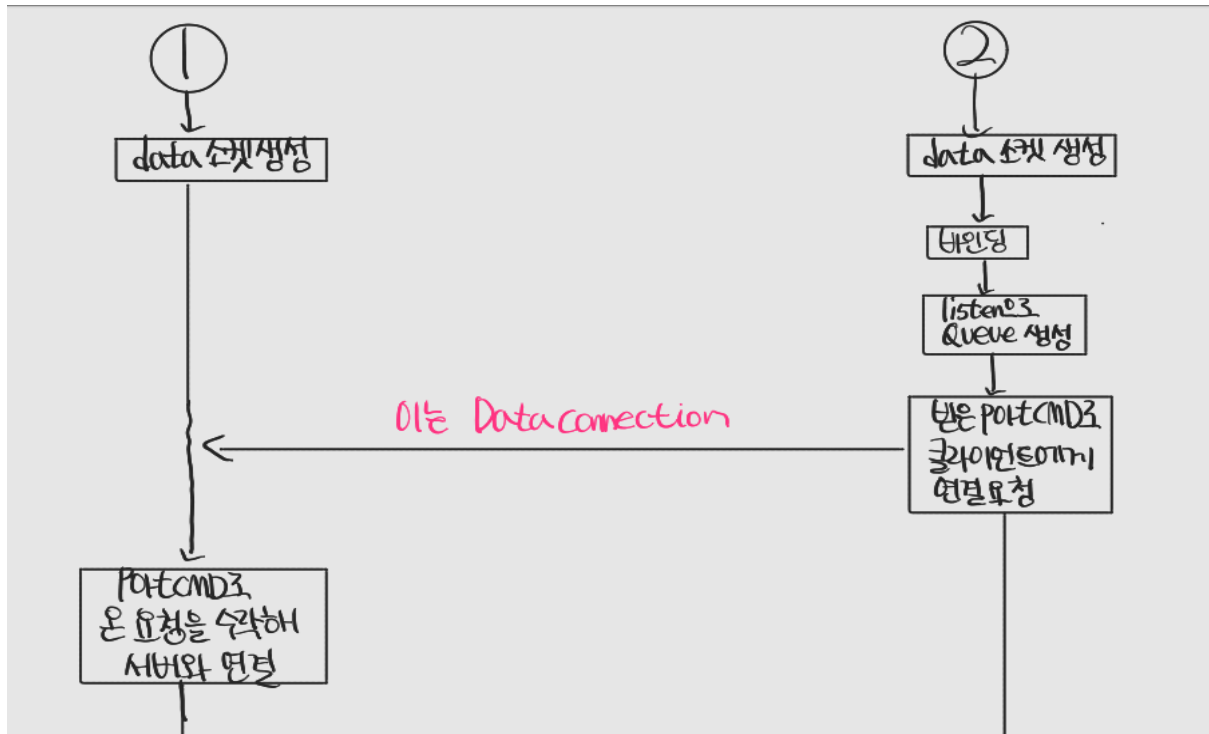
우선, 소켓 프로그래밍은 이 프로젝트의 기반이 된다. 이는 현대적인 데이터 통신에 있어 가장 효과적인 방법으로 간주된다. 소켓을 이용하여 클라이언트와 서버 간에 안정적이고 빠른 데이터 교환을 가능하게 함으로써, 다양한 응용 분야에서 중요한 역할을 할 것이다. FTP 서버는 파일 전송을 중심으로 하므로, 실시간 데이터 전송이 중요한 요소다. 뿐만 아니라, Port command 를 활용하여 연결을 관리하는 것도 이 프로젝트의 핵심 기능 중 하나이다. 이를 통해 서버가 클라이언트에게 연결 요청을 보내고 데이터를 안정적으로 전송함으로써 네트워크 통신의 안정성을 보장하고 데이터의 정확성을 유지할 수 있다. 더불어, 보안 기능을 추가하여 데이터의 무결성과 기밀성을 강화할 수 있을 것이다. 특히 FTP 서버는 파일 전송과 관련된 보안 문제에 대해 특히 주의를 기울여 구현해야 한다.

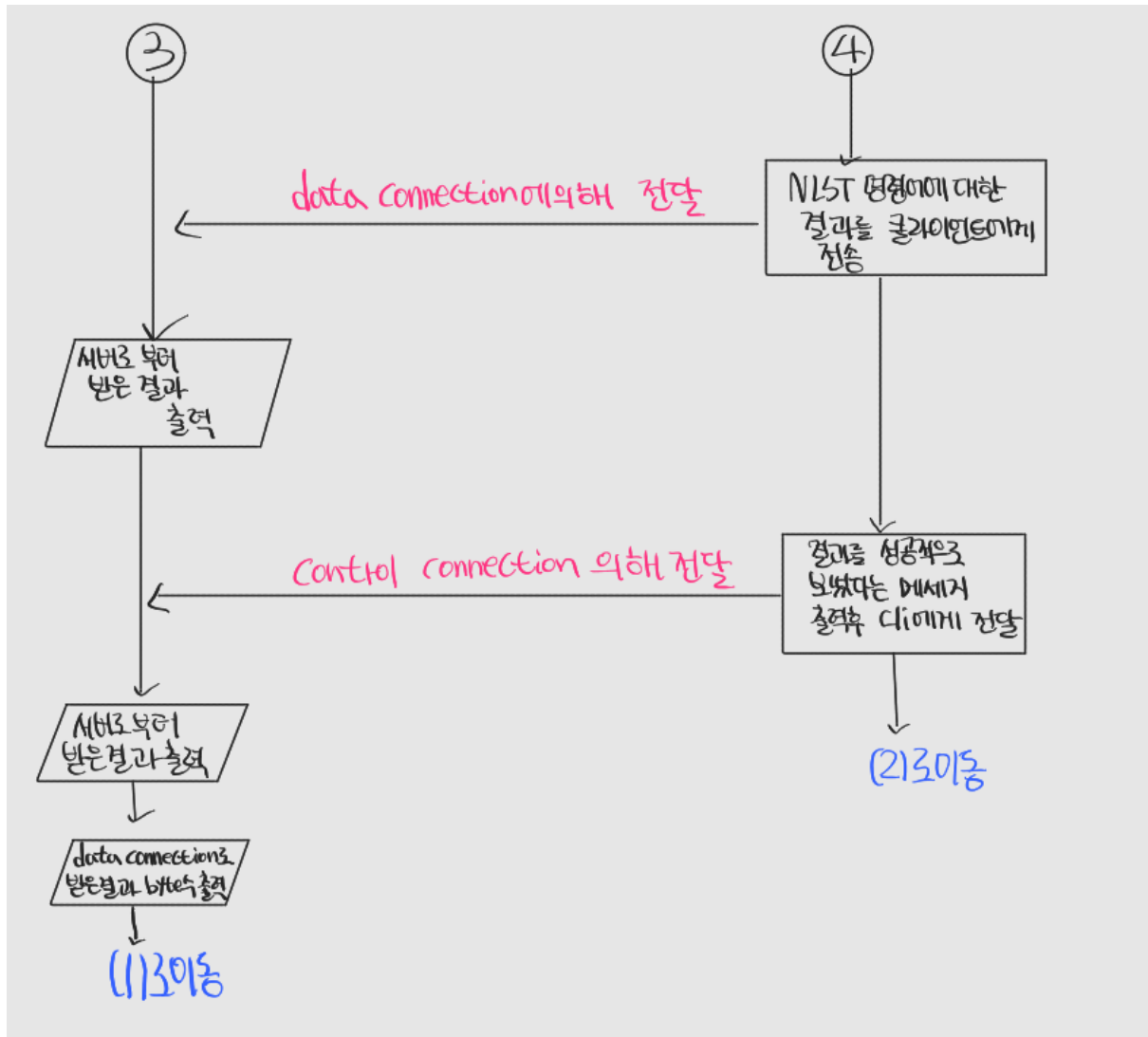
높은 이식성과 확장성을 갖춘 소켓 프로그래밍은 다양한 플랫폼과 언어에서 쉽게 구현될 수 있다. 이는 프로젝트의 유연성을 높이고 새로운 요구 사항에 대응하는 데 큰 장점이 된다. 또한, FTP 서버는 여러 사용자 및 다양한 파일 형식을 지원해야 하므로, 이식성과 확장성은 핵심적인 요소이다.

이 프로젝트는 안정성, 신뢰성, 그리고 보안성을 갖춘 FTP 서버를 구축함으로써 데이터 전송 및 파일 공유에 대한 필요성을 충족시킬 것이다.

Flow chart







Pseudo code

cli.c

```
5 void conv_cmd(char *input, char *output) {
6
7     char *token
8     char *tokens[BUF_SIZE]
9     int num_tokens = 0
10    char input_copy[BUF_SIZE]
11    strcpy(input_copy, input)
12
13    Tokenize the input string based on whitespace
14
15
16    if (num_tokens > 0 && token[0] == '1s'){
17        output = NLST
18        Attaching to the remaining factor output
19    }
20    else if(token[0]==quit){
21        | output= QUIT
22        | Attaching to the remaining factor output
23    }
24
25    else {
26        | Attaching to the remaining factor output
27    }
28 }
29
```

```
5
6 ✓ char* convert_str_to_addr(char *str, unsigned int *port) {
7     static char addr[32]
8     char copystr[MAX_BUFF]
9     char ip_part[4][4]
10    int ip[4], p1, p2
11
12    copystr=str
13
14    *,*,*,*, 형식에 대한 str을 ,기준으로 ip 배열에 파싱
15    남은 *,*를 ,을 기준으로 p1 p2로 파싱
16
17    파싱한 것들을 PORT *,*,*,*,*,*로 합쳐 addr에 저장
18
19    return addr
20 }
```

```

int main(int argc, char **argv) {
    int sockfd, data_sockfd, data_conn, n, port
    struct sockaddr_in servaddr, cliaddr, data_servaddr
    //MAX BUFF는 1024
    char buff[MAX_BUFF], ftp_cmd[MAX_BUFF], temp[MAX_BUFF], *host_ip
    socklen_t len
    unsigned int port_num

    if(인자가 3개가 아닌 경우){
        print "Usage: * <server_ip> <server_port>"
        eixt(fail)
    }

    sockfd=make_socket->AF_INET,SOCK_STREAM,0
    servaddr family=AF_INET
    servaddr addr= inet_addr(argv[1])
    servaddr port= 포트를 byte order로 변경

    서버와 연결 만약 오류시 print errno-> exit

```

```

while(1){
    print "> "
    cmd 입력
    conv_cmd를 통해 ftp command로 변경

    if(ftp cmd가 NLST일 경우){
        data_socket 생성
        생성 오류시 print errno->exit

        10000~30000사이의 port 난수 생성
        //data connetion 생성
        cliaddr family=AF_INET
        cliaddr addr= htonl(INADDR_ANY)
        cliaddr port= 포트를 byte order로 변경
        바인딩 후 만약 실패시 print errno->exit

        listen으로 queue생성 크기는 1.

        로컬 주소를 가져와 cliaddr에 저장

        IP를 .대신 ,로 변경
        난수 port에 2진수를 절반으로 나누어 port1 ,2에 저장

        host_ip=ip,port를 convert_str_to_addr함수에 전달
        host ip를 서버에 전달
        서버에게 buff 를 받아 출력

        서버에게 ftp command 전달
        서버에게 성공 메시지 전달받아 출력

        서버에 연결 요청을 수락== data_conn
        만약 수락 실패시 errno 출력 후 종료

```

```

    }
    else{
        서버에게 ftp_cmd 전달 //ftp cmd가 아닌 경우
        전달을 실패하였다면 errno출력 후 종료

        서버로 부터 결과를 받아서 출력
        if(ftp_cmd==QUIT){
            성공 메시지 read후 출력함
            break
        }
    }
}
}
소켓을 닫은 후 return;
}

```


srv.c

```
//이는 받은 경로가 2개 이상일 경우에 대한 변수
int path_count = 0;

void list_directory(int data_conn, const char *path) {
    if(받은 ls 명령어에 옵션이 있는 경우){
        print "550 Invalid option."
        "550 Invalid option"를 클라이언트에게 전달
        return
    }

    if(만약 경로가 두개 이상일 경우){
        print "501 Multiple directory paths are not supported."
        "501 Multiple directory paths are not supported." 를 클라이언트에게 전달
        return
    }

    if(만약 경로가 디렉토리가 아닌 경우){
        print "550 Failed to get directory information."
        "550 Failed to get directory information."를 클라이언트에게 전달
        return
    }

    if(경로가 디렉토리일 경우){
        디렉토리 오픈
        if(디렉토리를 열지 못했을 경우){
            print "550 Failed to open directory"
            "550 Failed to open directory" 를 클라이언트에게 전달
        }
        디렉토리를 순차적으로 읽어 buufer에 저장
        디렉토리 닫기

        buff를 dataconnetion을 통해 클라이언트에게 전달
    }
    else{
        print "550 Specified path is not a directory."
        "550 Specified path is not a directory."를 클라이언트에게 전달
    }

    if(만약 접근 권한이 없는 경우){
        print "550 Access to the path is denied."
        "550 Access to the path is denied." 를 클라이언트에게 전달
    }
}
```

```
char* convert_str_to_addr(const char *str, unsigned int *port) {
    받은 portcmd
    PORT *,*,*,*,*,* 형태를
    ,를 기준으로 IP와 port 1 2 로 파싱

    받은 port 1 2를 결합
    ex-> 127,0,0,1,15000 형태를 addr에 저장
    return addr
}
```

```

71 int main(int argc, char **argv) {
72     int sockfd, connfd, len, data_sockfd = -1, data_conn;
73     struct sockaddr_in servaddr, cliaddr, data_servaddr;
74     //MAX BUFF= 1024
75     char buff[MAX_BUFF], cmd[MAX_BUFF];
76     char *host_ip;
77     unsigned int port_num;
78
79     if(인자가 2개가 아니라면){
80         print "Usage: * <port>"
81         exit(fail)
82     }
83     소켓 생성(AF_INET,SOCK_STREAM,0)
84     만약 소켓 생성을 실패했다면 오류 출력후 종료
85
86     servaddr family=AF_INET
87     servaddr addr= htonl(INADDR_ANY)
88     servaddr port= 포트를 byte order로 변경
89     바인딩 후 오류 발생시 오류 출력 후 종료
90
91     listen함수로 queue생성 사이즈는 5
92     만약 생성 실패시 오류 출력 후 종료
93
94     클라이언트 연결 요청을 수락
95     연결 요청시 오류가 발생했다면 오류출력후 종료
96

```

```

while(1){

    buff 초기화
    클라이언트에게 받은 결과 출력

    if(buff가 PORT 커맨드면){
        host_ip=convert_str_to_addr호출로 인한 변경된 addr

        데이터 소켓 생성
        만약 소켓 생성을 실패한다면 오류 출력 후 종료

        data_servaddr family=AF_INET
        data_servaddr addr= inet_addr(argv[1])
        data_servaddr port= 포트를 byte order로 변경

        데이터 소켓을 클라이언트에게 연결
        만약 연결을 실패한다면 오류와 소켓을 닫아 종료

        print "200 Port command successful"
        "200 Port command successful"
        ->port커맨드를 받았다는 메시지를 클라이언트에게 전달
    }
}

```

```

121  else if(buf가 NLST 커맨드면){
122      NLST 명령에 경로가 있는지 검사
123
124      경로가 있다면 해당 경로로 설정
125      그것이 아니라면 해당 경로를 현재 디렉토리로 설정
126
127      print "150 Opening data connection for directory list."
128      "150 Opening data connection for directory list."
129      -> 데이터 연결을 열었다는 성공 메시지를 클라이언트에게 전달
130
131      list_directory 호출
132
133      print "226 Result is sent successfully."
134      "226 Result is sent successfully."
135      ->성공적으로 보낸 메시지를 클라이언트에게 전달받아
136      data socket닫음
137  }
138  else if(buf가 QUIT 커맨드면){
139      만약 QUIT뒤에 추가 인자가 있다면 Syntax err 클라이언트에게 전송
140
141      그것이 아니면 221 Goodbye print후 클라이언트로 보냄
142      break
143  }
144  else{
145      print "500 Syntax error, command unrecognized."
146      "500 Syntax error, command unrecognized."
147      ->알수 없는 명령어에 대한 오류 메시지를 클라이언트에게 전달
148  }
149  }
150  소켓을 모두 닫음
151  return
152  }

```

결과화면

```
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./cli 127.0.0.1 2222
>

kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./srv 2222
```

위 결과를 확인한다면 클라이언트와 서버가 연결되어 클라이언트는 입력을 할 수 있는 공간이 생겼음을 확인할 수 있다.

만약 ls 명령어를 입력할 경우는 다음과 같다.

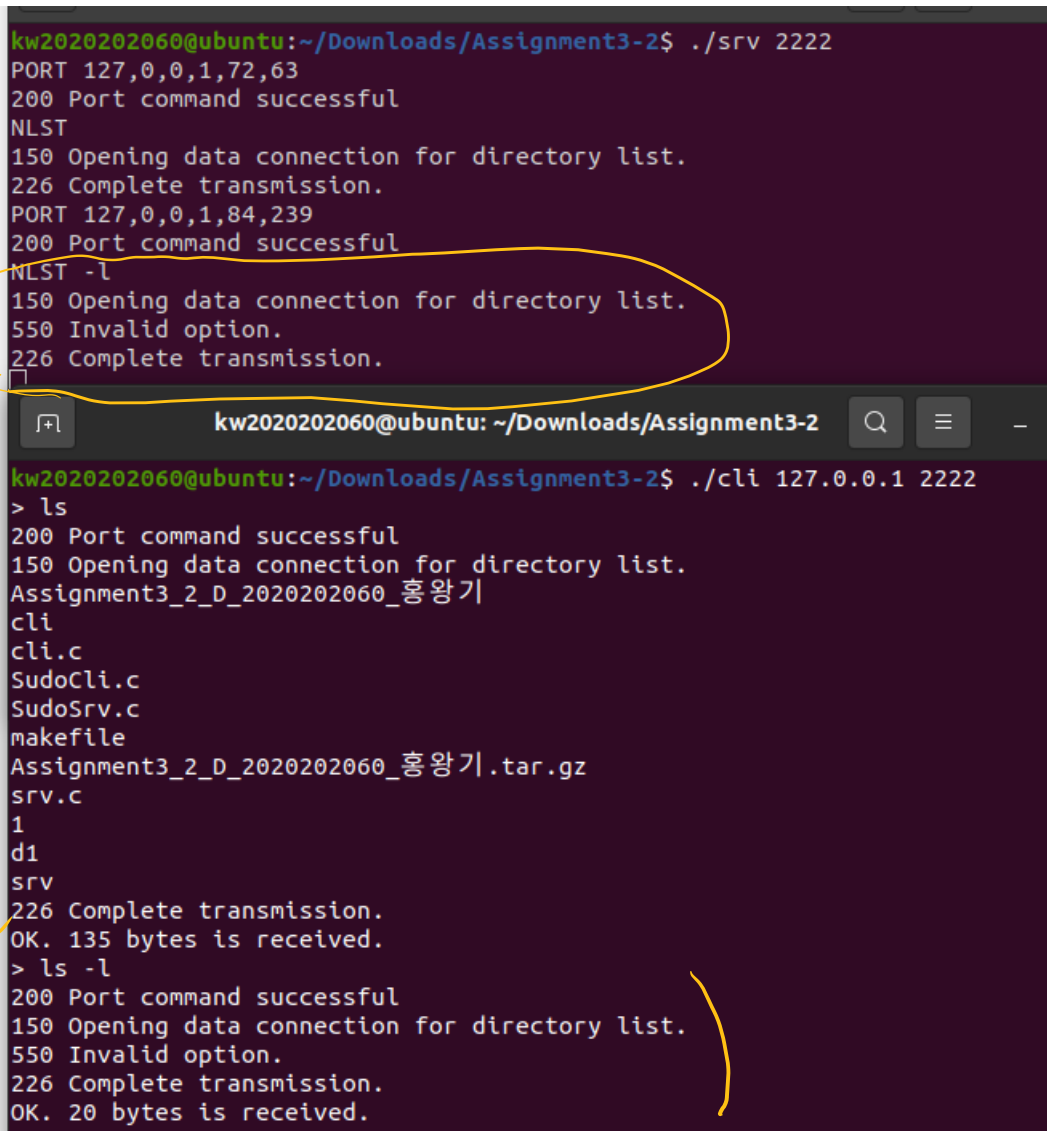
```
kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./srv 2222
PORT 127,0,0,1,72,63
200 Port command successful
NLST
150 Opening data connection for directory list.
226 Complete transmission.

kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./cli 127.0.0.1 2222
> ls
200 Port command successful
150 Opening data connection for directory list.
Assignment3_2_D_2020202060_홍왕기
cli
cli.c
SudoCli.c
SudoSrv.c
makefile
Assignment3_2_D_2020202060_홍왕기.tar.gz
srv.c
1
d1
srv
226 Complete transmission.
OK. 135 bytes is received.
>
```

위 결과를 확인한다면 ls 명령을 입력했을 때 port 커맨드와 ls에 대한 결과가 모두 올바른 것을 확인할 수 있다. 이때 성공 메시지와 결과에 대한 byte 수까지 출력됨을 볼 수 있다.

다음은 ls 에 대한 옵션이 있을 경우이다. 이번 과제에서는 ls 옵션 구현을 하지 않기에 오류 처리를 하였다.

위 결과를 확인한다면 ls 에대한 결과가 오류임을 볼 수 있다. 따라서 이 오류 메시지를 data connection 으로 전달되어 출력함을 확인할 수 있다.

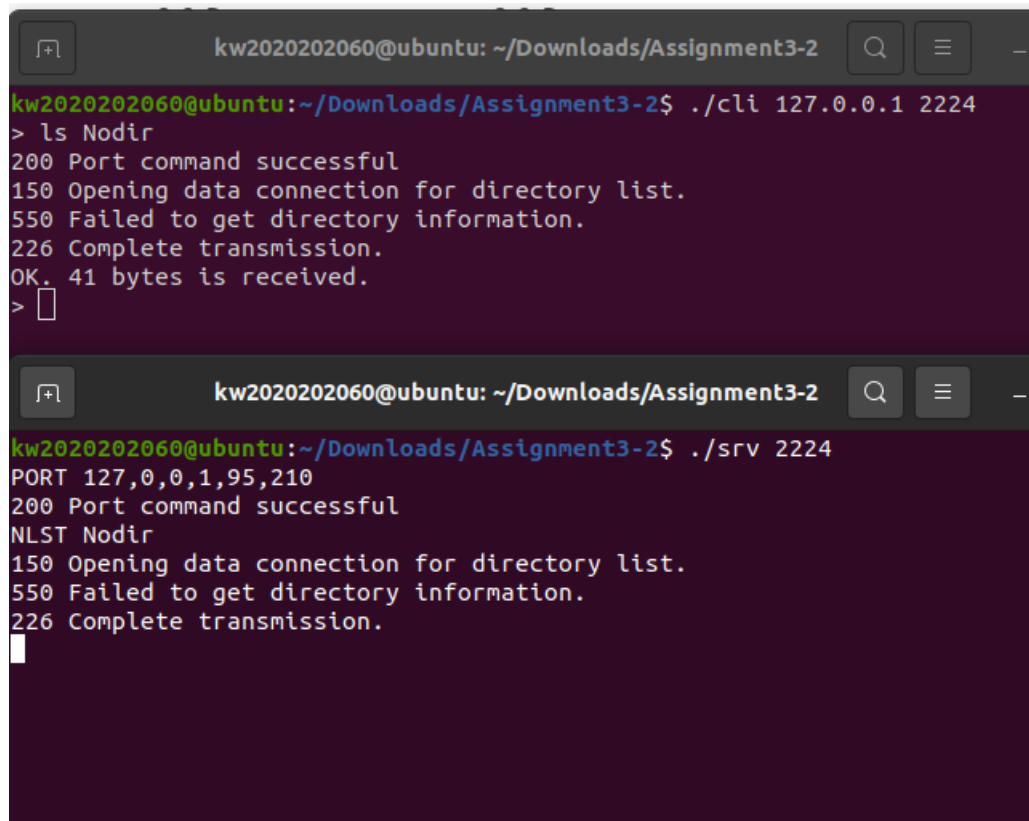


```
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./srv 2222
PORT 127,0,0,1,72,63
200 Port command successful
NLST
150 Opening data connection for directory list.
226 Complete transmission.
PORT 127,0,0,1,84,239
200 Port command successful
NLST -l
150 Opening data connection for directory list.
550 Invalid option.
226 Complete transmission.

kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./cli 127.0.0.1 2222
> ls
200 Port command successful
150 Opening data connection for directory list.
Assignment3_2_D_2020202060_홍왕기
cli
cli.c
SudoCli.c
SudoSrv.c
makefile
Assignment3_2_D_2020202060_홍왕기.tar.gz
srv.c
1
d1
srv
226 Complete transmission.
OK. 135 bytes is received.
> ls -l
200 Port command successful
150 Opening data connection for directory list.
550 Invalid option.
226 Complete transmission.
OK. 20 bytes is received.
```

The image shows a terminal window with two sessions. The top session is the server (./srv 2222) and the bottom is the client (./cli 127.0.0.1 2222). Yellow annotations highlight the '550 Invalid option.' error messages in both sessions, which occur when the 'ls -l' command is used. The error is sent over a 'data connection' as indicated by the '150 Opening data connection for directory list.' and '226 Complete transmission.' lines.

다음은 ls 에 디렉토리가 존재하지 않을 경우에 대한 결과이다.



The image shows two terminal windows. The top window shows the output of the command `./cli 127.0.0.1 2224` followed by `> ls Nodir`. The output indicates a successful port command, an attempt to open a data connection for a directory list, a failure to get directory information, and a complete transmission of 41 bytes. The bottom window shows the output of the command `./srv 2224` followed by `PORT 127,0,0,1,95,210` and `200 Port command successful`. It then shows `NLST Nodir`, an attempt to open a data connection for a directory list, a failure to get directory information, and a complete transmission.

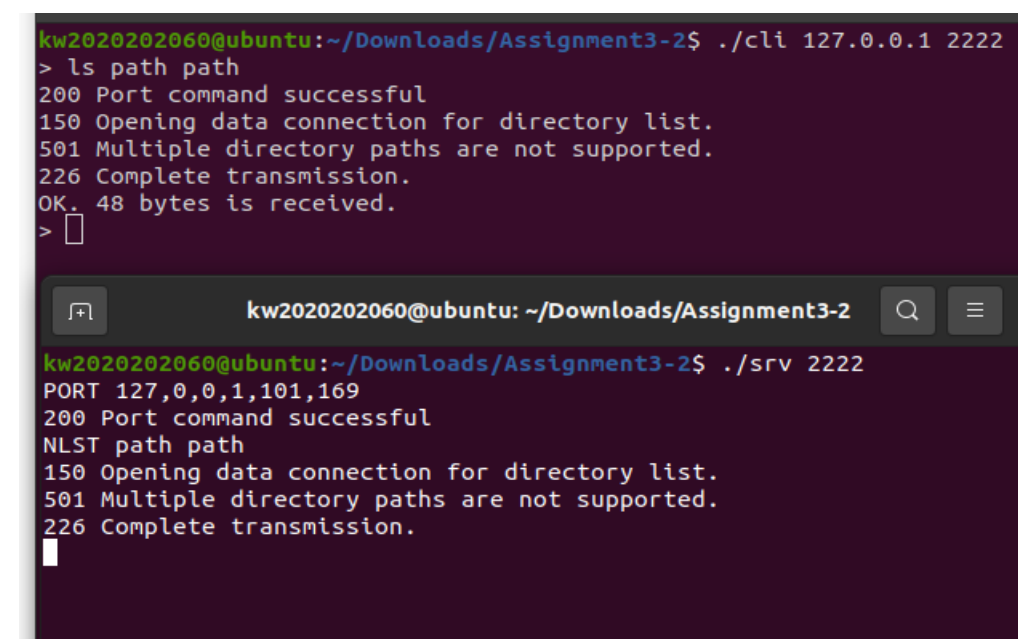
```
kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./cli 127.0.0.1 2224
> ls Nodir
200 Port command successful
150 Opening data connection for directory list.
550 Failed to get directory information.
226 Complete transmission.
OK. 41 bytes is received.
>

kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./srv 2224
PORT 127,0,0,1,95,210
200 Port command successful
NLST Nodir
150 Opening data connection for directory list.
550 Failed to get directory information.
226 Complete transmission.
```

결과를 확인한다면 받은 ls 명령어 결과가 존재하지 않는 디렉토리 이기에 클라이언트는 디렉토리 정보가 없기에 실패하였다는 메시지를 받은 것을 확인할 수 있다.

다음은 ls 에 대한 경로가 2 개 이상일 경우에 대한 검증이다.

(ls path path 입력)



The image shows two terminal windows. The top window shows the output of the command `./cli 127.0.0.1 2222` followed by `> ls path path`. The output indicates a successful port command, an attempt to open a data connection for a directory list, a failure due to multiple directory paths not being supported, and a complete transmission of 48 bytes. The bottom window shows the output of the command `./srv 2222` followed by `PORT 127,0,0,1,101,169` and `200 Port command successful`. It then shows `NLST path path`, an attempt to open a data connection for a directory list, a failure due to multiple directory paths not being supported, and a complete transmission.

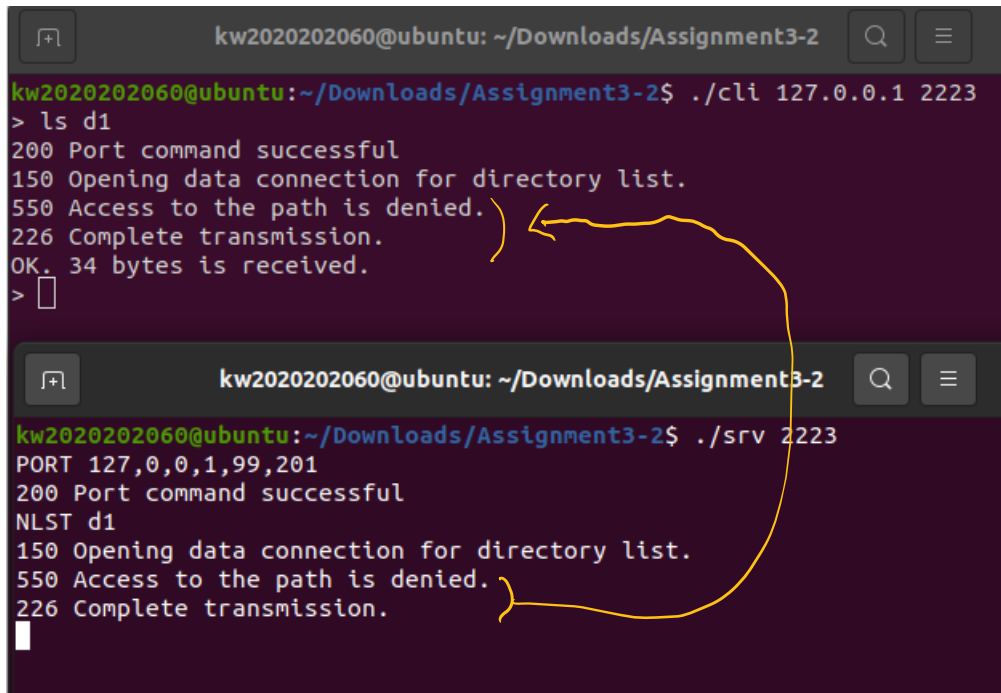
```
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./cli 127.0.0.1 2222
> ls path path
200 Port command successful
150 Opening data connection for directory list.
501 Multiple directory paths are not supported.
226 Complete transmission.
OK. 48 bytes is received.
>

kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./srv 2222
PORT 127,0,0,1,101,169
200 Port command successful
NLST path path
150 Opening data connection for directory list.
501 Multiple directory paths are not supported.
226 Complete transmission.
```

경로가 두개일 경우에 성공적으로 오류 메시지를 클라이언트에게 전달한 형태를 확인할 수 있다.

다음은 디렉토리에 접근 권한이 없을 때 검증 결과이다.

(ls d1(접근권한 x) 입력)



The image shows two terminal windows from a user 'kw2020202060' on an 'ubuntu' machine, located in the directory '~/Downloads/Assignment3-2'. The top window shows the output of the './cli 127.0.0.1 2223' command. The user enters 'ls d1', and the server responds with: '200 Port command successful', '150 Opening data connection for directory list.', '550 Access to the path is denied.', '226 Complete transmission.', and 'OK. 34 bytes is received.' A yellow arrow points from the '550 Access to the path is denied.' message to the bottom window. The bottom window shows the output of the './srv 2223' command. The server is listening on 'PORT 127,0,0,1,99,201'. It receives the command 'NLST d1' and responds with: '200 Port command successful', '150 Opening data connection for directory list.', '550 Access to the path is denied.', and '226 Complete transmission.' A yellow arrow points from the '550 Access to the path is denied.' message in the bottom window back to the top window.

```
kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./cli 127.0.0.1 2223
> ls d1
200 Port command successful
150 Opening data connection for directory list.
550 Access to the path is denied.
226 Complete transmission.
OK. 34 bytes is received.
>

kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./srv 2223
PORT 127,0,0,1,99,201
200 Port command successful
NLST d1
150 Opening data connection for directory list.
550 Access to the path is denied.
226 Complete transmission.
```

위를 확인하면 d1 디렉토리에 대한 ls을 할 경우 디렉토리 오픈을 실패하고 접근 권한이 없다는 오류 메시지가 출력됨을 확인할 수 있다.

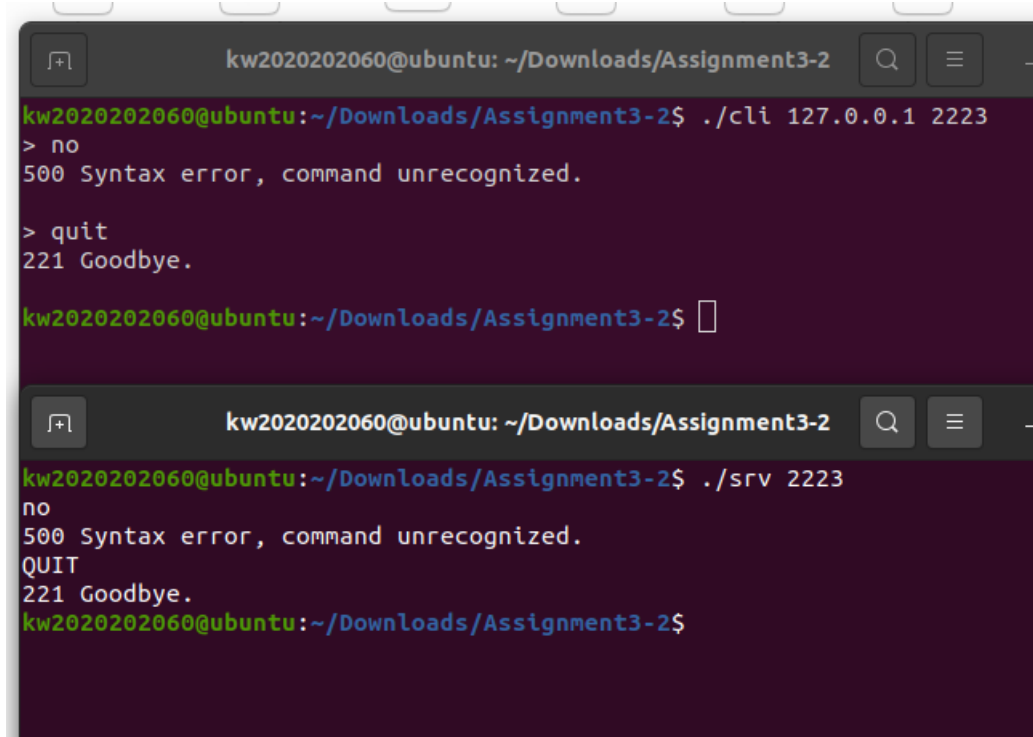
다음은 ls 경로가 올바르게 설정됐을 때 검증 결과이다.

```
kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./cli 127.0.0.1 2222
> ls ..
200 Port command successful
150 Opening data connection for directory list.
Assignment_1_2_D_2020202060_홍왕기.tar.gz
Assignment3-2
.a1.c.swp
.gostream-7PAFL2
Assignment2_3_D_2020202060_홍왕기.tar.gz
Assignment2-2
Assignment2_2_D_2020202060_홍왕기.tar.gz
Assignment1_3_D_2020202060_홍왕기.tar.gz
Assignment2-1
SrvCode.c
Assignment2_1_D_2020202060_홍왕기.tar.gz
Assignment1-3
CliCode.c
Assignment1_1_D_2020202060_홍왕기(1).tar.gz
Assignment1_1_D_2020202060_Hongwangki
Assignment3-1
226 Complete transmission.
OK. 428 bytes is received.
>

kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./srv 2222
PORT 127,0,0,1,72,202
200 Port command successful
NLST ..
150 Opening data connection for directory list.
226 Complete transmission.
```

위 결과를 확인한다면 ls ..을 입력했을 때 이전 경로인 Downloads 내용이 전체 출력되는 것을 확인할 수 있다.

다음은 올바른 명령어가 아닌 경우와 quit 를 입력했을 때 검증이다.



The image displays two screenshots of a terminal window. The top screenshot shows a user running the command `./cli 127.0.0.1 2223`. The prompt changes to `>`, and the user enters `no`. The terminal outputs `500 Syntax error, command unrecognized.`. The user then enters `quit`, and the terminal outputs `221 Goodbye.`. The bottom screenshot shows the user running the command `./srv 2223`. The prompt changes to `no`, and the user enters `QUIT`. The terminal outputs `500 Syntax error, command unrecognized.` and `221 Goodbye.`. Both screenshots show the terminal title bar as `kw2020202060@ubuntu: ~/Downloads/Assignment3-2`.

```
kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./cli 127.0.0.1 2223
> no
500 Syntax error, command unrecognized.

> quit
221 Goodbye.

kw2020202060@ubuntu:~/Downloads/Assignment3-2$

kw2020202060@ubuntu: ~/Downloads/Assignment3-2
kw2020202060@ubuntu:~/Downloads/Assignment3-2$ ./srv 2223
no
500 Syntax error, command unrecognized.
QUIT
221 Goodbye.
kw2020202060@ubuntu:~/Downloads/Assignment3-2$
```

위를 확인하면 올바른 명령어가 아닐 경우에 syntax 에러가 출력되고 quit 를 입력했을 때 성공적으로 종료한 것을 확인할 수 있다.

고찰

이 프로젝트를 구현하는 과정에서 여러 가지 오류가 발생했다. 가장 먼저, 데이터 연결 소켓에 대한 연결이 원활하게 이루어지지 않았다. 이를 해결하기 위해 다양한 시도를 하였고, 최종적으로 네트워크 설정과 소켓 초기화 과정에서의 문제를 찾아내어 수정했다.

또한 데이터를 주고받는 과정에서 아랍어 문자가 반복되어 출력되는 오류를 겪었다. 이 문제를 해결하기 위해 코드의 흐름을 따라가며 전체 과정을 출력해보았다. 각 단계마다 print 문을 삽입하여 어디서 문제가 발생하는지 확인한 결과, 버퍼 초기화 과정에서 문제가 있다는 것을 발견했다. 이 문제를 해결한 후, 버퍼 초기화의 중요성을 절실히 깨닫게 되었다.

이와 더불어, 데이터 및 제어 연결에 대해 양방향 소켓을 구현하는 과정에서도 많은 혼란함을 느꼈다. 양방향 소켓 통신을 구현하는 것은 단방향 통신보다 복잡한 작업이었으며, 특히 동시성 제어와 데이터 무결성 보장 측면에서 많은 고민이 필요했다. 이러한 문제들을 해결해 나가는 과정에서 네트워크 프로그래밍에 대한 이해도가 크게 향상되었고, 소프트웨어의 안정성을 높이는 방법에 대해 많은 것을 배웠다.

이 프로젝트를 통해 얻은 교훈은 다음과 같다. 첫째, 네트워크 연결 및 소켓 초기화 과정에서 발생할 수 있는 문제를 사전에 충분히 이해하고 대비하는 것이 중요하다는 점이다. 둘째, 데이터를 주고받는 과정에서 발생하는 문제를 해결하기 위해 디버깅을 철저히 하고, 코드의 흐름을 이해하는 것이 중요하다는 점이다. 셋째, 양방향 소켓 통신을 구현할 때는 동시성 문제와 데이터 무결성에 대한 고려가 필수적이라는 점이다.

결론적으로, 이번 프로젝트를 통해 네트워크 프로그래밍의 기초부터 고급 개념까지 폭넓게 학습할 수 있었고, 문제 해결 능력도 크게 향상되었다. 앞으로의 프로젝트에서도 이러한 경험을 바탕으로 더욱 효율적이고 안정적인 소프트웨어를 개발할 수 있을 것이라고 생각하였다.