시스템 프로그래밍 실습

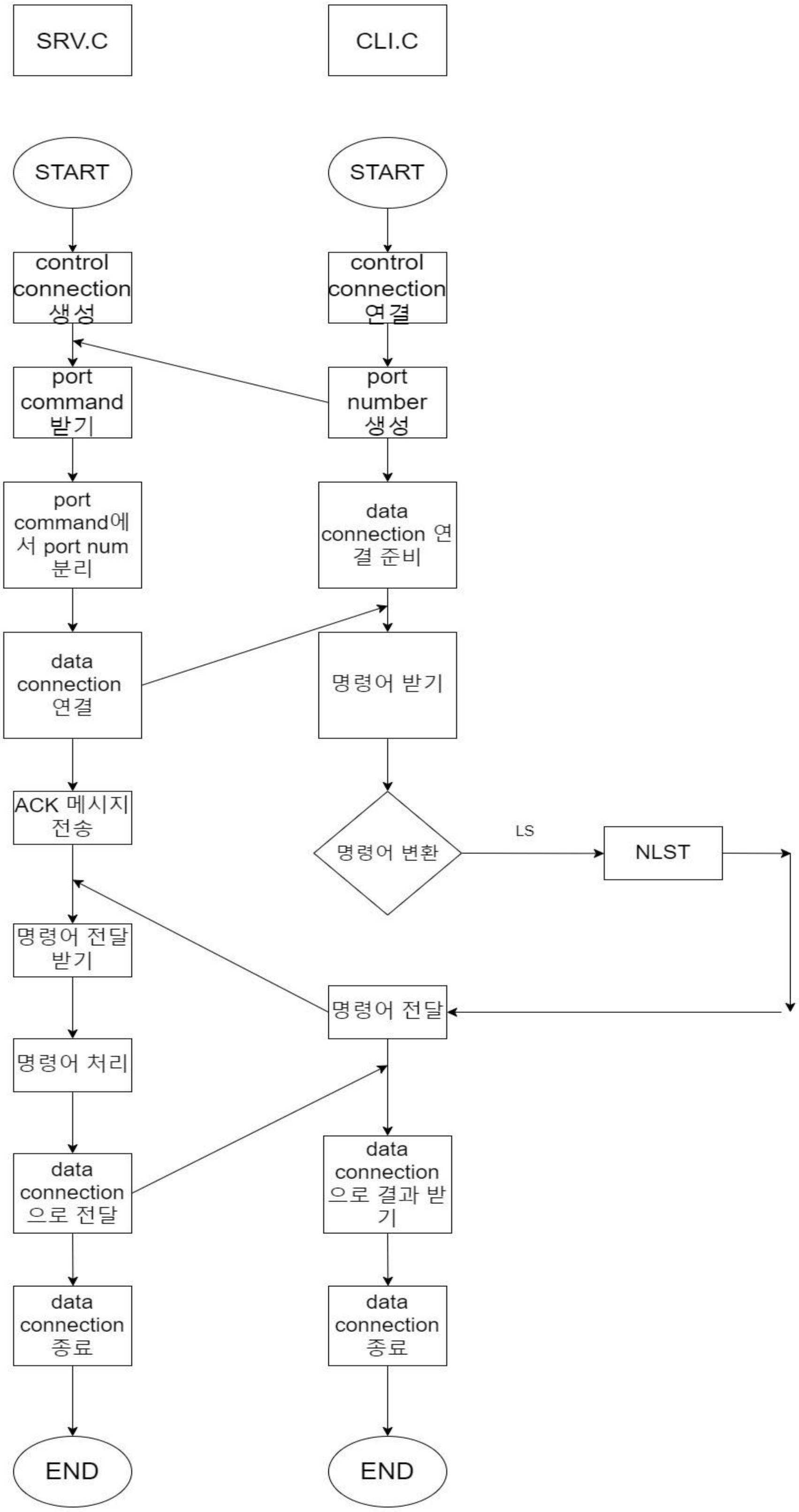
[Assignment3-2]

Class	:	[A]
Professor	:	[김태석 교수님]
Student ID	:	[2019202032]
Name	:	[이상현]

Introduction

해당 과제는 control connection 과 data connection 을 구현하는 과제이다. client 에서는 임의의 포트번호를 생성하고, 명령어 port 를 server 에 전달한다. server 에서는 전달 받은 명령어 port 에 대해 ack 를 전송하고, 해당 메시지로부터 port 번호와 ip address 를 분류하고, 해당 번호들을 이용하여 data connection 을 만든다. Data connection 을 통해 control connection 으로부터 받은 명령어의 결과를 전송하고, 한번 사용한 data connection 은 close 한다.

Flow chart


SRV.C
CLI.C
START
START
control connection 생성
control connection 연결
port command 받기
port number 생성
port command에서 port num 분리
data connection 연결 준비
data connection 연결
명령어 받기
ACK 메시지 전송
명령어 변환
LS
NLST
명령어 전달 받기
명령어 전달
명령어 처리
data connection으로 전달
data connection으로 결과 받기
data connection 종료
data connection 종료
END
END

Pseudo code

<cli.c>

```
char* convert_addr_to_str(unsigned long ip_addr, unsigned int port)

{

   int ip_num = 0;

    char *ptr = strtok(ip, ".");

    /************** start slicing *************/

    while(ptr != NULL) {

        strcat(addr, ptr);//strcat sliced string

        addr[strlen(addr)] = ',';

        ptr = strtok(NULL, ".");      //put ip address to port command

    }

    /*****************************************/


    /*********   convert to 16bit binary **********/

    char* binary = (char*)malloc(17);

    binary[16] = 0;

    for(int i = 15; i>= 0; i--) {

        binary[i] = (port&1) ? '1' : '0';

        port >>= 1;

    }

    /***********************************/


    /******** convert to MSB binary to decimal **********/
```

```
	int decimal1 = 0;
	int two = 1;
	for(int i = 7; i>=0; i--){
		decimal1 += two * (binary[i] - '0');
		two *= 2;
	}
	/***************************************/

/********* convert to LSB binary to decimal **********/
	int decimal2 = 0;
	two = 1;
	for(int i = 15; i>=8; i--){
		decimal2 += two * (binary[i] - '0');
		two *= 2;
	}
	/*****************************************************/

	return addr;		//return ip address
}
int main(int argc, char **argv)
{
	srand(time(NULL));
	/********************* 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);
    temp.sin_port = htons(atoi(argv[2]));
    connect(control_sockfd, (struct sockaddr*) &temp, sizeof(temp));     //connect with server
    /*************************************************************/


    /***************** make random port number *******************/
    int data_port = rand() %20000 + 10001;
    hostport = convert_addr_to_str(temp.sin_addr.s_addr, data_port);     //make port command
    /***********************************************************/



    write(control_sockfd, hostport, MAX_BUF);    //send port command



    /**************************************     prepare     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);
/*************************************************************************************************
**************/


    /*********receive command and convert command *******/
    read(STDIN_FILENO, buf, MAX_BUF);
    if(conv_cmd(buf, cmd_buf) < 0){
        write(1, "NO command₩n", 12);
    };
    /**************************************************/

    /********get message from control connection("200 PORT successful")********/
read(control_sockfd, buf, MAX_BUF);
write(1, buf, strlen(buf));
    /***************************************************************************/


    /**************send command***************/
    write(control_sockfd, cmd_buf, strlen(cmd_buf));
    /****************************************/
   /********get message from control connection("150 opening successful")********/
    read(control_sockfd, buf, MAX_BUF);
    write(1, buf, strlen(buf));
    /****************************************************************************/

    /****************** get result from data_connfd *********************/
    read(data_connfd, buf, MAX_BUF);
    write(1, buf, strlen(buf));
    /*********************************************************************/
```

```
        close(data_connfd);                                //close data_connection socket

        sleep(0.01);

    receive message from control connection("226 Result successful")

/*********************************************************************************************/

        printf("%ld bytes is received₩n", strlen(buf));

}
```

<srv.c>

```
char * convert_str_to_addr(char *str, unsigned int *port)

{
```

```
char *ptr = strtok(str, ",");

int num = 0;

/************** start slicing *************/

while(ptr != NULL) {

    if(num == 6) {        //get port number

        int two = 1;

        int number = atoi(ptr); //convert to integer

        for(int i = 7; i >= 0; i--) {

            int left = number % 2;

            number = number /2;

            *port += left * two;      //get port number

            two *= 2;

        }

    }

    else if(num == 5) { //get port number

        int two = 256;

        int number =atoi(ptr);   //convert to integer

        for(int i = 0; i<=7; i++){

            int left = number %2;

            number = number /2;

            *port += left * two;      //get port number

            two*= 2;

        }

    }
```

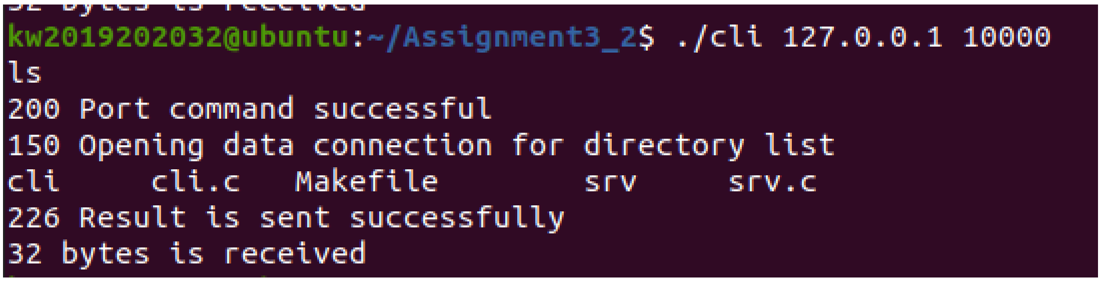
```
        else if(num >= 1){   //get ip address
            strcat(addr, ptr);
            addr[strlen(addr)] = '.';
        }
        num++;
        ptr = strtok(NULL, ",");
    }
    /*****************************************/
return addr;      //return ip address
}
int main(int argc, char **argv)
{
    /******************** prepare control connection *********************/
    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);
	/*************************************************************************/

	/*****************   receive random port number by control connection
	int n = read(connfd, temp, MAX_BUF);       //receive port command
	write(1, temp, MAX_BUF);
	host_ip = convert_str_to_addr(temp, (unsigned int *) &port_num);       //convert port number and ip address
	/*************************************************************************************/

	/******************** connect data connection ******************************/
	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);
```

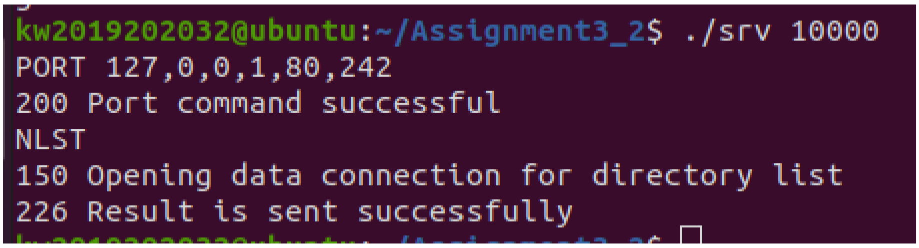
```
    if(connect(data_sockfd, (struct sockaddr*) &data_temp, sizeof(data_temp)) < 0)
printf("connection failed");      //connect with server
    /*******************************************************************************/



    /************* write message and send message ************/
    write(1, "200 Port command successful₩n", 29);
    write(connfd, "200 Port command successful₩n", MAX_BUF);
    /**********************************************************/
/******** get command from control connection ********/
    read(connfd, buf, MAX_BUF);
    buf[strlen(buf)] = '₩0';

    write(1, buf, strlen(buf));
    write(1, "₩n", 2);
    /*****************************************************/

    /************* write message and send message ************/
    write(1, "150 Opening data connection for directory list₩n", 48);
    write(connfd, "150 Opening data connection for directory list₩n", 48);
    /**********************************************************/
    /********* send result of command *********/
    char result_buff[MAX_BUF];
    cmd_process(buf, result_buff);   //command processing
```

```
        write(data_sockfd, result_buff, MAX_BUF);     //send result
        /*****************************************/

        close(data_sockfd); //close data connection


        /*********** write message and send message ("226 result successful") **********/
        write(1, "226 Result is sent successfully₩n", 33);
        write(connfd, "226 Result is sent successfully₩n", 33);
        /********************************************************************************/


}
```

결과화면

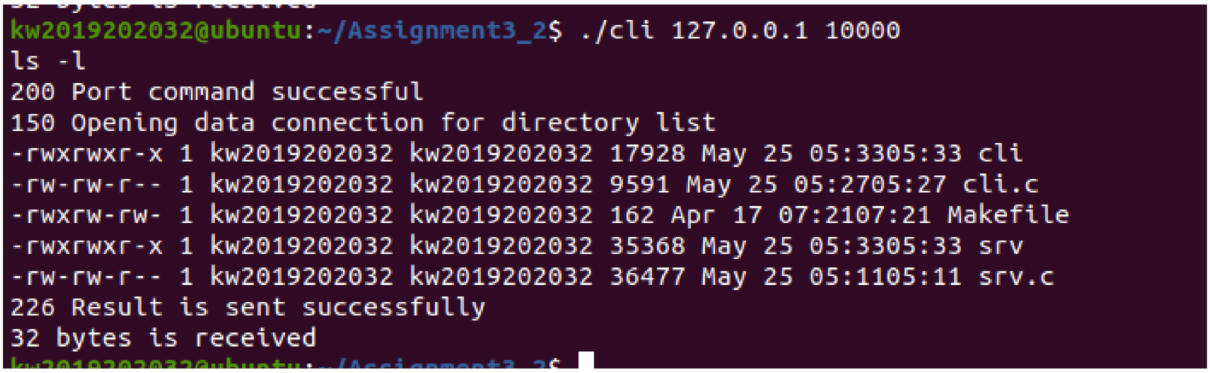


```
32 bytes is received
kw2019202032@ubuntu:~/Assignment3_2$ ./cli 127.0.0.1 10000
ls
200 Port command successful
150 Opening data connection for directory list
cli      cli.c    Makefile         srv      srv.c
226 Result is sent successfully
32 bytes is received
```

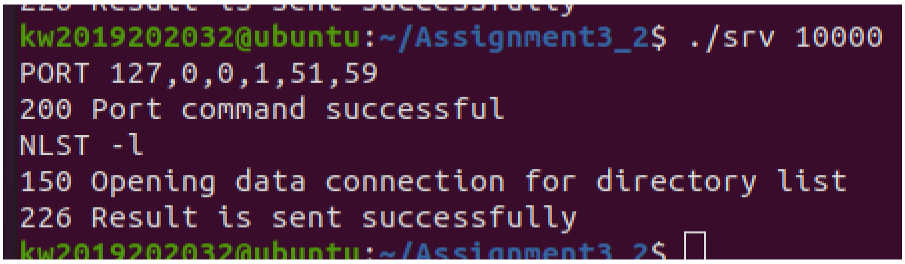


```
kw2019202032@ubuntu:~/Assignment3_2$ ./srv 10000
PORT 127,0,0,1,80,242
200 Port command successful
NLST
150 Opening data connection for directory list
226 Result is sent successfully
kw2019202032@ubuntu:~/Assignment3_2$ 
```

강의자료 속 예시 처럼 ls 명령어에 대해서 data connection 을 통해 결과를 주고 받는 것을 확인할 수 있다.



```
32 bytes is received
kw2019202032@ubuntu:~/Assignment3_2$ ./cli 127.0.0.1 10000
ls -l
200 Port command successful
150 Opening data connection for directory list
-rwxrwxr-x 1 kw2019202032 kw2019202032 17928 May 25 05:3305:33 cli
-rw-rw-r-- 1 kw2019202032 kw2019202032 9591 May 25 05:2705:27 cli.c
-rwxrw-rw- 1 kw2019202032 kw2019202032 162 Apr 17 07:2107:21 Makefile
-rwxrwxr-x 1 kw2019202032 kw2019202032 35368 May 25 05:3305:33 srv
-rw-rw-r-- 1 kw2019202032 kw2019202032 36477 May 25 05:1105:11 srv.c
226 Result is sent successfully
32 bytes is received
kw2019202032@ubuntu:~/Assignment3_2$ 
```



```
226 Result is sent successfully
kw2019202032@ubuntu:~/Assignment3_2$ ./srv 10000
PORT 127,0,0,1,51,59
200 Port command successful
NLST -l
150 Opening data connection for directory list
226 Result is sent successfully
kw2019202032@ubuntu:~/Assignment3_2$ 
```

다른 명령어에 대해서도 예시의 결과처럼 정상적으로 동작하는 것을 확인할 수 있다.

고찰

해당 과제를 진행하는 data connection 을 구축하기 위해서 client socket 이 server socket 역할을 하고, server socket 에서 client socket 역할을 하도록 하였다. 이 과정에서 많은 소켓들이 서로 read/write 를 수행하였기 때문에 특정 write 를 연결된 socket 에서 read 하지 못하는 상황들이 발생하였다. 이를 해결하기 위하여 read 하기에 앞서 sleep 함수를 사용하여 잠깐 delay 를 주었고, 이러한 방법을 통해 연결된 socket 에서 write 한 것을 정상적으로 read 할 수 있었다. 해당 과제를 통해 server 와 client 를 무조건적으로 분하여 보는 것이 아니라 역할에 따라 달라질 수 있다는 것을 유념애햐겠다는 생각을 가지게 되었다.

Reference

강의자료만 참고하였습니다.