System Programming

시스템 프로그래밍

(화5, 목6)

Assignment #4-2

Split Connection

김 태 석 교수님

컴퓨터정보공학부

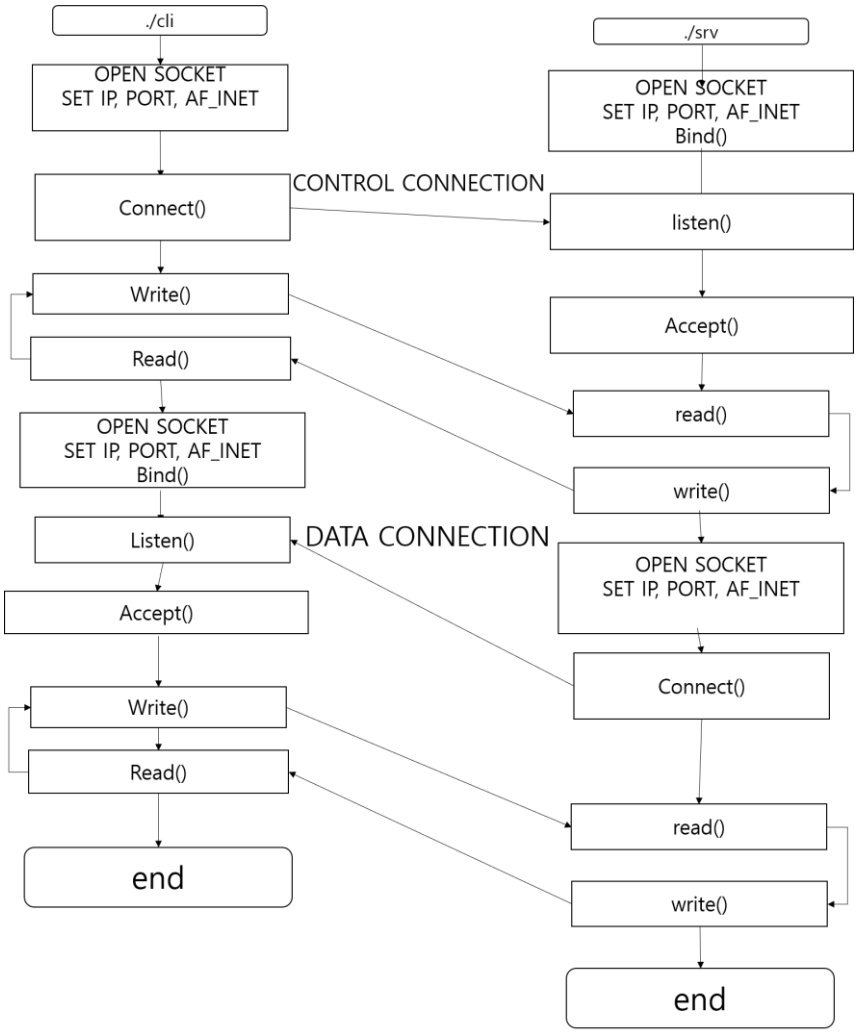
2017202037

오 민 혁

<Introduction>

이번 과제는 Spilt Connection을 구현하는 것이다. 이는 각각 다른 역할을 하는 control connection과 data connection 총 2개의 connection을 구현해야 한다. control connection은 이전 과제에서 구현했던 것처럼 client가 server에 접속하는 것이고, data connection은 server가 client에 접속하는 방식으로 구현해야 한다. Client는 control connection을 통해 port command와 ls 명령어를 server에게 전송한다. 그리고 server에서 이 명령어들을 받아서 결과값과 a numerical reply를 data connection을 통해 client에게 전달한다. 마지막으로 client가 data connection을 통해 받은 결과 값을 출력하면서 프로그램이 종료된다.

<Flow Chart>



<Source Code>

<cli.c>

```
////////////////////////////////////////////////////////////////////
// File Name : cli.c                                              //
// Date : 2020/06/13                                              //
// Os : Ubuntu 16.04.5 LTS                                        //
// Author : Oh Min Hyeok                                          //
// Student ID : 2017202037                                        //
// ---------------------------------------------------------- //
// Title : Assignment #4-2                                        //
// Description : ...                                              //
////////////////////////////////////////////////////////////////////

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/wait.h>
#include <string.h>
#include <time.h>
#define MAX_BUF 100
```

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


int main(int argc, char **argv){
	srand(time(NULL));
	char buf[MAX_BUF];
	int sockfd, client_fd,server_fd;
	struct sockaddr_in servaddr, temp,serv_addr;
	char * hostport;
	sockfd = socket(AF_INET, SOCK_STREAM, 0);// PF_INET = IPv4 Internet Protocol, SOCK_STREAM = STREAM(TCP Protocol)
	memset(&servaddr, 0, sizeof(servaddr)); // initialize
	servaddr.sin_family=AF_INET; // IPv4
	servaddr.sin_addr.s_addr=inet_addr(argv[1]); // STORE IP Address
	servaddr.sin_port=htons(atoi(argv[2])); // STORE PORT Number
	// server connection
	connect(sockfd,(struct sockaddr *)&servaddr, sizeof(servaddr));
	int n;
	write(STDOUT_FILENO, "> ", 2);
	n=read(STDIN_FILENO, buf, sizeof(buf));
	buf[n-1]='₩0';
	int randnum=(rand() % 20000) + 10001;
```

```
////////////////*************** data connection
***************////////////////////////////////
client_fd=socket(AF_INET,SOCK_STREAM,0);
memset(&temp,0,sizeof(temp));
temp.sin_family=AF_INET;
temp.sin_addr.s_addr=inet_addr(argv[1]);
temp.sin_port=randnum;// temporaily port number




hostport=convert_addr_to_str(temp.sin_addr.s_addr,temp.sin_port);
printf("converting to %s₩n",hostport);
write(sockfd,hostport,strlen(hostport));




if(bind(client_fd, (struct sockaddr *)&temp, sizeof(temp))<0){
        printf("Client: Can't bind local address.₩n"); // exception handling
        return 0;
}

listen(client_fd, 5);
int len=sizeof(serv_addr);
server_fd=accept(client_fd,(struct sockaddr*)&serv_addr,&len); // accept connect
request
```

```
//////////////////////////////////////////////////////////////////////////////////////////////
char buffer[MAX_BUF];
n=read(server_fd,buffer,sizeof(buffer)); // read from data connection
buffer[n]='₩0'; // remove trash value
printf("%s₩n",buffer);

write(sockfd,buf,strlen(buf)); // sent command to control connection

n=read(server_fd,buffer,sizeof(buffer));
printf("%s₩n",buffer);
size_t length = strlen(buffer); // result byte number.

char message[100]="226 Result is sent successfully.";
printf("%s₩n",message);
write(server_fd,message,sizeof(message));

printf("OK. %u bytes is received.₩n",length-48); // 'length -48' is byte number that remove message byte number.

close(sockfd);
return 0;
}
```

```
char * convert_addr_to_str(unsigned long ip_addr, unsigned int port){
	char * addr=malloc(sizeof(char)*MAX_BUF);
	//////////// network order 32bit big endian --->>> dotted deciman notation ///////////////
	struct in_addr tempaddr;
	tempaddr.s_addr=ip_addr;
	char * ttemp= inet_ntoa(tempaddr);
	char * ptr=strtok(ttemp,".");
	strcpy(addr,"PORT ");
	strcat(addr,ptr);
	strcat(addr,",");
	ptr=strtok(NULL,".");
	strcat(addr,ptr);
	strcat(addr,",");
	ptr=strtok(NULL,".");
	strcat(addr,ptr);
	strcat(addr,",");
	ptr=strtok(NULL,".");
	strcat(addr,ptr);
	strcat(addr,",");
	////////////////////////////////////////////////////////////////////////////////////////////
	int arr[16]={0, };
	int n=port,c=0,mok,nmg,i;
	/// ** decimal to binary ** ///
```

```
do{
        mok=n/2;
        nmg=n-mok*2;
        arr[c++]=nmg;
        n=mok;
}while(mok!=0);
//////////////////////////////
/////////// ** binary to decimal ** ////////
int first=0;
int decimal=0;
for(int i=8;i<=15;i++)
{
        if(arr[i]==1)
                decimal+=1<<first;
        first++;
}
char *dec=malloc(sizeof(char)*MAX_BUF);
sprintf(dec,"%d",decimal);
strcat(addr,dec);
strcat(addr,",");


first=0;
decimal=0;
for(int i=0;i<=7;i++)
```

```
        {
                if(arr[i]==1)
                        decimal+=1<<first;
                first++;
        }
        sprintf(dec,"%d",decimal);
        ////////////////////////////////////////
        strcat(addr,dec);


        return addr;
}
```

<srv.c>

```
/////////////////////////////////////////////////////////////////////
// File Name : srv.c                                               //
// Date : 2020/06/13                                               //
// Os : Ubuntu 16.04.5 LTS                                         //
// Author : Oh Min Hyeok                                           //
// Student ID : 2017202037                                         //
// ------------------------------------------------------------- //
// Title : Assignment #4-2                                          //
// Description : ...                                               //
/////////////////////////////////////////////////////////////////////
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <signal.h>
#include <string.h>
#include <dirent.h>

#define MAX_BUF 100
char * cmp_process(char *buff,char *result_buff);
char * convert_str_to_addr(char*str, unsigned int *port);

int main(int argc, char **argv) {
        char result_buff[MAX_BUF]; // RESULT BUFF
        bzero(result_buff, sizeof(result_buff));
        char buf[MAX_BUF]; // buf
        char temp[25];
        char *host_ip=NULL;
        unsigned int port_num;
        int n;
        int server_fd, client_fd;
```

```
struct sockaddr_in servaddr, cliaddr;

server_fd = socket(PF_INET, SOCK_STREAM, 0); // PF_INET = IPv4 Internet Protocol, SOCK_STREAM = STREAM(TCP Protocol)

memset(&servaddr, 0, sizeof(servaddr)); // initialize
servaddr.sin_family=AF_INET; // IPv4
servaddr.sin_addr.s_addr=htonl(INADDR_ANY); // SET IP Address automatically
servaddr.sin_port=htons(atoi(argv[1]));

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

listen(server_fd, 5);
int len=sizeof(cliaddr);
client_fd=accept(server_fd,(struct sockaddr*)&cliaddr,&len); // accept connect request

n=read(client_fd,temp,sizeof(temp));
temp[n]='₩0';
printf("%s₩n",temp);
host_ip=convert_str_to_addr(temp,(unsigned int*)&port_num);
//////////////////////////// ***** data connection *****
```

```
//////////////////////////////////////
	int sockfd;
	struct sockaddr_in sockaddr;
	sockfd = socket(AF_INET, SOCK_STREAM, 0);// PF_INET = IPv4 Internet Protocol,
SOCK_STREAM = STREAM(TCP Protocol)
	memset(&sockaddr, 0, sizeof(sockaddr)); // initialize
	sockaddr.sin_family=AF_INET; // IPv4
	sockaddr.sin_addr.s_addr=inet_addr(host_ip); // STORE IP Address
	sockaddr.sin_port=port_num; // STORE PORT Number
	// server connection
	connect(sockfd,(struct sockaddr *)&sockaddr, sizeof(sockaddr));
	////////////////////////////////////////////////////////////////////////////////////////////////////////
//
	char message[100]="200 Port command successful";
	printf("%s₩n",message);
	write(sockfd,message,strlen(message));
	n=read(client_fd,temp,sizeof(temp));
	temp[n]='₩0'; /// remove trash value
	printf("%s₩n",temp);

	strcpy(message,"226 Result is sent successfully.");
	write(sockfd,cmp_process(temp,result_buff),sizeof(result_buff));

	n=read(sockfd,message,sizeof(message));
	printf("%s₩n",message);
```

```
        close(client_fd);


}
char * convert_str_to_addr(char*str, unsigned int *port)
{
        ///////////// ***** Separate PORT Command ***** ////////////////////
        char * addr=malloc(sizeof(char)*20);
        char * ptr=strtok(str," ");


        ptr=strtok(NULL,",");
        ////////////////////// IP Address ////////////////////////////
        strcpy(addr,ptr);


        strcat(addr,".");


        ptr=strtok(NULL,",");
```

```
        strcat(addr,ptr);
        strcat(addr,".");
        ptr=strtok(NULL,",");
        strcat(addr,ptr);
        strcat(addr,".");
        ptr=strtok(NULL,",");
        strcat(addr,ptr);

        ////////////////////////////////////////////////////////////////
        ////////////////////////// PORT ////////////////////////////////
        ptr=strtok(NULL,",");
        unsigned int tmp=atoi(ptr)*256;
        ptr=strtok(NULL,",");
        tmp=tmp+atoi(ptr);
        *port=tmp;
        ////////////////////////////////////////////////////////////////
        return addr;
}

char * cmp_process(char *buff,char *result_buff){



        if(strcmp(buff,"ls")==0) { // ls command
```

```
char *cwd=(char*)malloc(sizeof(char)*1024); // dynamic allocate
DIR * dir =NULL; // directory pointer
struct dirent * entry =NULL; // directory struct
getcwd(cwd,1024); // current working directory
dir=opendir(cwd);

if(dir==NULL){ // exception handling
        strcat(result_buff,"error");

}
printf("%s₩n","150 Opening data connection for directory list.");

strcpy(result_buff,"150 Opening data connection for directory list.");
strcat(result_buff,"₩n");
while((entry=readdir(dir))!=NULL) // read directory
{

        char *tmp=(char*)malloc(sizeof(char)*1024);
        bzero((char*)&tmp, sizeof(tmp)); // initialize
        tmp=entry->d_name; // directory name
        int length=strlen(tmp);

        strcat(result_buff,tmp);
```

```
                strcat(result_buff,"₩n");


            }
            result_buff[strlen(result_buff)-1]='₩0';
            free(cwd); // free dynamic allocate memory
            closedir(dir); // close directory


        }
        else{
            strcat(result_buff,"input error");
        }



        return result_buff;
}
```

<Result Screen>