시스템 프로그래밍 실습

[Assignmen2-3]

Class : D 반(실습 2 금 56)

Professor : 최상호 교수님

Student ID : 2022202104

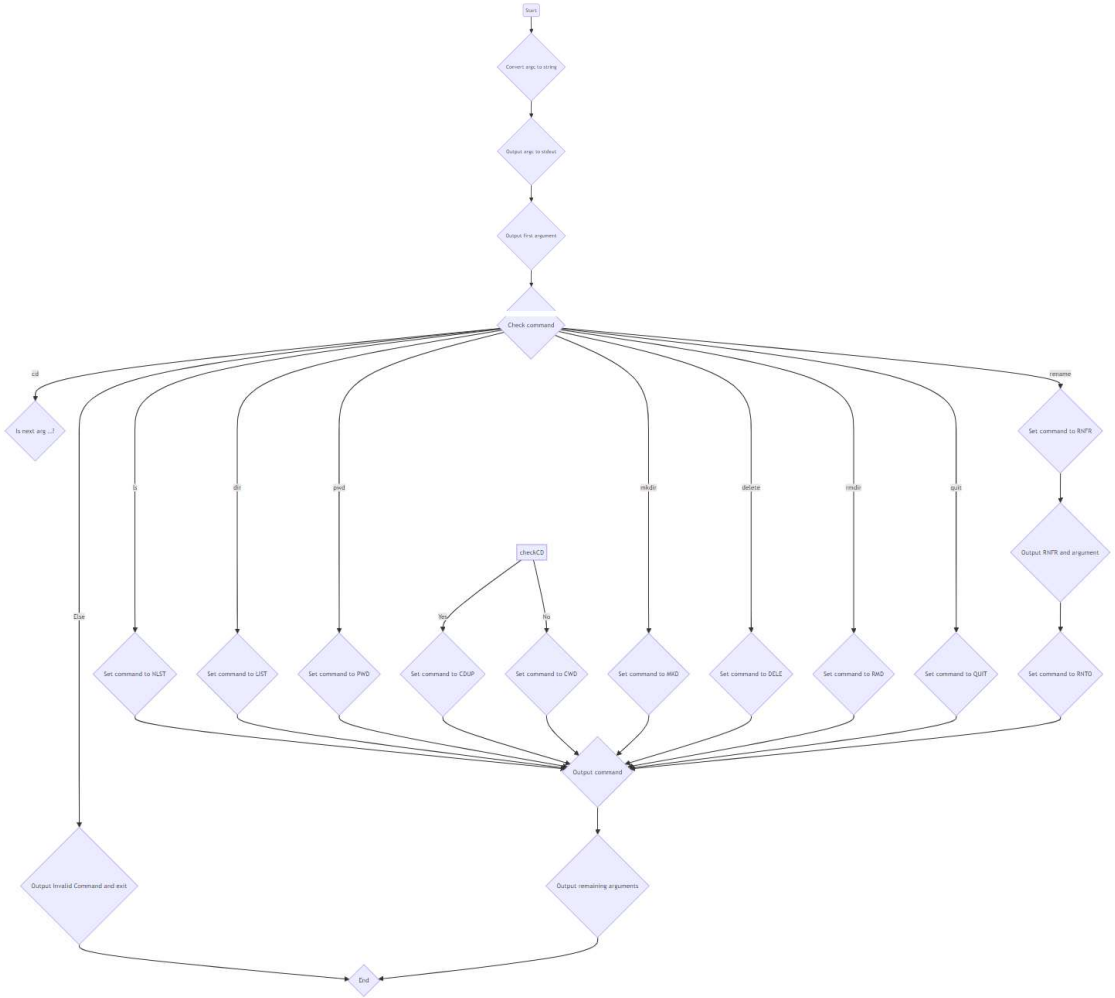
Name : 김유찬

Introduction

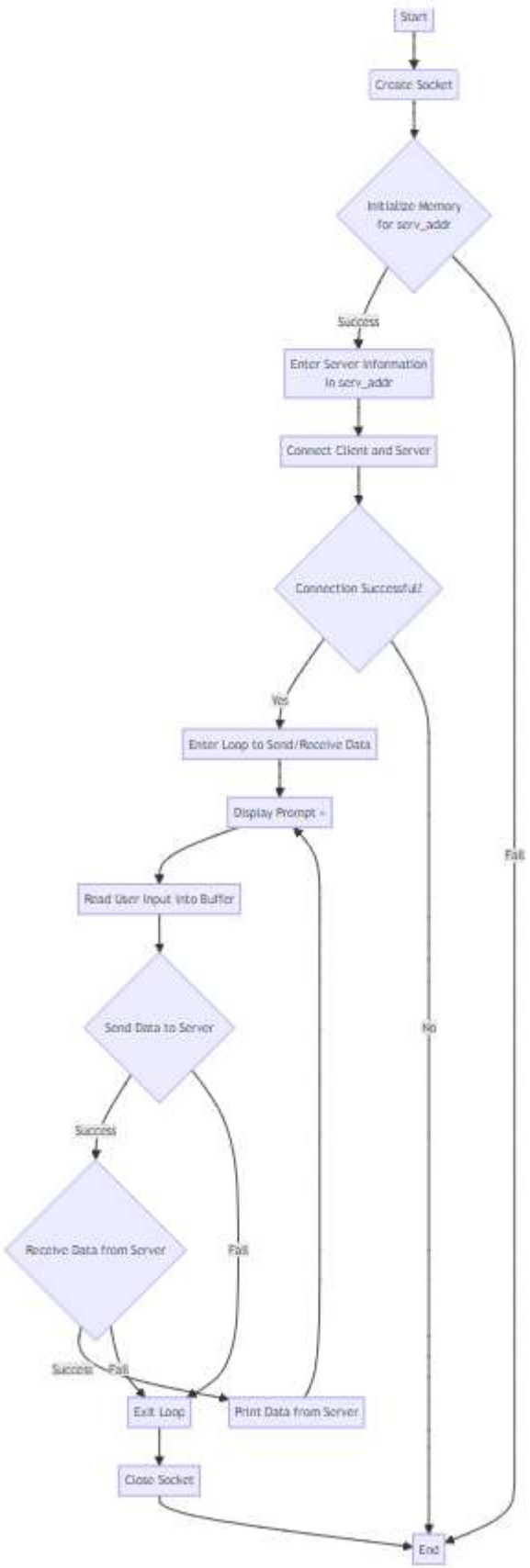
저번 과제에서는 Socket 과 fork 를 통해 클라이언트와 서버 간에 병렬 처리를 했다. 이번엔 client 가 리눅스 명령어를 FTP 명령어로 변환하고 서버는 FTP 명령어를 처리하여 그 결과를 다시 client 한테 넘겨주는 기능을 만들 것이고 또한 server 쪽에서 연결된 클라이언트 정보를 출력하는 기능도 추가한다. Ctnl+C 입력하면 프로세스에 SIGINT 라는 신호가 전달되는 이 신호를 이용하여 client 와 server 각각에서 접속이 끊어지고 프로세스가 종료될 수 있게 해보는 시간도 갖을 것이다.

Flow chart

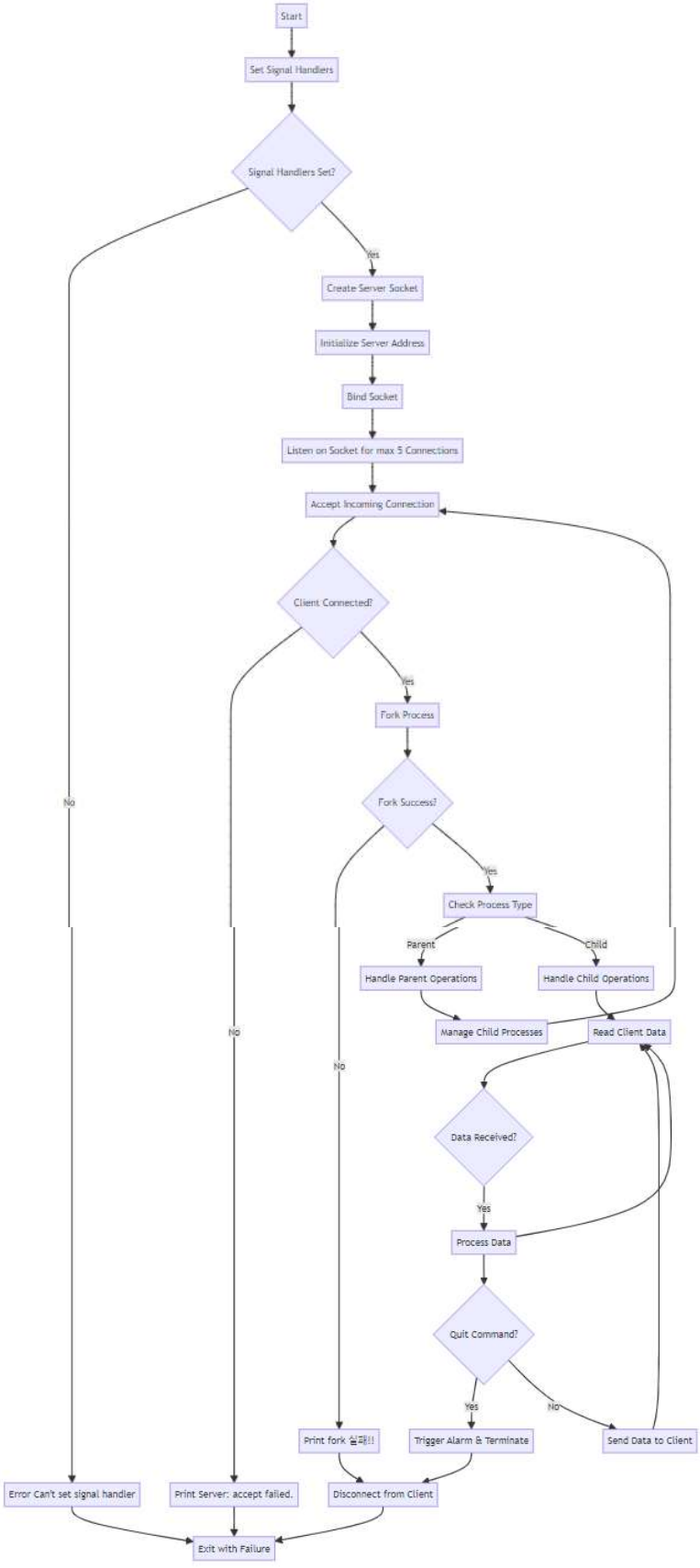
1) cli(convert to FTP command)



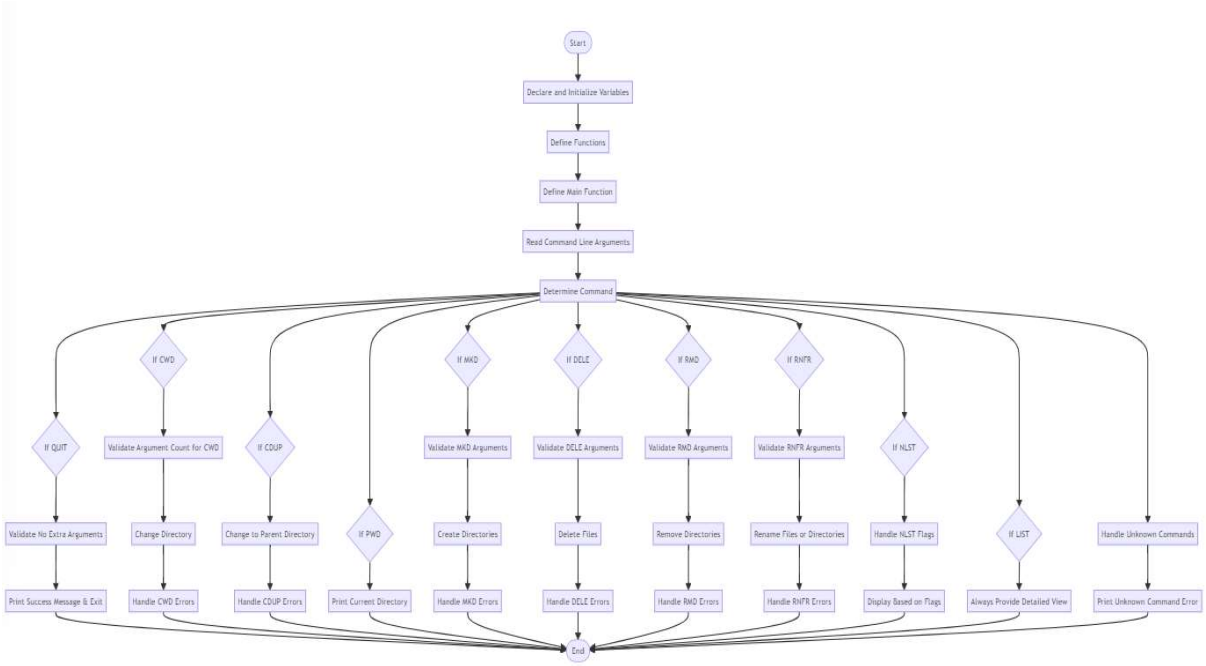
2) cli(socket 을 통해 server 와 연결)



3) srv(socket 을 통해 cli 와 연결)



4) srv(FTP 명령어 실행)



Pseudo code

1) cli(convert to FTP command)

```
int main(){
    char buffer[BUFFER_SIZE]
    ssize_t bytesRead
    char aa[SEND_SIZE]
    int idx = 0
    char FTP_instruc[SEND_SIZE] = ""

    if (the first argument is "rename")
    {
        Increment argc
    }

    Convert number of arguments to a string and store in aa
    Output the converted string to standard output

    Output the first argument to standard output
    Print a newline

    if (the current argument is "ls")
    {
        Set FTP_instruc to "NLST"
        Increment idx
    }
    else if (the current argument is "dir")
    {
        Set FTP_instruc to "LIST"
        Increment idx
    }
    else if (the current argument is "pwd")
    {
        Set FTP_instruc to "PWD"
        Increment idx
    }
    else if (the current argument is "cd")
```

```
    {
        if (there are more than 2 arguments and the next argument is "..")
        {
            Set FTP_instruc to "CDUP"
            Increment idx
        }
        else
        {
            Set FTP_instruc to "CWD"
            Increment idx
        }
    }
    else if (the current argument is "mkdir")
    {
        Set FTP_instruc to "MKD"
        Increment idx
    }
    else if (the current argument is "delete")
    {
        Set FTP_instruc to "DELE"
        Increment idx
    }
    else if (the current argument is "rmdir")
    {
        Set FTP_instruc to "RMD"
        Increment idx
    }
    else if (the current argument is "rename")
    {
        Set FTP_instruc to "RNFR"
        Output RNFR and the next argument
        Set FTP_instruc to "RNTO"
        Update idx to point to the third argument
    }
    else if (the current argument is "quit")
    {
```

```
        Set FTP_instruc to "QUIT"
        Increment idx
    }
    else
    {
        Print "Invalid command."
        Exit the program with failure
    }

    Output the determined FTP command

    for (int i = idx; i < argc; i++)
    {
        Output the argument
    }

}
```

2) cli(socket 을 통해 server 와 연결)

////////////////////////// Client-side Communication Setup //////////////////////
#define BUF_SIZE 256

```
int main(int argc, char *argv[])
{
	char buff[BUF_SIZE];
	int n;
	int sockfd;
	struct sockaddr_in serv_addr;

	sockfd = socket(AF_INET, SOCK_STREAM, 0);

	Initializes the memory of serv_addr.
	Enter server information in serv_addr.

	Connect the client and server.
	while(1) {
		write(STDOUT_FILENO, "> ", 2);
		read(STDIN_FILENO, buff, BUF_SIZE);

		if (Send the contents of the buffer to the client and succeed.) {
			if(Receives content from the client, stores it in the buffer, and succeeds.)
				printf("from server: %s", buff);
			else
				exit loop;
		} else
			 exit loop;
	}
	close endpoint;
	return 0;
}
```

3) srv(socket 을 통해 cli 와 연결)

```
//////////////////////////////////////
// Server Application Pseudocode //
//////////////////////////////////////
#define BUF_SIZE 256
char output[1024];
int length;

int client_info(struct sockaddr_in* cliaddr);

void sh_chld(int);

void sh_alrm(int);

int main(int argc, char *argv[]) {
	char buff[BUF_SIZE];
	int n;
	struct sockaddr_in server_addr, client_addr;
```

```
int server_fd, client_fd;
int len;
int port;

if (sh_alrm was called. But if an error occurs) {
	perror("Can't set signal handler");
	return 1;
}


if (The child process state changes and the sh_chld function is called. But if there is a problem) {
	perror("Can't set signal handler");
	return 1;
}

server_fd = socket(PF_INET, SOCK_STREAM, 0);

Initializes the memory of serv_addr.
Enter server information in serv_addr.
bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr));

Prepare Socket for Listening to Incoming max 5 Connections

while(1) {
	pid_t pid;
	len = sizeof(client_addr);
	client_fd = accept(server_fd, (struct sockaddr*)&client_addr, &len);

	if(Can't connect to client.){
		printf("Server: accept failed.₩n");
		return 0;
	}

	Create a new process

	if(If fork doesn't work){
		perror("fork 실패!!");
		Disconnect from the client.
	}

	else if(This is the parent process.){

				if (The client contents were printed. But if I couldn't do it)
					write(STDERR_FILENO, "client_info() err!!₩n", sizeof("client_info() err!!₩n"));

				length = sprintf(output, "Child Process ID : %d₩n", pid);
				write(STDOUT_FILENO, output, length);

				while (Check for any terminated child process without blocking);
```

```
            }
            else(){

                while(1){
                    if(Successfully receives the contents of the buffer from the client.){
                                                        The content received from the client is stored in the buffer.
                    }

                        if(!strncmp(buff, "QUIT", strlen("QUIT"))){
                            length = sprintf(output,"Child Process(PID: %d) will be terminated.₩n", getpid());
                                                                write(STDOUT_FILENO, output, length);

                            Activates sh_alrm after 1 second.
                            Disconnect from the client.
                            return 0;
                        }
                        else{
                            Send the contents in the buffer to the client.
                        }
                    }
                }
            }
            Disconnect from the client.
        }

        return 0;
}

void sh_chld(int signum) {
        printf("Status of Child process was changed.₩n");
        wait(NULL);
}

void sh_alrm(int signum) {
        printf("Child Process (PID : %d) will be terminated.₩n", getpid());
        exit(1);
}


int client_info(struct sockaddr_in* cliaddr){
        if(cliaddr->sin_family is not AF_INET)
            return -1;
              length = sprintf(output, "==========Client info==========₩n₩n");
              write(STDOUT_FILENO, output, length);
        length = sprintf(output, "client IP: %s₩n₩n₩n", inet_ntoa(cliaddr->sin_addr));
              write(STDOUT_FILENO, output, length);
        length = sprintf(output, "client port: %d₩n₩n", cliaddr->sin_port);
              write(STDOUT_FILENO, output, length);
              length = sprintf(output, "================================₩n");
```

```
            write(STDOUT_FILENO, output, length);
    return 1;
}
```

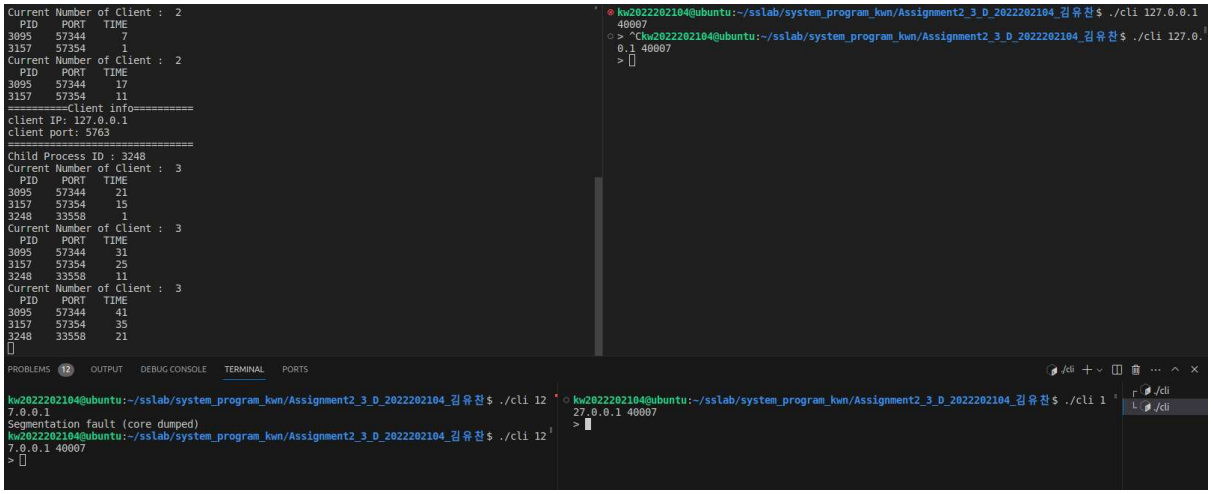
4) srv(FTP 명령어 실행)

```
int main{
    char buffer[BUFFER_SIZE];
    ssize_t bytesRead;
    char *cvalue = NULL;

    read(0, buffer, SEND_SIZE);
    int argc = atoi(buffer);

    int numStrings = SEND_SIZE;
    char **argv = malloc(numStrings * sizeof(char*));
    for (int i = 0; i < numStrings; i++) {
        argv[i] = malloc(256 * sizeof(char));
    }
    int i = 0;

    If (the command is QUIT) {
        Validate (no additional arguments are present)
        If (valid) {
            Print "QUIT success" and exit
        } Else {
            Report error and exit
        }
    }

    If (the command is CWD) {
        Validate (exactly one additional argument is present: the directory path)
        Attempt to change directory to provided path
        If (successful) {
            Update and display current directory
        } Else {
            Report error and exit
        }
    }
```

```
    If (the command is PWD) {
        Validate (no additional arguments are present)
        Fetch and print current working directory
        Handle errors appropriately
    }

    If (the command is MKD) {
        Validate (at least one additional argument: directory names to create)
        Attempt to create each directory
        Report success or failure for each directory
        Handle errors appropriately
    }

    If (the command is DELE) {
        Validate (at least one additional argument: file names to delete)
        Attempt to delete each file
        Report success or failure for each file
        Handle errors appropriately
    }

    If (the command is RMD) {
        Validate (at least one additional argument: directory names to remove)
        Attempt to remove each directory
        Report success or failure for each directory
        Handle errors appropriately
    }
```

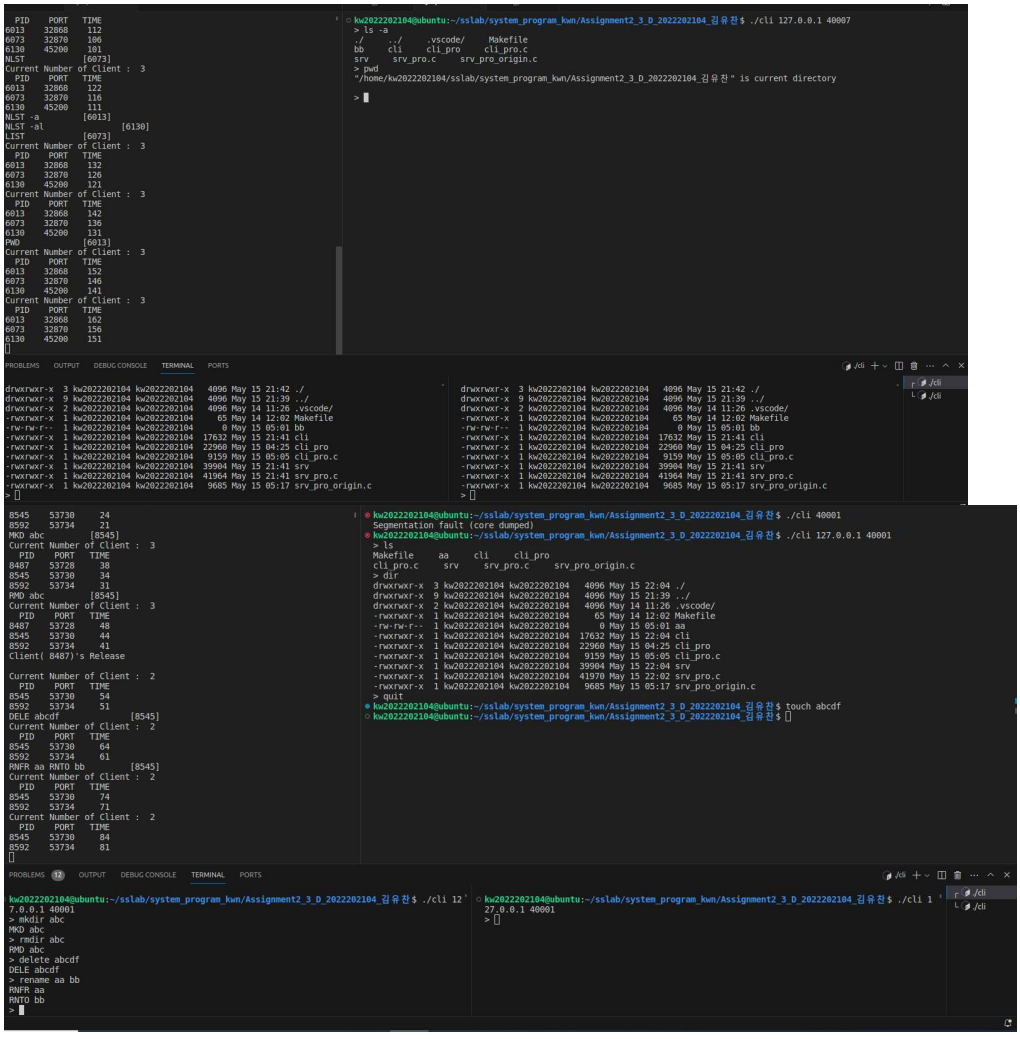
```
    If (the command is RNFR) {
        Validate (exactly two additional arguments: old and new names)
        Check that the new name does not exist
        Attempt to rename the file or directory
        Report success or failure
        Handle errors appropriately
    }

    If (the command is NLST or LIST) {
        Process flags and directory arguments
        Open specified directory
        Depending on flags, list all files or directories, including hidden ones if specified
        Display file details if requested
        Handle errors such as permission issues or non-existing directories
    }

    If (no valid command is recognized) {
        Report "This command does not exist."
    }
}
```

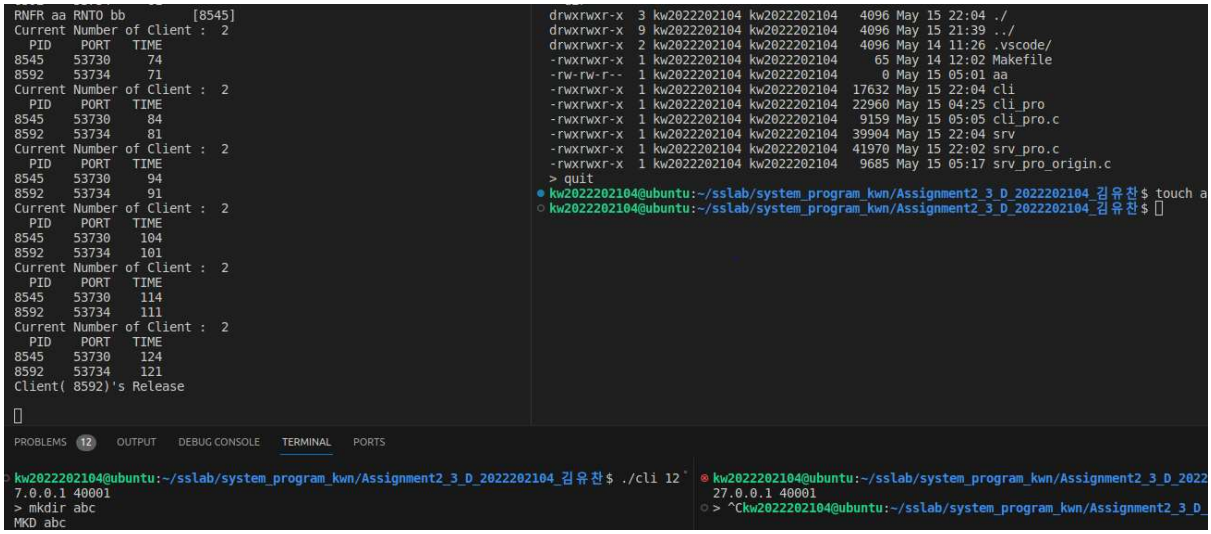
결과화면



1. Client 와 연결되었을 때 Client 정보를 server 쪽에서 출력하는 동시에 지금까지 연결된 Client 정보를 출력한다.

2. Server 쪽에서 10 초마다 연결하고 있는 Client 들의 정보를 출력한다. 만약 Client 와 연결이 되면 그 때 기준으로 다시 10 초마다 출력한다.





3. quit 나 Ctnl+C 를 client 에서 입력하면 server 쪽과 client 쪽에서 서로 연결을 끊는다.

4. ls, pwd, dir, cd, mkdir, delete, rmdir, rename 의 명령어를 client 에서 FTP 명령어로 바꾸고 server 에서 그 명령어 받아 실행하고 client 한테 실행결과를 전달하여 client 에서 결과가 출력한다.

고찰

지금까지 한 과제를 하나로 묶고 Sigint 시스템콜을 이용해서 Ctnl+C 입력을 받았을 때 다른 작업을 할 수 있도록 만들었다. Client 는 QUIT 명령어처럼 server 와의 연결을 끊고 process 를 종료하는 작업을 하고 linked list 를 만들어서 client 정보를 저장하여 server 에서 10 초마다 혹은 연결될 때마다 연결중인 client 정보를 출력하게 만들었다. 과제를 하나로 묶는 작업을 제외하고 크게 새로운 개념이 추가되거나 어려운 작업은 없었고 지금까지 했던 것을 정리할 수 있는 시간이 되었다. 하지만 같은 내용이지만 다른 방식으로 접근해서 했기 때문에 코드가 서로 달라서 일일이 수정하고 그 수정하면서 에러난 부분을 찾는 것이 많이 힘들었다. 처음에 코딩할 때 충분히 검토하고 결정해서 그 방식대로 계속 해야한다.

Reference

시스템프로그래밍 실습 강의자료