

시스템 프로그래밍 실습

**[FTP2-2]**

Class : D

Professor : 최상호 교수님

Student ID : 2020202092

Name : 강현민

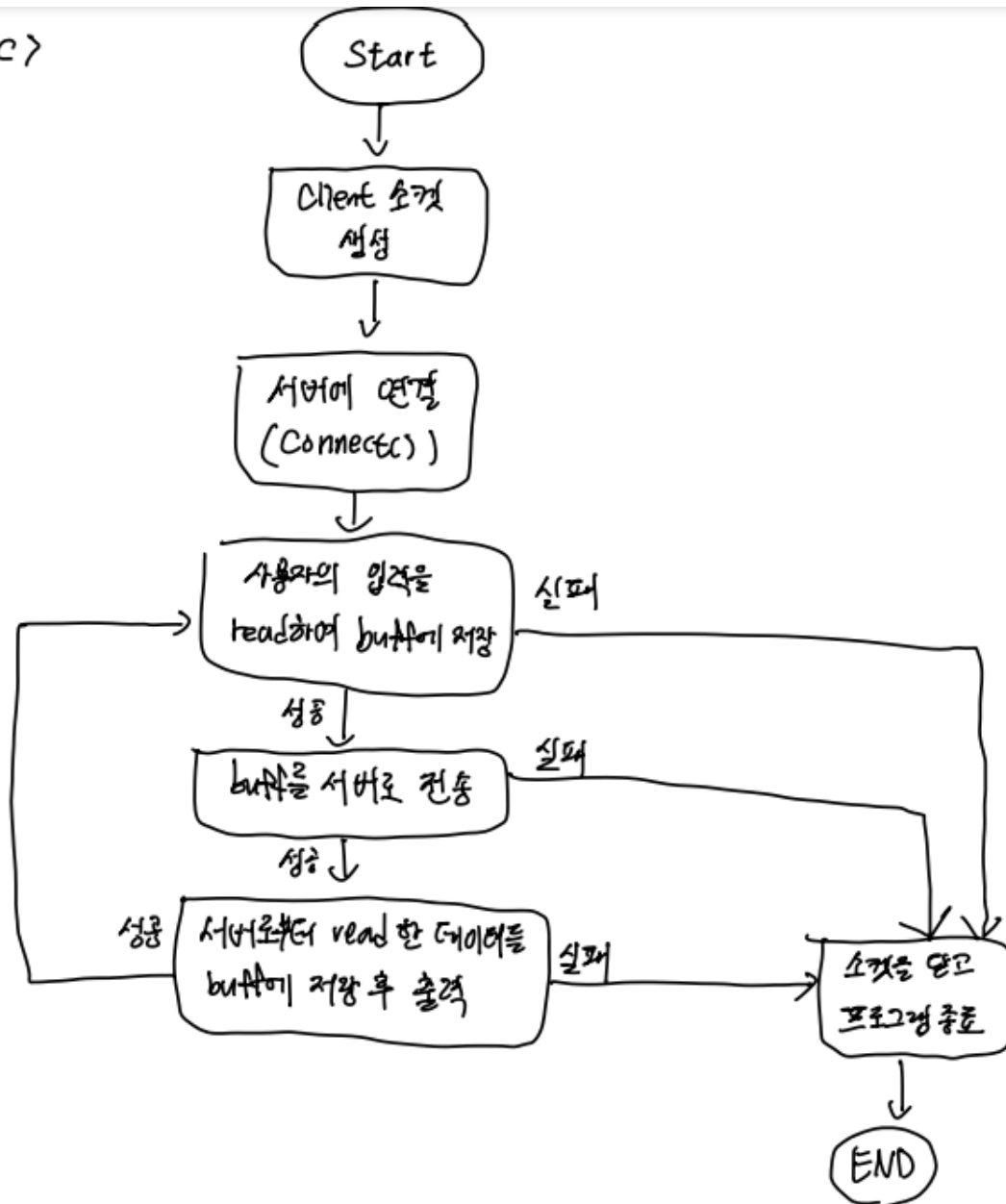
# Introduction

이번 [FTP 2-2](#) 과제는 FTP 서버 구현을 최종 목표로 하는 프로젝트의 두번째 단계인 소켓 프로그래밍 중 fork 함수 및 시그널 관련 함수를 이용하는 과제로, 클라이언트에서 문자열을 입력하면 문자열이 서버로 전송되고, 서버에서 이 문자열을 다시 클라이언트로 전송하여 출력한다. 이 때 "QUIT" 문자열이 입력될 경우, 서버에서 SIGALRM 시그널을 호출하여 1 초 뒤에 해당 프로세스가 종료되고, 이에 따라 SIGCHLD 시그널을 호출하여 자식 프로세스가 종료된 것을 알린다. 이번 과제를 통해 fork 함수 및 시그널 함수 사용 용법을 익혀 최종적으로 모든 명령어에 대해 응용을 할 수 있도록 하는 것을 목표로 한다.

# Flow chart

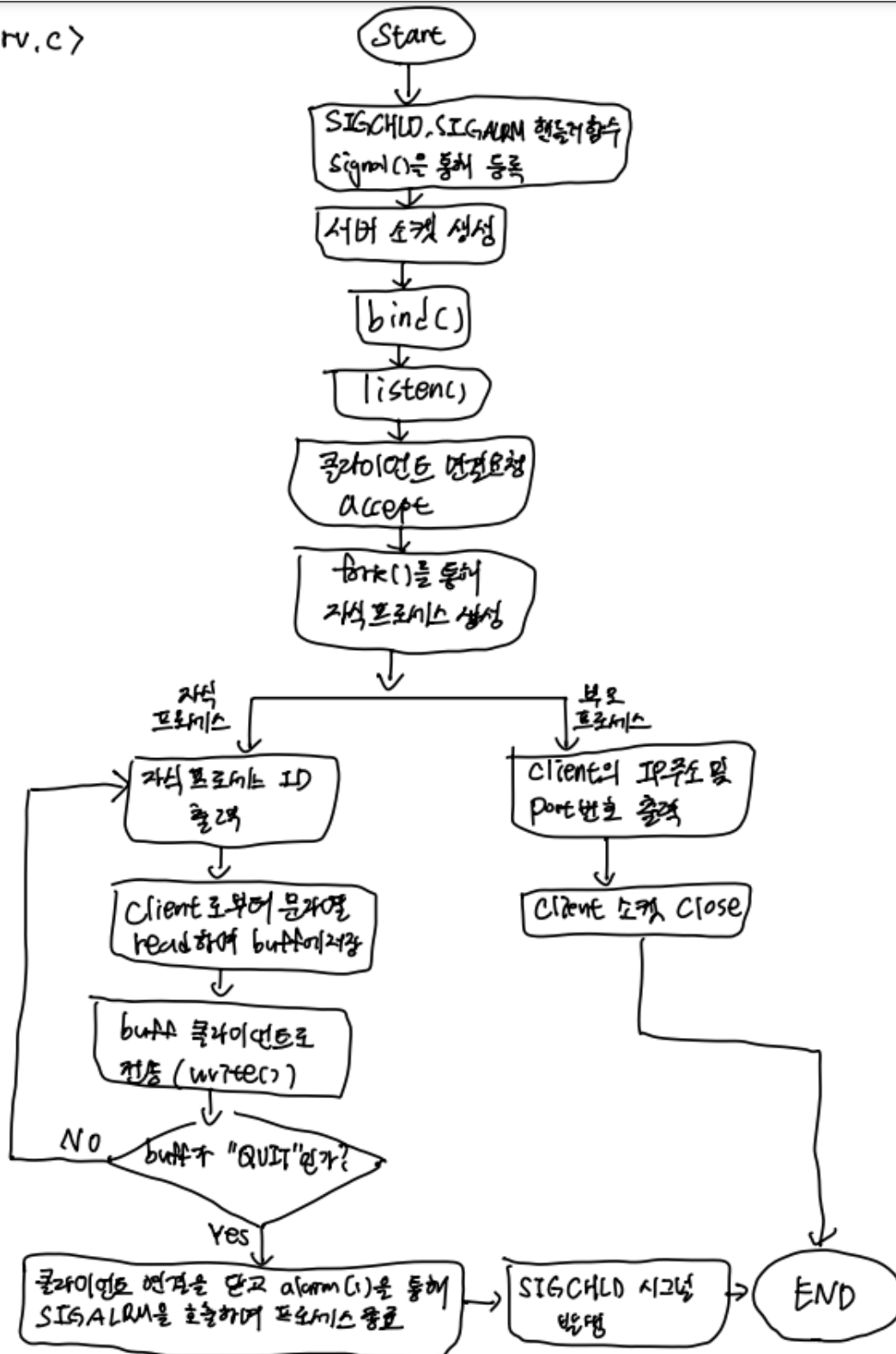
<cli.c>

<cli.c>

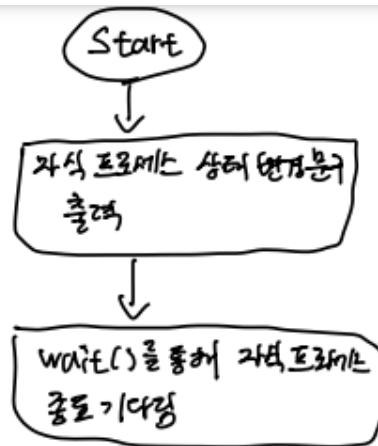


<srv.c>

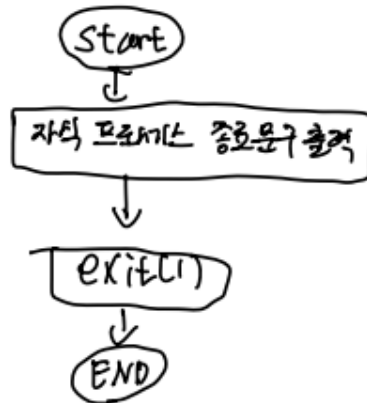
<srv.c>



<sh-child>



<sh-alm>



# Pseudo code

<cli.c>

```
Int main(int argc, char** argv) {
```

```
    If IP address and port number are not inputted
```

```
        Print error message
```

```
        Return 1;
```

```
    Declare buff, sockfd, struct sockaddr_in
```

```
    Create socket
```

```
    Store AF_INET, IP address and port number into struct
```

```
    Connect to server
```

```
    While(1)
```

```
        Read user input and store into buff
```

```
        Send message to the server
```

```
        Receive response from the server
```

```
    Close sockfd
```

```
    Return 0;
```

<srv.c>

Void sh\_chld(int);

Void sh\_alm(int);

Void client\_info(const struct sockaddr\_in\* cliaddr)

Print client IP address and port number

Int main(int argc, char\*\* argv) {

Declare buff, client\_fd, server\_fd, struct;

Signal(SIGCHLD, sh\_chld);

Signal(SIGALRM, sh\_alm);

Bind the socket to the address

Listen for incoming connections

While (1) {

Accept incoming connections

Fork ( );

If child process

Print child process ID;

While (1)

Read from client

write back to client

if buff is "QUIT"

close client connection

Trigger SIGALRM in 1 second using alarm(1);

While (1);

Close client\_fd

Exit(0);

Else if parent process

Client\_info( );

Else perror(fork failed);

Close client socket

Return 0;

Void sh\_chld(int signum)

Print "Status of Childprocess was changed."

Wait();

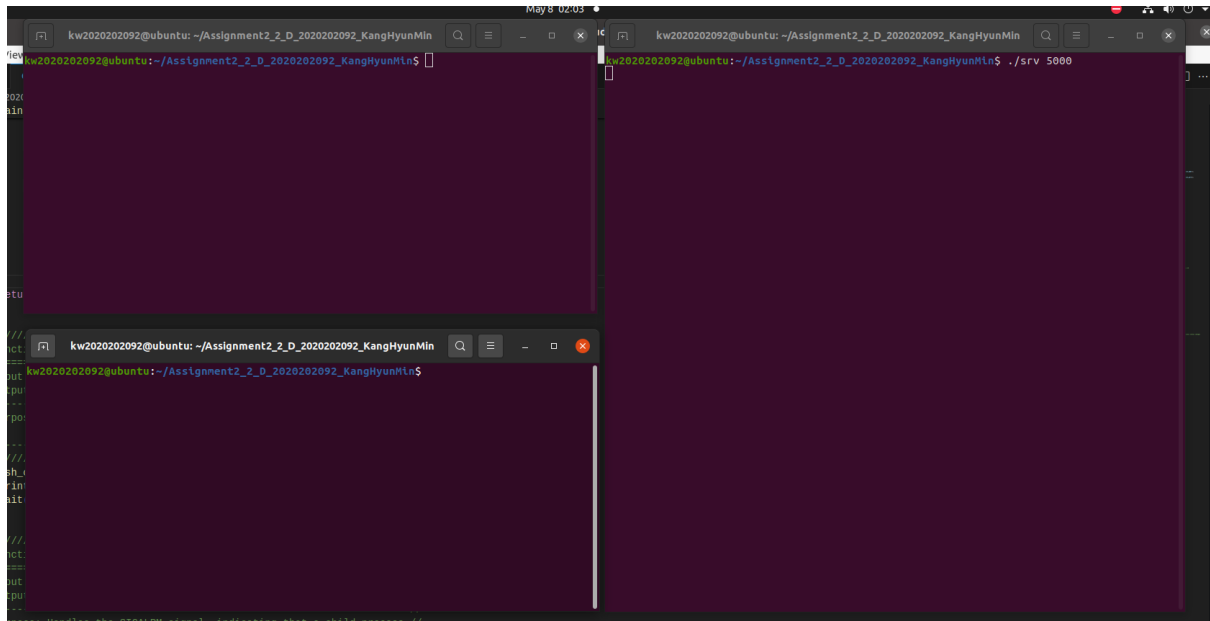
Void sh\_alrm(int signum)

Print "child process will be terminated."

Exit(1);

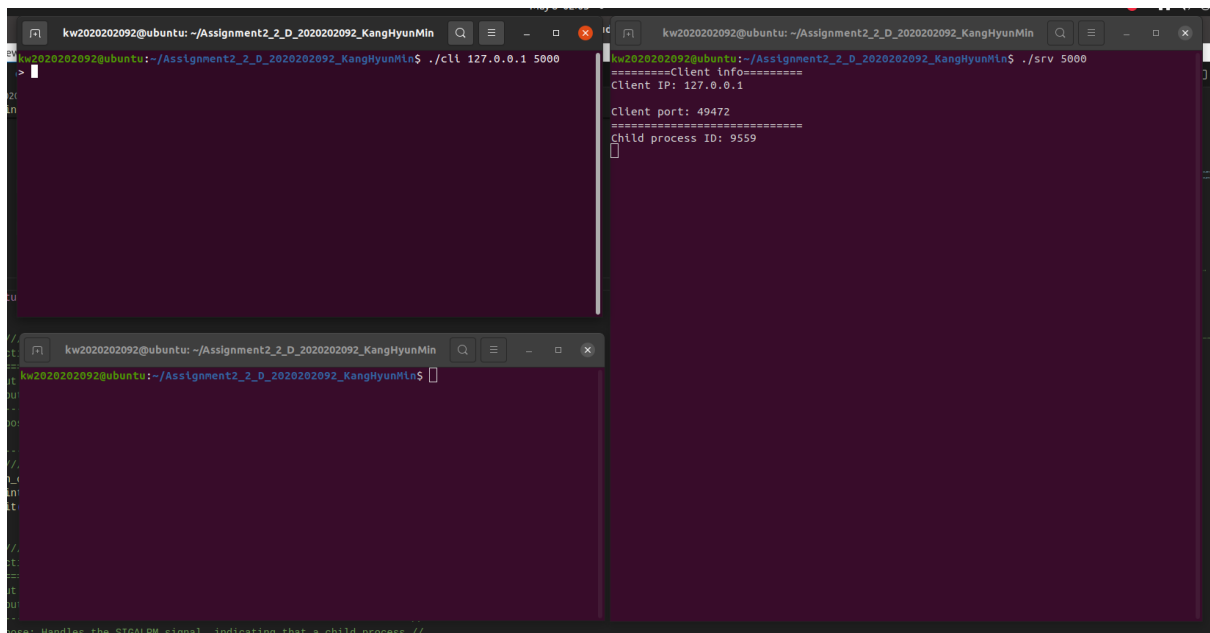


## 결과화면



The image shows three terminal windows. The top-left window shows the prompt `kw2020202092@ubuntu: ~/Assignment2_2_D_2020202092_KangHyunMin`. The top-right window shows the command `./srv 5000` being executed. The bottom window shows the command `./cli 127.0.0.1 5000` being executed.

먼저 포트 번호 5000 을 인자로 하여 서버를 실행한다.



The image shows three terminal windows. The top-left window shows the command `./cli 127.0.0.1 5000` being executed. The top-right window shows the output of the server program: `====Client info=====`, `Client IP: 127.0.0.1`, `Client port: 49472`, and `Child process ID: 9559`. The bottom window shows the prompt `kw2020202092@ubuntu: ~/Assignment2_2_D_2020202092_KangHyunMin`.

그 후 IP address 를 127.0.0.1, 포트 번호를 동일하게 5000 으로 하여 클라이언트를 실행하여 서버에 연결한다.

```
kw2020202092@ubuntu: ~/Assignment2_2_D_2020202092_KangHyunMin
kw2020202092@ubuntu:~/Assignment2_2_D_2020202092_KangHyunMin$ ./cli 127.0.0.1 5000
> this is test1
from server: this is test1
>

kw2020202092@ubuntu:~/Assignment2_2_D_2020202092_KangHyunMin$ ./srv 5000
=====Client info=====
Client IP: 127.0.0.1
Client port: 49472
=====
Child process ID: 9559
[]

kw2020202092@ubuntu:~/Assignment2_2_D_2020202092_KangHyunMin$
```

이후 클라이언트에 “this is test1” 문자열을 입력했을 때, 이 문자열이 서버로 전송되고 다시 서버로부터 클라이언트로 전송되어 동일한 문자열이 클라이언트에 출력되는 것을 확인할 수 있다.

```
kw2020202092@ubuntu:~/Assignment2_2_D_2020202092_KangHyunMin
kw2020202092@ubuntu:~/Assignment2_2_D_2020202092_KangHyunMin$ ./cli 127.0.0.1 5000
> this is test1
from server: this is test1
> QUIT
from server: QUIT
>

kw2020202092@ubuntu:~/Assignment2_2_D_2020202092_KangHyunMin$ ./srv 5000
=====Client info=====
Client IP: 127.0.0.1
Client port: 49472
=====
Child process ID: 9559
Child Process(PID : 9559) will be terminated.
Status of Child process was changed.
[]

kw2020202092@ubuntu:~/Assignment2_2_D_2020202092_KangHyunMin$
```

그 후 QUIT 을 입력했을 때, SIGALRM 이 호출되어 9559 를 ID 로 하는 프로세스가 종료된다는 메시지가 출력되고, 종료되는 것을 확인할 수 있다.

```
kw2020202092@ubuntu: ~/Assignment2_2_D_2020202092_KangHyunMin$ ./cli 127.0.0.1 5000
> this is test1
from server: this is test1
2> QUIT
from server: QUIT
>

kw2020202092@ubuntu: ~/Assignment2_2_D_2020202092_KangHyunMin$ ./srv 5000
=====Client info=====
Client IP: 127.0.0.1

Client port: 49472
=====
Child process ID: 9559
Child Process(PID : 9559) will be terminated.
Status of Child process was changed.
=====Client info=====
Client IP: 127.0.0.1

Client port: 43076
=====
Child process ID: 9566
[]

kw2020202092@ubuntu: ~/Assignment2_2_D_2020202092_KangHyunMin$ ./cli 127.0.0.1 5000
>
from server: this is test2
> QUIT
from server: QUIT
>
```

그 후 다른 클라이언트를 같은 인자로 하여 실행했을 때, 9566 을 ID 로 하는 프로세스가 생성 되는 것을 확인할 수 있다.

```
kw2020202092@ubuntu: ~/Assignment2_2_D_2020202092_KangHyunMin$ ./cli 127.0.0.1 5000
> this is test1
from server: this is test1
2> QUIT
from server: QUIT
>

kw2020202092@ubuntu: ~/Assignment2_2_D_2020202092_KangHyunMin$ ./srv 5000
=====Client info=====
Client IP: 127.0.0.1

Client port: 49472
=====
Child process ID: 9559
Child Process(PID : 9559) will be terminated.
Status of Child process was changed.
=====Client info=====
Client IP: 127.0.0.1

Client port: 43076
=====
Child process ID: 9566
Child Process(PID : 9566) will be terminated.
Status of Child process was changed.
[]

kw2020202092@ubuntu: ~/Assignment2_2_D_2020202092_KangHyunMin$ ./cli 127.0.0.1 5000
> this is test2
from server: this is test2
> QUIT
from server: QUIT
>
```

그 후 위의 결과화면과 동일하게 문자열을 하나 입력한 후 QUIT 을 입력했을 때, 동일한 결과가 출력되는 것을 확인할 수 있다.

## 고찰

이번 [FTP 2-2](#) 과제를 통해 fork 함수와 시그널 관련 함수를 사용하여 클라이언트와 서버 구조의 데이터 전송을 구현하는 코드를 작성했는데, fork 함수의 기본 사용 방법 및 프로세스의 반환 값을 이용하여 자식 프로세스와 부모 프로세스의 동작을 나누어 구현하는 방법에 대해서 학습할 수 있었다. 또한, signal() 함수를 통해 시그널 handler를 등록하는 방법에 대해 학습할 수 있었으며, alarm 함수를 통해 SIGALRM 시그널을 발생시킨 후, 이에 따라 시그널 핸들러 함수가 호출되는 과정을 코드로 구현함으로써 시그널 관련 함수의 사용 방법 및 구현 방법에 대해 학습하고 응용할 수 있는 기회가 되었다.

## Reference