시스템 프로그래밍 실습

[Assignment2-2]

Class : **[A]**

Professor : **[김태석 교수님]**

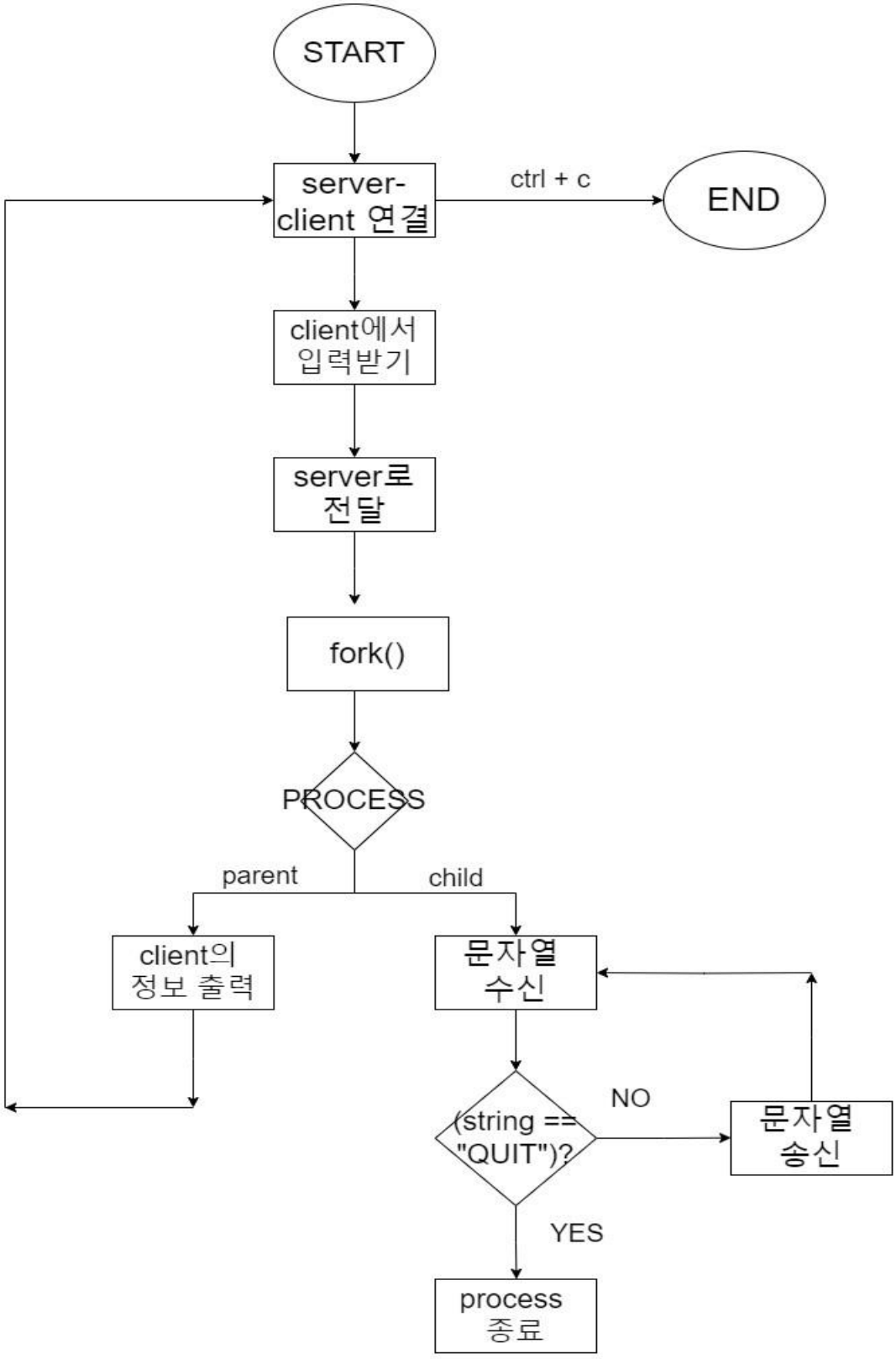
Student ID : **[2019202032]**

Name : **[이상현]**

Introduction

해당 과제는 client 와 server 소켓을 연결하고 client 에서는 사용자로부터 입력을 받는다. server 에서는 fork 를 통해 child process 를 만든다. Parent process 에서는 연결된 client 의 정보를 출력하고 다음 client 와 연결을 준비한다. Child process 에서는 client 로부터 문자열을 주고받는 과정을 반복한다. 만약 quit 이 입력으로 들어올 경우 해당 client 와 연결을 종료하고 sigalrm 을 호출하고 1 초뒤에 해당 프로세스도 종료한다. 또한 새로운 process 가 생성될 때마다 해당 process 의 pid 를 출력한다.

Flow chart


START
server-client 연결
ctrl + c
END
client에서 입력받기
server로 전달
fork()
PROCESS
parent
child
client의 정보 출력
문자열 수신
(string == "QUIT")?
NO
문자열 송신
YES
process 종료

Pseudo code

<srv.c>

```
int main(int argc, char **argv)
{
    /*              prepare server socket and connect with client socket              */
      server_fd = socket(PF_INET, SOCK_STREAM, 0);
      int opt = 1;
      setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR, &opt, sizeof(opt));

      memset(&server_addr, 0, sizeof(server_addr));
      server_addr.sin_family = AF_INET;
      server_addr.sin_addr.s_addr = htonl(INADDR_ANY);//set address
      server_addr.sin_port = htons(atoi(argv[1]));//set port
      ////////////////////////////////////////////////////////////////////////////////

      if(bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0){   //bind socket
            printf("Server: Can't bind local address₩n");
            return 0;
      }
      listen(server_fd, 5);     //listen from client
/***************************** communicate with client *****************************/
      while(1)
      {
```

```
client_fd = accept(server_fd, (struct sockaddr *)&client_addr, &len);

/************************* Parent process *********************/
if((pid = fork())> 0) {
      print client information
      int ret = waitpid(pid, 0, 0);//wait for child process to be terminated
      sh_chld(0);//signal handling
}
/*************************************************************/


/************************* child process ***********************/
else{
      printf("Child Process ID : %d₩n", getpid());       //print child process ID
      while(1){
          receive string;
            /********** if received string is QUIT *************/
            if(!strcmp(temp, "QUIT")){
                  sh_alrm(getpid());    //signal handling
                  break;
            }
            /***************************************************/
```

```
                write(client_fd, buff, strlen(buff));     //write(send) string to client
            }
        }
        /*****************************************************************/
    }
}
```

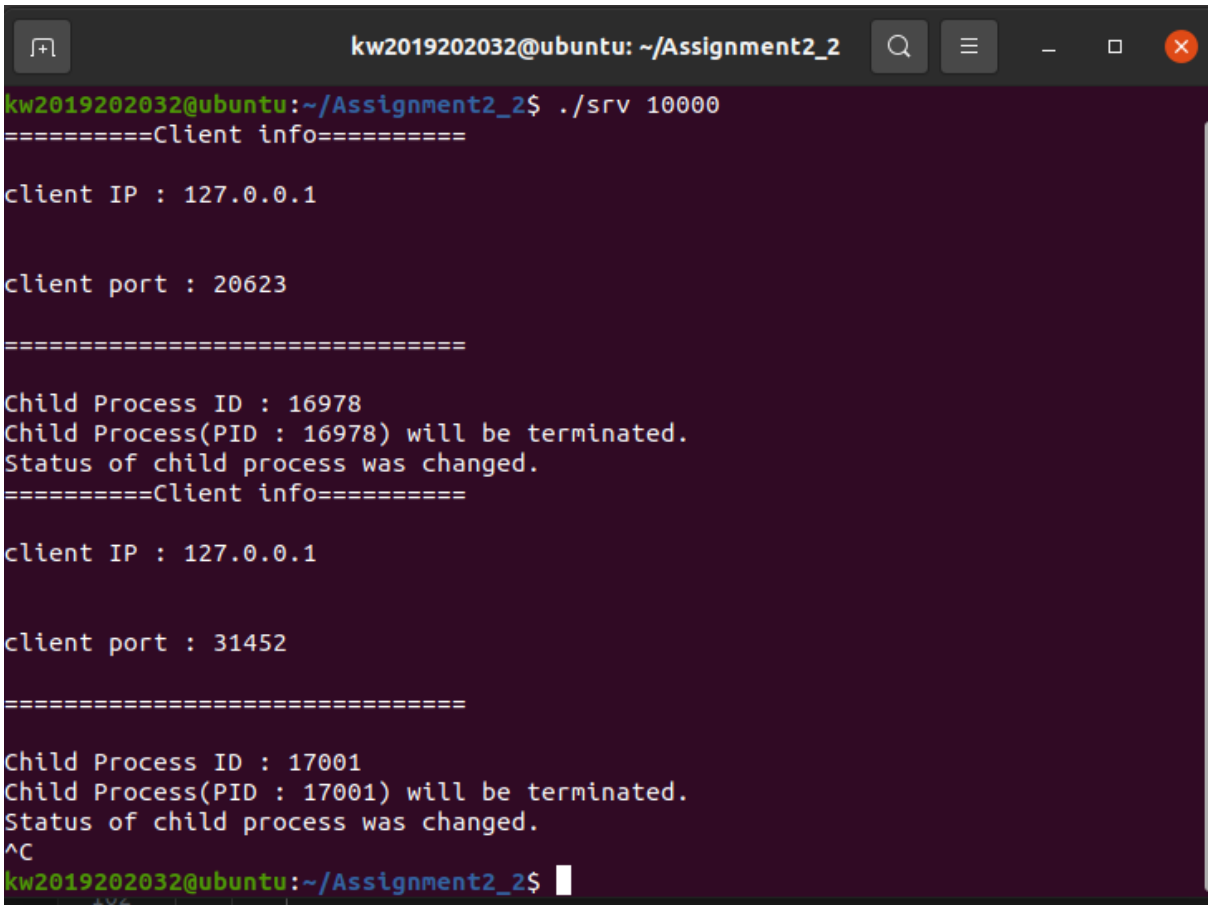
<cli.c>

```
int main(int argc, char**argv)
{
    /********************* prepare client socket **********************/
    sockfd = socket(AF_INET, SOCK_STREAM, 0);

    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = inet_addr(argv[1]);
    serv_addr.sin_port = htons(atoi(argv[2]));
    connect(sockfd, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
    /*****************************************************************/

    while(1){
        receive string from user;
        if(write(sockfd, buff, BUF_SIZE) > 0){   /* send string to server */
            memset(buff, 0, BUF_SIZE);
            if((n = read(sockfd, buff, BUF_SIZE)) > 0) { /* receive string from server */
```

```
                buff[strlen(buff)] = '₩0';
                printf("from server:%s", buff);
                memset(buff, 0, BUF_SIZE);
            }
            else       /* error handling for read */
            {
                close(sockfd);
                shutdown(sockfd, SHUT_RDWR);
                exit(0);
            }
        }
        else /* write error handling */
        {
            close(sockfd);
            exit(0);
        }
    }
    close(sockfd);
    return 0;
}
```

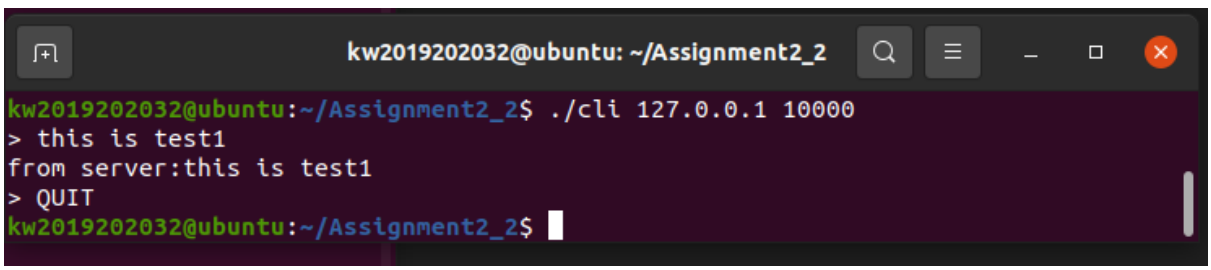
결과화면

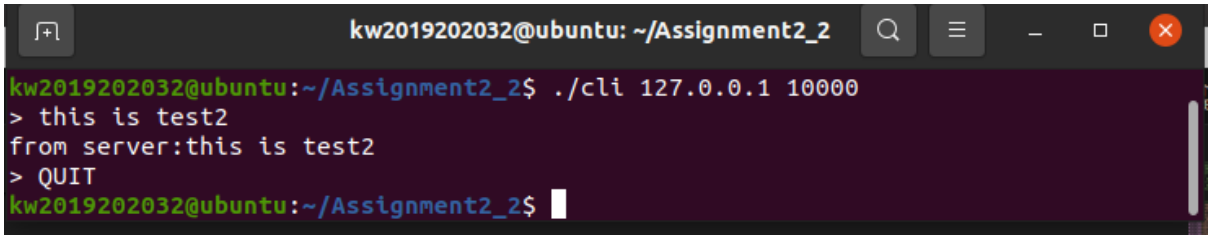
<server>



다음과 같이 client 의 연결을 확인하고 child process 로부터 SIGALRM 을 통해 child process 를 종료하고, parent process 로부터 SIGCHLD 를 통해 child process 가 종료되었음을 알려주는 것을 확인할 수 있다.

<client>





client 에서 server 와 연결하고 문자열을 송수신한 뒤 QUIT 라는 문장을 통해 server 의 child process 를 종료시키는 것을 확인할 수 있다.

고찰

해당 과제는 server 에서 client 와 연결이 되면 먼저 부모 프로세스에서 client 의 정보를 출력하고, 이후 child process 에서 문자열을 주고받은 후, child process 가 종료되면 parent process 에서 child process 의 상태가 변경됐다는 문장을 출력하고 다시 다음 연결을 준비해야한다. 이 과정에서 가장 어려웠던 점은 처음에는 부모 process 에서 정보를 먼저 출력한 뒤, child process 의 동작이 끝난 뒤 어떻게 다시 연결을 받을 것인가에 대한 부분이었다. 이 부분에 대해서 고민한 결과 wait 함수를 사용하여 child process 가 종료되기를 기다린 후, child process 가 종료되면 child process 의 상태를 회수하는 방식으로 해결하였다. 해당 과제를 통해 client 로부터 server 에 많은 접속 요청이 들어온다고 하더라도 fork 를 사용해 multithread 로 해결할 수 있을 것이라는 생각을 하게되었다.

Reference

강의자료만 참고하였습니다.