시스템 프로그래밍 실습

7차 과제

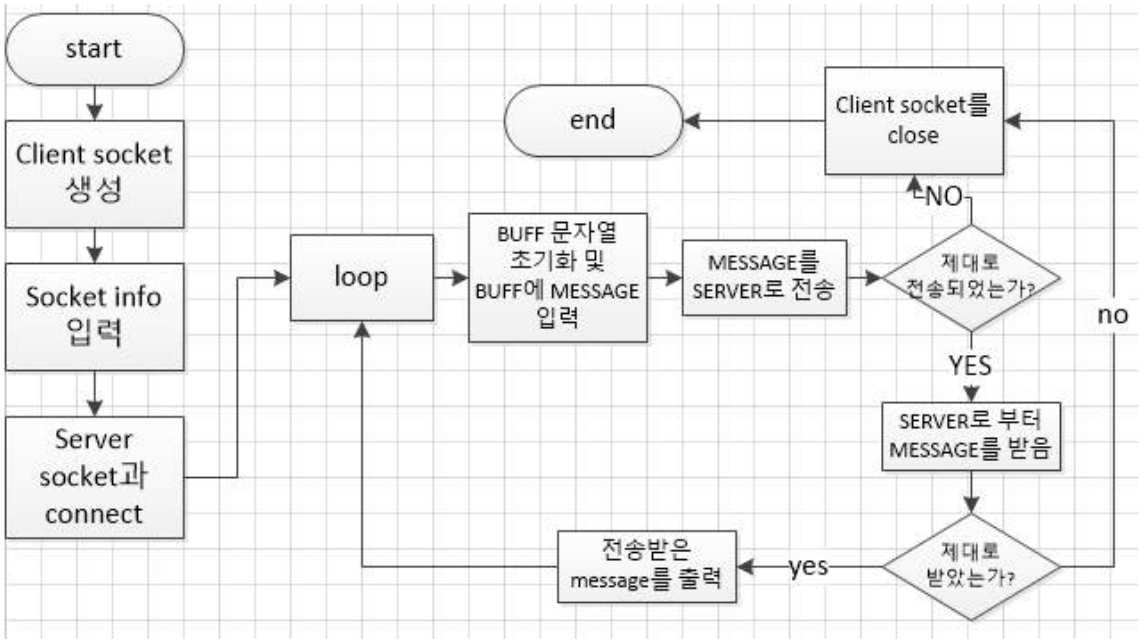
실습 일시 : 화 1,2
담당 교수님 : 김태석 교수님
학번 : 2013722095
이름 : 최재은
실습 번호 : practice #2-2

■ Introduction

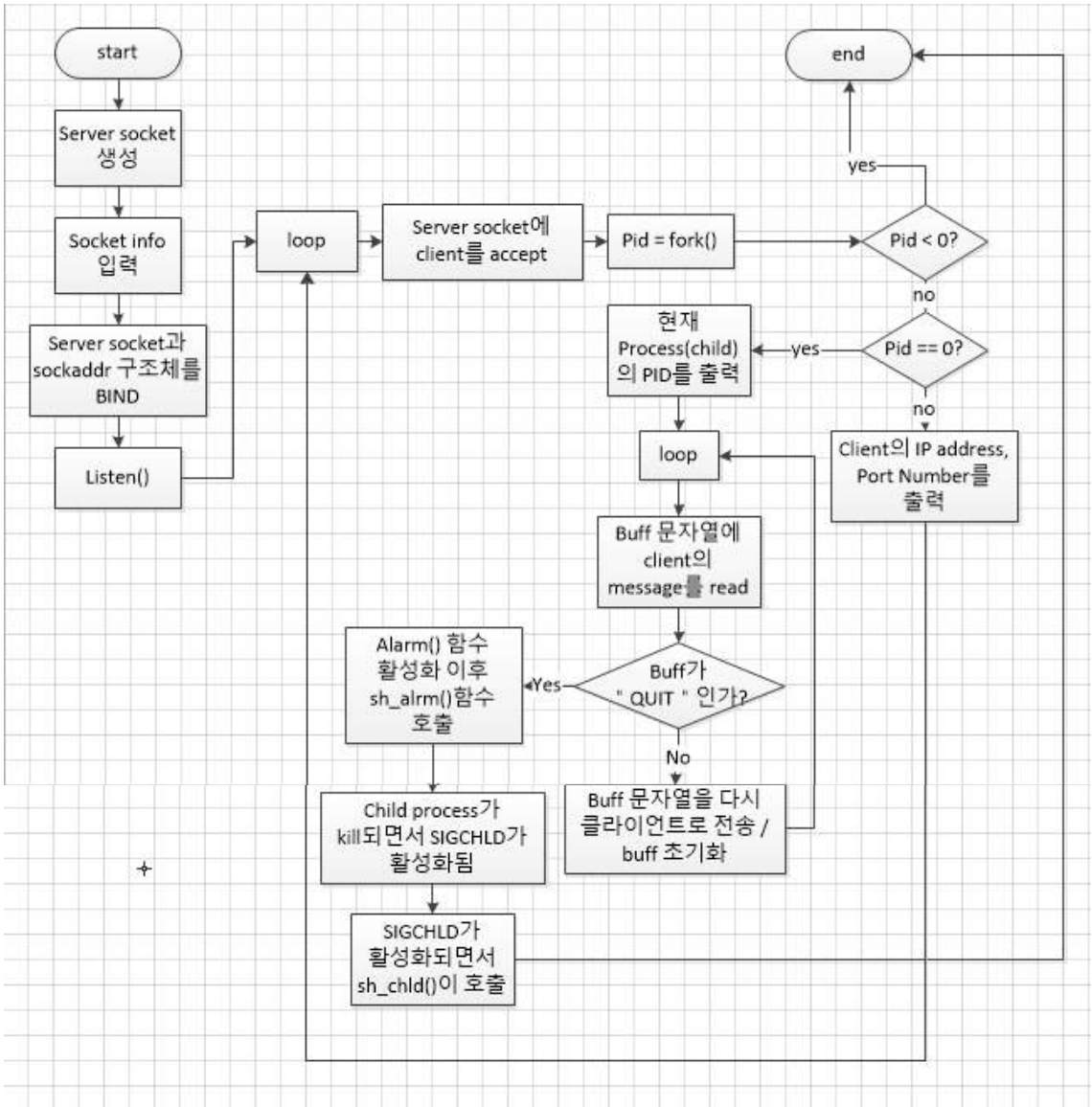
Client / Server 통신을 활용하여 client가 server로 전송한 메시지를 다시 받아서 출력하는 echo ftp를 구현해 본다.

■ Flow Chart

1. cli.c



2. srv.c


start
Server socket 생성
Socket info 입력
Server socket과 sockaddr 구조체를 BIND
Listen()
loop
Server socket에 client를 accept
Pid = fork()
Pid < 0?
yes
end
no
Pid == 0?
yes
현재 Process(child)의 PID를 출력
no
Client의 IP address, Port Number를 출력
loop
Buff 문자열에 client의 message를 read
Buff가 " QUIT " 인가?
Yes
Alarm() 함수 활성화 이후 sh_alrm()함수 호출
No
Buff 문자열을 다시 클라이언트로 전송 / buff 초기화
Child process가 kill되면서 SIGCHLD가 활성화됨
SIGCHLD가 활성화되면서 sh_chld()이 호출

■ Source Code

1. cli.c

```
#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 <netinet/in.h>
#include <signal.h>
#include <string.h>
#define BUF_SIZE 256

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

        /* creat client socket*/
        sockfd = socket(AF_INET, SOCK_STREAM, 0);

        /* set socket information */
        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 with server socket */
        connect(sockfd,(struct sockaddr *)&serv_addr,sizeof(serv_addr));

        while(1)
        {
                memset(buff, 0, BUF_SIZE); // initialize buff array
                write(STDOUT_FILENO, ">", 2);
                read(STDIN_FILENO, buff, BUF_SIZE); // recieve sentence from user

                if(write(sockfd, buff, BUF_SIZE) > 0)      // pass buff to server
                {
                        if(read(sockfd, buff, BUF_SIZE) > 0)
                        // recieve sentence from server
                                printf("from server : %s", buff);
                        else
                                break;
                }
                else
                        break;
        }
```

```
        close(sockfd); // close client socket
        return 0;
}
```

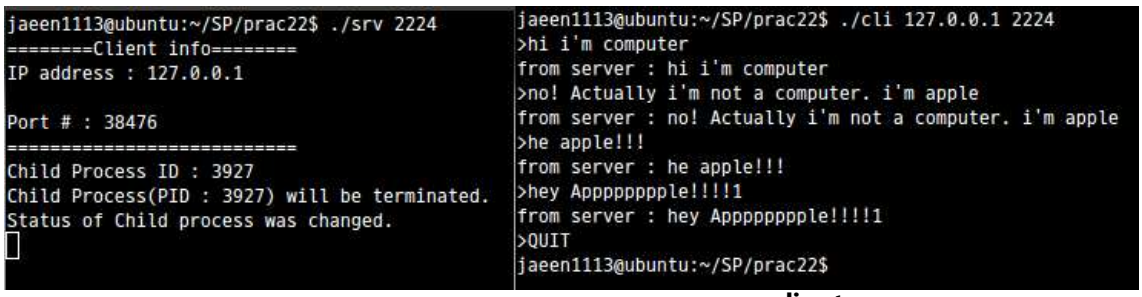
2. srv.c

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

#define BUF_SIZE 256

void sh_chld(int); // signal handler for SIGCHLD
void sh_alrm(int); // signall handler for SIGALRM
int client_info(struct sockaddr_in*); // print client's info

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;

        signal(SIGALRM,sh_alrm); // if alarm() function is activated, excute sh__alrm() function
        signal(SIGCHLD, sh_chld); // if child process' is changed, excute sh_chld() function

        /* create server socket */
        server_fd = socket(PF_INET, SOCK_STREAM, 0);

        /* set server socket's information */
        memset(&server_addr, 0 , sizeof(server_addr));
        server_addr.sin_family = AF_INET;
        server_addr.sin_addr.s_addr = htons(INADDR_ANY);//htonl(INADDR_ANY);
        server_addr.sin_port = htons(atoi(argv[1]));

        /* bind server socket with address */
        bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr));

        /* prepare connection */
        listen(server_fd, 5);
```

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


		if((pid = fork())< 0)//create child process
		{
			printf("fork() err ₩n");
			exit(1);
		}

		else if(pid == 0) // child process
		{
			/* print child's process id */
			printf("Child Process ID : %d₩n", getpid());

			while(1)
			{

				n = read(client_fd, buff, BUF_SIZE);		// recieve message

				if(!strcmp(buff, "QUIT₩n")) // if message is 'QUIT'
				{
					alarm(2); // activate alarm function after 2sec
					break;			// break loop
				}

				write(client_fd, buff, n); // message re-transfer
				memset(buff, 0, BUF_SIZE); // initialize buff array
			}
		}

		else  // parent process
		{

			client_info(&client_addr); // print child process' infomation
		}

		close(client_fd);			// close client socket
	}

	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* clientaddr)
{
		char temp[200];
		char ip[20] ;
		int port;
		if(inet_ntoa(clientaddr->sin_addr) < 0) return -1;
		else strcpy(ip, inet_ntoa(clientaddr->sin_addr));
		if(ntohs(clientaddr->sin_port) < 0) return -1;
		else port = ntohs(clientaddr->sin_port);

		sprintf(temp, "========Client info========₩nIP address : %s₩n₩nPort # : %d
₩n============================₩n", ip, port);
		write(STDOUT_FILENO, temp, strlen(temp)); // print input info
		return 0;

}
```

■ Result



<server> **<client>**

- server와 client가 연결되면 client의 IP address, port number가 출력 된다
- fork를 통해 child process가 생기면서 그 PID를 출력해주고, client에서 입력한 메시지가 다시 돌아오는 echo 기능을 수행하는 것을 볼 수 있다.
- QUIT라는 명령을 치면 클라이언트를 종료하면서 서버에서는 종료할 child process에 관한 문구를 출력한다.