TI DSP, MCU 및 Xilinx Zynq FPGA
프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 윤연성
whatmatters@naver.com

//실행 ls
// op_client.c   op_server.c
// gcc -o op_server op_server.c
// gcc -o op_client op_client.c
// ./op_server 7777

~~~~~~~~~서버영역에서 컴파일 할떄

ys@ys-Z20NH-AS51B5U:~/my_proj/27day$ gcc -o op_server op_server.c
ys@ys-Z20NH-AS51B5U:~/my_proj/27day$ ./op_server 7777

~~~~~~~~~클라이언트영역에서 컴파일 할때

ys@ys-Z20NH-AS51B5U:~/my_proj/27day$ gcc -o op_client op_client.c
ys@ys-Z20NH-AS51B5U:~/my_proj/27day$ ./op_client 127.0.0.1 7777
Conneted .....
Operand Cnt: 3
Operand 1:1
Operand 2:3
Operand 3:5
Operator : +
OPeration result : 9

서버에서 계산하여 클라이언트로 넘김

클라이언트는 값을 받아서 출력만 해주어야됨
보안성 때문
모든 연산은 서버에서 해줘야됨

Socket()

#include <sys/types.h>
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
	socket(PF_INET, SOCK_STREAM, 0);

int domain : 소켓의 통신영역 결정
int type : 생성될 소켓이 어떤 방식으로 데이터를 전송할지를 결정 SOCK_STREAM/ DGRAM

int protocol : 사용될 종단 간 프로토콜 명시 0 은 앞의 인자를 확인하여 자동으로 선택

bind() bind 는소켓주소할당 sockaddr_in 구조체를 해당주소와 포트를 소켓에 연결

if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr))==-1)

int socket 소켓 식별자 serv_sock
struct *localAddress 구조체의주소(포인터로접근) (sap)&serv_addr
socklen_t addressSize 구조체의 크기 sizeof(serv_addr)

listen()

주소연결 (bind)를 하면 서버소켓은 주소를 가지게되며
클라이언트로부터 연결을 대기하게 하기위해

즉 연결요청을 대기모드로 설정 (단 절대 송수신에는 사용되지않음 클라이언트를 연결하기위한 새로운 소켓을 획득하는 수단으로만 사용)

queue 로 동작 선입선출

if(listen(serv_sock, 5)== -1)

listen(.5) 연결요청을 하는 클라이언트 최대 개수를 5 개로 함
listen(, 1) 1 개의 클라이언트만 가능 통신중 다른 클라이언트가 들어와도 대기하지 못함

Connetc()
서버는 클라이언트가 접속하기를 수동으로 기다리는데 반해 클라이언트는 연결을 직접시도함 bind 랑 비슷함

```
if(connect(sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)
    err_handler("connect() error");
```

int socket 소켓 식별자 serv_sock
struct *localAddress 구조체의주소(포인터로접근) (sap)&serv_addr
socklen_t addressSize 구조체의 크기 sizeof(serv_addr)

```
if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr))==-1)
```

bind 랑 비슷함

client 동작순서

socket() 소켓생성

구조체에 접속할 서버주소 설정

connect()연결요청

accept()
연결을 위해 새로운 소켓을 생성
clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size);

server 동작순서

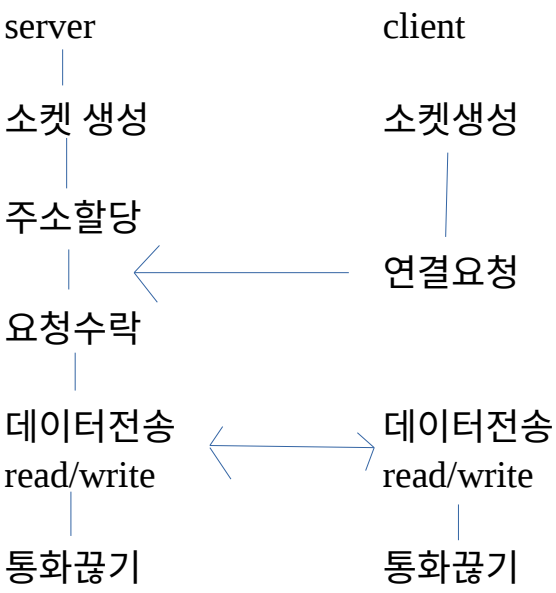
server socket 생성

sockadd_in 구조체에 주소설정(ip.port)

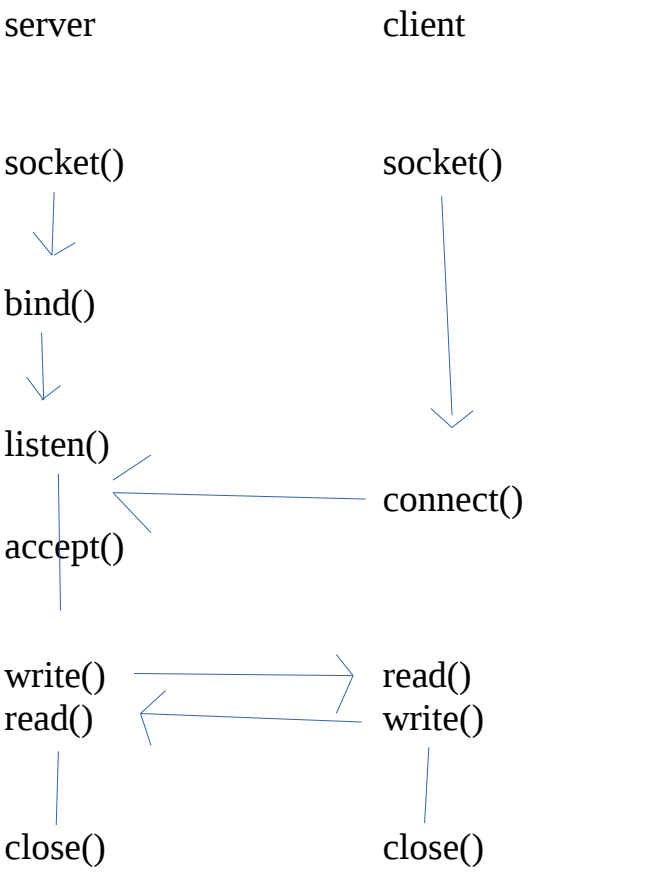
bind() 소켓에 구조체 값 할당

listen() 서버소켓에 대기모드, 클라이언트 최대 접속수 설정 하고 나옴
(nonblocking)
accept() 연결요청한 클라이언트와 통신할 송수신용 소켓생성
- 연결요청이 들어올때까지 대기함

네트워크 통신



리눅스 프로그래밍



소켓생성

소켓주소할당

bind 를 해서 서버소켓에 주소를 가지고
클라이언트로 부터 연결을 대기함

~~~~~~~~~~~~~~~서버~~~~~~~~~~~~~~~~

```
#include <stdio.h>                  //모든연산은 서버에서 해주는게 좋음
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>

typedef struct sockaddr_in       si;
typedef struct sockaddr* sap;

#define BUF_SIZE         1024
#define OPSZ       4

void err_handler(char* msg)
{
        fputs(msg, stderr);
        fputc('\n', stderr);
        exit(1);
}


int calculate(int opnum, int *opnds, char op) //opnum 3 입력하면 1 부터 3 까지
{

        int result = opnds[0],i;   //주의!

        switch(op)
        {
                case'+':
                        for(i=1; i<opnum; i++)
                                result += opnds[i];
                        break;

                case'-':
                        for(i=1; i<opnum; i++)
                                result -= opnds[i];
                        break;

                case'*':
                        for(i=1; i<opnum; i++)
```
~~~~~~~~~~~~~~~

```
                        result *= opnds[i];
                    break;

        }
        return result;  //결과값 리턴 이 값은 리절트에 저장

}


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

        int serv_sock, clnt_sock;
        char opinfo[BUF_SIZE];

        int result, opnd_cnt, i;
        int recv_cnt, recv_len;

        si serv_addr, clnt_addr;
        socklen_t clnt_addr_size;

        if(argc != 2)
        {
                printf("use: %s <port>\n", argv[0]);
                exit(1);
        }
                //서버소켓 생성
        serv_sock = socket(PF_INET, SOCK_STREAM, 0);
                //통신영역, 데이터통신방식 ,프로토콜

        if(serv_sock == -1)
                err_handler("socket() error");

        memset(&serv_addr,0,sizeof(serv_addr));
        serv_addr.sin_family = AF_INET;
        serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
        serv_addr.sin_port = htons(atoi(argv[1]));


        if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr))==-1) // bind 소켓주소
할당
                                //sockaddr_in 구조체를 해당주소와 포트를 소켓에 연결
```

```
		//소켓 식별자 serv_sock 구조체의주소(포인터로접근) (sap)&serv_addr 구조체의크기 sizeof(serv_addr)
		err_handler("bind() error");
		//연결요청 대기상태로 진입
	if(listen(serv_sock, 5)== -1)
		err_handler("listen() error");		//listen 은 한번만 쓸수있음
		//listem 연결대기 상태	queue 이고 listen( .5)는 연결요청을하는 클라이언트의 최대 갯수를 5 개로 한다는 뜻
		//연결요청 수락
	clnt_addr_size = sizeof(clnt_addr);

	for(i=0;i<5;i++)  //for 문이 있는 이유는 여러명이 접속할수있으니까
	{
		opnd_cnt = 0;
		clnt_sock = accept(serv_sock, (sap)&clnt_addr, &clnt_addr_size);
		read(clnt_sock, &opnd_cnt,1);

		recv_len=0;
		while((opnd_cnt * OPSZ +1) >  recv_len)
		{
			recv_cnt = read(clnt_sock, &opinfo[recv_len], BUF_SIZE-1);
			//읽은바이트수
			recv_len += recv_cnt;
		} //중간에 끊겼을 경우를 대비해 while 로 우회까지  (끊김방지)

		result = calculate(opnd_cnt, (int*)opinfo, opinfo[recv_len-1]);//계산부분
			//opnd_cnt
		write(clnt_sock, (char*)&result, sizeof(result)); //리절트를 &해줘서 리절트값을 출력   ~~~~~ 데이터전송

		close(clnt_sock);				//연결종료
	}
	close(serv_sock);
	return 0;
}
```

~~~~~~~~~~~~~~~~클라이언트 ~~~~~~~~~~~~~~~~~~~~~~``

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

typedef struct sockaddr_in       si;
typedef struct sockaddr *sap;

#define BUF_SIZE        1024
#define ALT_SIZE        4
#define       OPSZ        4

void err_handler(char *msg)
{
        fputs(msg,stderr);
        fputc('\n', stderr);
        exit(1);
}

int main(int argc, char **argv)
{
        int i, sock, result, opnd_cnt;
        char opmsg[BUF_SIZE] = {0};
        si serv_addr;

        if(argc != 3)
        {
                printf("use : %s <IP> <port>\n", argv[0]);
                exit(1);
        }

        sock = socket(PF_INET, SOCK_STREAM,0);              //서버접속을 위한 소
켓생성

        if(sock == -1)
                err_handler("socket() error");

        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]));
```
~~~~~~~~~~~~~~~~

```
//소켓만들고 커넥트 하는 구간은 server 이랑 똑같음
	if(connect(sock, (sap)&serv_addr,sizeof(serv_addr)) == -1)
		err_handler("connect() error");			//서버로부터 연결요청
	else
		puts("Conneted .......");

	fputs("Operand Cnt: ", stdout);
	scanf("%d", &opnd_cnt);

	opmsg[0] = (char)opnd_cnt;

	for(i=0; i<opnd_cnt ; i++)
	{
		printf("Operand %d:", i+1);
		scanf("%d", (int *)&opmsg[i * OPSZ +1]);	//숫자값 입력하라는 소리
	}

	fgetc(stdin);			//getc 니까 리드해서 1 바이트만 읽음
	fputs("Operator : ", stdout);
	scanf("%c", &opmsg[opnd_cnt * OPSZ +1]);
	write(sock, opmsg, opnd_cnt * OPSZ +2);		//서버쪽으로 데이터가 날라
와서
							//계산은 서버가 하는것임!
	read(sock, &result, ALT_SIZE);

	printf("OPeration result : %d\n", result);
	close(sock);

	return 0;
}
```