Xilinx Zynq FPGA, TI DSP,MCU기반의 프로그래밍 및 회로 설계 전문가 과정

강사 - Innov (이상훈)

gcccompil3r@gmail.com

학생 - 이유성

dbtjd1102@naver.com

basic_server.c

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

typedef struct sockaddr_in si;//IPv4주소체계에서 사용하는 구조체
typedef struct sockaddr *sap;//

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

int main(int argc, char **argv)
{
        int serv_sock; //
        int clnt_sock;

        si serv_addr;
        si clnt_addr;  //si poket address
        socklen_t clnt_addr_size;//길이 값,

        char msg[] = "Hello Network Programming"; //전달하려는 메시지

        if(argc !=2)//2개아니면
        {
                printf("use : %s <port>\n",argv[0]);//포트번호 입력하라 포트번호 7777(통로)
// (웹)80=www ,10 20ftp업로드 , 22=ssh다운로드   포트번호 (특정역할 service)  7777우리가 만든 전용 커스텀. 즉service번호
                exit(1);
        }

        serv_sock = socket(PF_INET,SOCK_STREAM,0); //1번째 인자 :ipv4를쓴다(ip통신 프로토콜  ,2번째 데이터 전송 형태를 지정 ,3번째인자 0(특정 프로토콜 사용하기 위한 변수 )

        if(serv_sock == -1) //socket =(파일) 리턴값= 파일 디스크립터(오픈이랑 결과값 같음)

                err_handler("socket() error");//
        //멤셋부터 4줄(포켓사이즈(만드는 형식)은 형태 똑같으니까 초기화  패턴 익히자.다르게 쓸 일이 없다.
        memset(&serv_addr , 0 , sizeof(serv_addr));
//멤셋 : 일종의 초기화 , 1번인자에 3번 인자만큼 2번인자를 입력한다.
         //serv_addr 구조체 si라는 si sockaddr_in 이라는 정보 3개
                //사이즈를 0으로                        //포켓 페밀리(port서비스  포켓 어드레스(어떤 ip주소든지 다 받겠다

        serv_addr.sin_family = AF_INET; //어떤 서비스 인지 셋팅   IPv4주소체계
        serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);//addr.s_addr: IP를 받는다
        serv_addr.sin_port = htons(atoi(argv[1])); //INADDR_ANY :사용 가능한 랜카드의 ip주소 ....값은 디파인을 통해 0으로 설정되어있음.
```

```
//변수3개  패밀리:af_inet ==ipv4형태의 주소를 사용하겠다)  어드레스:127.0.0.01로컬 어드레스
포트:내가 입력한 포트를 쓰겠다.그 포트 = 통로(7777) (클라이언트) 로 다른사람이 들어올 수 있다.

        if(bind(serv_sock , (sap)&serv_addr,sizeof(serv_addr)) ==-1 )//bind=서버의 ip주소 셋팅하는
것
//첫번째 인자 : 주소를 입력할 파일 디스크립터. sockaddr을 통해 정보를 받아옴.
//자기 주소로 정하겠다
                err_handler("bind() error");
        if(listen(serv_sock , 5) == -1)//listen:소켓을 연결 요청 대기 상태로 둔다. 최대 5명 받겠다  ..
연결하고자 하는 파일 디스크립터를 주고으녀결 요청하는 대기 큐의 크기를 전달한다. 실제 접속을
기다리는 위
                err_handler("listen() error");

        clnt_addr_size = sizeof(clnt_addr); //32 클라이언트 사이즈 저장
        clnt_sock = accept(serv_sock , (struct sockaddr *)&clnt_addr,&clnt_addr_size);//서버
소켓이 클라이언트의 접속을 허용 / 1번 인자:전달받을 파일 디스크립터 2번인자:연결요청 한
클라이언트의 정보 3번 인자는 전달받을 클라이언트 정보 크기
                //실제로 클라이언트가 기다리는 구간 listen을 기다림 (패턴을 봐라 해석하지말고 쓸
때없이)
                // 실제로 클라이언트 접속허용이 이루어지는구간 accept  .. listen -> accept넘어옴
                //accept하면  클라이언트의 주소가 잡힌다
        //어쎕트 성공하면 접속에 성공한 클라이언트의 소켓 파일디스크립터를 반환.치
        //그러면 똑같은클라이언트의  파일 디스크립터를 받음.
        if(clnt_sock ==-1)
                err_handler("accept() error");

        write(clnt_sock,msg,sizeof(msg));// socket은 파일(원격에 있는 파일) 네트워크 원격으로
동기화(semaphore)
        //파일 디스크립터 받으면 읽고 쓰기 다 가능, 보낼 msg를 연결된 fd에 쓴다
        //메세지 적어 원격에 있는 클라이언트에서 메시지가 나온다
        close(clnt_sock);
        close(serv_sock);

        return 0;
}
```

read_client.c

```
#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;

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

int main(int argc ,char **argv)
{
	int sock;
	int str_len = 0;
	si serv_addr;
	char msg[32] = {0};
	int idx = 0, read_len = 0;

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


	sock = socket(PF_INET,SOCK_STREAM, 0 );

	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]));

	if(connect(sock , (sap)&serv_addr, sizeof(serv_addr)) == -1)//커넥트를하면 서버의
listen에서 받는다 accept한다
// accept하면 서버랑 클라이언트랑 통신이 된다. 서버가 write해서 메시지 하나 보냄
		err_handler("connect() error");  //서버에 연결

	while(read_len = read(sock , &msg[idx++],1)) //추가됨.  예외적 , 16 -(끊힘) > 12 , 4로
읽어옴.  총길이는 read로 먼저 계산( 들어오다가 폭격..)문자 1개씩 읽어옴.
	{
		if(read_len == -1)
			err_handler("read() error !");

		str_len +=read_len;
	}
	printf("msg from serv :%s\n",msg);
	printf("read count : %d\n" ,str_len);
```

```
        close(sock);
        return 0;
}
```

inet_aton.c

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

typedef struct sockaddr_in si;

void err_handler(char *msg){

	write(2,msg,strlen(msg));
	exit(1);

}

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

	char *addr = "127.124.73.31";
	si addr_inet;

	if(!inet_aton(addr,&addr_inet.sin_addr))
		err_handler("Conversion Error!");
	else
		printf("Network Ordered integer Addr: %#x\n",addr_inet.sin_addr.s_addr);

	return 0;

}

//address to network
//기본적으로 빅엔디안(순서대로 저장)
//네트워크 주소로 바꾼다는 것은 기준을 둔다는 것(빅 엔디안)
//인텔은 무조건 리틀엔디안
//이 예제는 리틀을 빅으로 바꾼것임
```

inet_ntoa.c

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

typedef struct sockaddr_in si;

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

        si addr1,addr2;

        char *str;
        char str_arr[32]={0};

        addr1.sin_addr.s_addr = htonl(0x10203040);
        addr2.sin_addr.s_addr = htonl(0x12345678);

        str = inet_ntoa(addr1.sin_addr); //network to address
        strcpy(str_arr,str);
        printf("Not 1: %s\n",str);

        inet_ntoa(addr2.sin_addr);
        printf("Not 2: %s\n",str);
        printf("Not 3: %s\n",str_arr);

                return 0;

}

//network to address
//다시 크로스매칭 시키는 것임
```