TI DSP,MCU 및 Xilinux Zynq FPGA

프로그래밍 전문가 과정

이름	문지희
학생 이메일	mjh8127@naver.com
날짜	2018/3/29
수업일수	26 일차
담당강사	Innova Lee(이상훈)
강사 이메일	gcccompil3r@gmail.com

목차

1.signal

1. sigaction

```
#include<stdio.h>
#include<signal.h>

struct sigaction act_new;
struct sigaction act_old;

void sigint_handler(int signo)
{
        printf("Ctrl + C₩n");
        printf("If you push it one more time then exit₩n");
        sigaction(SIGINT,&act_old,NULL); //SIGINT 이면 act_old 가 동작한다.
}
int main(void)
{
        act_new.sa_handler=sigint_handler;
        sigemptyset(&act_new.sa_mask);//시그널 셋을 초기화.(어떤 시그널도 막지 않는다)

        sigaction(SIGINT,&act_new, &act_old);//sigint 가 들어오면 act_new 동작,이전에
동작시켰던 signal 의 정보를 &act_old 에  받아온다.(이전 핸들러를 &act_old 에 넣어줌)

        while(1)
        {
                printf("sigaction test₩n"); //문자열을 계속 출력한다.
                sleep(1);
        }
```

```
        return 0;
}//ctrl+c 두번 누르면 꺼짐
```

```
sigaction test
sigaction test
^CCtrl + C
If you push it one more time then exit
sigaction test
sigaction test
sigaction test
sigaction test
^C
```

sigaction( )
:시그널을 취급할 방법을 선택할 수 있다.

형태 : int sigaction(int signo, const struct sigaction *act, struct sigaction *oact);
인자 :
- signo : 행동을 지정할 시그널
- act : 지정하고 싶은 행동
- oact : 나중에 복구를 위해 현재 값을 저장한다.

Sigemptyset( )
: 시그널 셋을 초기화 할 때 사용
형태 : int sigemptyset(sigset_t *set);
성공시 0반환, 실패시 -1 반환

2.다른 프로세스에서 시그널을 날리는 예제

프로세스간 시그널 통신

-test.c

```
#include<stdio.h>
#include<signal.h>

void gogogo(int voidv)
{
        printf("SIGINT Accur!₩n"); //문자열 출력
        exit(0);
}

int main(void)
{
        signal(SIGINT,gogogo); //SIGINT 가 오면 gogogo 실행하겠다고 등록한다.

        for(;;)//무한루프
        {
                printf("kill Test₩n");
                sleep(2);
        }

        return 0;
}
```

-kill.c

```
#include<stdio.h>
#include<unistd.h>
#include<signal.h>
```

```
int main(int argc,char *argv[])
{
	if(argc<2)//인자 2개보다 작으면
		printf("Usage ./exe pid₩n");
	else
		kill(atoi(argv[1]),SIGINT);//문자열을 숫자로 바꿔주고, sigint 에 해당하는 시그널이 날라감
	return 0;
}
```

~결과

```
xeno@xeno-NH:~/proj/0329$ ./test & //&를 붙이면 PID 가 같이 뜬다
[1] 6163
xeno@xeno-NH:~/proj/0329$ kill Test
kill Test
kill Test
SIGINT Accur!

xeno@xeno-NH:~/proj/0329$ ./kill 6163
```

2. Thread

-Thread

~소스코드

```
#include<stdio.h>
#include<pthread.h>

void *task1(void *X) //void *는 어떤것이든지 인자로 받고 리턴하겠다는 의미
{
printf("Thread A Compelte₩n");
}

void *task2(void *X)
{
printf("Thread B Complete₩n");
}

int main(void)
{
pthread_t ThreadA, ThreadB;//쓰레드 지정할 때 pthread_t 로 지정. <pthread.h>안에있음
pthread_create(&ThreadA, NULL, task1, NULL); //task1을 threadA 가 구동시킬 것이라고 쓰레드의
생김새를 만듦
pthread_create(&ThreadB, NULL, task2, NULL);//task2를 ThreadB 가 구동시킬 것
                                                                    //NULL 은 아직 신경 x

pthread_join(ThreadA, NULL);//실제 메모리에 올리는 구간, ThreadA 실행
pthread_join(ThreadB, NULL);//ThreadB 실행

return 0;
```

```
}
```

~결과

```
xeno@xeno-NH:~/proj/0329$ gcc 2.c -lpthread
xeno@xeno-NH:~/proj/0329$ ./a.out
Thread A Compelte
Thread B Complete
```

-그래픽카드와 dsp

그래픽카드와 dsp 는 용도가 비슷하지만 dsp 는 하나를 고속처리하고 그래픽카드는 BW 가 넓고 속도가 느리지만 병렬처리가 우세하다.

3.Network

1. 네트워크

-server.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 serv_sock;
        int clnt_sock;

        si serv_addr;
        si clnt_addr;
```

```
	socklen_t clnt_addr_size;
	char msg[] = "hello Netsork Programming";//전달할 문자열
	if(argc != 2)//인자 2개아니면
	{
		printf("Use : %s <port>₩n",argv[0]); //포트번호 입력하라는 문자열출력
							//포트- 80번이 www 20,21이 ftp  22이 ssh
							//포트번호를 알면 포트번호의 특정역할을 알 수
있음.
							//웹브라우저는 80
							// 이 예제는 서비스번호가 7777, 웹브라우저는 80
		exit(1);
	}
//네트워크는 원격 IPC
	serv_sock = socket(PF_INET, SOCK_STREAM,0);
	//소켓도 파일이어서 파일 디스크립터가 넘어온다.

	if(serv_sock==-1)//서버소켓(파일 디스크립터)이 -1이면 오류
		err_handler ("socket()error");

	memset(&serv_addr, 0, sizeof(serv_addr));
	//serv_addr 은 구조체 si
	//어떤걸 열 것인가<??
	serv_addr.sin_family = AF_INET;//패턴 외우기<?
	serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);//어떤 아이피주소든 다 받겠다.
	serv_addr.sin_port = htons(atoi(argv[1]));

//127.0.0.1 ->로컬호스트(나)



	if(bind(serv_sock, (sap)&serv_addr,sizeof(serv_addr)) == -1)
	//bind : 서버의 ip 주소 셋팅, 127.0.0.1이 셋팅된다.
```

```
                err_handler("listen() error");
        if(listen(serv_sock, 5)==-1)
        //5명 받겠다는 의미
        //클라이언트를 기다리고
                err_handler("listen() error");
        clnt_addr_size = sizeof(clnt_addr);//30
        clnt_sock = accept(serv_sock, (struct sockaddr *)&clnt_addr, &clnt_addr_size);
        //accept : 클라이언트의 접속을 기다.림 실제 기다리는 구간은 listen 에서 기다린다.
        //클라이언트의 접속을 허용
        //accept 하면 클라이언트의 &clnt_addr 잡힌다, 파일디스크립터가 넘어옴, (원격에 있는 파일)
        if(clnt_sock == -1)
                err_handler("accept() error");


        write(clnt_sock, msg, sizeof(msg));//메세지 적음.클라이언트 소켓으로 날라감
//서버가 가지고 있던 메세지가 클라이언트로 보낸다
        close(clnt_sock);
        close(serv_sock);

        return 0;
}
```

-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;
	si serv_addr;
	char msg[32];

	if(argc !=3)//3개인 이유:어디로 접속하는지 알아야 하기 때문
	{
		printf("use : %s <ip> <port> ₩n", argv[0]);
		exit(1);

	}
	//사설아이피 : 192.168.0.~

	sock = socket(PF_INET, SOCK_STREAM, 0);
	//네트워크상의 파일디스크립터를 얻음
	//통신을 할 수 있는 파일 디스크립터를 얻음(fd)
	if(sock == -1)
		err_handler("socket() error");
```

```
        memset(&serv_addr, 0, sizeof(serv_addr));//서버어드레스를 초기화
        serv_addr.sin_family=AF_INET;//패턴임 기억
        serv_addr.sin_addr. s_addr = inet_addr(argv[1]);//입력한 아이피주소입력
        serv_addr.sin_port = htons(atoi(argv[2]));

        if(connect (sock, (sap)&serv_addr, sizeof(serv_addr))==-1)
        //서버 adr 로 연결(서버의 아이피값이랑 포트가 들어있음)
                err_handler("connet() error");

        str_len=read(sock,msg,sizeof(msg)-1);
        //msg 에 서버에서 보낸 메세지가 저장됨??

        if(str_len == -1)
                err_handler("read() error!");

        printf("msg from serv : %s₩n",msg);
        close(sock);
        return 0;
}
```

~결과

```
xeno@xeno-NH:~/proj/0329$ ./serv 7777

xeno@xeno-NH:~/proj/0329$ ./clnt 127.0.0.1 7777
msg from serv : hello Netsork Programming
```

네트워크

1. cs : client,server
2. 토콜로지 : 네트워크 구성도(그래프 알고리즘)
3. TCP/IP 프로토콜

TCP/IP : 서로 다른 시스템을 가진 컴퓨터들을 서로 연결하고 데이터를 전송하는 데 사용하는 통신 프로토콜들의 집합

-IP : 인터넷 프로토콜

종류

(1) 공인 IP : WAN 통신을 하려면 필요하다.

WAN : 광역통신망, 둘 이상의 LAN 이 넓은 지역에 걸쳐 연결되어 있는 네트워크를 말한다.

(2) 사설 IP : 공유기

NAT : 사설 IP 주소를 공인 IP 주소로 바꿔주는데 사용하는 통신망의 주소 변환기이다. 인터넷의 공인 IP 주소는 한정되어 있기 때문에 가급적 이를 공유할 수 있도록 하는 것이 필요한데 NAT 를 이용하면 사설 IP 주소를 사용하면서도 이를 공인 IP 주소와 상호변환 할 수 있도록 하여 공인 IP 주소를 다수가 함께 사용할 수 있도록 함으로써 이를 절약할 수 있는 것이다.

```
xeno@xeno-NH:~/proj/0329$ ifconfig
enp1s0      Link encap:Ethernet   HWaddr 8c:89:a5:61:5f:2d   → MAC 주소
            inet addr:192.168.0.39   Bcast:192.168.0.255   Mask:255.255.255.0
             → IP 주소
            inet6 addr: fe80::d23e:3bf2:109d:ec72/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST   MTU:1500   Metric:1
            RX packets:47825 errors:0 dropped:0 overruns:0 frame:0
            TX packets:30781 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:46702947 (46.7 MB)   TX bytes:3512077 (3.5 MB)

lo          Link encap:Local Loopback
            inet addr:127.0.0.1   Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING   MTU:65536   Metric:1
```

```
          RX packets:4022 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4022 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:559771 (559.7 KB)  TX bytes:559771 (559.7 KB)
```

ipv6 : ipv4보다 업그레이드 된 IP 주소, 센서 네트워크 구축을 하기 위해 나타났다.
1번 : 게이트웨이
255번 : 브로드캐스트

MAC 주소 : 이더넷의 물리적인 주소이다. 일반적으로 48비트 길이이다.

라우터 : 서로 다른 네트워크를 연결해주는 장치
스위치 : 몇개의 분기 중 어딘가를 하나를 선택하기 위한 분기명령

스위치 내에 통신을 할 때에는 브로드캐스트를 하여 통신한다. 스위치 대역폭을 넓히면 계속 신호를 보내는 것을 제거할 수 있다.
라우터를 통해 통신을 할 때에는 통신할 곳 까지 경로를 따라서 데이터를 공유한다.

2.socket

~소스코드

```
#include<stdio.h>
#include<fcntl.h>
#include<unistd.h>
#include<sys/socket.h>

int main(void)
{
          int fd[3];
          int i;
          //소켓도 파일 디스크립터를 만든다.
          fd[0]=socket(PF_INET,SOCK_STREAM,0);//tcp : 연결형 서비스를 지원하는 프로토콜이다.
```

```
	fd[1]=socket(PF_INET,SOCK_DGRAM,0);//udp : 정보를 주고 받을 때 한쪽에서 일방적으로
보내는 방식의 프로토콜
	fd[2]=open("test.txt", O_CREAT | O_WRONLY | O_TRUNC);
	//socket 이나 open 이나 모두 파일이다.
	for(i=0;i<3;i++)
		printf("fd[%d] = %d₩n",i,fd[i]);
	for(i=0;i<3;i++)
		close(fd[i]);

	return 0;
}
```

~결과
fd[0] = 3
fd[1] = 4
fd[2] = 5

socket 도 open 처럼 파일 디스크립트를 반환하고 socket 도 파일이다.

3. 라우터의 우회로

```
#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;
	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)
		err_handler("connect() error");

	while(read_len = read(sock, &msg[idx++], 1))//4바이트를 받아온다
	{
		if(read_len == -1)
			err_handler("read() error!");
		str_len += read_len;
	}
	//중간에 라우터가 끊겨도 우회로를 만들어 주는 것

	printf("msg from serv : %s₩n",msg);
	printf("read count, %d₩n",str_len);
	close(sock);

	return 0;

}
```

~결과

```
xeno@xeno-NH:~/proj/0329$ ./clnt 127.0.0.1 7777
msg from serv : hello Netsork Programming
read count, 65306
```

4.endian

~소스코드

```
#include<stdio.h>

int main(void)
{
	unsigned short host_port = 0x5678;
	unsigned short net_port;
	unsigned long host_addr = 0x87654321;
	unsigned long net_addr;

	net_port = htons(host_port); //빅 엔디안 방식으로 net_port 에 저장
	net_addr=htonl(host_addr); // 리틀 엔디안 방식으로 net_addr 에 저장

	printf("Host ordered Port : %0x₩n", host_port);
	//5678
	printf("Netsork Ordered Port : %0x₩n", net_port);
	//7856
	//풀어헤친것,메모리에 실제 배치되는 순서로 바꾸는것
	printf("Host Ordered Address : %0lx₩n", host_addr);
	//87654321
	printf("Netsork Ordered Address : %0lx₩n", net_addr);
	//21436587
	return 0;
}
```

~결과

Host ordered Port : 5678
Netsork Ordered Port : 7856 //데이터를 저장할 때에는 크로스 매칭을 통해 뒤집어서 배치한다.
Host Ordered Address : 87654321
Netsork Ordered Address : 21436587

```
최상위 비트를 가장 높은 주소에 저장됨 -> big endian format
```

Big endian format : 데이터의 최상위 비트가 가장 높은 주소에 저장되므로 그냥 보기에는 역으로 보인다. 바이트 단위로 주소가 할당되는 컴퓨터 기억장소에서 바이트 이상의 큰 데이터가 저장되는 형식의 하나이다.

Little endian format : 최상위 자리의 바이트부터 앞의 주소에 차례대로 기억되며, 바이트 단위로 주소가 할당되는 컴퓨터에서 두 바이트 이상의 데이터를 기억장소에 저장하는 형식의 하나이다.

5.endian 예제

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

int main(int argc,char **argv)
{
	char *addr1= "3.7.5.9";
	char *addr2 = "1.3.5.7";

	unsigned long conv_addr = inet_addr(addr1);
	if(conv_addr == INADDR_NONE) //변환에 실패할 경우
		printf("Error₩n");
	else
		printf("Network Ordered integer Addr : %#lx₩n",conv_addr);

	conv_addr = inet_addr(addr2);
	if(conv_addr == INADDR_NONE)
		printf("Error₩n");
	else
		printf("Network Ordered integer Addr : %#lx₩n",conv_addr);

	return 0;
}
```

~결과

Network Ordered integer Addr : 0x9070503
Network Ordered integer Addr : 0x7050301
크로스 매칭 되어 값이 출력된다.

inet_addr( )
: 네트워크 바이트 순서(Big Endian)과 Dotted-Decimal Notation 간의 상호 변환 함수
형태 : unsigned long inet_addr(const char *string)
성공 시 Big Endian 32비트 값 반환, 실패 시 INDDR_NONE 반환