Xilinx Zynq FPGA, TI DSP MCU 기반의

프로그래밍 및 회로 설계 전문가

강사 이상훈

(Innova Lee)

Gcccompil3r@gmail.com

학생 김민호

minking12@naver.com

```c
#include<time.h>
#include<string.h>
#include<stdlib.h>
#include<unistd.h>
#include<stdio.h>
#include<signal.h>
#include<pthread.h>
#include<sys/socket.h>
#include<fcntl.h>
#include<arpa/inet.h>

//sigaction1
struct sigaction act_new;//구조체 sigaction act new
struct sigaction act_old;//구조체 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);//act_old에 0을 전달하니 종료가 된다. 안에 뭐가 없으니까..
}

int main()
{
    act_new.sa_handler=sigint_handler;//구조체 act_new 에있는 sa_handler(signum번호를 가지는 시그널이 발생했을 때 실행된 함수를 설정한다)에 sigint_handler함수를 넣는다.
                                      //(수업시간)시그널 두번째 변수와 같다 , 첫번째 변수는 시그널번호
    sigemptyset(&act_new.sa_mask);// 인자로 주어진 &act_new.sa_mask 포함되어 있는 모든 시그널을 비운다.
                                  //(수업시간)특정시그널들을 막을수있다 ...empty->막지 않는다 시그널을 막아야하는 경우-> 우선순위가 높은 처리를 할때.

    sigaction(SIGINT,&act_new,&act_old);//sigaction은 특정 시그널의 수신에 대해서 취할 액션을 설정하거나 변경하기 위해서 사용된다.ctrl +c 를 하면 act_new가 실행
                                        //act_old에는 0이 들어간다. sigaciont은 성공하면  0을 반환하고실패시  -1 을 반환. sigaction은 전에 실행했던것을 맨 마지막 오른쪽에 리턴을 하는데 성공시 0 실패시
                                                                                                                                                                  //-1을 반환한다
                                        //(수업시간)인트럽트 시그널이 들어오면 act_new 구조체 실행,시그엑션은 엑트올드에 전달한다. 주소니까 &로 보낸애들을 변경 시킬수있다.
    while(1)
    {
        printf("sigaction test\n");//1초 간격으로 프린트문을 띄운다.
        sleep(1);
    }
    return 0;
}
```

```c
/*
//test.c                        ./test & 백그라운드에서 실행 ps -ef로 pid 본뒤 킬한다.

void gogogo(int voidv)
{
    printf("SIGINT Accur!\n");// 입력받고 Accur 실행후 종료
    exit(0);
}
int main()
{
    signal(SIGINT,gogogo);// 시그인트 전달시  해해해 함수  실행

    for(;;)
    {
        printf("kill test\n");//2초 간격마다 프린트문 띄움.
        sleep(2);
    }
    return 0;
}
*/

/*
//kill.c                    ./kill pid[숫자]

int main(int argc,char *argv[])
{
    if(argc<2)
        printf("usage : ./exe pid\n");
    else
        kill(atoi(argv[1]),SIGINT);//    ./kill [pid] 시그인트(ctrl+c)전달
    return 0;
}
```

```c
/*

//프로세스는 독립되어있다. 스레드는 종속적 (메모리를 공유 그래서 임계영역이 발생 해결책으로 락)
//thread.c    gcc 0329.c -lpthread

void *task1(void *X)
{
    printf("Thread A Complete\n");
}

void *task2(void *X)//보이드 포인터는 무엇이든 리턴 , 인자도 무엇이던지 받을수있다.
{
    printf("Thread B complete\n");
}


int main()
{
    pthread_t ThreadA,ThreadB;//스레드 A와 B 2개를 만든다.

    pthread_create(&ThreadA,NULL,task1,NULL);//(수업시간)null 은 앞뒤로 넣지 않겠다 .. fork 랑 비슷한 느낌 &주소를 받으니까  ThreadA 에 뭘 채워 넣는다 ,
                                             //무언가를 구동시킬 함수, ThreadA 를 구동시키면 task1 작동
    pthread_create(&ThreadB,NULL,task2,NULL);//ThreadB 를 구동시키면 task2는 작동한다를 등록(스레드 생김만 ...)

    pthread_join(ThreadA,NULL);//join 을 하면 실제 메모리에 올린다. 그리고 task1 이 실행되서 프린트문을 띄운다.
    pthread_join(ThreadB,NULL);//''

    return 0;

```

```c

//소켓은 파일이다.
int main()
{
    int fd[3];
    int i;

    fd[0]=socket(PF_INET,SOCK_STREAM,0);//소켓 옵션 tcp 3번, PF_INET=ipv4
    fd[1]=socket(PF_INET,SOCK_DGRAM,0);//           udp 4번
    fd[2]=open("test.txt",O_CREAT|O_WRONLY|O_TRUNC);//test.txt 파일을 만든다. 5번

    for(i=0;i<3;i++)
        printf("fd[%d]=%d\n",i,fd[i]); // 인덱스 번호를 띄운다.

    for(i=0;i<3;i++)
        close(fd[i]);

    return 0;
}

```

```
#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);
   fputs('\n',stderr);
   exit(1);
}

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

   si serv_addr;//si 는 소켓 어드레스 인 구조체
   si clnt_addr;
   socklen_t clnt_addr_size;//클라이언트 어드레스의 길이값 32 비트 4 바이트

   char msg[]="Hello network programming";//전달하려는 문자열 서버에 세팅이 되어있다.

   if(argc!=2)//2 개가 아니면 오류 포트번호는 통로
   {
        printf("use : %s <port>\n",argv[0]);
        exit(1);
   }

   serv_sock=socket(PF_INET,SOCK_STREAM,0);//잘입력했으면 소켓(여기로) 소켓은 파일이다.  네트
워크는 원격 IPC, serv sock 은 fd 가 넘어온다, tcp 형식,소켓=open fd 를 리턴한다.

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

   memset(&serv_addr,0,sizeof(serv_addr));//serv_addr 을 0 으로 초기화
   //서버 어드레스 구조체 si 라는 구조체 si 는 맵핑이 되어있는데 소크어드레스 인으로 3 개의 멤버가 들어가있다
소켓 패밀리 소켓 어드레스 소켓 포트
   serv_addr.sin_family=AF_INET;//tcp 소켓만들때 쓰는 형식 <패턴>
   serv_addr.sin_addr.s_addr=htonl(INADDR_ANY);//inaddr any  어떤 아이피던 받겠다. 자기 자신이
되겠다
   serv_addr.sin_port=htons(atoi(argv[1]));//서비스 번호=포트번호 어떤 서비스를 열것인가

   if(bind(serv_sock, (sap)&serv_addr,sizeof(serv_addr))==-1)//127.0.0.1 이 세팅
```

```
        err_handler("bind() error");//네트웤에서 bind 는 서버에 아이피를 세팅한다는 것.
    //서버에 아이피가 세팅된다

    if(listen(serv_sock,5)==-1)//5 명이상 받으면 안된다.
        err_handler("listen() error");//5 명받겠다는것 , accep 이 아니고 사실상 여기서 기다린다. 기다리고 있다가 클라이언트가 접속하면 accep 로 접속 허용을 해준다

    clnt_addr_size=sizeof(clnt_addr);//32
    clnt_sock=accept(serv_sock, (struct sockaddr *)&clnt_addr, &clnt_addr_size);//클라이언트의 주소와 사이즈 accept 는 서버 소켓이 클라이언트의 접속을 기다린다, 클라이언트의 어드레스,클라이언트 어드레스의 사이즈가 넘어간다 accept 에 & 니까 accept 안에서 뭔가 저 부분에 값들을 채워 넣는다.

    if(clnt_sock==-1)
        err_handler("accept() error");//서버 소켓이 클라이언트의 접속을 기다린다. ,접속 허용은 여기서, accept 를 하게 되는 순간 클라이언트 어드레스가 잡힌다. printf clnt addr 하면 접속자의 어드레스 주소가 나온다.

    write(clnt_sock,msg,sizeof(msg));//클라이언트 소켓에 쏘고있다. msg  문자열을 원격에 있는 클라이언트한테 write 한다 서버가 가지고 있 소켓은 파일 원격에 있는 ... 통신을 하려고 IPC 를 쓰고 있다
                                     // 클라이언트 소켓에 날아가고 있다. 원격에 있는 클라이언트한테 라이트 하고 있다.
    close(clnt_sock);
    close(serv_sock);

    return 0;
}
```

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

//basic_client.c

typedef struct sockaddr_in si;
typedef struct sockaddr * sap;


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

int main(int argc,char **argv)
{
   int sock;
   int str_len;
   si serv_addr;
   char msg[32];

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

   sock=socket(PF_INET,SOCK_STREAM,0);//네트워크상 fd 값을 가지고 온다.내가 통신을 할수있는 fd 를 가져온다 그걸 가져와서 sock 에 저장 sock = fd

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

   memset(&serv_addr,0,sizeof(serv_addr));//서버 어드레스 초기화
   serv_addr.sin_family=AF_INET;//패턴
   serv_addr.sin_addr.s_addr=inet_addr(argv[1]);//tcp 포맷 위아래 줄이 ... 우리가 입력한 아이피 주소 아이피 주소와 포트번호가 세팅 된다.
   serv_addr.sin_port=htons(atoi(argv[2]));//포트번호 7777

   if(connect(sock, (sap)&serv_addr, sizeof(serv_addr))==-1)//sock 이 fd 값 서버 어드레스에 연결 아이피값과 포트번호가 들어있다.
        err_handler("connect() error");//connect 하면 리슨에서 받는다-> accept 로->서버와 클라이언트가 통신
```

```
    str_len=read(sock,msg,sizeof(msg)-1);//read 는 블로킹 함수 들어올때까지 움직이지 않는다. sock 파일
디스크립터 msg 에 서버에서 보낸 hello ~ 가 들어가있다.

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

    printf("msg from serv: %s\n",msg);
    close(sock);

    return 0;
}
```