# TI DSP,Xilinx zynq FPGA,MCU 및 Xilinx zynq FPGA 프로그래밍 전문가 과정

강사-INNOVA LEE(이상훈)
Gcccompil3r@gmail.com
학생-윤지완
Yoonjw7894@naver.com

```
#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 int one more time then exit\n");
sigaction (SIGINT , &act_old,NULL);/ctrl +c 를 누르면 아무것도 동작이 안한다.
그래서 ctrl +c 를 두번을 누르면 꺼지는 이유다.
}
int main(void)
{

act_new.sa_handler=sigint_handler;
sigemptyset(&act_new.sa_mask);//아무것도 막지 않는다는 뜻
sigaction(SIGINT, &act_new,&act_old);//signal 의 두번째 주소 SIGINT 가 되면 act_new 가 동
작이 되고 act_new sa.handler 가 동작이 되고 맨 위에
while(1)
{
printf("sigaction test\n");
sleep(1);
}
return 0;
```

다른 사람코드를 빠르게 분석하는 방범
&를 보라. &는 주소를 보내는 것이고 그것이 포인터인데
&로 보낸애들은 변경시킬수 가 있다 도는 값이 seting 이 되어있다.
함수를 out put 한개만 가능하지만 포인터는 여러개 가능하다.

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

void gogogo(int voidv)

{

printf("SIGINT Accur!\n");
exit(0);


}
int main(void)
{

signal(SIGINT ,gogogo);
for(;;)
```

```
{
printf("kill test\n");
sleep(2);
}
return 0;
}

#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<signal.h>
int main( int argc, char *argv[])
{
if(argc<2)
printf("user:./exe pid\n");
else
kill(atoi(argv[1]),SIGINT);
return 0;
}


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

void  *task1(void *X)//void 포인터 무엇이든 리턴 가능 인자도 아무거나 받을 수 있다.
{
printf("thread a complete\n");
}


void *task2(void *X)
{
printf("thread b COMPLETE\n");
}

int main(void)
{

pthread_t ThreadA, ThreadB;//threada 와 b 에 값을 넣겠다는 것
pthread_create(&ThreadA ,NULL, task1,NULL);
//task1 을 threada 가 구동시키는 것, tread 의 형태만 만들어 놓은것 동작 ㄴㄴ
pthread_create(&ThreadB ,NULL, task2,NULL);

pthread_join(ThreadA,NULL);//실질적으로 여기서 thread 의 동작이 이뤄진다.
pthread_join(ThreadB,NULL);
return 0;
}
```

컴파일 방법:gcc test.c -lpthread
cpu 는 병렬처리가  아닌 순차처리의 특화가 되어있다.

# <네트워크>

**1.cs(client server)**
**2.토폴로지(위상수학 ㄴㄴ)**
**:네트워크 구성도(그래프 알고리즘)**
**이 두가지가 기본이다.**
**Tcp/ip 프로토콜:4 계층**
**(dSI 7 LOYER)**
**[라우터,스위치,OS}**
***인터넷**
**:랜선 ,IP(INTERNET PROTOCALL)**
**[ifconfig 를 치고  이 3 가지를 중점적으로 봐야한다->ipv4,ipv6,nat]**
**0 번:게이트 웨이**
**255 번:브로드 캐스트**
**ip 의 종류**
**1)공인 ip-wan 통신**
**2)사설 ip(공유기)**
**ipv4=32 비트  ipv6=센서 네트회로**
**MAC 통신**
**랜카드의 고유 번호,식별자 이 흔적은 스위치에 남겨진다.**
**MAC 주소를 보고 어디로 보낼지 결정하는게 스위치**
**A 통신에서 C 통신으로 데이터를 전달할라는데 이때 스위치에게 데이터를 보내고싶다고 전달하면 스위치는 브로드 캐스팅을 한다.스위치는 계속해서 전기신호를 보내는데 이때 a/dc 로 디지털 신호로 바꿔주어 이 값이 메모리에 쌓이는데 메모리를 처리하기도전에 데이터가 계속 오니까 메모리가 나가는 것이다.**
**이것이 디도스이다.**
**만약 A 통신이 다른 스위치가 관리하는 B 통신에게 데이터를 전달하고 싶은데 이 스위치는 권한 밖이라 라우터에게 B 통신의 고유 IP 를 보내고 라우터가 다른 라우터에게 고유 IP 를 보내고**
**주소를 찾아가는 식이다.**
**네트위킹=느린 IPC 하지만 외부랑 통신이 가능하다.**
**네트워크란 원격 IPC 이다.**

# <client>

```
#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){
        printf("use: %s <IP> <port>\n", argv[0]);
        exit(1);
   }

   sock = socket(PF_INET, SOCK_STREAM, 0);소켓에 리턴되는 값은 fd(file discript)와 같다.

   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)//sap 자기 자신의 데이터 discript
        err_handler("connect() error");

   str_len = read(sock, msg, sizeof(msg) -1);//read 는 blocking 이라 드러오기전까지 아무것도
안한다. 서버에서 보낸 메시지를 받아서  sock 에 저장한다.서버에서 listen 에서 받고 aceept 가
받고 write 가 되어 메시지를 던잘하는 방식이다.

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

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

```
    close(sock);

    return 0;
}
```

# <server>

```
#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;//32 바이트

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

    if(argc != 2){
        printf("use: %s <port>\n", argv[0]);//port 는 통로이다. 20 번 upload 21 번 download
        exit(1);
    }

    serv_sock = socket(PF_INET, SOCK_STREAM, 0);//socket 도 파일이다 ,tcp 소켓을 사용하겠다. 리턴은 fd 이다

    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);//어떤 ip 든 다받겠다. 127.0.0.1 에서 127 은 localhost.
    serv_addr.sin_port = htons(atoi(argv[1]));
```

```
   if(bind(serv_sock, (sap)&serv_addr, sizeof(serv_addr)) == -1)//bind=ip 주소를 세팅하는것이다. 서버에 ip 가 셋팅이 된다.
        err_handler("bind() error");

   if(listen(serv_sock, 5) == -1)//5 명만 받겠다. listen 에서 클라이언트를 기다리고있는다.
        err_handler("listen() error");

   clnt_addr_size = sizeof(clnt_addr);
   clnt_sock = accept(serv_sock, (struct sockaddr *)&clnt_addr, &clnt_addr_size);
//accept 는 소켓이 서버 클라이언트를 기다린다. 소켓은 원격 ip
   if(clnt_sock == -1)
        err_handler("accept() error");

   write(clnt_sock, msg, sizeof(msg));//원격에 있는 client 에게 메시지를 보낸다.
   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>
#include<fcntl.h>


int main(void)
{
   int fd[3];
int i;
fd[0]= socket(PF_INET , SOCK_STREAM,0);//tcp
fd[1]= socket(PF_INET , SOCK_DGRAM,0);//udp
fd[2]= open("test.txt",O_CREAT | O_WRONLY | O_TRUNC);
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;
}

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include<fcntl.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 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)
      err_handler("connect() error");
while(read_len = read(sock,&msg[idx++],1))
{
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;
}
```

위에 코드와 거의다 동일하지만 데이터가 전송되다가 중간에 데이터를 손실되지않게 만드는 코드

```
#include <stdio.h>


int main(void)
{
unsigned short host_port = 0x5687;//2byte
unsigned short net_port ;//2byte
unsigned long host_addr = 0x87654321;/4byte
unsigned long net_addr;//4byte
net_port = htons(host_addr);
net_addr = htonl(host_addr);
printf("host orfered port:%#x\n",host_port);
printf("network ordered port:%#x\n",net_port);
printf("host ordered address:%#lx\n",host_addr);
printf("network ordered adress:%#lx\n",net_addr);
return 0;
}
```
//다른 랜디안 끼리 정보를 공유하면 정보가 꼬이는데 그것을 방지하는 코드

```
#include <stdio.h>
#include<arpa/inet.h>
int main(int argc,char **argv)
{
char *addr1= "3.7.5.9";//big radian 과 little radian 는 각자 메모리에 들어가는 방식이 틀린데
```
그것을 설정해놓은것이다.
```
char *addr2 = "1.3.5.7";
unsigned long conv_addr = inet_addr(addr1);
if(conv_addr == INADDR_NONE)
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;
}
```