

TI DSP, MCU 및 Xilinx Zynq FPGA

프로그래밍 전문가 과정

강사 - Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 - 하성용
accept0108@naver.com

26 일차
- 네트워크 프로그래밍 1 일차

sigaction1.c

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

struct sigaction act_new;
struct sigaction act_old;

//시그널의 동작을 처리하는 함수
void sigint_handler(int signo) //시그널을 실행할때
{
    printf("Ctrl +C\n");

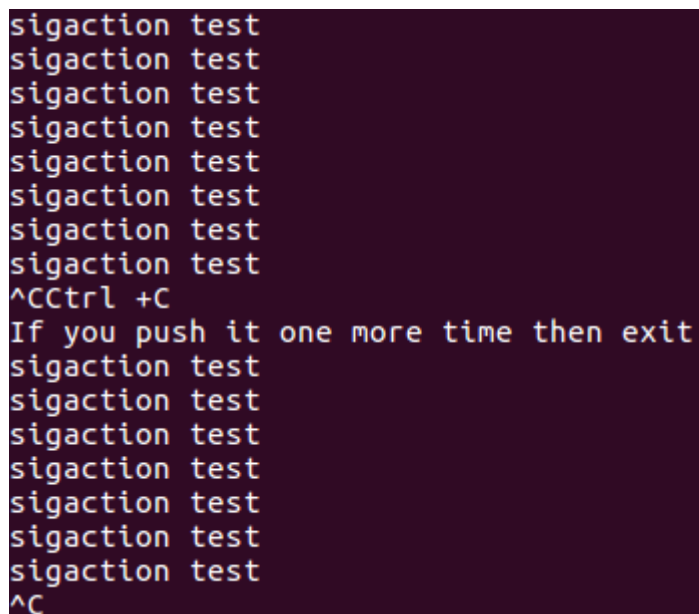
    printf("If you push it one more time then exit\n");
    sigaction(SIGINT, &act_old, NULL); //NULL 은 무시, 즉 ctrl+c 의 원래기능실행
}

int main(void)
{
    act_new.sa_handler = sigint_handler;

    sigemptyset(&act_new.sa_mask);

    sigaction(SIGINT, &act_new, &act_old);

    while(1)
    {
        printf("sigaction test\n");
        sleep(1); //1 초 있다가
    }
    return 0;
}
```



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

프로세스가 프로세스를 kill 하게 만들기

kill.c

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

int main(int argc, char *argv[])
{
    if(argc <2)
        printf("Usage : ./exe pid\n");
    else
        kill(atoi(argv[1]),SIGINT);

    return 0;
}
```

thread.c

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

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

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

int main(void)
{
    pthread_t ThreadA, ThreadB;

    pthread_create(&ThreadA, NULL, task1, NULL);
    pthread_create(&ThreadB, NULL, task2, NULL);

    pthread_join(ThreadA, NULL);
    pthread_join(ThreadB, NULL);

    return 0;
}
```

test.c

```
#include<stdio.h>
#include<signal.h>
#include<string.h>
#include<stdlib.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;
}
```

1. CS(Client Server)
2. 토폴로지(위상수학 X)
네트워크 구성도(그래프 알고리즘)
3. TCP/IP 프로토콜 ←---┐
이론-(OSI 7layer) |
버클리 ↑ → 4 계층 ----┘
[라우터, 스위치, OS]리눅스, 유닉스 최적화됨

* 인터넷
LAN 선,[IP]
if config
「 ipv4, ipv6, NAT
IP의 종류
1. 공인 IP(WAN 통신)
2. 사설 IP(공유기) 」
MAC 통신
스위치 장비

원격 IPC

```
gcc -o clnt basic_client.c
gcc -o serv basic_server.c
터미널 2개 띄우고
./serv 이후 ./clnt
./serv 7777
./clnt 127.0.0.1 7777
```

basic_client.c

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

//sockaddr 은 IP 주소+포트번호로 구성되었다
typedef struct sockaddr_in si; //si 를 'sockaddr'으로 정의
typedef struct sockaddr* sap; //sap 를 'sockaddr 포인터'로 정의

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

//argc 는 포트번호 입력하라는것
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);

    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");
    str_len = read(sock, msg, sizeof(msg) -1);
    if(str_len == -1)
        err_handler("read() error!");
    printf("msg from serv : %s\n", msg);
    close(sock);

    return 0;
}
```

basic_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 Network Programming";
    if(argc != 2)
    {
        printf("use: %s <port>\n",argv[0]);
        exit(1);
    }

    serv_sock = socket(PF_INET, SOCK_STREAM,
                      0);

    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)
        err_handler("bind()error");

    if(listen(serv_sock,5)== -1)
        err_handler("listen()error");

    clnt_addr_size= sizeof(clnt_addr);
    clnt_sock=accept(serv_sock,(struct sockaddr
    *)&clnt_addr,&clnt_addr_size);

    if(clnt_sock== -1)
        err_handler("accept()error");
```

```

        write(clnt_sock,msg,sizeof(msg));
        close(clnt_sock);
        close(serv_sock);

    return 0;
}

```

그래서 서버가 가지고있던 메시지가 뭐였는지
클라이언트에게서 메세지나왔는데
이 메시지가 제대로 클라이언트에게 잘 전달되었다는뜻

192.168.0 = 사설 ip 주소

소켓하면 파일디스크래터를 얻는데
네트워크상의 파일디스크래터

네트워크에서 소켓은 파일의 오픈과같음

sock 은 fd. 자기자신의 파일디스크래터
그걸 서버어드레스에 연결함
커넥트를하고 실제로받는건 서버의 리슨에서받음
받아서 엑셉트

통신이되니까 str_len write 했으니 read(=블로킹함수)하고
속은 파일디스크래터, msg 에는 파일디스크래터가 들어오고
printf 에 msg 값이 들어오고 Hello Network Programming 가 출력됨

소켓이 fd 라는걸 증명하는 예제
소크 스트림은 ftp
소크 디그림은 udp

```

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

int main(void)
{
    int fd[3];
    int i;

    fd[0] = socket(PF_INET, SOCK_STREAM, 0);
    fd[1] = socket(PF_INET, SOCK_DGRAM, 0);
    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;
}

```

새로만들었을때 text.txt 를지우고 만들면 3 4 5 가 제대로 쓰지만
안지우고만들면 -1 오류가 뜬다

소켓도 파일디스크래터를 만든다는뜻
즉, 소켓은 파일이다

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)
        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;
}

```

리슨은 실제로 클라이언트를 기다리는곳

리슨하고 커넥트를하고 엑셉트
 커넥트가 엑셉트와 리스산이에들어가는거?
 리슨이 기다리면
 어셉트에서 보내줄지말지 결정하고 커넥트에 연결

최단거리 데이터를 옮기다 중간에 끊기면 우회
 처음데이터의 길이를 아는건 처음 커넥트를 할때 길이를 구해줌

```

msg from serv: Hello Network Programming
read count: 26

```

str_len 에는 12 가 들어오고 리드로 올라가서 남은 4 바이트를 받아와서 리드엔은 4
 str_len 에 넣어서 손실된데이터가 없게
 중간에 끊기는 상황을 고려

```

use: ./a.out <IP> <port>

```

ex) ./a.out 192.168.0.25 7777

ex) 네이버
 서버에 클라이언트가 붙는것

이걸만들고나면 네트워크를 연동한 3 6 9 게임을 만들것

convert_endian.c

정보가 꼬이지말라고 바꿔주는것

inet_addr.c

```

#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)
        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;
    }

```

```

Network Ordered Integer Addr : 0x9050703
Network Ordered Integer Addr : 0x7050301

```

빅엔디와 리드엔디와 통신을 한다고 가정하였을때
 서로 같은규칙을 안하고 보게되었을때 꼬이게된다
 그래서 네트워크형식에 맞게 변경을 시켜놓으면
 030507 을 리드엔디를 메모리형식으로 바꿔서 출력하면 9050703 으로 나오게된다

ex) 03.07.05.09 크로스매칭
 메모리 저장할때

->

09 05 07 03

이렇게 잘못된주소로 가게되니 애초에 메모리형식으로 바꾸어놓고
 변경방식대로 읽어들었을때 꼬일일이 없게된다