TI DSP, MCU 및 Xilinx Zynq FPGA 프로그래밍 전문가 과정

강사 – Innova Lee(이상훈)
gcccompil3r@gmail.com
학생 – 최대성
c3d4s19@naver.com

2018.03.29. – 026일차 수업

다른 프로세스로 시그널 보내기

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

---

int sigaction([signum], [act], [oldact]) 함수

-> signal() 함수와 같은 동작 + 추가 설정 가능,

성공시 0, 실패시 -1 반환

-> [signum]을 받으면 [act]구조체 내용 실행하고

[oldact]구조체에 이전 sigaction()에서 실행된 내용을

넣고 만약 이전 실행 함수가 없으면 NULL을 넣는다.

[act]와 [oldact]는 <signal.h> 헤더파일에 정의된

struct sigaction*형 인자를 받는다

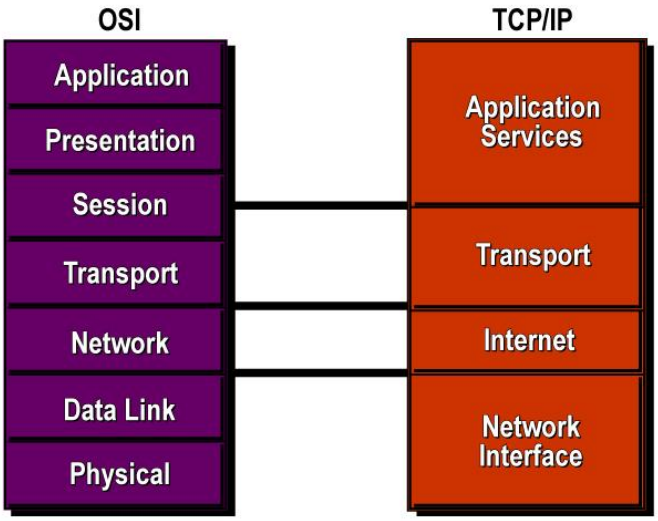
```
#include <stdio.h>
#include <signal.h>

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);
}

int main(){
        // sigaction.sa_handler -> 시그널 핸들러
함수 설정
        act_new.sa_handler = sigint_handler;
        // sigaction.sa_mask -> 특정 시그널 막음
        sigemptyset(&act_new.sa_mask);
        sigaction(SIGINT, &act_new, &act_old);


        while(1){
                printf("sigaction test₩n");
                sleep(1);
        }
        return 0;
}
```

간단한 thread 사용 예제

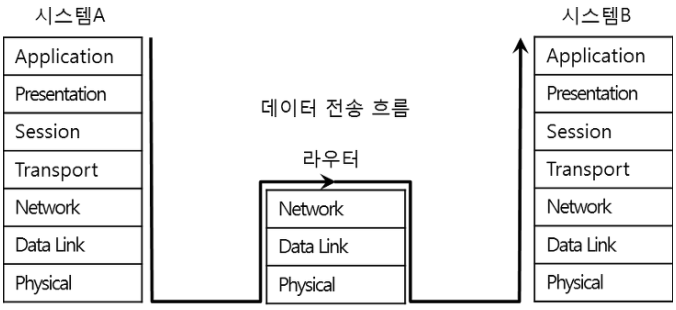
```
#include <stdio.h>
#include <pthread.h>

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

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

int main(){
        pthread_t ThreadA, ThreadB;
        //Thread 생성(생성만 하고 실행X)
        pthread_create(&ThreadA, NULL, task1,
NULL);
        pthread_create(&ThreadB, NULL, task2,
NULL);
        //Thread 실행
        pthread_join(ThreadA, NULL);
        pthread_join(ThreadB, NULL);

        return 0;
}
```

네트워크 프로그래밍

OSI(7계층) ----> TCP/IP 프로토콜(4계층)

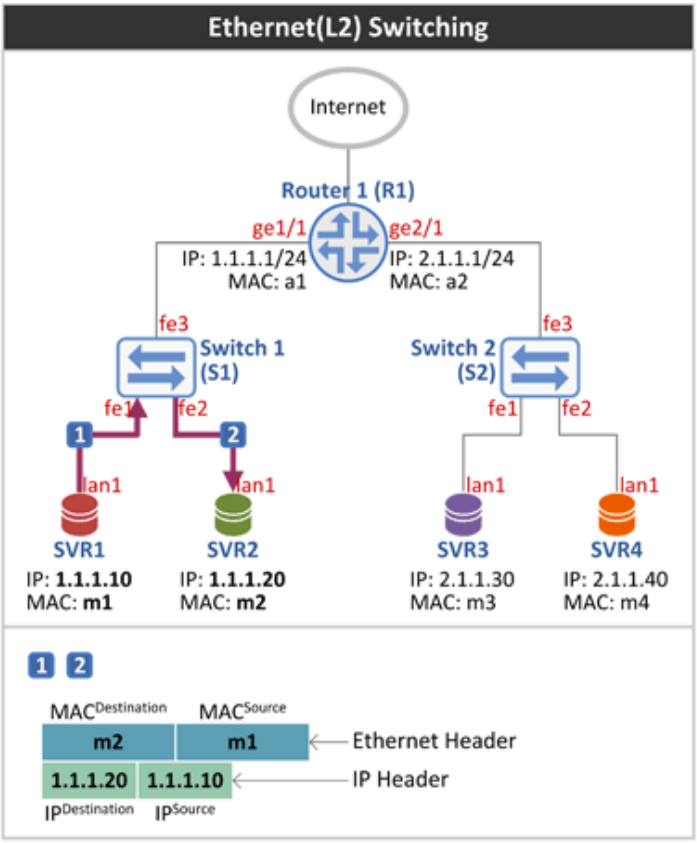


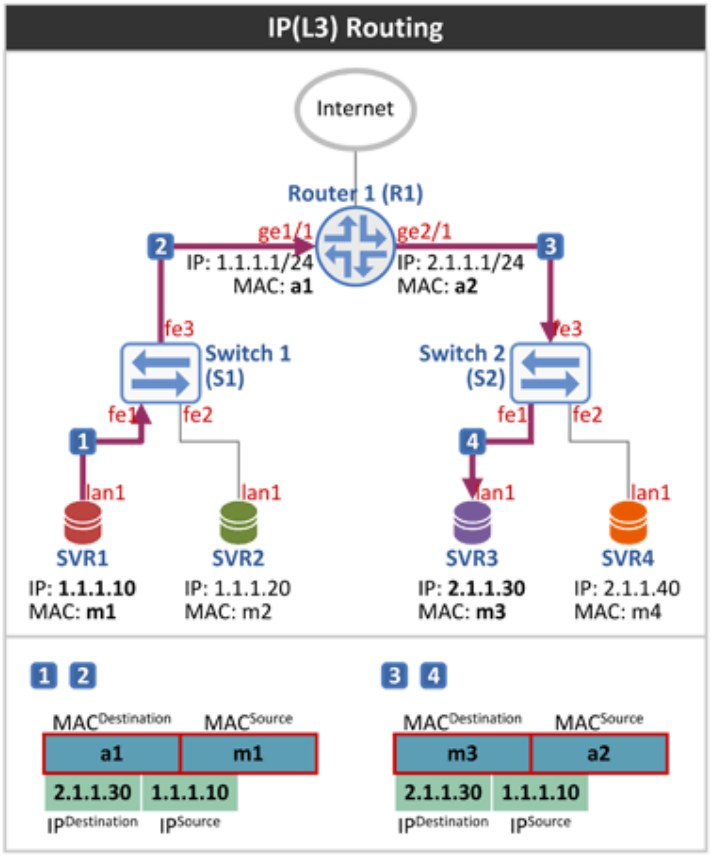
구분	주요 대상	내용	프로토콜
Application	응용 S/W	통신 서비스	FTP,HTTP
Presentation	데이터변환	압축, 암호화	H.264,MPEG2
Session	세션체결	전송방식결정	반이중,전이중
Transport	segment	에러/경로제어	TCP,UDP
Network	datagram	논리주소/경로설정	RIP,BGP
Data Link	frame	물리주소/경로설정	ARQ
Physical	bit	전기신호제어	물리접속



MAC 주소 -> 랜카드 고유 주소

IP의 종류 -> 공인IP(WAN 통신), 사설IP(공유기)





Switching, Routing 참고 자료 링크

https://www.netmanias.com/ko/post/blog/5501/arp-bridging-ip-routing-network-protocol/switching-and-routing-part-1-ethernet-switching

---

int socket(int [domain], int [type], [0]);

-> 소켓 식별자(네트워크상의 Descriptor) 반환, 실패시 -1 반환

[domain] -> 인터넷을 통해 통신할지, 같은 시스템 내에서 프로세스 끼리 통신할지의 여부를 설정

PF_INET -> IPv4 사용
PF_INET6 -> IPv6 사용

[type] -> 데이터의 전송 형태를 지정

SOCK_STREAM -> TCP/IP 프로토콜
SOCK_DGRAM -> UDP/IP 프로토콜

사용 예제

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

int main(){
    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;
}
```

Server – Client사이 TCP/IP 통신

<basic_server.c 파일>

```
/*
컴파일 방법
gcc -o serv basic_client.c
gcc -o clnt basic_client.c

./serv 7777
./clnt 127.0.0.1 7777

*/
```

```
#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;
}
```

<basic_client.c 파일>

```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))
// 1byte씩 읽기
    {
        if (read_len == -1){ //read() 실패경우
            err_handler("read() error!");
            idx--;
        }
        str_len += read_len;
    }

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

void* memset(void* [dest], int [c], size_t [count]);

[dest]의 [count]byte만큼을 [c]로 설정해준다

AF_INET, PF_INET 차이

AF -> 주소 체계 / PF -> 프로토콜 체계

참고 링크: http://mintnlatte.tistory.com/263

int bind(int [sockfd], struct sockaddr* [myaddr], socklen_t [addrlen]);

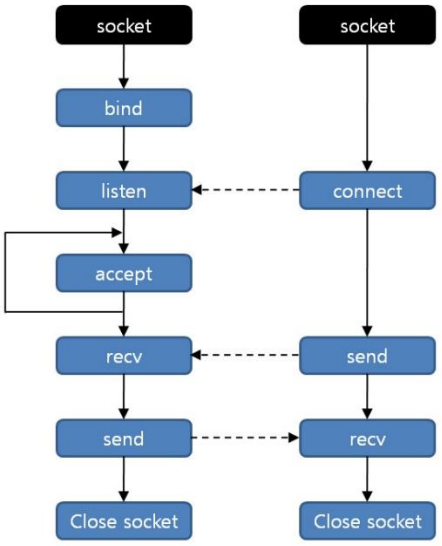
[sockfd] -> 소켓 식별자 (socket descriptor)
[myaddr] ->
[addrlen] -> [myaddr] 구조체의 크기

int listen(int [s], int [backlog]);
[s] -> 소켓 식별자 (socket descriptor)
[backlog] -> 대기 메시지 큐(Queue)의 개수

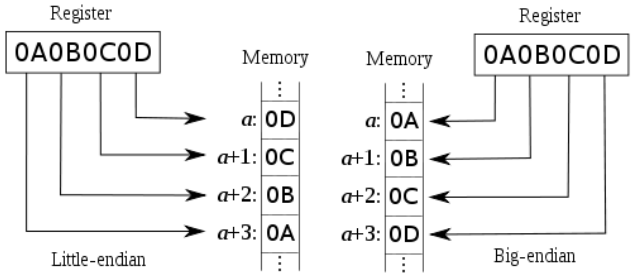
(listen 함수에서 계속 대기하다가 접속 받음)

int accept(int [s], struct sockaddr* [addr], socklen_t* [addrlen])
[s] -> 소켓 디스크립터 (socket file descriptor)
[addr] -> Client 주소 정보를 가지고 있는 포인터
[addrlen] -> [addr]포인터가 가르키는 구조체의 크기

(accept 함수에서는 허가할지 말지만 결정함)



CPU마다 바이트를 배열하는 방법(Endian)이 다르다



네트워크에서는 Big-Endian 기준으로 사용하고
Little-Endian 방식은 데이터를 Big-Endian 방식으로 변환하여 네트워크에 보내고 역변환하여 데이터를 가져온다.

htons()

-> short형 데이터의 host byte 순서를 network byte 순서로 (host to network short)

htonl()

-> long형 데이터의 host byte 순서를 network byte 순서로 (host to network long)

ntohs()

-> short형 데이터의 network byte 순서를 host byte 순서로 (network to host short)

ntohl()

-> long형 데이터의 network byte 순서를 host byte 순서로 (network to host long)

```c
#include <stdio.h>

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

    net_port = htons(host_port); // 2byte 처리
host to network short
    net_addr = htonl(host_addr); // 4byte 처리
host to network long

    printf("Host Ordered Port: %#x₩n", host_port);
    printf("Network Ordered Port: %#x₩n",
net_port);
    printf("Host Ordered Address: %#x₩n",
host_addr);
    printf("Network Ordered Address: %#x₩n",
net_addr);

    return 0;
}
```

```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;
}
```