Socket

네트워크의 연결 도구

- Socket API 서로 다른/같은 호스트 상에서 실행되는 두개 이상의 프로세스들이 컴퓨터 네트워크를 사용하여 통신하기 위해 제공되는 도구
- 소켓과 입출력 파일 디스크립터를 이용해 소켓 입출력 가능 소켓 외의 입출력 - stdin/out, file, terminal...
- 파일 디스크립터 열린 파일들을 식별하기 위한 정보 (int)
- 기본 입출력 함수

    1. open

    ```
    open(path, oflag)
    ```

        - oflag O_RDONLY, O_APPEND, O_CREATE, O_TRUNC.. 등

    2. read

    ```
    read(fd, buf, count)
    ```

    fd에서 count만큼 buf에 읽는다

    3. write

    ```
    write(fd, buf, size)
    ```

    fd에서 size만큼 buf에 쓴다

    4. close

    ```
    close(fd)
    ```

    열려있는 fd 닫기

▶ 실습 01:io-syscall.c

```c
#include <stdio.h>
#include <fcntl.h>
//#include <stdlib.h>

int main(int argc, char** argv) {
    int fD, writeLen, readLen;
    char rBuff[BUFSIZ];
```

```c
    if(argc != 2) {
        fprintf(stderr, "Usage: %s [Filename] \n ",argv[0]);
        return 0;
    }

    //fd=0: stdin
    readLen = read(0, rBuff, BUFSIZ-1);
    //readLen 리턴값 -> 읽은 만큼의 사이즈
    if(readLen == -1)
    {
        fprintf(stderr, "Read Error \n");
        return 0;
    }

    printf("Total reading data: %d\n", readLen);

    //개행 추가
    rBuff[readLen] = '\0';

    //해당 파일을 read, right 권한으로 열고 / 없으면 새 파일 생성 / 있으면 초기화 후 다시
    fD = open(argv[1],O_WRONLY|O_CREAT|O_TRUNC);
    if(fD == -1)
    {
        fprintf(stderr, "Open Error \n");
        return 0;
    }

    //null 포함 +1
    writeLen = write(fD, rBuff, readLen+1);
    if(writeLen == -1)
    {
        fprintf(stderr, "Write Error \n");
        return 0;
    }
    printf("Total writing data: %d\n", readLen);

    close(fD);
    return 0;
}
```



```c
socket(int domain, type, protocol)
// 프로토콜 체계(IPv4/6), 전송방식 (SOCK_STREAM / SOCK_DGRAM), 프로토콜
(IPPROTO_TCP/UDP)
```

▶ 실습 02: socket-syscall.c

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

int main()
{
    int sD1, fD1, sD2, fD2;

```

```
    //IPv4, TCP
    sD1 = socket(PF_INET, SOCK_STREAM, 0);
    fD1 = open("test",O_RDONLY);
    sD2 = socket(PF_INET, SOCK_STREAM, 0);
    fD2 = open("test",O_RDONLY);

    printf("sD1:%d, fD1:%d, sD2:%d, fD2:%d \n",sD1, fD1, sD2, fD2);

    close(sD1);
    close(fD1);
    close(sD2);
    close(fD1);

    return 0;
}

                                                    Colored by Color Scripter
```

통신 프로토콜

pid - 프로세스 식별할 수 있지만 Host 내에서만 가능함 port - 서로다른 호스트 사이에서 프로세스 식별, app과 전송 계층 사이에 존재

프로토콜 체계

PF_INET - Progocol Famillly AF_INET - Address Famillly

PF_INET = AF_INET //ipv4 PF_INET6 = AF_INET6 //ipv6

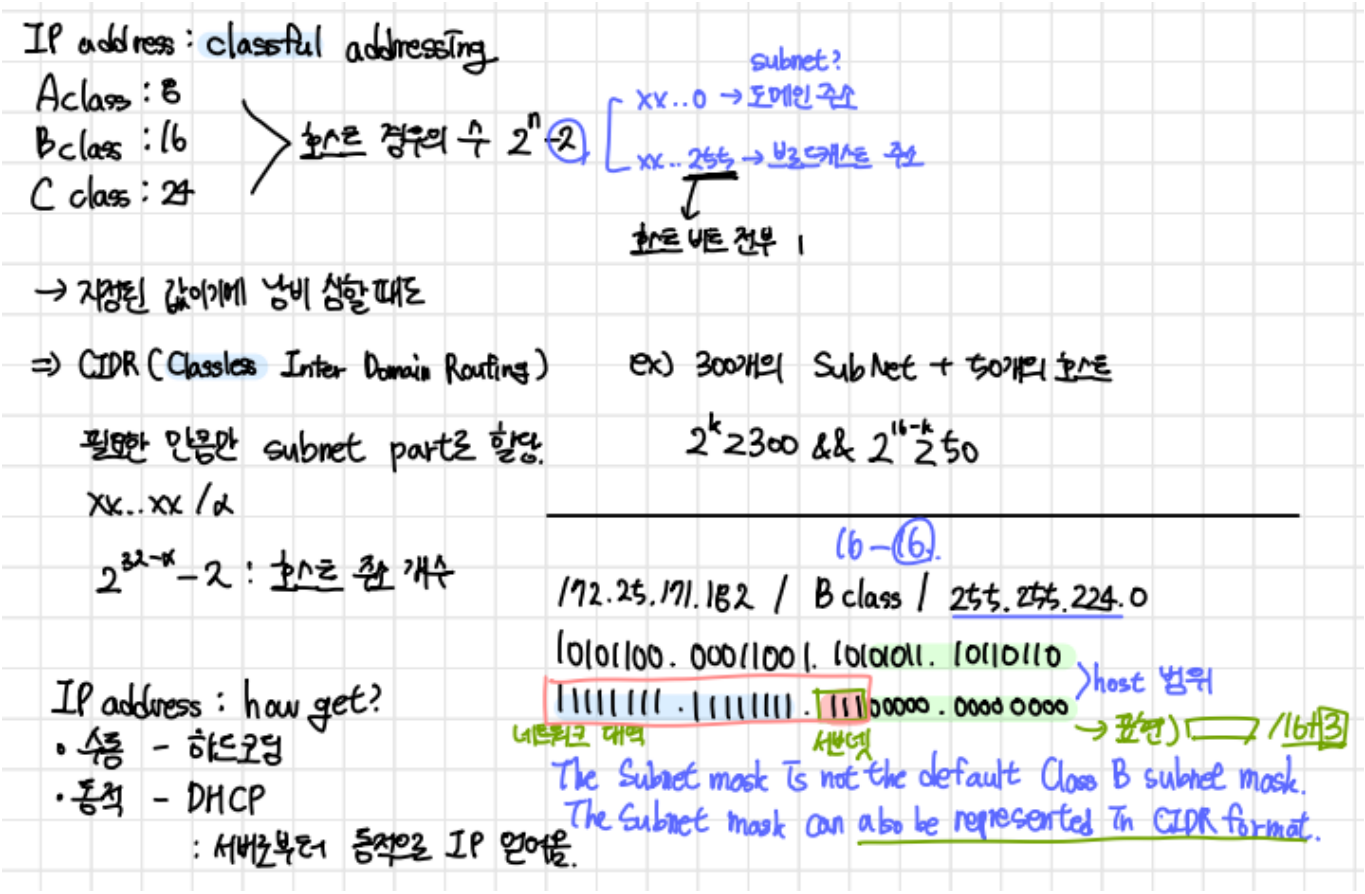
--> 요즘에는 동일하게 사용하는중..

Socket API

- 호스트 식별용 IP주소, 프로세스 식별용 port 필요가 필요함

IP

IPv4 - 32bit / IPv6 - 128bit



class 단위 말고 CIDR로 비트단위로 할당함 10.10.1.32/27: 상위 27비트는 네트워크 주소

- Lookback 인터페이스 127.0.0.1 주소 - 호스트 사용 (loallhost)

소켓 주소

- 구조체

```
sockaddr(AF_INET, port, internet address)
```

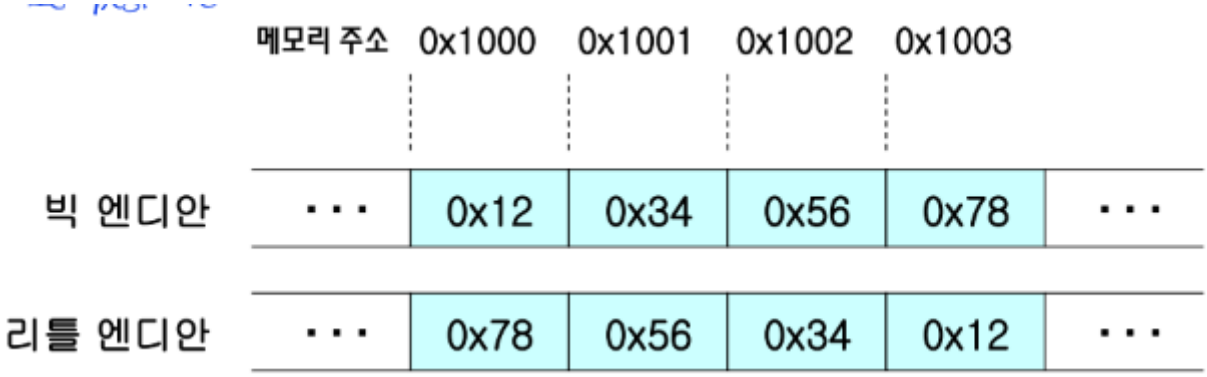
- bind 소켓에 주소를 설정하는 함수

```
bind (fd, const struct sockaddr *addr, len)
```

바이트 순서

메모리에 데이터를 저장할 때 바이트 순서

- 빅엔디안: 상위 바이트 값이 메모리의 작은 주소 (네트워크 바이트)
- 리틀 엔디안: 하위 바이트 값이 메모리의 작은 주소 (호스트 바이트, 보통 사용)



-> 바이트 순서를 고려하지 않으면 오류가 나도 메모리의 어디를 봐야할지 모르게 때문에..

- IP -> 빅엔디안 변경
- Prot -> 빅엔디안 변경
- app에서 주고받는 데이터 -> 빅엔디안 / 리틀엔디안 통일 필요

1. 엔디안 변경

```
htonl //host to network long
htons //host to network short
ntohs //network to host long
ntohs //network to host short
```

2. IPv4의 점 형식.. 192.168.137.1

```
inet_addr //IP to network
inet_network //IP to host
int inet_aton //IPv4 (196.168.137.2) -> network
char* inet_ntoa //network -> IPv4

//차이점
ip = 127.0.0.1
inet_addr(ip) = inet_aton(ip) 까지와 기능은 동일하지만
inet_addr은 변환해서 직접 넣어줘야하고
inet_aton은 파라미터에 소켓의 주소를 넣을 수 있음
```

3. IPv6를 지원하기 위해 새롭게 바이트 변환 함수..

```
inet_pton //ASCII string to numeric form
inet_ntop //network to string
```

▶ 실습 03:ip-uint.c

```cs
#include <netinet/in.h>
#include <stdio.h>
#include <arpa/inet.h>
```

```c

int main(int argc, char** argv)
{
    uint32_t ipInInt;
    uint32_t ipInInt2;

    char *ipInStr = "127.0.0.1";
    //IPv4 to host
    ipInInt = inet_network(ipInStr);

    printf("String: %s, Decimal: %u, Hex: %X \n",
             ipInStr, ipInInt, ipInInt);

    //host(little) to network(big)
    ipInInt = htonl(ipInInt);
    printf("After htonl - Decimal: %u, Hex: %X \n",
            ipInInt, ipInInt);

    //IP to network
    ipInInt2 = inet_addr(ipInStr);
    printf("after inet_addr: %u, Hex: %x \n", ipInInt2, ipInInt2);

    return 0;
}
```


inet_network() + htonl() == inet_addr() // IP to host / host to network -> IP to network

▶ 실습 04: address-resolution.c

```c
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int printAddr(struct sockaddr_in *);
int main(char argc, char** argv)
{
    char *sampleIP = "127.0.0.1";
    int port = 9002;

    struct sockaddr_in sockAddr1, sockAddr2, sockAddr3;

    sockAddr1.sin_family = AF_INET; //IPv4
    sockAddr1.sin_addr.s_addr = inet_addr(sampleIP); //IP to network
    sockAddr1.sin_port = htons(port); //host to network

    sockAddr2.sin_family = AF_INET;
    inet_aton(sampleIP,&(sockAddr2.sin_addr)); //IP to network 후 파라미터로 저장
    sockAddr2.sin_port = htons(port);

    sockAddr3.sin_family = AF_INET;
    inet_pton(AF_INET, sampleIP, &(sockAddr3.sin_addr)); //IPv4를 선택하고 IP to netw
    sockAddr3.sin_port = htons(port);

    //3개 모두 동일한 결과
    printAddr(&sockAddr1);
    printAddr(&sockAddr2);
```

```
    printAddr(&sockAddr3);

    //network to IPv4
    printf("============ntoa==============\n");
    printf("IP:%s \n",inet_ntoa(sockAddr1.sin_addr));
    printf("IP:%s \n",inet_ntoa(sockAddr2.sin_addr));
    printf("IP:%s \n",inet_ntoa(sockAddr3.sin_addr));

    return 0;
}
int printAddr(struct sockaddr_in *myAddr)
{
    int port;
    char txt[INET_ADDRSTRLEN];

    //network to host (이미 myAddr에 network 형태로 저장되어있기 때문)
    port = ntohs(myAddr->sin_port);

    //network to IP
    //IPv4, socket의 IP, 저장할 공간, 사이즈
    inet_ntop(AF_INET,&(((struct sockaddr_in *)myAddr)->sin_addr),
        txt,sizeof(struct sockaddr_in));
    printf("IP:%s, Port:%d \n",txt, myAddr->sin_port);
    return 0;
}

```

호스트 이름과 IP 주소

- gethostbyname() 호스트 이름을 이용해 IP주소 조회
- hostent 구조체 name, aliases, addrtype, len, addr_list; //도메인 이름, 별칭, AF_INET.., address list

▶ 실습 05: gethostbyname.c

```c
#include <stdio.h>
#include <netdb.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <stdlib.h>
#include <arpa/inet.h>
#include <string.h>
void errProc(const char *);
int main(int argc, char **argv)
{
    //호스트 구조체
    struct hostent *ent;
    //주소 구조체
    struct in_addr **res;
    int i = 0;
    if(argc != 2)
    {
        fprintf(stderr,"Usage: %s <hostname> \n", argv[0]);
        return -1;
```

```c
    }

    //호스트 이름을 이용해 IP 주소 조회
    ent = gethostbyname(argv[1]);
    if(ent == NULL) errProc("gethostbyname");

    //host의 addr_list 가져오기
    res = (struct in_addr **)ent->h_addr_list;

    //호스트 이름
    printf("hostname: %s \n",ent->h_name);

    //호스트의 addr_list 모두 출력
    while(res[i] != NULL)
    {
        //network to IPv4
        printf("%s ", inet_ntoa(*res[i]));
        i++;
    }
printf("\n");

}

void errProc(const char *str)
{
    fprintf(stderr,"%s: %s\n",str,strerror(errno));
    exit(errno);
}


```



```
ssoxong@DESKTOP-1CG42BG:/mnt/c/users/leeso/NP
/week03$ ./gethostbyname www.naver.com
hostname: www.naver.com.nheos.com
223.130.195.200 223.130.200.107
```

해당 domain(host)에 접근할 수 있는 모든 ip 출력