



# 윤성우의 열혈 TCP/IP 소켓 프로그래밍

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

## Chapter 03. 주소체계와 데이터 정렬



## Chapter 03-1. 소켓에 할당되는 IP주소와 PORT번호

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

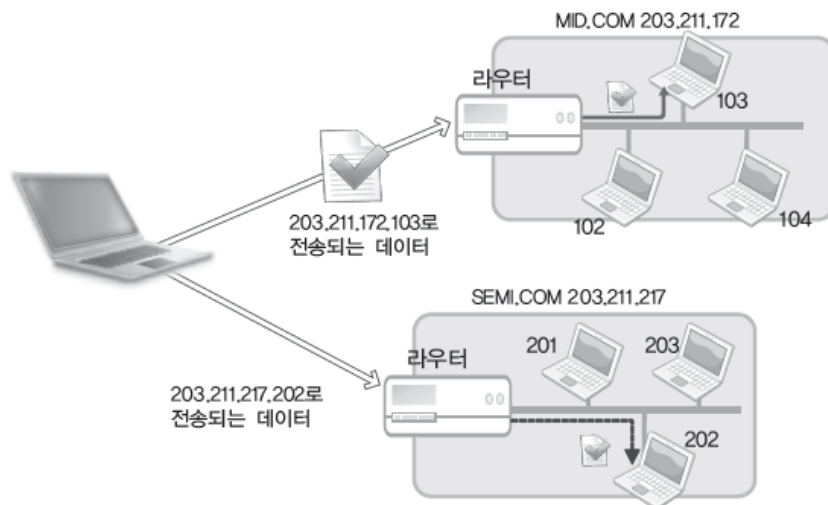
# 인터넷 주소(Internet Address)

## ▶ 인터넷 주소란?

- ▶ 인터넷상에서 컴퓨터를 구분하는 목적으로 사용되는 주소.
- ▶ 4바이트 주소체계인 IPv4와 16바이트 주소체계인 IPv6가 존재한다.
- ▶ 소켓을 생성할 때 기본적인 프로토콜을 지정해야 한다.
- ▶ 네트워크 주소와 호스트 주소로 나뉜다. 네트워크 주소를 이용해서 네트워크를 찾고, 호스트 주소를 이용해서 호스트를 구분한다.



IPv4 인터넷 주소의 체계



인터넷 주소의 역할

# 클래스 별 네트워크 주소와 호스트 주소의 경계

- 클래스 A의 첫 번째 바이트 범위      0이상 127이하
- 클래스 B의 첫 번째 바이트 범위      128이상 191이하
- 클래스 C의 첫 번째 바이트 범위      192이상 223이하



**달리 말하면...**

- 클래스 A의 첫 번째 비트는 항상 0으로 시작
- 클래스 B의 첫 두 비트는 항상 10으로 시작
- 클래스 C의 첫 세 비트는 항상 110으로 시작

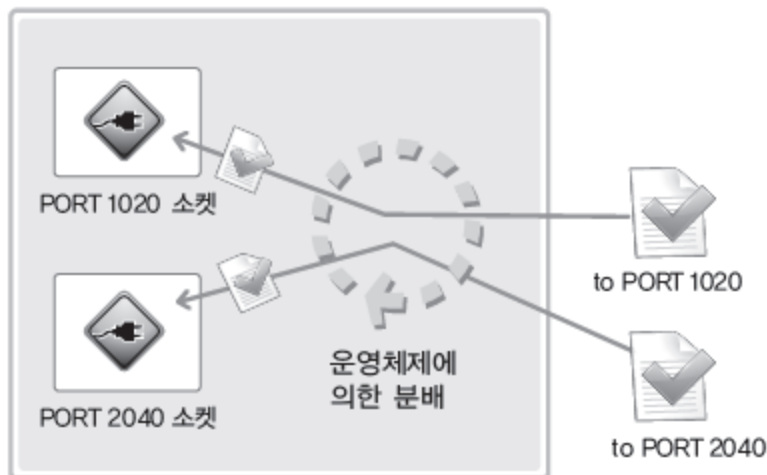
**때문에 첫 번째 바이트 정보만 참조해도 IP주소의 클래스 구분이 가능하며,  
이로 인해서 네트워크 주소와 호스트 주소의 경계 구분이 가능하다.**



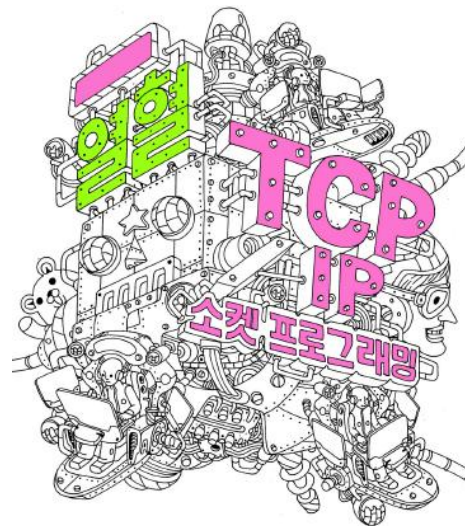
# 소켓의 구분에 활용되는 PORT번호

## ▶ PORT번호

- ▶ IP는 컴퓨터를 구분하는 용도로 사용되며, PORT번호는 소켓을 구분하는 용도로 사용된다.
- ▶ 하나의 프로그램 내에서는 둘 이상의 소켓이 존재할 수 있으므로, 둘 이상의 PORT가 하나의 프로그램에 의해 할당될 수 있다.
- ▶ PORT번호는 16비트로 표현, 따라서 그 값은 0 이상 65535 이하
- ▶ 0~1023은 잘 알려진 PORT(Well-known PORT)라 해서 이미 용도가 결정되어 있다.



PORT번호에 의한 소켓의 구분과정



## Chapter 03-2. 주소정보의 표현

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

# IPv4 기반의 주소표현을 위한 구조체

```
struct sockaddr_in
{
    sa_family_t    sin_family; 주소체계
    uint16_t       sin_port; PORT번호
    struct in_addr sin_addr; 32비트 IP주소
    char           sin_zero[8]; 사용되지 않음
};
```

IP주소와 PORT번호는 구조체 `sockaddr_in`의 변수에 담아서 표현한다.

```
struct in_addr
{
    in_addr_t      s_addr;
};
```

32비트 IPv4 인터넷 주소

자료형 이름	자료형에 담길 정보
int8_t	signed 8-bit int
uint8_t	unsigned 8-bit int (unsigned char)
int16_t	signed 16-bit int
uint16_t	unsigned 16-bit int (unsigned short)
int32_t	signed 32-bit int
uint32_t	unsigned 32-bit int (unsigned long)
sa_family_t	주소체계(address family)
socklen_t	길이정보(length of struct)
in_addr_t	IP주소정보, uint32_t로 정의되어 있음
in_port_t	PORT번호정보, uint16_t로 정의되어 있음

# 구조체 sockaddr\_in의 멤버에 대한 분석

- ▶ 멤버 sin\_family
  - ▶ 주소체계 정보 저장

- ▶ 멤버 sin\_port
  - ▶ 16비트 PORT번호 저장
  - ▶ 네트워크 바이트 순서로 저장

주소체계(Address Family)	의 미
AF_INET	IPv4 인터넷 프로토콜에 적용하는 주소체계
AF_INET6	IPv6 인터넷 프로토콜에 적용하는 주소체계
AF_LOCAL	로컬 통신을 위한 유닉스 프로토콜의 주소체계

- ▶ 멤버 sin\_addr
  - ▶ 32비트 IP주소정보 저장
  - ▶ 네트워크 바이트 순서로 저장
  - ▶ 멤버 sin\_addr의 구조체 자료형 in\_addr 사실상 32비트 정수자료형

- ▶ 멤버 sin\_zero
  - ▶ 특별한 의미를 지니지 않는 멤버
  - ▶ 반드시 0으로 채워야 한다.

0으로 채워야 하는 멤버 sin\_zero의 존재 이유를 이해할 필요가 있다!





## 구조체 sockaddr\_in의 활용의 예

```
struct sockaddr_in serv_addr;
. . . .
if(bind(serv_sock, (struct sockaddr*) &serv_addr, sizeof(serv_addr))==-1)
    error_handling("bind() error");
. . . .
```

☞ 구조체 변수 `sockaddr_in`은 `bind` 함수의 인자로 전달되는데, 매개변수 형이 `sockaddr`이므로 형 변환을 해야만 한다.

```
struct sockaddr
{
    sa_family_t    sin_family;    // 주소체계(Address Family)
    char           sa_data[14];   // 주소정보
};
```

☞ 구조체 `sockaddr`은 다양한 주소체계의 주소정보를 담을 수 있도록 정의되었다. 그래서 IPv4의 주소정보를 담기가 불편하다. 이에 동일한 바이트 열을 구성하는 구조체 `sockaddr_in`이 정의되었으며, 이를 이용해서 쉽게 IPv4의 주소정보를 담을 수 있다.



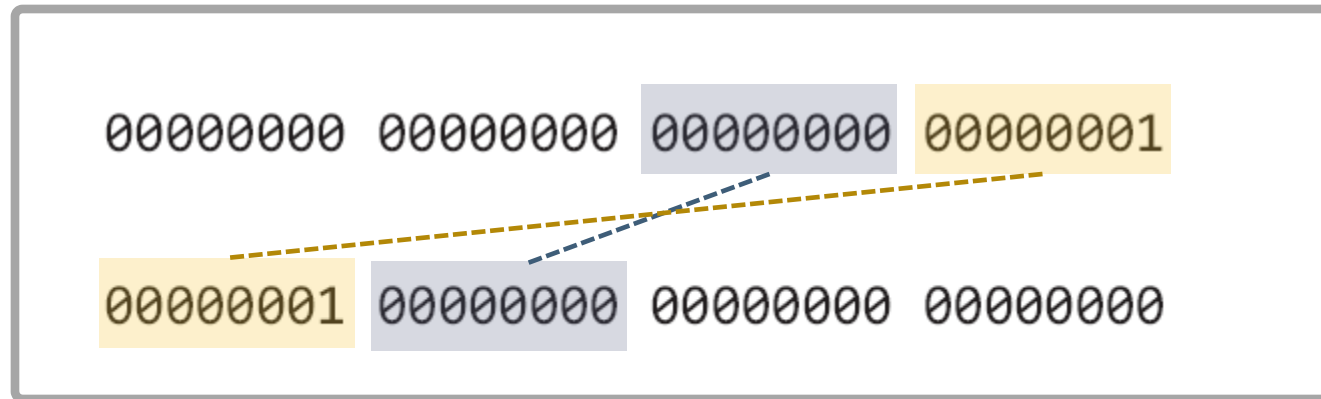


## Chapter 03-3. 네트워크 바이트 순서와 인터넷 주소 변환

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

# CPU에 따라 달라지는 정수의 표현

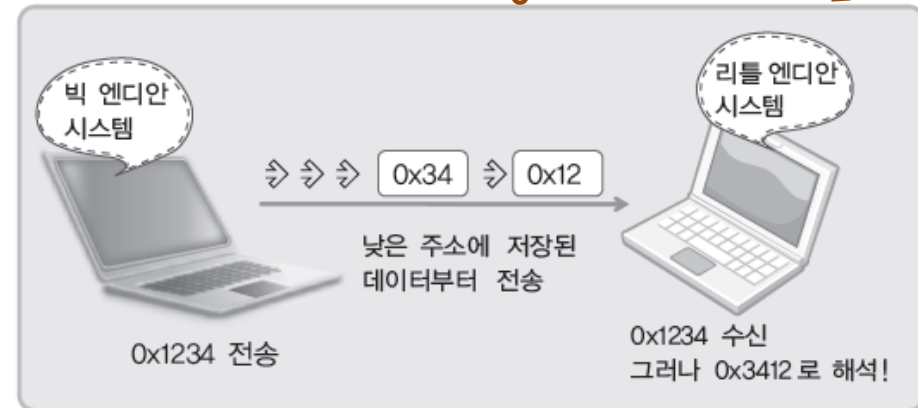
정수 1을 저장하는 두 가지 방법 : CPU가 적응하는....



CPU에 따라서 상위 바이트를 하위 메모리 주소에 저장하기도 하고, 상위 바이트를 상위 메모리 주소에 저장하기도 한다. 즉, CPU마다 데이터를 표현 및 해석하는 방식이 다르다!

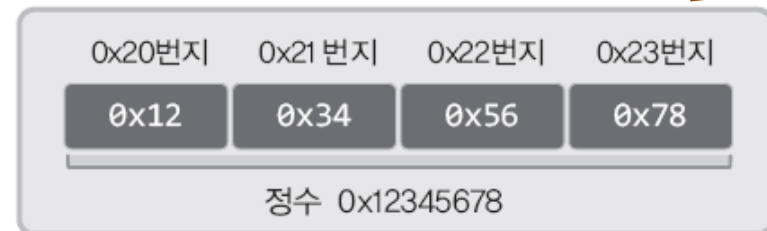
# 바이트 순서(Order)와 네트워크 바이트 순서

## 데이터 송수신 과정에서의 문제

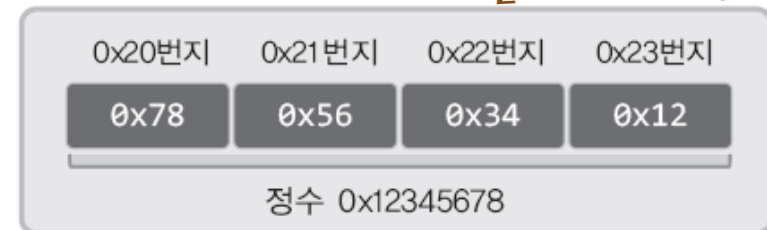


- ▶ 빅 엔디안(Big Endian)
  - ▶ 상위 바이트의 값을 작은 번지수에 저장
- ▶ 리틀 엔디안(Little Endian)
  - ▶ 상위 바이트의 값을 큰 번지수에 저장

## 빅 엔디안 모델



## 리틀 엔디안 모델



- ▶ 호스트 바이트 순서
  - ▶ CPU별 데이터 저장방식을 의미함
- ▶ 네트워크 바이트 순서
  - ▶ 통일된 데이터 송수신 기준을 의미함
  - ▶ 빅 엔디안이 기준이다!

# 바이트 순서의 변환

## 바이트 변환 함수

```
unsigned short htons(unsigned short);
unsigned short ntohs(unsigned short);
unsigned long htonl(unsigned long);
unsigned long ntohl(unsigned long);
```

- htons에서 h는 호스트(host) 바이트 순서를 의미
- htons에서 n은 네트워크(network) 바이트 순서를 의미
- htons에서 s는 자료형 short를 의미
- htonl에서 l은 자료형 long을 의미

이 기준을 적용하면 위 함수가 의미하는 바를 이해할 수 있다.

# 바이트 변환의 예

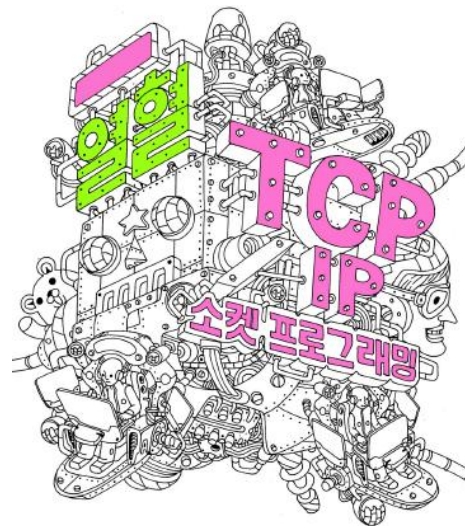
```
int main(int argc, char *argv[])
{
    unsigned short host_port=0x1234;
    unsigned short net_port;
    unsigned long host_addr=0x12345678;
    unsigned long net_addr;

    net_port=htons(host_port);
    net_addr=htonl(host_addr);

    printf("Host ordered port: %#x \n", host_port);
    printf("Network ordered port: %#x \n", net_port);
    printf("Host ordered address: %#lx \n", host_addr);
    printf("Network ordered address: %#lx \n", net_addr);
    return 0;
}
```

실행 결과

```
root@my_linux:/tcpip# gcc endian_conv.c -o conv
root@my_linux:/tcpip# ./conv
Host ordered port: 0x1234
Network ordered port: 0x3412
Host ordered address: 0x12345678
Network ordered address: 0x78563412
```



## Chapter 03-4. 인터넷 주소의 초기화와 할당

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

# 문자열 정보를 네트워크 바이트 순서의 정수로 변환

```
#include <arpa/inet.h>
```

```
in_addr_t inet_addr(const char * string);
```

➔ 성공 시 빅 엔디안으로 변환된 32비트 정수 값, 실패 시 INADDR\_NONE 반환

“211.214.107.99” 와 같이 점이찍힌 10진수로 표현된 문자열을 전달하면, 해당 문자열 정보를 참조해서 IP주소정보를 32비트 정수형으로 반환!

```
int main(int argc, char *argv[])
{
    char *addr1="1.2.3.4";
    char *addr2="1.2.3.256";
    unsigned long conv_addr=inet_addr(addr1);
    if(conv_addr==INADDR_NONE)
        printf("Error occured! \n");
    else
        printf("Network ordered integer addr: %#lx \n", conv_addr);

    conv_addr=inet_addr(addr2);
    if(conv_addr==INADDR_NONE)
        printf("Error occuredded \n");
    else
        printf("Network ordered integer addr: %#lx \n\n", conv_addr);
    return 0;
}
```

실행 결과

```
root@my_linux:/tcip# gcc inet_addr.c -o addr
root@my_linux:/tcip# ./addr
Network ordered integer addr: 0x4030201
Error occuredded
```



# inet\_aton

```
#include <arpa/inet.h>

int inet_aton(const char * string, struct in_addr * addr);
```

→ 성공 시 1(true), 실패 시 0(false) 반환

- string 변환할 IP주소 정보를 담고 있는 문자열의 주소 값 전달.
- addr 변환된 정보를 저장할 in\_addr 구조체 변수의 주소 값 전달.

```
int main(int argc, char *argv[])
{
    char *addr="127.232.124.79";
    struct sockaddr_in addr_inet;

    if(!inet_aton(addr, &addr_inet.sin_addr))
        error_handling("Conversion error");
    else
        printf("Network ordered integer addr: %#x \n",
            addr_inet.sin_addr.s_addr);
    return 0;
}
```

기능상으로 inet\_addr 함수와 동일하다. 다만 in\_addr형 구조체 변수에 변환의 결과가 저장된다는 점에서 차이를 보인다.

## 실행 결과

```
root@my_linux:/tcip# gcc inet_aton.c -o aton
root@my_linux:/tcip# ./aton
Network ordered integer addr: 0x4f7ce87f
```

# inet\_ntoa

```
#include <arpa/inet.h>
```

```
char * inet_ntoa(struct in_addr adr);
```

➔ 성공 시 변환된 문자열의 주소 값, 실패 시 -1 반환

**inet\_aton** 함수의 반대기능 제공! 네트워크 바이트 순서로 정렬된 정수형 IP주소정보를 우리가 눈으로 쉽게 인식할 수 있는 문자열의 형태로 변환.

```
struct sockaddr_in addr1, addr2;
char *str_ptr;
char str_arr[20];

addr1.sin_addr.s_addr=htonl(0x1020304);
addr2.sin_addr.s_addr=htonl(0x1010101);

str_ptr=inet_ntoa(addr1.sin_addr);
strcpy(str_arr, str_ptr);
printf("Dotted-Decimal notation1: %s \n", str_ptr);

inet_ntoa(addr2.sin_addr);
printf("Dotted-Decimal notation2: %s \n", str_ptr);
printf("Dotted-Decimal notation3: %s \n", str_arr);
return 0;
```

## 실행결과

```
root@my_linux:/tcip# gcc inet_ntoa.c -o ntoa
root@my_linux:/tcip# ./ntoa
Dotted-Decimal notation1: 1.2.3.4
Dotted-Decimal notation2: 1.1.1.1
Dotted-Decimal notation3: 1.2.3.4
```

# 인터넷 주소의 초기화

## 일반적인 인터넷 주소의 초기화 과정

```
struct sockaddr_in addr;
char *serv_ip="211.217.168.13";      // IP주소 문자열 선언
char *serv_port="9190";              // PORT번호 문자열 선언
memset(&addr, 0, sizeof(addr));      // 구조체 변수 addr의 모든 멤버 0으로 초기화
addr.sin_family=AF_INET;             // 주소체계 지정
addr.sin_addr.s_addr=inet_addr(serv_ip); // 문자열 기반의 IP주소 초기화
addr.sin_port=htons(atoi(serv_port)); // 문자열 기반의 PORT번호 초기화
```

## 서버에서 주소정보를 설정하는 이유!

"IP 211.217.168.13, PORT 9190으로 들어오는 데이터는 내게로 다 보내라!"

## 클라이언트에서 주소정보를 설정하는 이유!

"IP 211.217.168.13, PORT 9190으로 연결을 해라!"

# INADDR\_ANY



```
struct sockaddr_in addr;  
char *serv_port="9190";  
memset(&addr, 0, sizeof(addr));  
addr.sin_family=AF_INET;  
addr.sin_addr.s_addr=htonl(INADDR_ANY);  
addr.sin_port=htons(atoi(serv_port));
```

현재 실행중인 컴퓨터의 IP를 소켓에 부여할때 사용되는 것이 INADDR\_ANY이다. 이는 서버 프로그램의 구현에 주로 사용된다.



# Chapter 01의 예제 실행방식의 고찰

---



`./hserver 9190`

서버의 실행방식, 서버의 리스닝 소켓 주소는 `INADDR_ANY`로 지정을 하니, 소켓의 PORT번호만 인자를 통해 전달하면 된다.

`./hclient 127.0.0.1 9190`

클라이언트의 실행방식, 연결할 서버의 IP와 PORT번호를 인자로 전달한다.

127.0.0.1은 루프백 주소라 하며, 이는 클라이언트를 실행하는 컴퓨터의 IP주소를 의미한다.

루프백 주소를 전달한 이유는, 서버와 클라이언트를 한 대의 컴퓨터에서 실행시켰기 때문이다.



# 소켓에 인터넷 주소 할당하기

```
#include <sys/socket.h>
```

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

➔ 성공 시 0, 실패 시 -1 반환

- sockfd 주소정보를(IP와 PORT를) 할당할 소켓의 파일 디스크립터.
- myaddr 할당하고자 하는 주소정보를 지니는 구조체 변수의 주소 값.
- addrlen 두 번째 인자로 전달된 구조체 변수의 길이정보.

```
int serv_sock;
```

```
struct sockaddr_in serv_addr;
```

```
char *serv_port="9190";
```

```
/* 서버 소켓(리스닝 소켓) 생성 */
```

```
serv_sock=socket(PF_INET, SOCK_STREAM, 0);
```

```
/* 주소정보 초기화 */
```

```
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(serv_port));
```

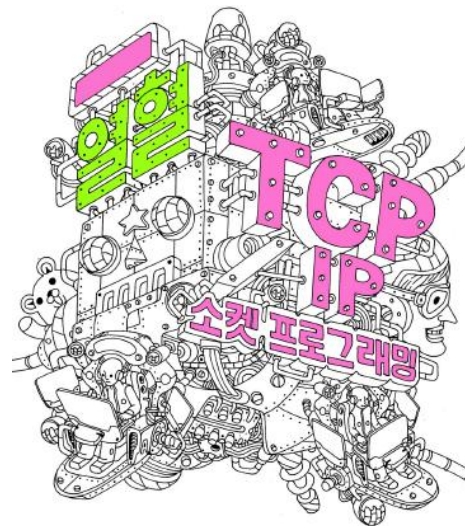
```
/* 주소정보 할당 */
```

```
bind(serv_sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr));
```

```
.....
```

**서버 프로그램에서의 일반적인**

**주소할당의 과정!**



## Chapter 03-5. 윈도우 기반으로 구현하기

윤성우 저 열혈강의 TCP/IP 소켓 프로그래밍 개정판

# 함수 htons, htonl의 윈도우 기반 사용 예

```
int main(int argc, char *argv[])
{
    WSADATA wsaData;
    unsigned short host_port=0x1234;
    unsigned short net_port;
    unsigned long host_addr=0x12345678;
    unsigned long net_addr;

    if(WSAStartup(MAKEWORD(2, 2), &wsaData)!=0)
        ErrorHandling("WSAStartup() error!");

    net_port=htons(host_port);
    net_addr=htonl(host_addr);

    printf("Host ordered port: %#x \n", host_port);
    printf("Network ordered port: %#x \n", net_port);
    printf("Host ordered address: %#lx \n", host_addr);
    printf("Network ordered address: %#lx \n", net_addr);
    WSACleanup();
    return 0;
}
```

실행 결과

```
Host ordered port: 0x1234
Network ordered port: 0x3412
Host ordered address: 0x12345678
Network ordered address: 0x78563412
```



# 함수 inet\_addr, inet\_ntoa의 윈도우 기반 사용 예

```

/* inet_addr 함수의 호출 예 */
{
    char *addr="127.212.124.78";
    unsigned long conv_addr=inet_addr(addr);
    if(conv_addr==INADDR_NONE)
        printf("Error occurred! \n");
    else
        printf("Network ordered integer addr: %#lx \n", conv_addr);
}

/* inet_ntoa 함수의 호출 예 */
{
    struct sockaddr_in addr;
    char *strPtr;
    char strArr[20];

    addr.sin_addr.s_addr=htonl(0x1020304);
    strPtr=inet_ntoa(addr.sin_addr);
    strcpy(strArr, strPtr);
    printf("Dotted-Decimal notation3 %s \n", strArr);
}

```

실행 결과

Network ordered integer addr: 0x4e7cd47f  
Dotted-Decimal notation3 1.2.3.4

# 윈도우에서의 소켓 주소할당

```

SOCKET servSock;
struct sockaddr_in servAddr;
char *servPort="9190";

/* 서버 소켓 생성 */
servSock=socket(PF_INET, SOCK_STREAM, 0);

/* 주소정보 초기화 */
memset(&servAddr, 0, sizeof(servAddr));
servAddr.sin_family=AF_INET;
servAddr.sin_addr.s_addr=htonl(INADDR_ANY);
servAddr.sin_port=htons(atoi(servPort));

/* 주소정보 할당 */
bind(servSock, (struct sockaddr*)&servAddr, sizeof(servAddr));
. . . . .

```

**리눅스에서의 소켓 주소할당과 차이가 없다.**

# WSAStringToAddress

```
#include <winsock2.h>

INT WSAStringToAddress(
    LPTSTR AddressString, INT AddressFamily, LPWSAProtocolInfo lpProtocolInfo,
    LPSOCKADDR lpAddress, LPINT lpAddressLength
);
```

➔ 성공 시 0, 실패 시 SOCKET\_ERROR 반환

- AddressString IP와 PORT번호를 담고 있는 문자열의 주소 값 전달.
- AddressFamily 첫 번째 인자로 전달된 주소정보가 속하는 주소체계 정보전달.
- lpProtocolInfo 프로토콜 프로바이더(Provider) 설정, 일반적으로 NULL 전달.
- lpAddress 주소정보를 담은 구조체 변수의 주소 값 전달.
- lpAddressLength 네 번째 인자로 전달된 주소 값의 변수 크기를 담고 있는 변수의 주소 값 전달.

주소정보를 나타내는 문자열을 가지고, 주소정보 구조체 변수를 적절히 채워 넣을 때 호출하는 함수  
IPv6 기반에서도 사용이 가능! 단, 이 함수를 사용하면, 윈도우에 의존적인 코드가 구성 됨

# WSAAddressToString

```
#include <winsock2.h>
```

```
INT WSAAddressToString(
    LPSOCKADDR lpsaAddress, DWORD dwAddressLength,
    LPWSAProtocolInfo lpProtocolInfo, LPTSTR lpszAddressString,
    LPDWORD lpdwAddressStringLength
);
```

➔ 성공 시 0, 실패 시 SOCKET\_ERROR 반환

- lpsaAddress 문자열로 변환할 주소정보를 지니는 구조체 변수의 주소 값 전달.
- dwAddressLength 첫 번째 인자로 전달된 구조체 변수의 크기 전달.
- lpProtocolInfo 프로토콜 프로바이더(Provider) 설정, 일반적으로 NULL 전달.
- lpszAddressString 문자열로 변환된 결과를 저장할 배열의 주소 값 전달.
- lpdwAddressStringLength 네 번째 인자로 전달된 주소 값의 배열 크기를 담고 있는 변수의 주소 값 전달.

**WSAStringToAddress 함수와 반대의 기능을 제공**



# WSAStringToAddress & WSAAddressToString 의 사용 예



```
char *strAddr="203.211.218.102:9190";
char strAddrBuf[50];
SOCKADDR_IN servAddr;
int size;

WSADATA wsaData;
WSAStartup(MAKEWORD(2, 2), &wsaData);

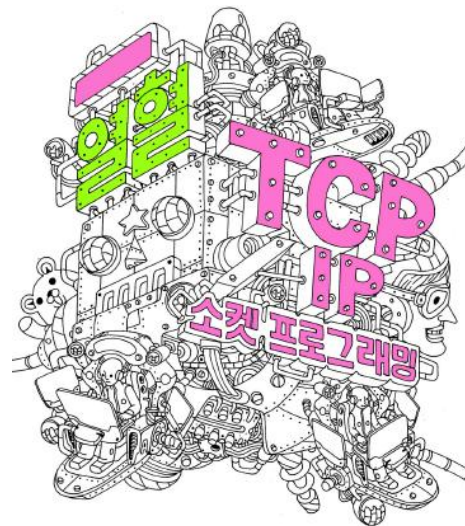
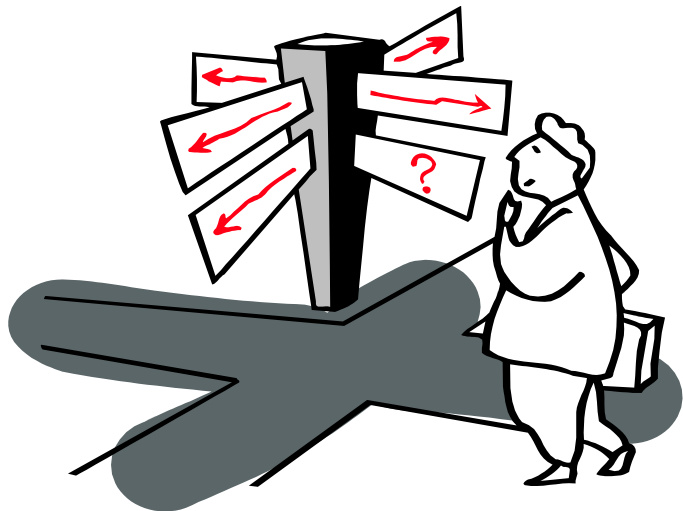
size=sizeof(servAddr);
WSAStringToAddress(
    strAddr, AF_INET, NULL, (SOCKADDR*)&servAddr, &size);

size=sizeof(strAddrBuf);
WSAAddressToString(
    (SOCKADDR*)&servAddr, sizeof(servAddr), NULL, strAddrBuf, &size);

printf("Second conv result: %s \n", strAddrBuf);
WSACleanup();
return 0;
```

실행 결과

Second conv result: 203.211.218.102:9190



Chapter 03이 끝났습니다. 질문 있으신지요?