

Global KPU!

글로벌 경쟁력을 갖춘 산업기술 명문대학  
세계를 향해 더 큰 미래를 펼쳐갑니다

# 3 소켓 구조 구조체 다루기

네트워크 게임 프로그래밍

- ❖ 소켓 주소 구조체의 정의와 초기화 방법을 익힌다.
- ❖ 바이트 정렬 함수의 필요성과 사용법을 익힌다.
- ❖ IP 주소 변환 함수를 익힌다.
- ❖ 도메인 이름 시스템의 동작 원리를 이해하고 이름 변환 함수를 익힌다.



# 소켓 주소 구조체 (1)

## ❖ 소켓 주소 구조체

- 네트워크 프로그램에서 필요로 하는 주소 정보를 담고 있는 구조체로, 다양한 소켓 함수의 인자로 사용
- 프로토콜 체계에 따라 다양한 형태가 존재
  - 예) TCP/IP ⇨ SOCKADDR\_IN{} 또는 SOCKADDR\_IN6{}  
IrDA ⇨ SOCKADDR\_IRDA{}
- 기본형은 SOCKADDR 구조체임



# 소켓 주소 구조체 (2)

## ❖ SOCKADDR 구조체 – 기본형

- ws2def.h 에 정의

```
typedef struct sockaddr {  
    u_short  sa_family;  
    char     sa_data[14];  
} SOCKADDR;
```

- sa\_family
  - 주소 체계를 나타내는 16비트 정수 값  
예) TCP/IP 프로토콜 ⇨ AF\_INET 또는 AF\_INET6
- sa\_data
  - 해당 주소 체계에서 사용할 주소 정보
  - 주소 체계에 따라 필요한 정보가 상이하므로 바이트 배열로 선언  
예) TCP/IP 프로토콜 ⇨ IP 주소와 포트 번호



# 소켓 주소 구조체 (3)

## ❖ SOCKADDR\_IN 구조체 - IPv4 전용

```
typedef struct sockaddr_in {  
    short          sin_family;      // AF_INET  
    u_short        sin_port;       // 포트 번호  
    struct in_addr  sin_addr;      // IPv4 주소  
    char           sin_zero[8];    // 0으로 설정  
} SOCKADDR_IN;
```

- sin\_family
  - 주소 체계를 의미. AF\_INET / AF\_INET6 값을 사용.
- sin\_port
  - 포트번호를 의미, 부호없는 16비트 정수 값
- sin\_addr
  - IP 주소를 의미, 32비트 구조체



## 소켓 주소 구조체 (4)

### ❖ SOCKADDR\_IN6 구조체 - IPv6 전용

```
typedef struct sockaddr_in6 {  
    short          sin6_family;      // AF_INET6  
    u_short        sin6_port;        // 포트 번호  
    u_long         sin6_flowinfo;    // 대부분 0으로 설정  
    struct in6_addr sin6_addr;       // IPv6 주소  
    u_long         sin6_scope_id;    // 대부분 0으로 설정  
} SOCKADDR_IN6;
```

- sin\_family
  - 주소 체계를 의미. AF\_INET / AF\_INET6 값을 사용.
- sin6\_port
  - 포트번호를 의미, 부호없는 16비트 정수 값
- sin6\_addr
  - IP 주소를 의미, 128비트 구조체



# 소켓 주소 구조체 (5)

## ❖ IN\_ADDR 구조체 - IPv4 주소 저장용 - inaddr.h 정의

```
typedef struct in_addr {  
    union {  
        struct { u_char s_b1, s_b2, s_b3, s_b4; } S_un_b;  
        struct { u_short s_w1, s_w2; } S_un_w;  
        u_long S_addr;  
    } S_un;  
#define s_addr S_un.S_addr  
} IN_ADDR;
```

- IPv4 주소를 담는 in\_addr 구조체는 동일 메모리 영역을 각각 8bit (S\_un\_b), 16bit(S\_un\_w), 32bit(S\_addr) 단위로 접근할 수 있도록 만든 공용체(S\_un)임
- 보통은 32bit 단위로 접근하므로 S\_un.S\_addr 필드를 사용
- 매크로를 통해 재정의된 s\_addr를 사용하면 편리함



# 소켓 주소 구조체 (6)

## ❖ IN6\_ADDR 구조체 - IPv6 주소 저장용

```
typedef struct in6_addr {  
    union {  
        u_char Byte[16];  
        u_short Word[8];  
    } u;  
} IN6_ADDR;
```

- in\_addr 구조체와 달리 IPv6 주소를 담는 in6\_addr 구조체는 단순한 바이트 또는 워드 배열로 정의되어 있음.





## 소켓 주소 구조체 (7)

### ❖ 소켓 주소 구조체 크기 비교

구조체 이름	전체 크기(바이트 단위)
SOCKADDR	16
SOCKADDR_IN	16
SOCKADDR_IN6	28
SOCKADDR_IRDA	32
SOCKADDR_BTH	30



# 소켓 주소 구조체 (8)

## ❖ 소켓 주소 구조체 사용 예 (2 type)

- 반드시 SOCKADDR 포인터형으로 변환(type casting)해야 함
- 예1) 응용프로그램이 소켓 주소 구조체를 초기화하고 소켓 함수를 넘겨주는 경우

```
// 소켓 주소 구조체를 초기화한다.  
SOCKADDR_IN addr;  
...  
  
SocketFunc(..., (SOCKADDR *)&addr, sizeof(addr), ...);
```

- 예2) 소켓함수가 소켓 주소 구조체를 입력받아 내용을 채우면, 응용 프로그램이 이를 출력등의 목적으로 사용하는 경우

```
SOCKADDR_IN addr;  
SocketFunc(..., (SOCKADDR *)&addr, sizeof(addr), ...);  
  
// 소켓 주소 구조체를 사용한다.  
...
```

# 바이트 정렬 함수 (1)

## ❖ 바이트 정렬

### ■ 메모리에 데이터를 저장할 때 바이트 순서

- 빅 엔디안(big-endian), 리틀 엔디안(little-endian)
- 시스템에서 사용하는 바이트 정렬 방식은 CPU와 운영체제에 따라 상이함.
- 예) 0x12345678 을 메모리 0x1000번지에 저장할 때 두 바이트 정렬 방식 비교

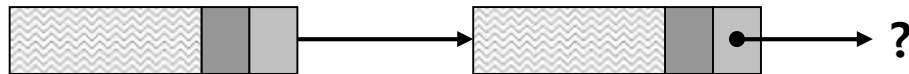
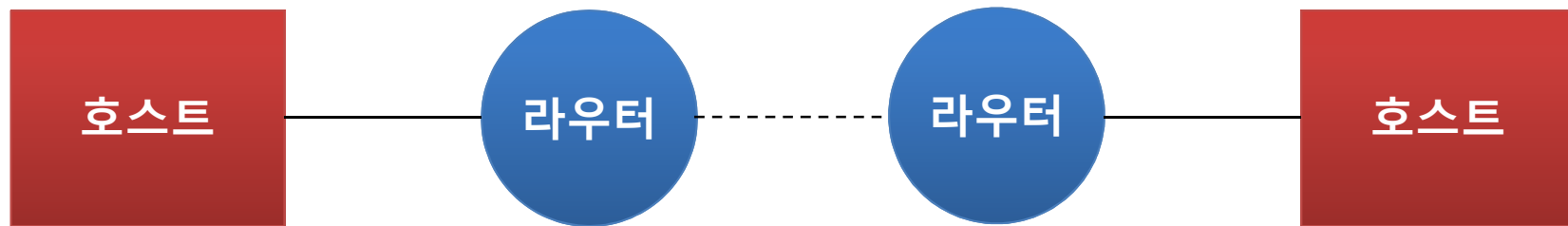
		0x1000	0x1001	0x1002	0x1003	
		⋮	⋮	⋮	⋮	
빅 엔디안	...	0x12	0x34	0x56	0x78	...
리틀 엔디안	...	0x78	0x56	0x34	0x12	...

파일에 데이터를 저장하고 읽어오는 경우나 네트워크를 통해 데이터를 송신하고 수신하는 경우에는 바이트 정렬 방식에 유의해야 한다.

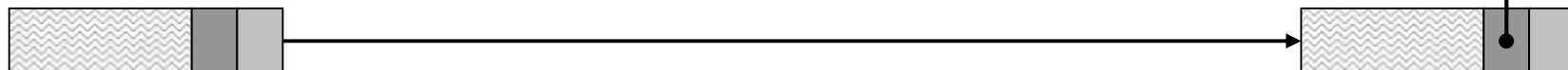
빅 엔디안과 리틀 엔디안..  
무엇이 좋을까요?

## 바이트 정렬 함수 (2)

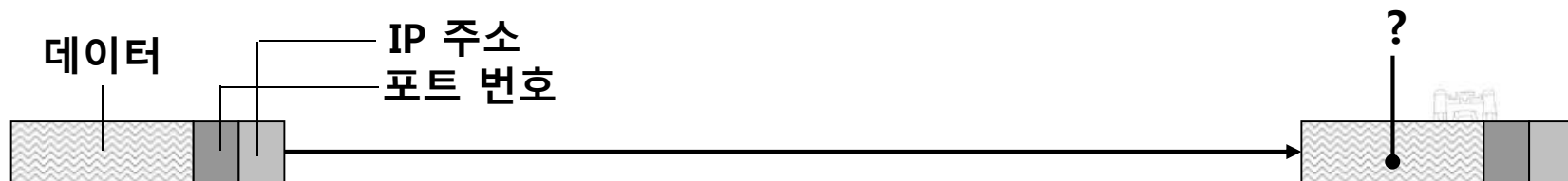
❖ 바이트 정렬 방식을 고려해야 하는 경우



(a) IP 주소에서 문제가 발생하는 상황



(b) 포트 번호에서 문제가 발생하는 상황



(c) 응용 프로그램 데이터에서 문제가 발생하는 상황



# 바이트 정렬 함수 (3)

## ❖ 바이트 정렬 방식을 고려해야 하는 경우

### ■ 프로토콜 구현을 위해 필요한 정보

- 호스트와 라우터가 IP 주소의 바이트 정렬 방식을 약속하지 않으면 IP 주소 해석이 달라져 라우팅에 문제 발생
- (a) IP 주소 ⇨ 빅 엔디안으로 통일
- (b) 포트 번호 ⇨ 빅 엔디안으로 통일

### ■ 응용 프로그램이 주고 받는 데이터

- 서버와 클라이언트를 동시에 개발시에는 바이트 정렬 방식을 통일하고 클라이언트만 개발시에는 서버의 바이트 정렬 방식을 따른다.
- (c) 빅 엔디안 또는 리틀 엔디안으로 통일

## 참고

### ❖ 네트워크 바이트 정렬 (network byte ordering)

: 빅 엔디안 방식

### ❖ 호스트 바이트 정렬 (host byte ordering)

: 시스템이 사용하는 고유한 바이트 정렬 방식



# 바이트 정렬 함수 (4)

## ❖ 바이트 정렬 함수(유닉스 호환)

```
u_short htons(u_short hostshort);    // host-to-network-short
u_long htonl(u_long hostlong);       // host-to-network-long
u_short ntohs(u_short netshort);     // network-to-host-short
u_long ntohl(u_long netlong);        // network-to-host-long
```

## ❖ 바이트 정렬 함수(윈속 확장)

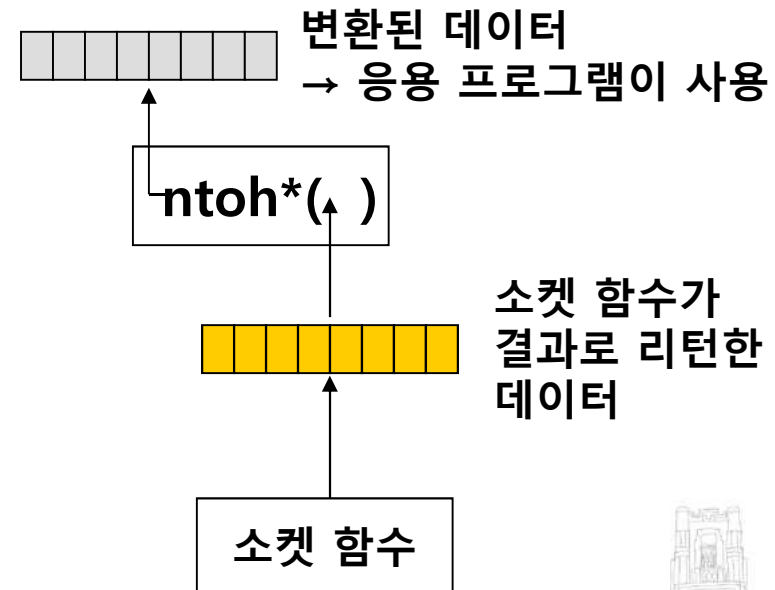
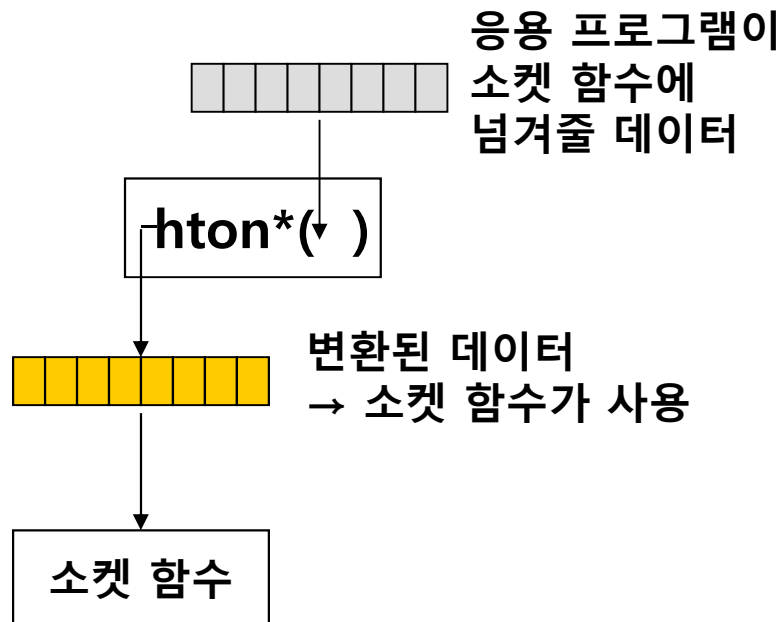
```
int WSAHtons(SOCKET s, u_short hostshort, u_short *lpnetshort);
int WSAHtonl(SOCKET s, u_long hostlong, u_long *lpnetlong);
int WSANTohs(SOCKET s, u_short netshort, u_short *lpnetshort);
int WSANTohl(SOCKET s, u_long netlong, u_long *lpnetlong);
```



# 바이트 정렬 함수 (5)

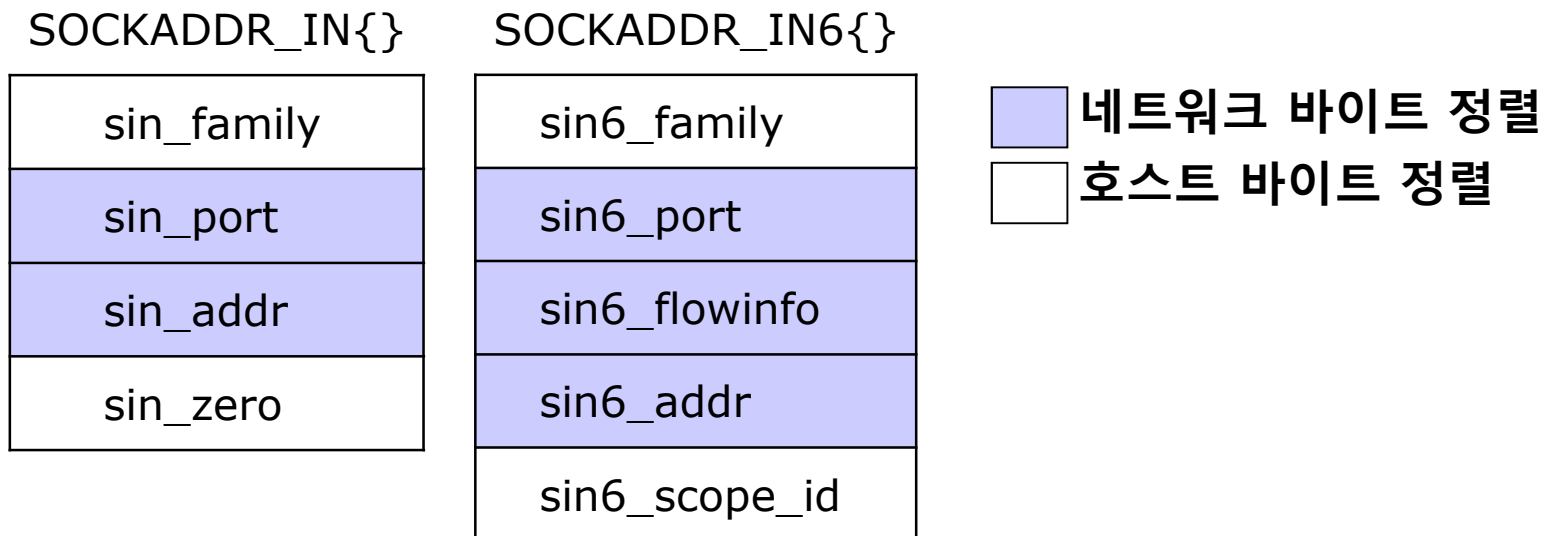
## ❖ 바이트 정렬 함수 사용 상황

- `hton*()` 함수는 응용프로그램이 소켓 함수에 데이터를 넘겨주기 전에 호출되며, `ntoh*()` 함수는 소켓 함수가 결과로 리턴한 데이터를 응용 프로그램이 출력등의 목적으로 이용하기 전에 호출



# 바이트 정렬 함수 (6)

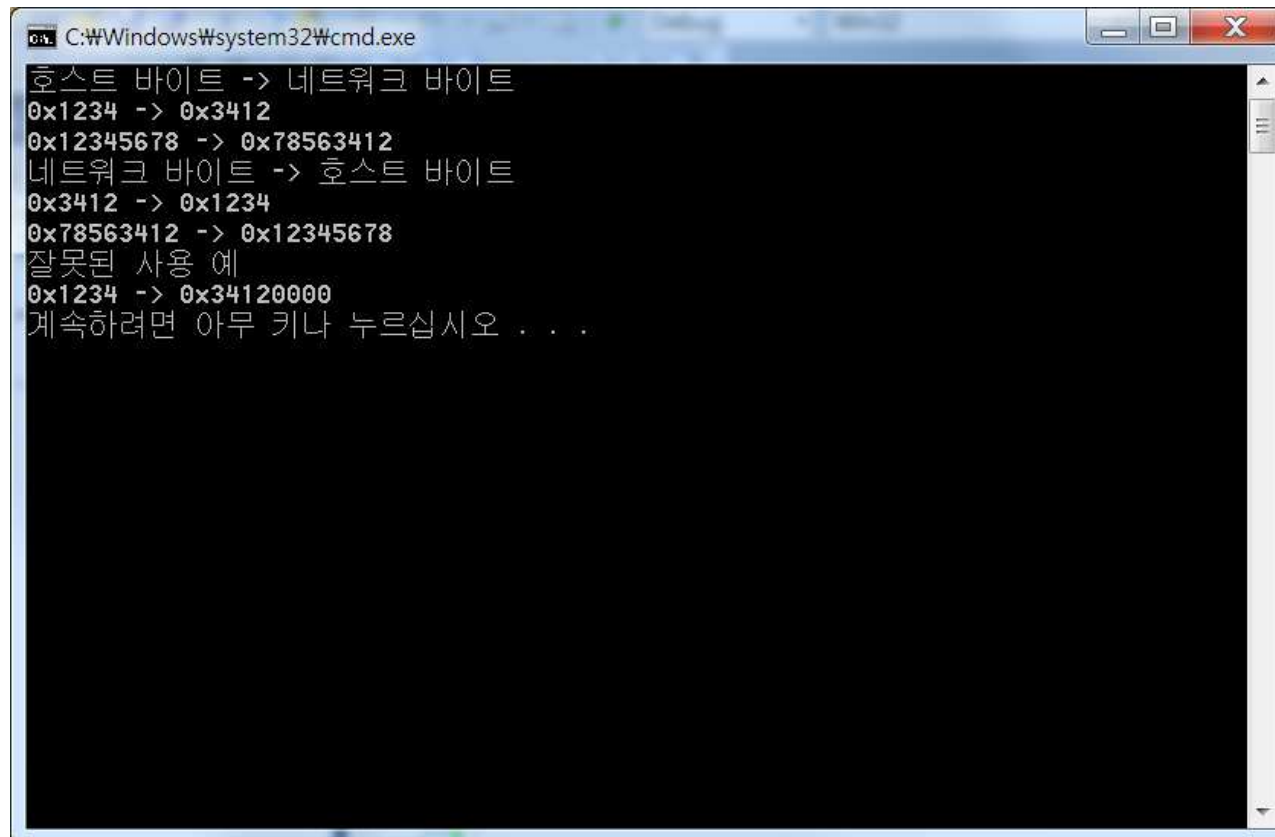
## ❖ SOCKADDR\_IN/SOCKADDR\_IN6 구조체의 바이트 정렬 방식





# 예제 실습

## 실습 3-1 바이트 정렬 함수 연습 p. 71

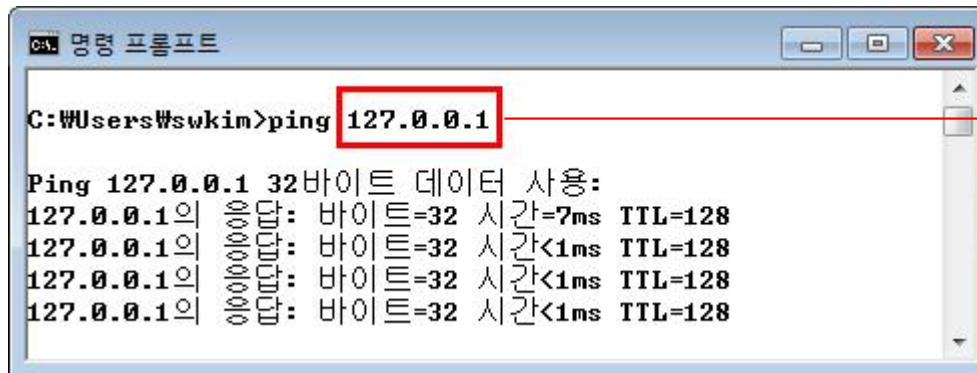


```
C:\Windows\system32\cmd.exe
호스트 바이트 -> 네트워크 바이트
0x1234 -> 0x3412
0x12345678 -> 0x78563412
네트워크 바이트 -> 호스트 바이트
0x3412 -> 0x1234
0x78563412 -> 0x12345678
잘못된 사용 예
0x1234 -> 0x34120000
계속하려면 아무 키나 누르십시오 . . .
```



# IP 주소 변환 함수 (1)

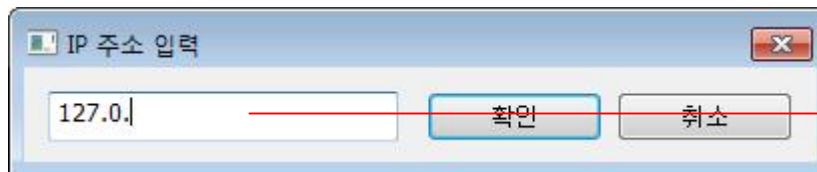
## ❖ IP 주소 변환 예



```
C:\Users\swkim>ping 127.0.0.1

Ping 127.0.0.1 32바이트 데이터 사용:
127.0.0.1의 응답: 바이트=32 시간=7ms TTL=128
127.0.0.1의 응답: 바이트=32 시간<1ms TTL=128
127.0.0.1의 응답: 바이트=32 시간<1ms TTL=128
127.0.0.1의 응답: 바이트=32 시간<1ms TTL=128
```

→ 명령행 인자

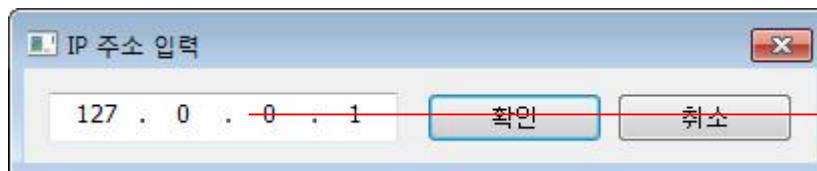


IP 주소 입력

127.0.

확인 취소

→ 편집 컨트롤



IP 주소 입력

127 . 0 . 0 . 1

확인 취소

→ IP 주소 컨트롤



# IP 주소 변환 함수 (2)

## ❖ IPv4 주소 변환 함수

```
unsigned long inet_addr (const char *cp);
```

- 문자열 -> 숫자
- 문자열 형태로 IPv4 주소 입력  
⇒ 32비트 숫자(네트워크 바이트 정렬)로 리턴

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

- 숫자 -> 문자열
- 32비트 숫자(네트워크 바이트 정렬)로 IPv4 주소 입력  
⇒ 문자열 형태로 리턴



# IP 주소 변환 함수 (3)

## ❖ IPv4 또는 IPv6 주소 변환 함수(문자열 ⇒ 숫자)

- inet\_addr()/inet\_ntoa() 함수와 달리 IPv4와 IPv6 주소 변환을 모두 지원

```
int WSAStringToAddress (  
    // 문자열 형식의 IP 주소  
    LPTSTR AddressString,  
    // AF_INET 또는 AF_INET6  
    INT AddressFamily,  
    // NULL  
    LPWSAProtocolInfo IpProtocolInfo,  
    // IP 주소(숫자)를 저장할 구조체; SOCKADDR_IN 또는 SOCKADDR_IN6  
    LPSOCKADDR IpAddress,  
    // 주소 구조체의 길이  
    LPINT IpAddressLength  
);
```



# IP 주소 변환 함수 (4)

## ❖ IPv4 또는 IPv6 주소 변환 함수(숫자 ⇒ 문자열)

- inet\_addr()/inet\_ntoa() 함수와 달리 IPv4와 IPv6 주소 변환을 모두 지원

```
int WSAAddressToString (  
    // 숫자 형식의 IP 주소; SOCKADDR_IN 또는 SOCKADDR_IN6  
    LPSOCKADDR IpsaAddress,  
    // 주소 구조체의 길이  
    DWORD dwAddressLength,  
    // NULL  
    LPWSAProtocolInfo IpProtocolInfo,  
    // IP 주소(문자열)를 저장할 버퍼  
    LPTSTR IpszAddressString,  
    // 버퍼의 길이  
    LPDWORD lpdwAddressStringLength  
);
```

# IP 주소 변환 함수 (5)

## ❖ 바이트 정렬 함수와 IP 주소 변환 함수 사용 예 ①

- 응용 프로그램이 소켓 주소 구조체를 초기화하고, 소켓 함수에 넘겨주는 경우
- 바이트 정렬 함수와 IP 주소 변환 함수를 SOCKADDR\_IN (IPv4 소켓 주소 구조체) 구조체에 사용하는 경우
- SocketFunc() 는 임의의 소켓 함수를 나타냄

// 소켓 주소 구조체를 초기화한다.

```
SOCKADDR_IN addr;
```

```
ZeroMemory(&addr, sizeof(addr));          /* 0으로 채운다. */
```

```
addr.sin_family = AF_INET;
```

```
addr.sin_addr.s_addr = inet_addr("147.46.114.70");
```

```
addr.sin_port = htons(9000);
```

// 소켓 함수를 호출한다.

```
SocketFunc(..., (SOCKADDR *)&addr, sizeof(addr), ...);
```



# IP 주소 변환 함수 (6)

## ❖ 바이트 정렬 함수와 IP 주소 변환 함수 사용 예 ②

- 소켓 함수가 소켓 주소 구조체를 입력으로 받아 내용을 채우면, 응용 프로그램이 이를 출력 등의 목적으로 사용하는 경우
- 바이트 정렬 함수와 IP 주소 변환 함수를 SOCKADDR\_IN (IPv4 소켓 주소 구조체) 구조체에 사용하는 경우
- SocketFunc() 는 임의의 소켓 함수를 나타냄

// 소켓 함수를 호출한다.

```
SOCKADDR_IN addr;  
int addrlen = sizeof(addr);  
SocketFunc(..., (SOCKADDR *)&addr, &addrlen, ...);
```

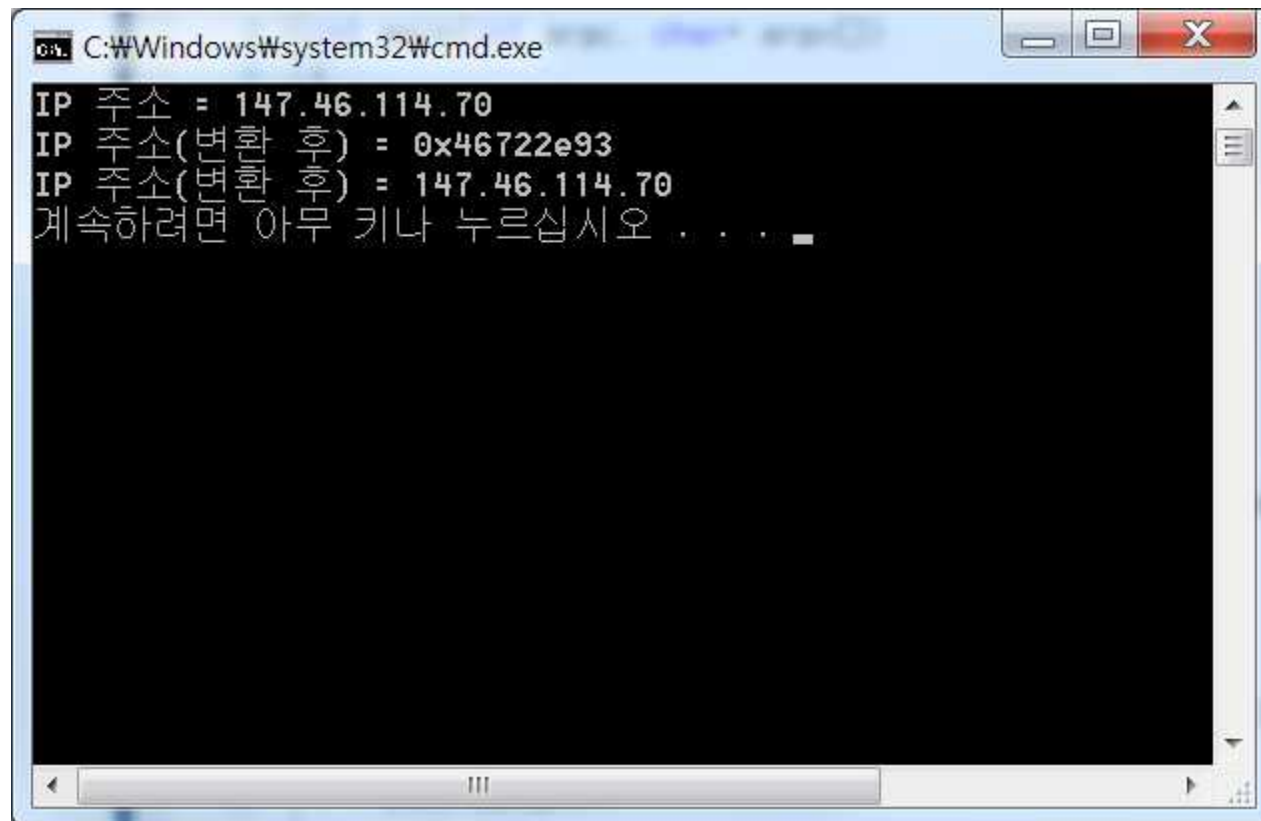
// 소켓 주소 구조체를 사용한다.

```
printf("IP 주소=%s, 포트 번호=%d\n",  
       inet_ntoa(addr.sin_addr), ntohs(addr.sin_port));
```



# 예제 실습

## 실습 3-2 IP 주소 변환 함수 활용 p76



```
C:\Windows\system32\cmd.exe
IP 주소 = 147.46.114.70
IP 주소(변환 후) = 0x46722e93
IP 주소(변환 후) = 147.46.114.70
계속하려면 아무 키나 누르십시오 . . .
```



# 도메인 이름 시스템과 이름 변환 함수 (1)

## ❖ 도메인 이름

- IP 주소처럼 호스트나 라우터의 고유한 식별자로 사용
- IP 주소보다 기억하고 사용하기 쉬움

## ❖ 도메인 이름 ⇨ IP 주소 변환 예

```
C:\명령 프롬프트

C:\Users\swkin>ping www.example.com

Ping www.example.com [192.0.32.10] 32바이트 데이터 사용:
192.0.32.10의 응답: 바이트=32 시간=244ms TTL=128
192.0.32.10의 응답: 바이트=32 시간=258ms TTL=128
192.0.32.10의 응답: 바이트=32 시간=270ms TTL=128
192.0.32.10의 응답: 바이트=32 시간=280ms TTL=128
```

```
C:\명령 프롬프트

C:\Users\swkin>ping -6 www.example.com

Ping www.example.com [2620:0:2d0:200::10] 32바이트 데이터 사용:
2620:0:2d0:200::10의 응답: 시간=578ms
2620:0:2d0:200::10의 응답: 시간=169ms
2620:0:2d0:200::10의 응답: 시간=156ms
2620:0:2d0:200::10의 응답: 시간=164ms
```

# 도메인 이름 시스템과 이름 변환 함수 (2)

## ❖ 도메인 이름 ⇔ IP 주소 변환 함수

- 두 함수 모두 hostent 구조체형 포인터를 리턴
- 종종 NULL 값이 리턴되므로 예외 처리 필요 => WSAGetLastError()

/\* 도메인 이름 → IP 주소(네트워크 바이트 정렬) \*/

```
struct hostent *gethostbyname (  
    const char *name           // 도메인 이름  
);
```

/\* IP 주소(네트워크 바이트 정렬) → 도메인 이름 \*/

```
struct hostent *gethostbyaddr (  
    const char *addr, // IP 주소(네트워크 바이트 정렬)  
    int len,          // IP 주소의 길이  
    int type           // 주소 체계(AF_INET 또는 AF_INET6)  
);
```

## ❖ hostent 구조체

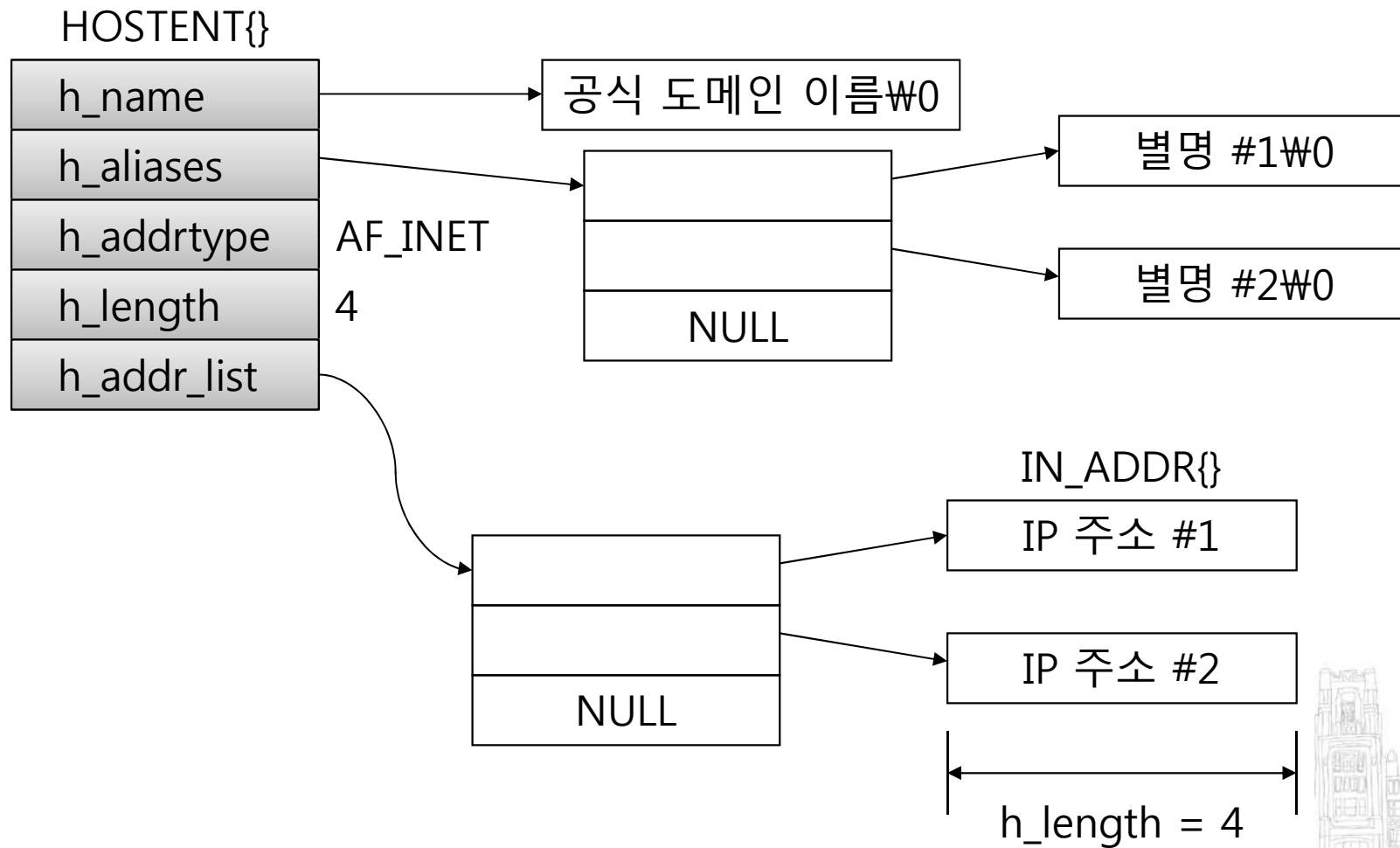
```
typedef struct hostent {  
    char*      h_name;           // official name of host  
    char**     h_aliases;       // alias list  
    short      h_addrtype;      // host address type  
    short      h_length;        // length of address  
    char**     h_addr_list;     // list of addresses  
#define h_addr h_addr_list[0] // address, for backward compatibility  
} HOSTENT;
```

- h\_name: 공식 도메인 이름
- h\_aliases: 공식 도메인 이름 이외의 별명
- h\_addrtype: 주소 체계. AF\_INET, AF\_INET6
- h\_length: IP 주소의 길이. 4 or 16 bytes
- h\_addr\_list: 네트워크 바이트 정렬된 IP 주소. 한 호스트가 여러 IP를 가질 경우 이 포인터를 통해 모든 IP 주소를 얻을 수 있음. 특정 호스트에 접속할때는 보통은 0번째 IP에 접근 (h\_addr\_list[0])



# 도메인 이름 시스템과 이름 변환 함수 (4)

## ❖ hostent 구조체 - IPv4를 사용하는 경우



산업기술의 최고 명문대학  
**한국산업기술대학교**

The diagram illustrates the structure of the `HOSTENT` structure and its pointers. The `HOSTENT` structure is defined as follows:

```
struct HOSTENT {
    char *h_name;
    char **h_aliases;
    int h_addrtype;
    int h_length;
    char **h_addr_list;
};
```

The diagram shows the following components and their relationships:

- HOSTENT structure:** A vertical stack of fields: `h_name`, `h_aliases`, `h_addrtype`, `h_length`, and `h_addr_list`.
- h\_name:** Points to a box labeled "공식 도메인 이름w0" (Official domain name w0).
- h\_aliases:** Points to a vertical stack of three boxes. The bottom box is labeled "NULL". The top two boxes point to "별명 #1w0" (Alias #1 w0) and "별명 #2w0" (Alias #2 w0).
- h\_addrtype:** Labeled as "AF\_INET6".
- h\_length:** Labeled as "16".
- h\_addr\_list:** Points to a vertical stack of three boxes. The bottom box is labeled "NULL". The top two boxes point to "IP 주소 #1" (IP address #1) and "IP 주소 #2" (IP address #2).
- IN6\_ADDR structure:** A vertical stack of two boxes labeled "IP 주소 #1" and "IP 주소 #2". A horizontal double-headed arrow below these boxes is labeled "h\_length = 16".



# 도메인 이름 시스템과 이름 변환 함수 (6)

## ❖ 사용자 정의 함수 ①

```
// 도메인 이름 -> IPv4 주소
BOOL GetIPAddr(char *name, IN_ADDR *addr)
{
    HOSTENT *ptr = gethostbyname(name);
    if(ptr == NULL){
        err_display("gethostbyname()");
        return FALSE;
    }
    if(ptr->h_addrtype != AF_INET)
        return FALSE;
    memcpy(addr, ptr->h_addr, ptr->h_length);
    return TRUE;
}
```

- gethostbyname() 함수를 호출해 HOSTENT 구조체형 포인터를 얻음
- 오류가 발생하면 오류를 출력하고 FALSE를 리턴
- AF\_INET 주소 체계가 아니면 FALSE를 리턴
- IP 주소를 복사한 후 TRUE를 리턴



# 도메인 이름 시스템과 이름 변환 함수 (7)

## ❖ 사용자 정의 함수 ②

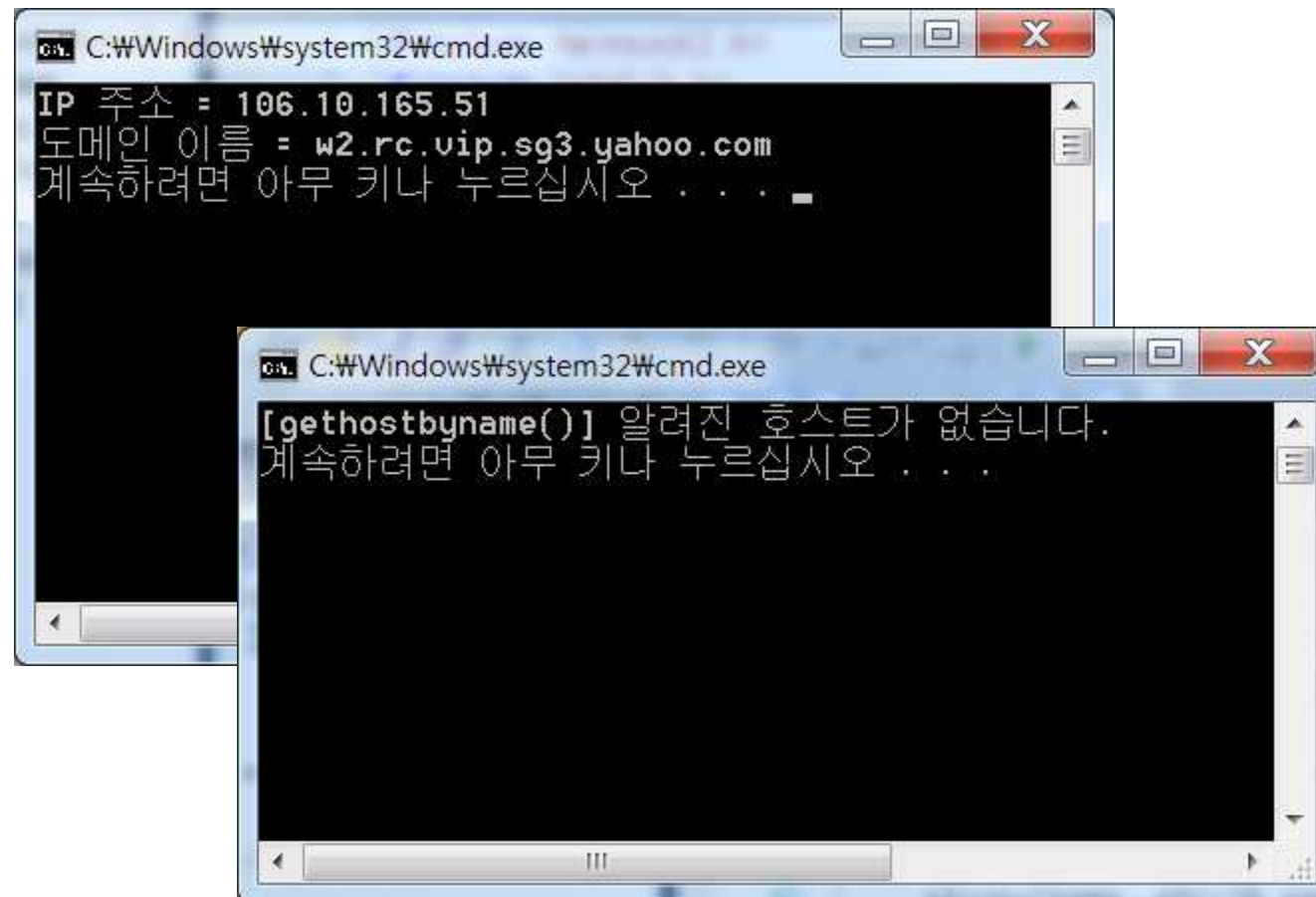
```
// IPv4 주소 -> 도메인 이름
BOOL GetDomainName(IN_ADDR addr, char *name, int namelen)
{
    HOSTENT *ptr = gethostbyaddr((char *)&addr, sizeof(addr), AF_INET);
    if(ptr == NULL){
        err_display("gethostbyaddr()");
        return FALSE;
    }
    if(ptr->h_addrtype != AF_INET)
        return FALSE;
    strncpy(name, ptr->h_name, namelen);
    return TRUE;
}
```

- gethostbyaddr() 함수를 호출해 HOSTENT 구조체형 포인터를 얻음
- 오류가 발생하면 오류를 출력하고 FALSE를 리턴
- AF\_INET 주소 체계가 아니면 FALSE를 리턴
- 도메인 이름을 복사한 후 TRUE를 리턴



# 예제 실습

## 실습 3-3 이름 변환 함수 활용 p82

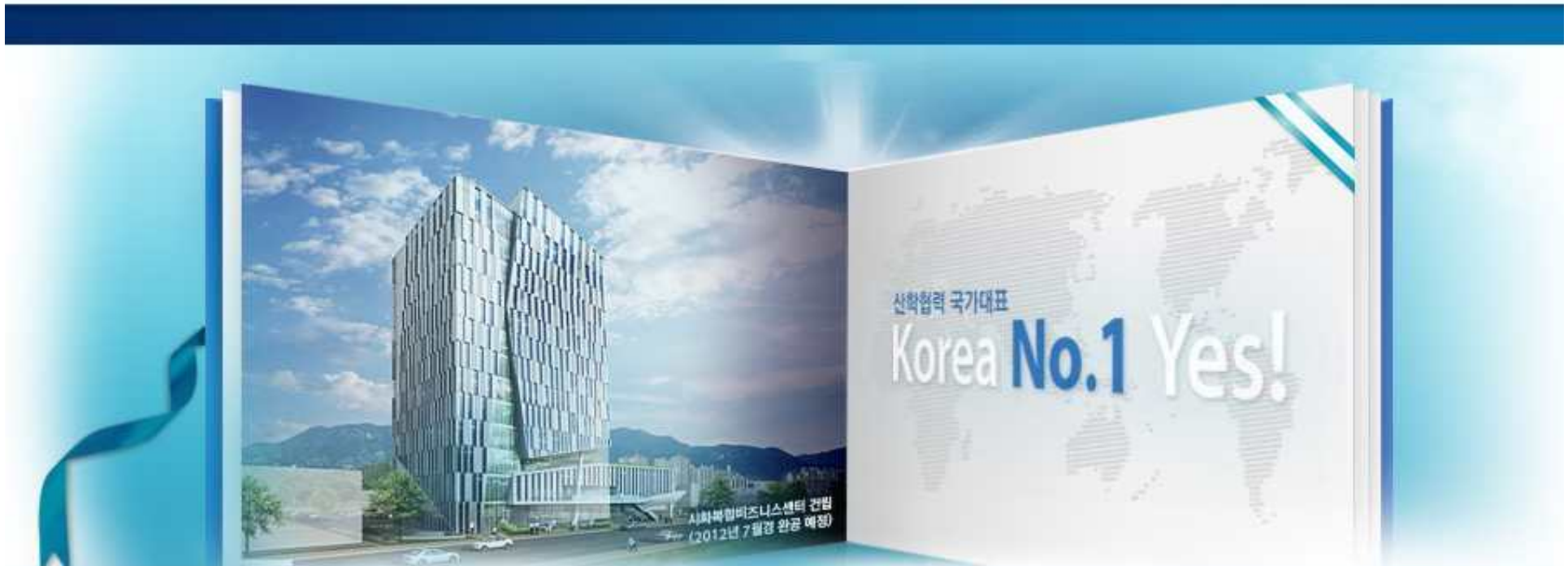


The image shows two overlapping Windows command prompt windows. The top window displays the output of a command, showing an IP address and a domain name. The bottom window displays the output of a command, showing an error message in Korean.

```
C:\Windows\system32\cmd.exe
IP 주소 = 106.10.165.51
도메인 이름 = w2.rc.vip.sg3.yahoo.com
계속하려면 아무 키나 누르십시오 . . .
```

```
C:\Windows\system32\cmd.exe
[gethostbyname()] 알려진 호스트가 없습니다.
계속하려면 아무 키나 누르십시오 . . .
```





# Thank You !

[oasis01@gmail.com](mailto:oasis01@gmail.com) / [rhqudtn75@nate.com](mailto:rhqudtn75@nate.com)