산업기술의 최고 명문대학
한국산업기술대학교

3 소켓 구소 구소체 다루기

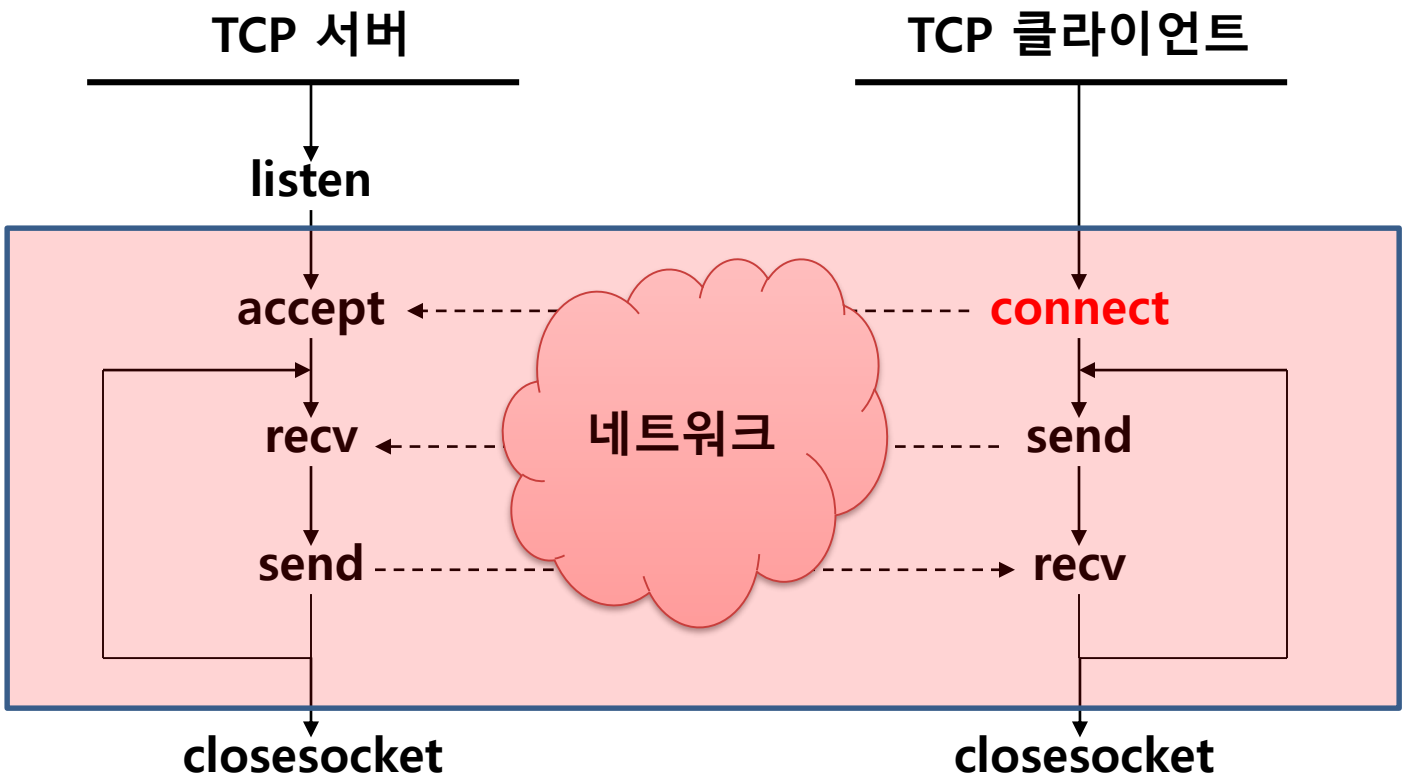
네트워크 게임 프로그래밍

Contents

- 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

- 네트워크 통신을 위해 필요한 것?
  - IP? port? Domain?



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{} ; 적외선 무선통신용
- 기본형은 SOCKADDR 구조체임
- 실제 프로그래밍에서는 응용 프로그램이 사용할 프로토콜의 종류에 맞는 별도의 소켓 주소 구조체를 사용

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_IN;
```

- 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 구조체는 다양한 형태로 초기화(선언)할 수 있고 소켓 함수에 인자로 전달할 수 있지만 교재에서는 해당 형태로만 사용 예정

❖ 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(SocketFunc()SMS 임의의 소켓함수임)

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

// 소켓 주소 구조체를 사용한다.
...
```

네트워크 바이트 정렬

❖ 네트워크 바이트 정렬

- 컴퓨터와 네트워크 마다 사용하는 언어 형식이 나름. 즉, 바이트 순서가 다름
- 컴퓨터 입장에서는 모든 비트(혹은 바이트)의 나열이라고 볼수 있음
- cpu에 따라서 레지스터에서 읽어오는 방식이 다름 (호스트 바이트 오더링이 다름)
- 네트워크 바이트 정렬은 빅엔디안으로 표준화(통일)함
- 다른 컴퓨터간 인지하는 바이트 정렬이 다를경우 받는 메시지의 내용 해석에 오류가 발생
- 소켓은 바이트 오더 함수를 제공
  - 윈도우즈는 확장 바이트 오더 함수를 제공
  - WSAHtons / WSAHtonl / WSANtohs / WSANtohl

네트워크 바이트 정렬

❖ 해결방안

- 모든 수신 메시지에 대한 바이트 오더링을 고려해야 함 (호스트 바이트 정렬(오더링))
- 단, 네트워크는 무조건 빅 엔디안으로 통일되어 있음 (네트워크 바이트 정렬(오더링))
  - 따라서, 수신단에서 필요한 호스트 바이트 오더링으로 변경하여 사용(소켓 바이트 오더 함수 사용)
  - 네트워크 빅 엔디안으로 통일하고,
  - 이 외의 경우에는 소프트웨어적으로 바이트 오더를 조정함
- 송수신 메시지의 경우에는 xml등 근래 다양한 메시지 포맷을 사용하면 됨

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 0x10000번지에 저장할 때 두 바이트 정렬 방식 비교

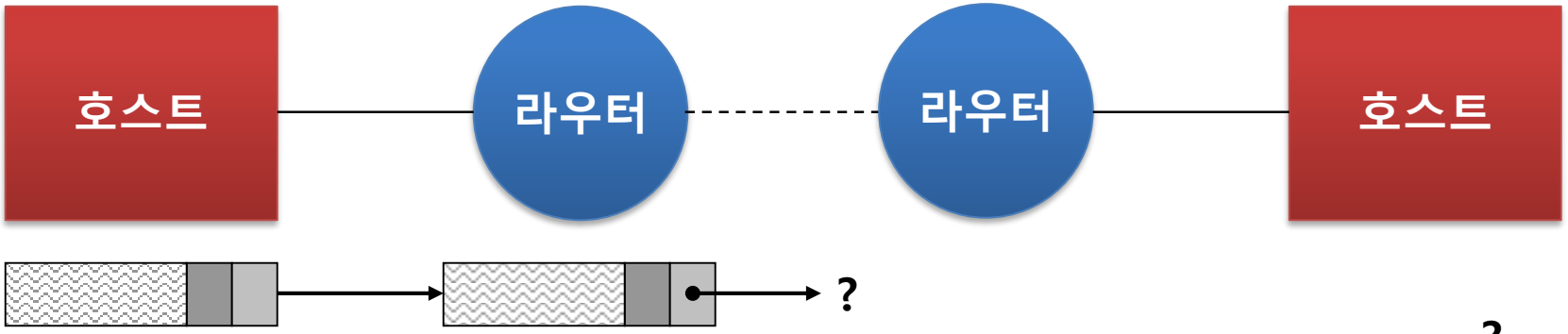
	0x1000	0x1001	0x1002	0x1003	
빅 엔디안	・・・	0x12	0x34	0x56	0x78	・・・
리틀 엔디안	・・・	0x78	0x56	0x34	0x12	・・・

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

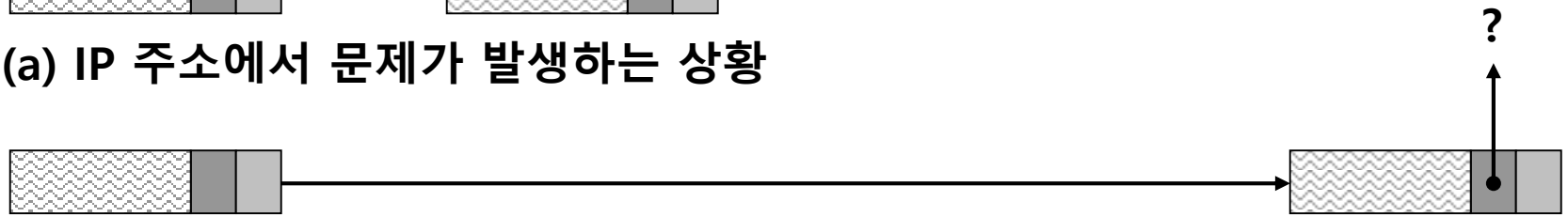
빅 엔디안과 리틀 엔디안..
무엇이 좋을까요?

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)

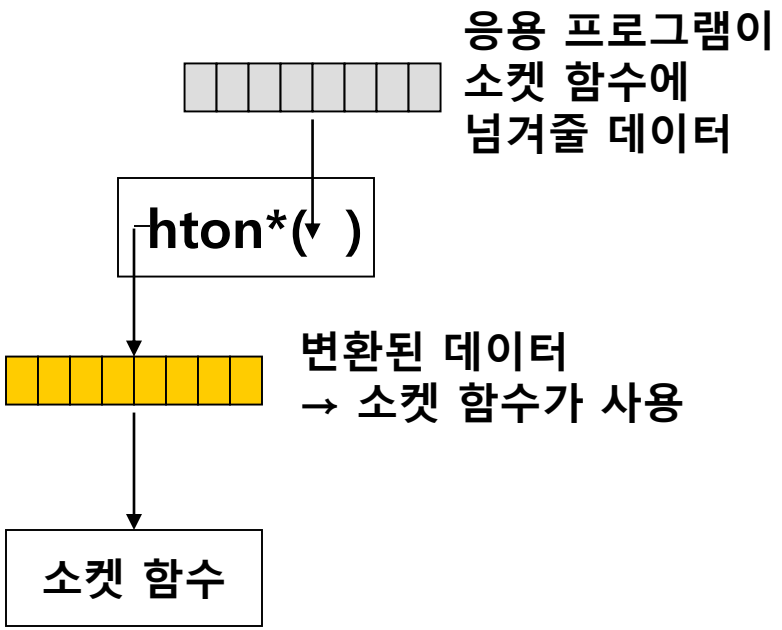
- 소켓 디스크립터
- 변환된 결과를 리턴값이 아닌 세번째 인자로 전달

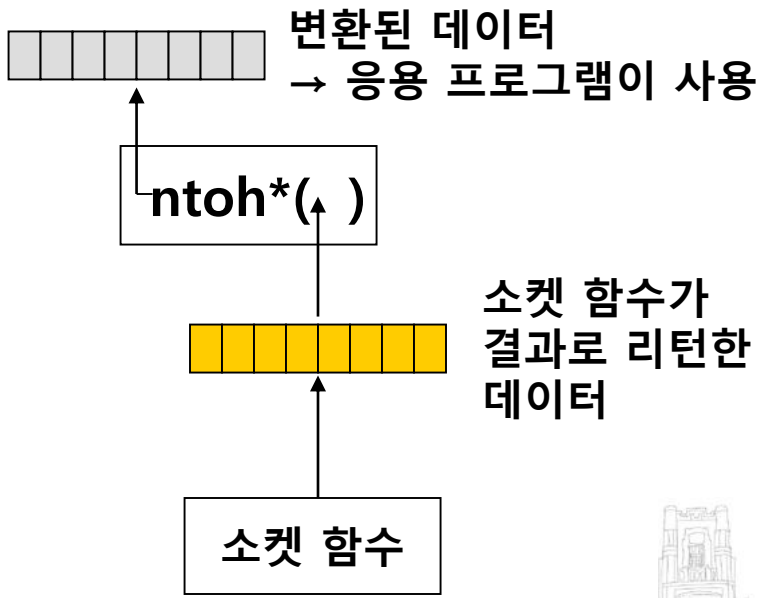
```
int WSAHtons(SOCKET s, u_short hostshort, u_short *lpnetshort);
int WSAHtonl(SOCKET s, u_long hostlong, u_long *lpnetlong);
int WSANtohs(SOCKET s, u_short netshort, u_short *lphostshort);
int WSANtohl(SOCKET s, u_long netlong, u_long *lphostlong);
```

바이트 정렬 함수 (5)

❖ 바이트 정렬 함수 사용 상황

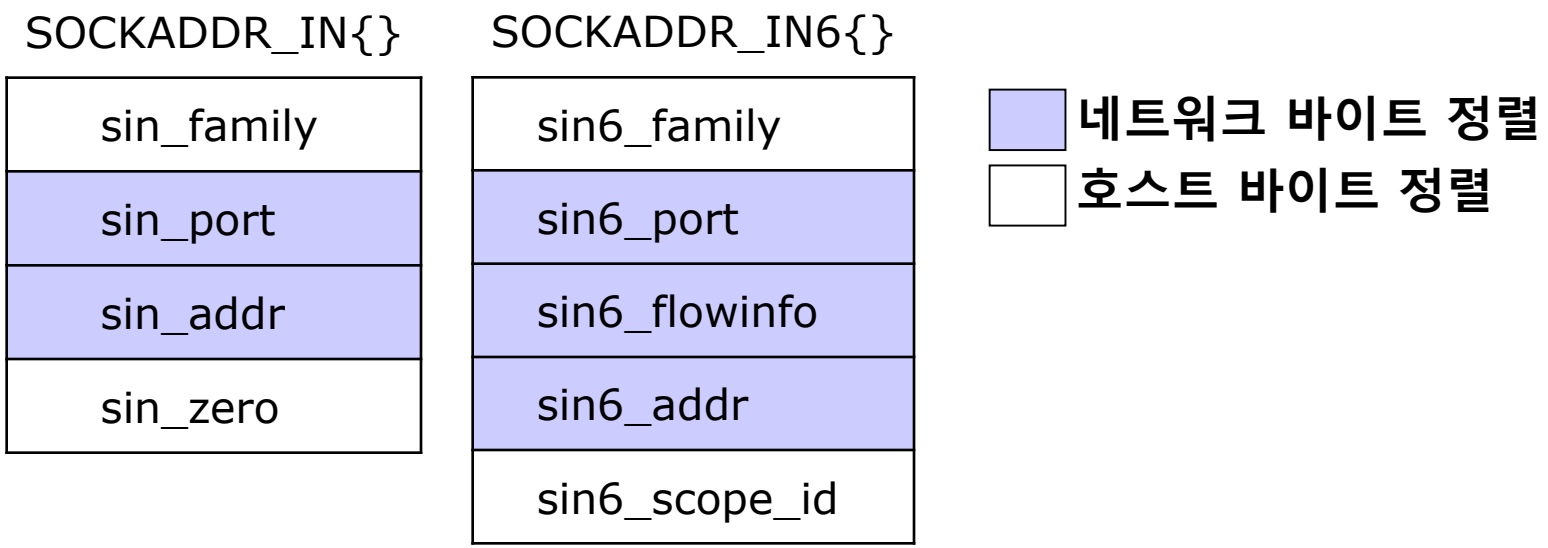
- hton*() 함수는 응용프로그램이 소켓 함수에 데이터를 넘겨주기 전에 호출되며, ntoh*() 함수는 소켓 함수가 결과로 리턴한 데이터를 응용 프로그램이 출력등의 목적으로 이용하기 전에 호출





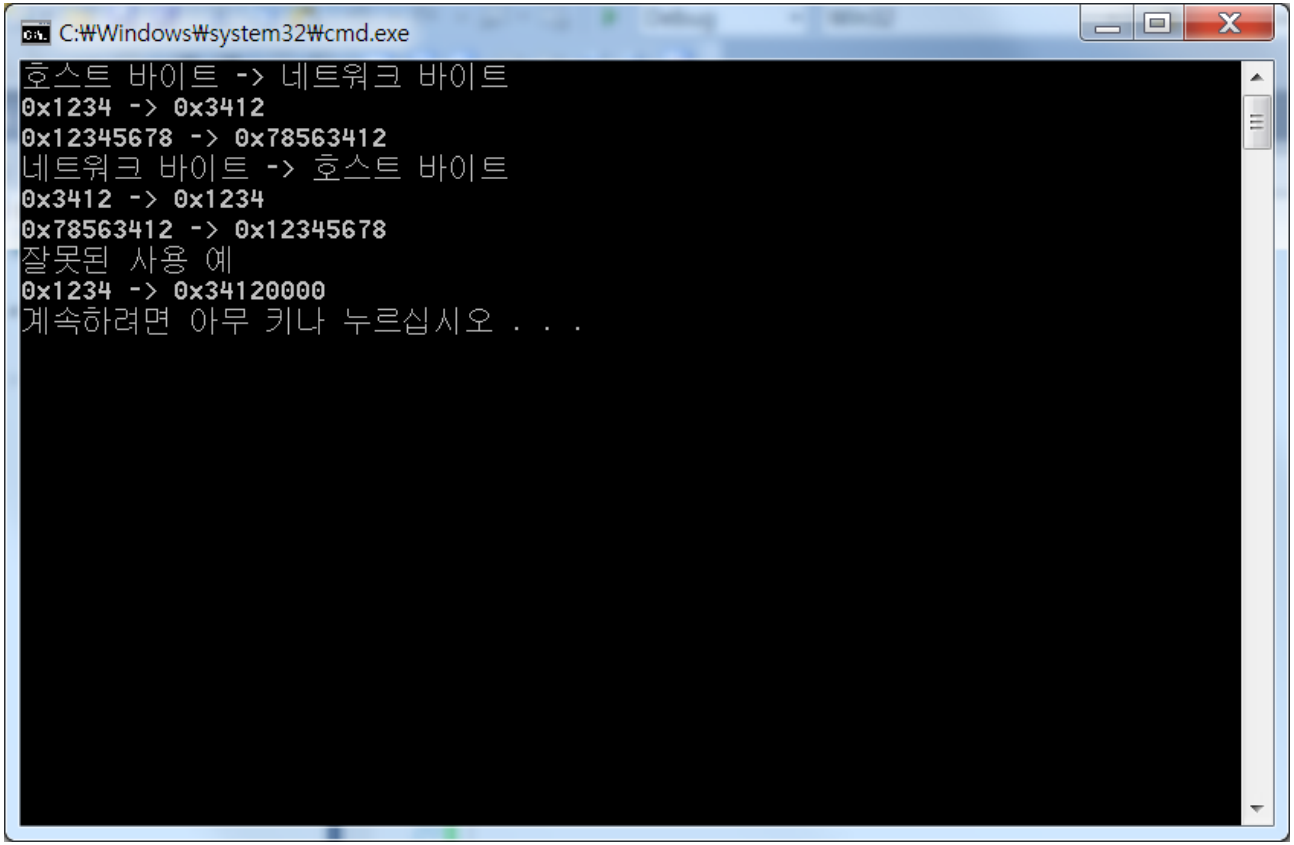
바이트 정렬 함수 (6)

❖ **SOCKADDR_IN/SOCKADDR_IN6 구조체의 바이트 정렬 방식**



예제 실습

실습 3-1
바이트 정렬 함수 연습
p. 71



예제 실습

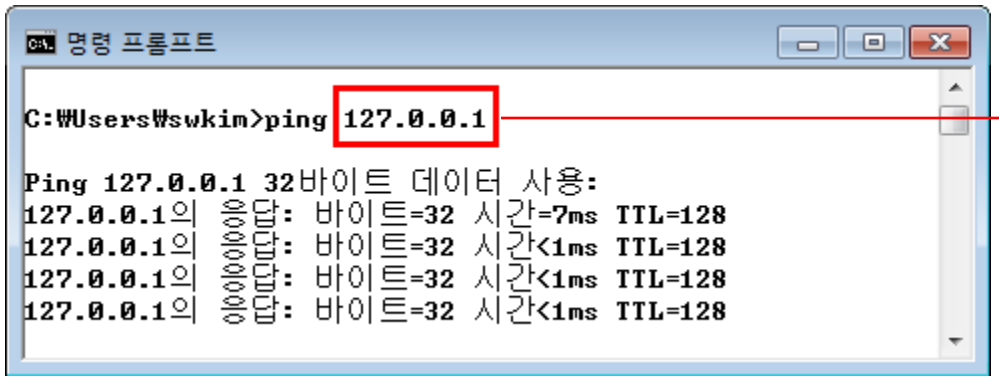
실습

WSAHton~~() 와 WSANtoh~~() 함수로 바꾸시오.

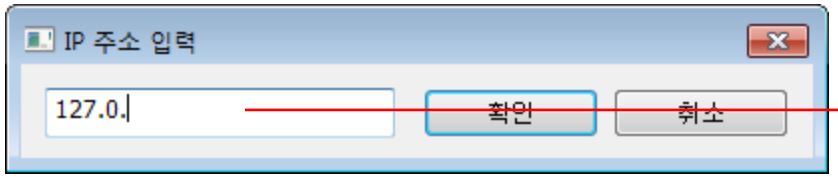
```
SOCKET s = socket(AF_INET, SOCK_STREAM, 0);
 if(s == SOCKET_ERROR)
            return 1;
 u_short x1 = 0x1234;
 u_long  y1 = 0x12345678;
 u_short x2, x3;
 u_long  y2, y3;

// 호스트 바이트 -> 네트워크 바이트
 printf("[호스트 바이트 -> 네트워크 바이트]\n");
 WSAHtons(s, x1, &x2);
 printf("0x%x -> 0x%x\n", x1, x2);
 WSAHtonl(s, y1, &y2);
 printf("0x%x -> 0x%x\n", y1, y2);

// 네트워크 바이트 -> 호스트 바이트
 printf("\n[네트워크 바이트 -> 호스트 바이트]\n");
 WSANtohs(s, x2, &x3);
 printf("0x%x -> 0x%x\n", x2, x3);
 WSANtohl(s, y2, &y3);
 printf("0x%x -> 0x%x\n", y2, y3);
```

IP 주소 변환 함수 (1)

❖ IP 주소 변환 예

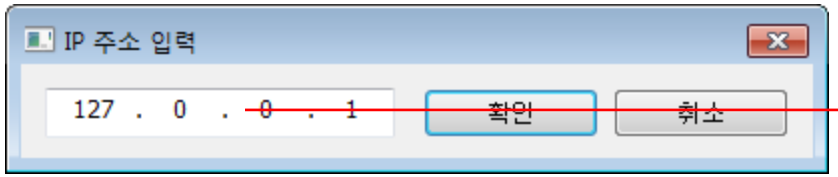


명령행 인자



편집 컨트롤

⇒ 문자열 형태로 전달받으므로, 숫자로 변환해야 함



IP 주소 컨트롤

=> IPv6에서 사용 불가능

IP 주소 변환

❖ IP 주소 변환

- 컴퓨터는 이진 데이터만 해석 가능
- 인터넷 주소(문자열) -> 이진 데이터 (네트워크 바이트) 로 변경
- 혹은 이진 데이터 (네트워크 바이트) -> 인터넷 주소(문자열)로 변경할 필요가 있음
- 이 모두 소켓 함수로 지원

IP 주소 변환 함수 (2)

❖ IPv4 주소 변환 함수

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

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

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

- 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)

```
int WSAStringToAddress (
    // 문자열 형식의 IP 주소
    LPTSTR AddressString,
    // AF_INET 또는 AF_INET6
    INT AddressFamily,
    // NULL
    LPWSAPROTOCOL_INFO lpProtocolInfo,
    // IP 주소(숫자)를 저장할 구조체; SOCKADDR_IN 또는 SOCKADDR_IN6
    LPSOCKADDR lpAddress,
    // 주소 구조체의 길이
    LPINT lpAddressLength
);
```

IP 주소 변환 함수 (4)

❖ IPv4 또는 IPv6 주소 변환 함수(숫자 ⇒ 문자열)

```
int WSAAddressToString (
  // 숫자 형식의 IP 주소; SOCKADDR_IN 또는 SOCKADDR_IN6
  LPSOCKADDR lpsaAddress,
  // 주소 구조체의 길이
  DWORD dwAddressLength,
  // NULL
  LPWSAPROTOCOL_INFO lpProtocolInfo,
  // IP 주소(문자열)를 저장할 버퍼
  LPTSTR lpszAddressString,
  // 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

```
// 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)

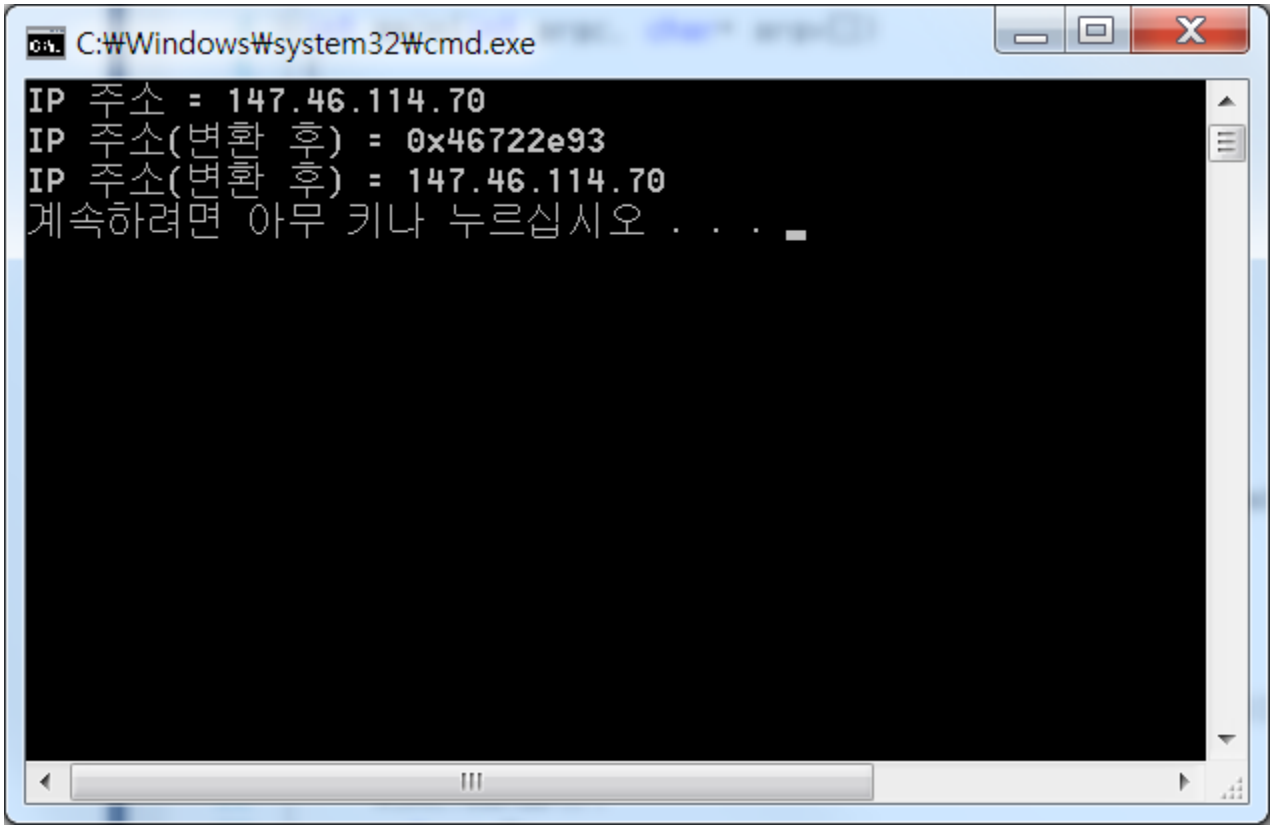
❖ 바이트 정렬 함수와 IP 주소 변환 함수 사용 예 ②

- 소켓 함수가 소켓 주소 구조체를 입력으로 받아 내용을 채우면, 응용 프로그램이 이를 출력 등의 목적으로 사용하는 경우

```
// 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



도메인 주소 변환

❖ 도메인 주소 변환

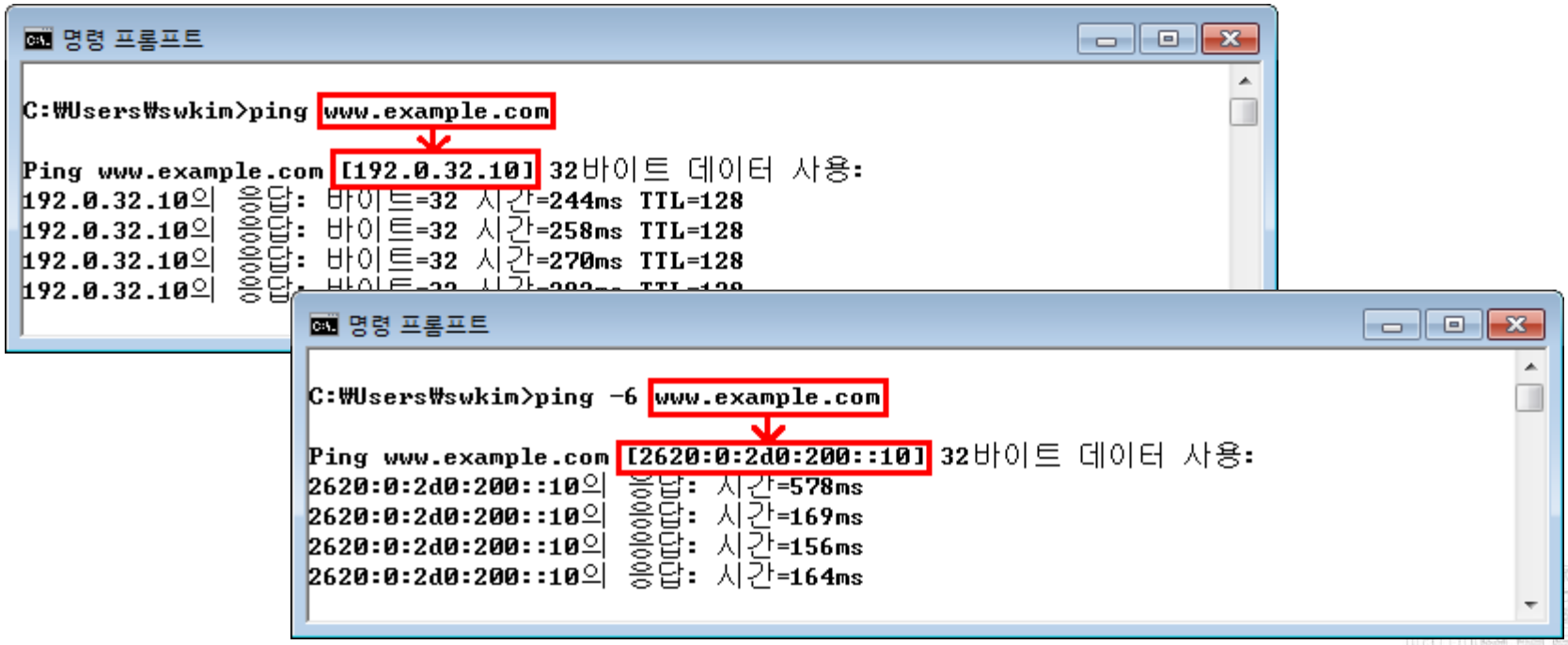
- 도메인 주소는 사람이 가장 쉽게 이해할 수 있는 인터넷 주소
- 인터넷에서는 이진 인터넷 주소만 사용
  - 도메인을 사용하기 위해서는 변환 시스템이 필요함 (DNS)
- gethostbyname() 함수 지원
  - 호스트의 이름, 별칭, 호스트 주소 타입, 주소 길이, 주소 리스트(하나의 도메인에서 여러 ip를 사용)
- 도메인 주소 변환 함수
  - 인터넷 주소 관리 체계
  - 인터넷 주소 변환 시스템
  - 도메인 주소 변환 시스템

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



도메인 이름 시스템과 이름 변환 함수 (2)

❖ 도메인 이름 ⇔ IP 주소 변환 함수

- 두 함수 모두 hostnet 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)
);
```

도메인 이름 시스템과 이름 변환 함수 (3)

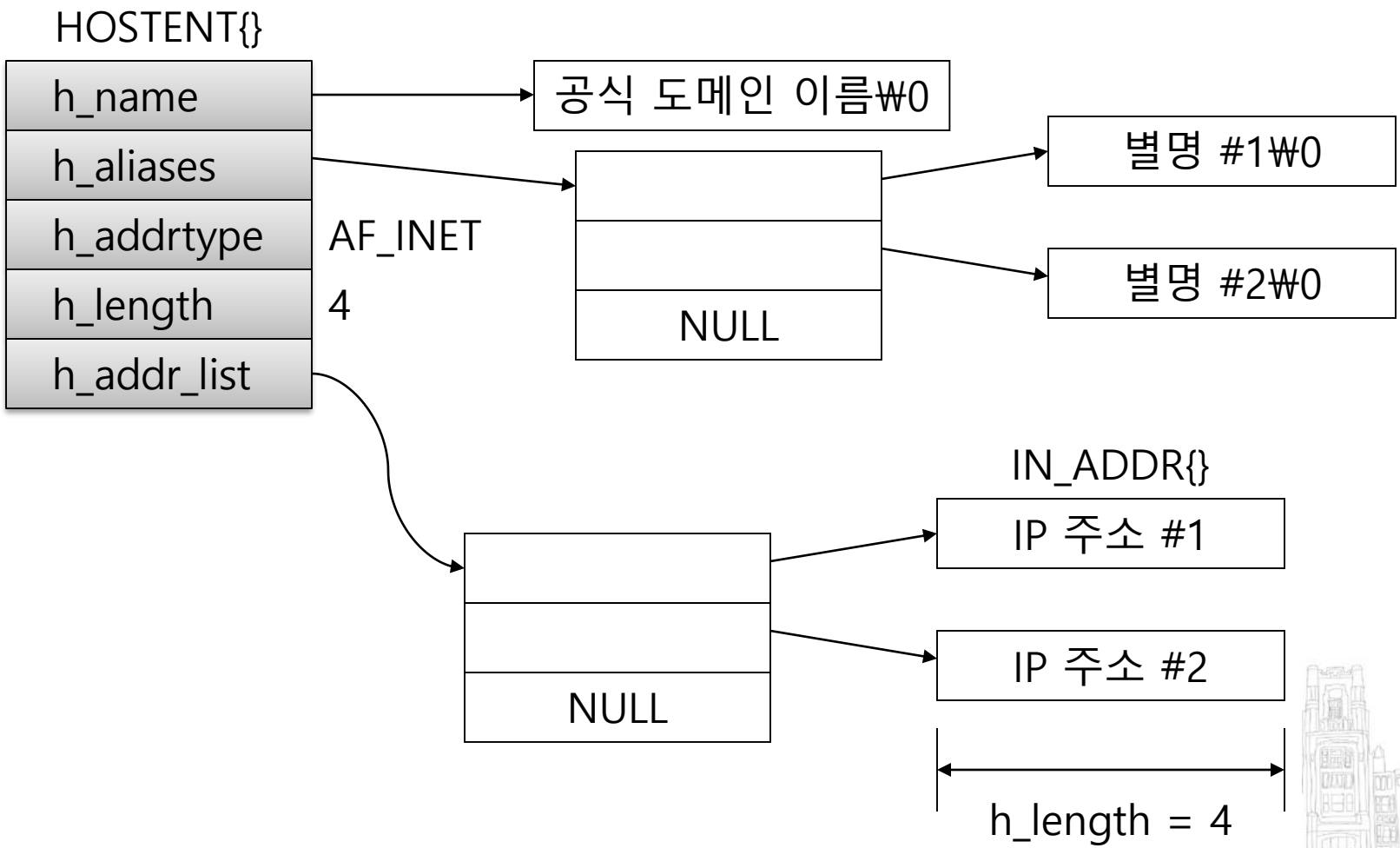
❖ hostent 구조체

```
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])

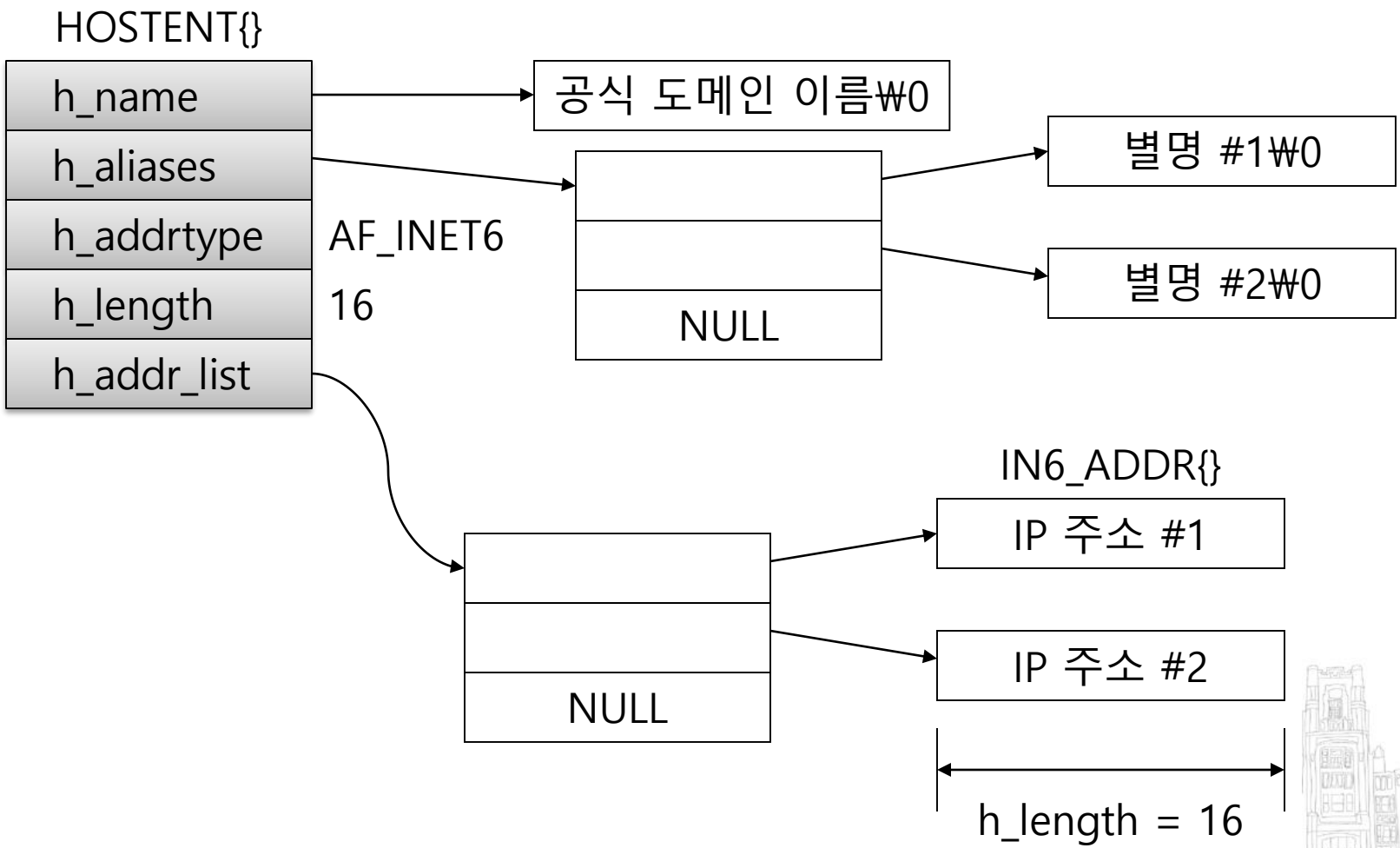
도메인 이름 시스템과 이름 변환 함수 (4)

❖ **hostent 구조체 - IPv4를 사용하는 경우**



도메인 이름 시스템과 이름 변환 함수 (5)

❖ **hostent 구조체 - IPv6를 사용하는 경우**



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;
}
```

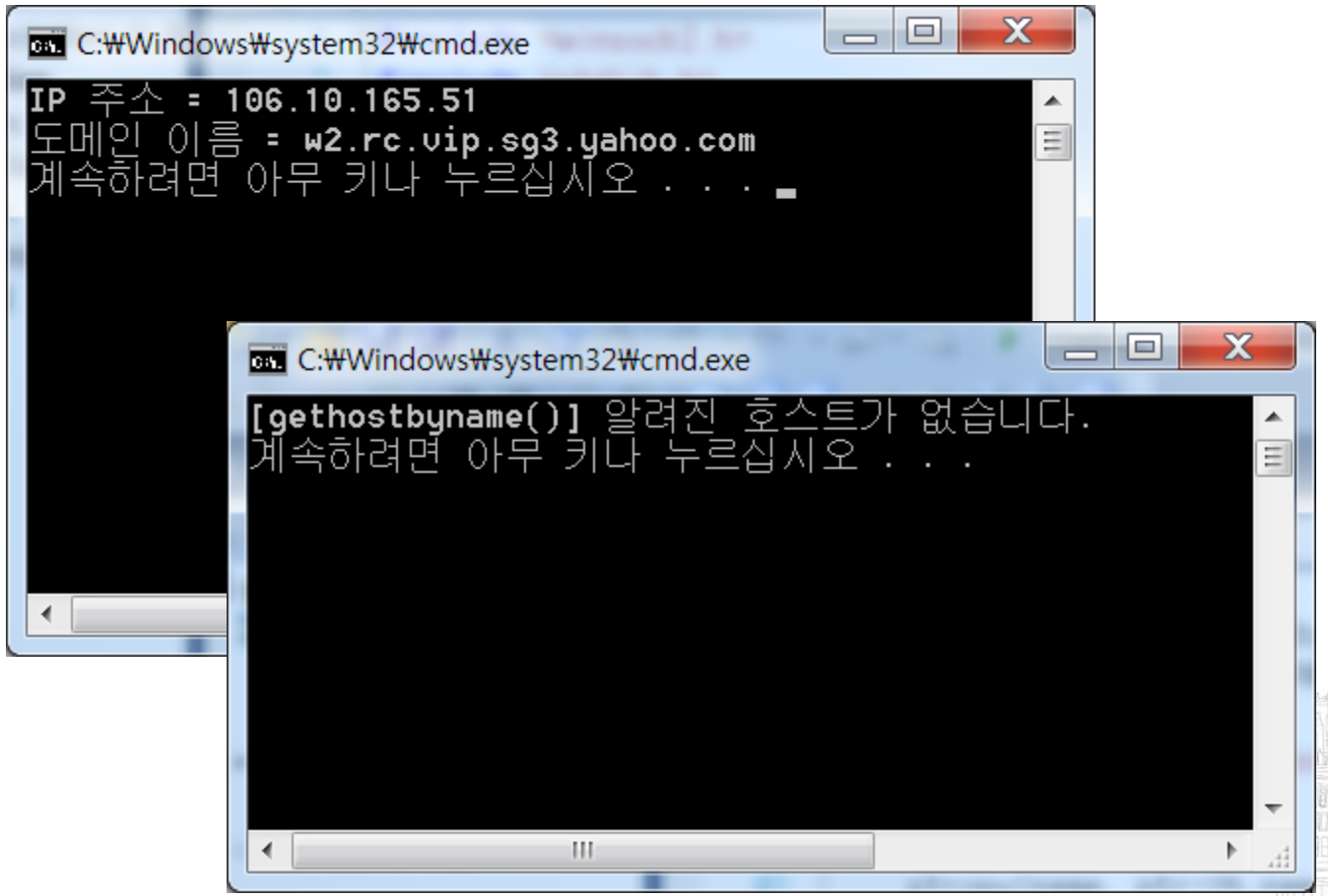
도메인 이름 시스템과 이름 변환 함수 (7)

❖ 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;
}
```

예제 실습

실습 3-3
이름 변환 함수 활용



Thank You !

oasis01@gmail.com / rhqudtn75@nate.com