

Chapter 03 소켓 주소 구조체 다루기

학습목표

- 소켓/주소 구조체의 정의와 초기화 방법을 익힌다.
- 바이트 정렬 함수의 필요성과 사용법을 익힌다.
- IP 주소 변환 함수를 익힌다. ~ 10진수 ↔ 2진수로 바뀌는 방식
- 도메인 이름 시스템의 원리와 이름 변환 함수를 익힌다.
↑ IP주소

목차

- 01 소켓 주소 구조체
- 02 바이트 정렬 함수
- 03 IP 주소 변환 함수
- 04 DNS와 이름 변환 함수

01 소켓 주소 구조체



소켓 주소 구조체 (1)

■ 소켓 주소 구조체

- 네트워크 프로그램에서 필요한 주소 정보를 담는 구조체
- 다양한 소켓 함수의 인수로 사용
- 프로토콜 체계에 따라 다양한 소켓 주소 구조체가 존재
- 기본형은 sockaddr 구조체

→ TCP에서는 IP 주소와 포트번호

예) IPv4, IPv6
32bit 128bit

```
#include <winsock2.h>
#include <sys/socket.h>

struct sockaddr {
    ① unsigned short    sa_family;
    ② char              sa_data[14];
};
```

윈도우

리눅스

login을 예시로 세션은 ID와 password, 주소 정보와 포트번호는 미리 약속을 클라이언트에게 있는 것을 담와 보낸다.

소켓 주소 구조체 (2)

■ sockaddr 구조체

- 응용 프로그램이 사용할 프로토콜의 종류에 따라 별도의 소켓 주소 구조체가 정의되어 있음
 - TCP/IP ⇒ sockaddr_in{} 또는 sockaddr_in6{} *크기가 다르다*
 - TCP/IP 프로토콜을 위한 ^{두 번째 네트워킹 라이브러리} 소켓 주소 구조체는 IP 버전에 따라 두 종류가 제공
 - 블루투스 ⇒ sockaddr_bth{}

소켓 주소 구조체 (3)

■ sockaddr_in 구조체 - IPv4 전용

```
/* IPv4 소켓 주소 구조체 */
```

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

윈도우

```
#include <netinet/in.h>
```

리눅스

```
struct sockaddr_in {
```

① short sin_family;

→ 주소 체계 (AF_INET)

② unsigned short sin_port;

→ 포트번호

③ struct in_addr sin_addr;

→ IP주소

char sin_zero[8]; // 항상 0으로 설정

```
};
```

→ 확장을 위해 남겨둔 멤버, 사용하지 않음

소켓 주소 구조체 (4)

■ sockaddr_in6 구조체 - IPv6 전용

```
/* IPv6 소켓 주소 구조체 */  
#include <ws2tcpip.h>   윈도우  
#include <netinet/in.h> 리눅스  
struct sockaddr_in6 {  
    ① short          sin6_family;  
    ② u_short        sin6_port;  
    u_long           sin6_flowinfo; // 대부분 0으로 설정  
    ③ struct in6_addr sin6_addr;  
    u_long           sin6_scope_id; // 대부분 0으로 설정  
};
```


소켓 주소 구조체 (5)

■ in_addr 구조체와 in6_addr 구조체

- IPv4 주소를 담는 in_addr 구조체는 32비트 주소를 담는 s_addr 필드를 가짐
- IPv6 주소를 담는 in6_addr 구조체는 운영체제에 따라 공용체/배열로 구성

소켓 주소 구조체 (6)

■ 소켓 주소 구조체 크기 비교

소켓 주소 구조체	전체 크기(바이트 단위)
sockaddr	16
sockaddr_in	16
sockaddr_in6	28
sockaddr_bth	30

} 사이즈가 크다 → 인자로 &를 넘겨준다.

소켓 주소 구조체 (7)

■ 소켓 주소 구조체 사용 예

■ 예1)

```
struct sockaddr_in addr;  
... // 소켓 주소 구조체를 초기화한다.  
SocketFunc(..., (struct sockaddr *)&addr, sizeof(addr), ...);
```

↓
소켓 주소 구조체의 시작 주소로 넘겨줌 (+타입캐스팅)

■ 예2)

```
struct sockaddr_in addr;  
int addrlen = sizeof(addr);  
SocketFunc(..., (struct sockaddr *)&addr, &addrlen, ...);  
... // 소켓 주소 구조체를 사용한다.
```

02 바이트 정렬 함수



바이트 정렬 함수 (1)

■ 바이트 정렬

- 메모리에 데이터를 저장할 때 바이트의 배치 순서

- 빅 엔디언(Big-endian), 리틀 엔디언(Little-endian)

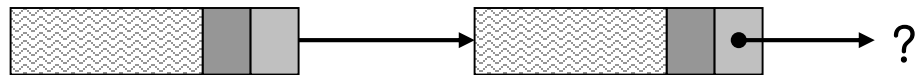
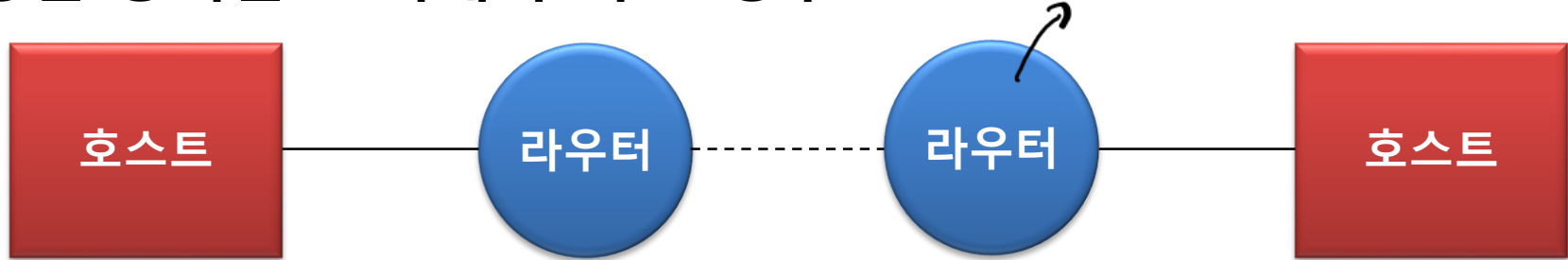
→ 앞에서부터 값을 채워 넣는다.

→ 뒤에서부터 값을 채워 넣는다

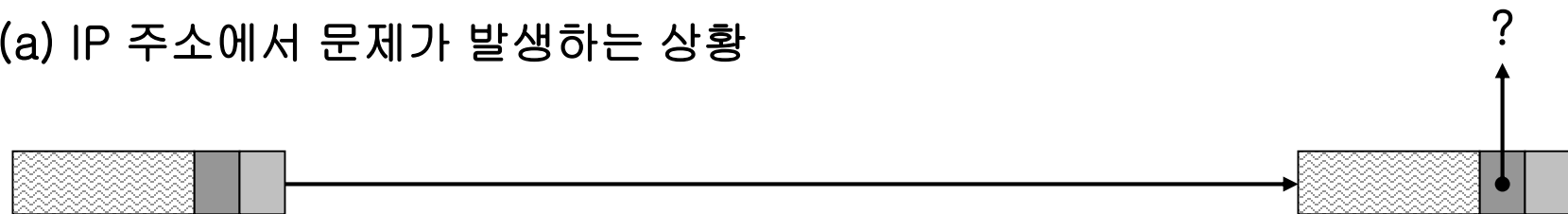
		0x1000	0x1001	0x1002	0x1003	
		⋮	⋮	⋮	⋮	
빅 엔디언	...	0x12	0x34	0x56	0x78	...
리틀 엔디언	...	0x78	0x56	0x34	0x12	...

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

- (a) IP 주소 \Rightarrow 빅 엔디언
- (b) 포트 번호 \Rightarrow 빅 엔디언

) \rightarrow 빅 엔디언 방식으로 통일

■ 응용 프로그램이 주고받는 데이터

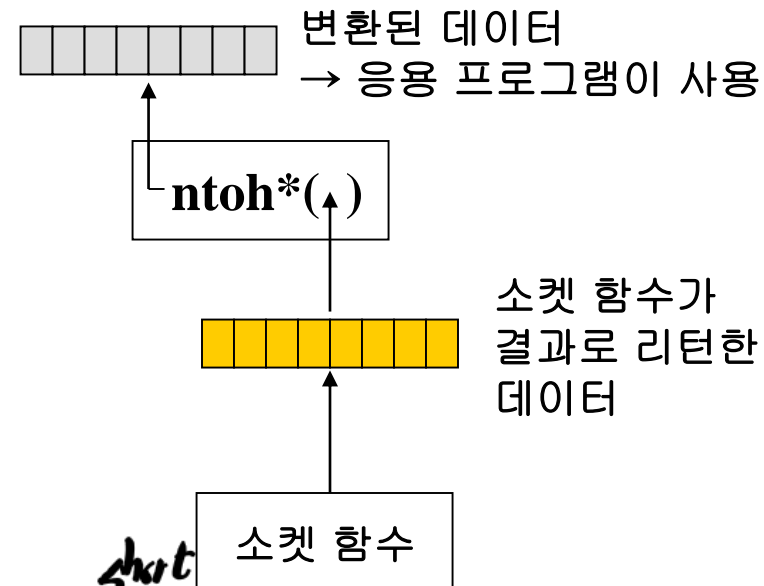
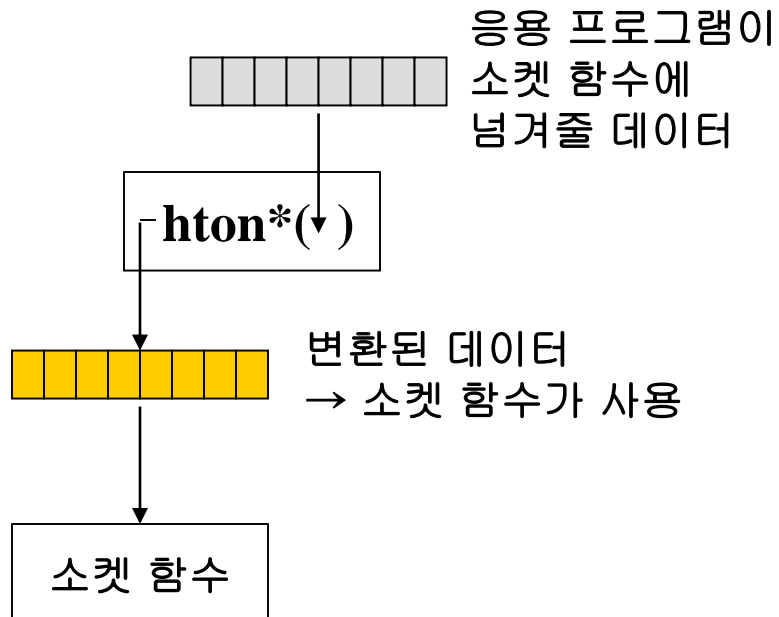
- (c) 빅 엔디언 또는 리틀 엔디언으로 통일 \rightarrow 개발자가 알아서 통일

의 엔디언을 호스트 바이트 라고 할

바이트 정렬 함수 (4)

■ 바이트 정렬 함수 사용 상황

- hton*() 함수는 응용 프로그램이 소켓 함수에 데이터를 넘겨주기 전에 호출 *→ 보낼 때*
- ntoh*() 함수는 소켓 함수가 결과로 리턴한 데이터를 응용 프로그램이 출력 등의 목적으로 사용하기 전에 호출 *→ 받을 때*



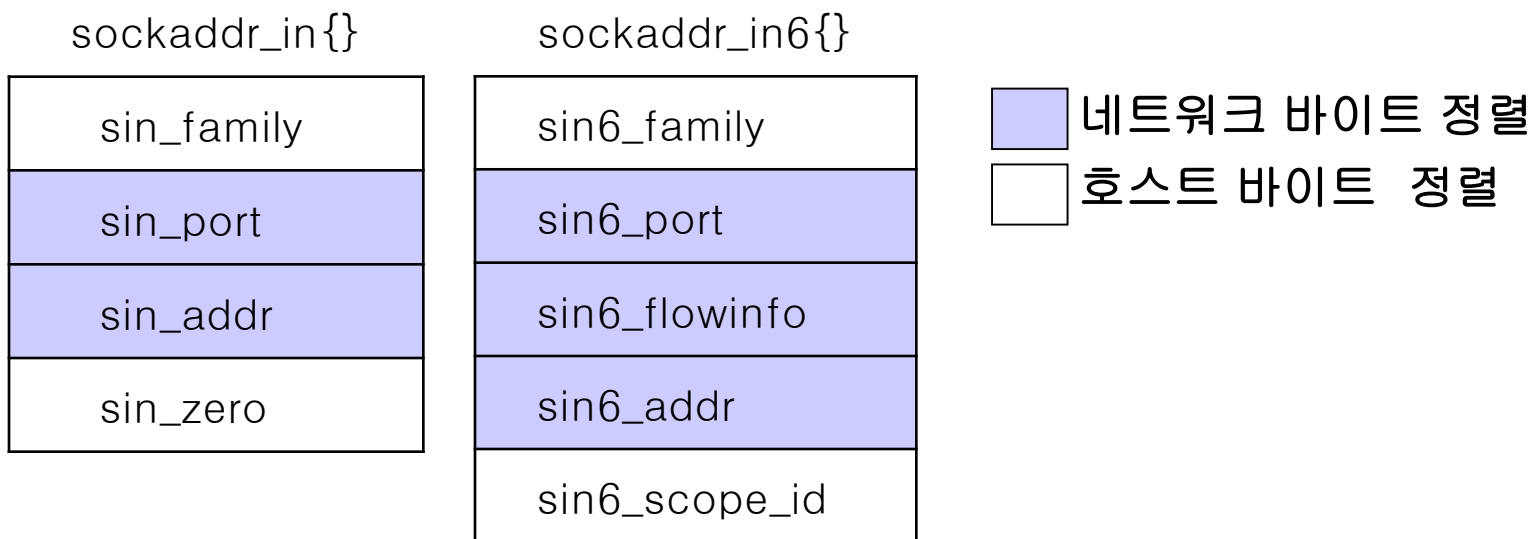
short
↑
htons → 2 byte 크기의 데이터를 변환

바이트 정렬 함수 (5)

htonl → 4byte 지
↳ long

//

■ sockaddr_in/sockaddr_in6 구조체의 바이트 정렬 방식



↳ CPU의 생김새를 봐서 회사

intel, ARM - little endian

IBM, Motorola - Big endian

AMD → Big endian 일지 little인지 바꿀지 꼭 세

바이트 정렬 함수 (6)

■ 실습 3-1 바이트 정렬 함수 연습

- ByteOrder.cpp
- [윈도우] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter03/ByteOrder/ByteOrder.cpp>
- [리눅스] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter03/ByteOrder.cpp>

03 IP 주소 변환 함수



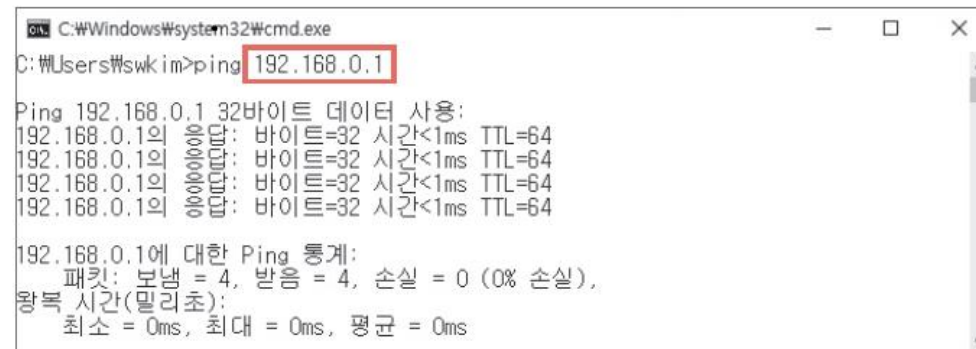
IP 주소 변환 함수 (1)

ping: host에게 4번 신호를 보내 시간을 측정

■ IP 주소 입력 ①

- 네트워크 프로그램에서 IP 주소를 입력받을 때는 명령행 인수를 사용하거나 운영체제가 제공하는 입력용 위젯을 이용
- 이때 IP 주소를 32비트(IPv4) 또는 128비트(IPv6) 숫자로 변환해야 함

10진수 → 2진수로 바꾸기 컴퓨터에게 알려줘야 함

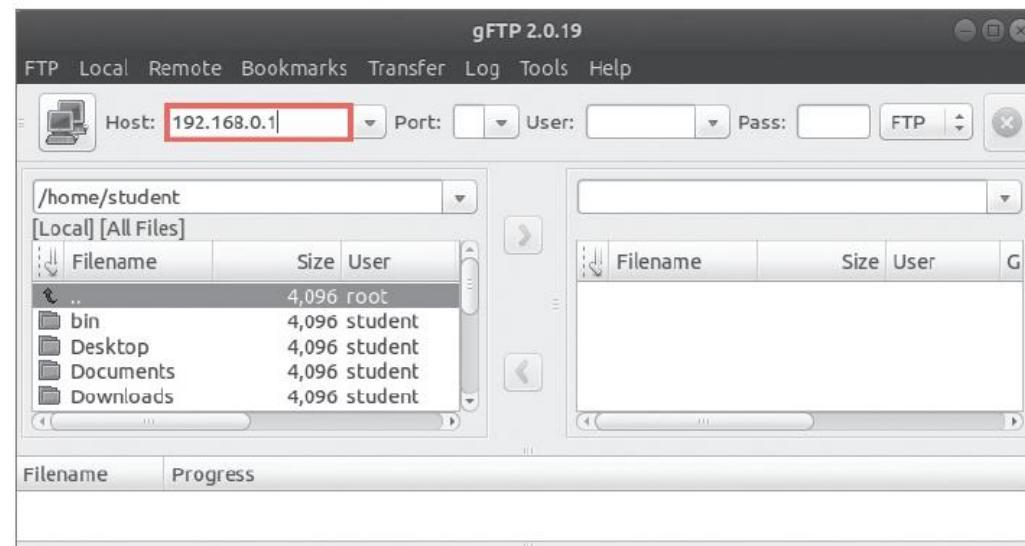


```
C:\Windows\system32\cmd.exe
C:\Users\swkim>ping 192.168.0.1

Ping 192.168.0.1 32바이트 데이터 사용:
192.168.0.1의 응답: 바이트=32 시간<1ms TTL=64
192.168.0.1의 응답: 바이트=32 시간<1ms TTL=64
192.168.0.1의 응답: 바이트=32 시간<1ms TTL=64
192.168.0.1의 응답: 바이트=32 시간<1ms TTL=64

192.168.0.1에 대한 Ping 통계:
    패킷: 보냄 = 4, 받음 = 4, 손실 = 0 (0% 손실),
    왕복 시간(밀리초):
        최소 = 0ms, 최대 = 0ms, 평균 = 0ms
```

(a) 명령행 인수



(b) 입력용 위젯

그림 3-6 IP 주소 입력: 명령행 인수와 입력용 위젯

IP 주소 변환 함수 (2)

■ IP 주소 입력 ②

- IP 주소 전용 위젯을 사용하면 IP 주소를 입력받아 문자열 또는 32비트 숫자(네트워크 바이트 정렬)를 얻을 수 있음
- IPv6에는 사용할 수 없고, 콘솔 응용 프로그램에서 사용하기에는 부적절

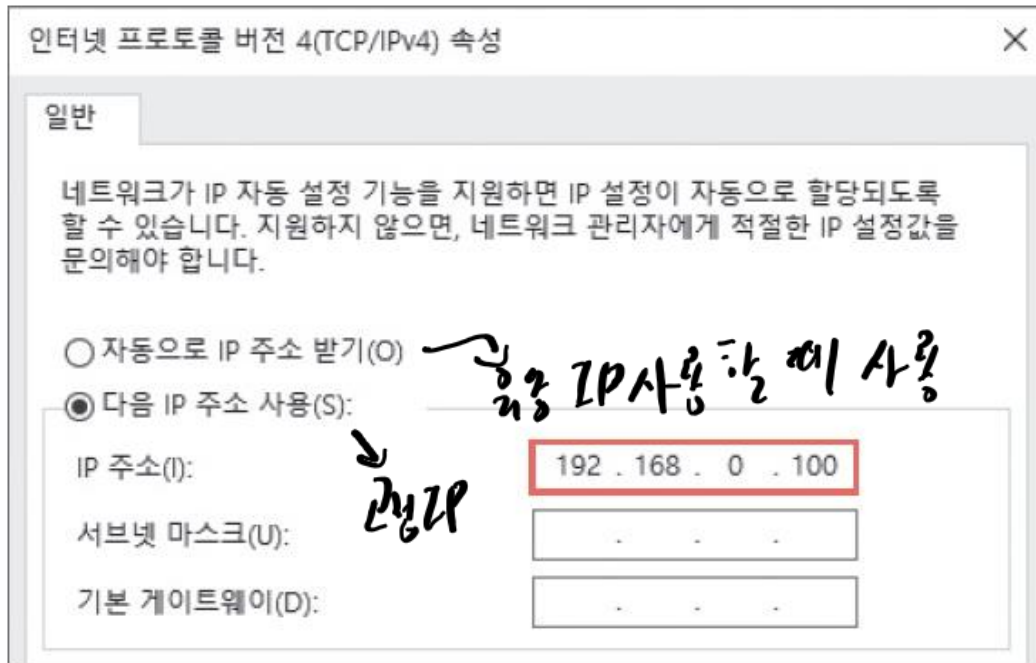


그림 3-7 IP 주소 전용 위젯

IP 주소 변환 함수 (3)

■ IP 주소 변환 함수

→ IPv4, IPv6 등 다양한 해커

→ 10진수 혹은 16진수
IP4 IP6

윈도우

```
#include <ws2tcpip.h>
int inet_pton(int af, const char *src, void *dst);
const char *inet_ntop(int af, const void *src, char *dst, size_t size);
```

리눅스

```
#include <arpa/inet.h>
int inet_pton(int af, const char *src, void *dst);
const char *inet_ntop(int af, const void *src, char *dst, socklen_t size);
```

IP 주소 변환 함수 (4)

■ 바이트 정렬 함수와 IP 주소 변환 함수 사용 예 ①

- 응용 프로그램이 소켓 주소 구조체를 초기화하고, 소켓 함수에 넘겨주는 경우

```
// 소켓 주소 구조체를 초기화한다.
struct sockaddr_in addr;           // IPv4용 소켓 주소 구조체
memset(&addr, 0, sizeof(addr));    // 0으로 채운다. 서버에 데이터를 전송
addr.sin_family = AF_INET;        // 주소 체계: IPv4
inet_pton(AF_INET, "147.46.114.70", &addr.sin_addr); // IP 주소: 문자열 → 숫자
addr.sin_port = htons(9000);      // 포트 번호: 호스트 바이트 정렬 → 네트워크 바이트 정렬

// 소켓 함수를 호출한다.
SocketFunc(..., (struct sockaddr *)&addr, sizeof(addr), ...); // 소켓 함수 호출
```

각 함수의 바이트 순서로 바꿔서 넘겨준다.

IP 주소 변환 함수 (5)

■ 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.
```

```
struct sockaddr_in addr; // IPv4용 소켓 주소 구조체
```

```
int addrlen = sizeof(addr); // 리눅스에서는 int 대신 socklen_t 타입을 사용해야 한다.
```

```
SocketFunc(..., (struct sockaddr *)&addr, &addrlen, ...); // 소켓 함수 호출 → 상대 소켓정보는 있는 거.
```

```
// 소켓 주소 구조체를 사용한다.
```

```
char ipaddr[INET_ADDRSTRLEN]; // 문자열 형태의 IPv4 주소를 담을 버퍼
```

```
inet_ntop(AF_INET, &clientaddr.sin_addr, ipaddr, sizeof(ipaddr)); // IP 주소: 숫자 → 문자열 → 받아서 포스트 바이트정렬을 해준다.
```

```
printf("\n[TCP 서버] 클라이언트 접속: IP 주소=%s, 포트 번호=%d\n",
```

```
ipaddr, ntohs(clientaddr.sin_port)); // 소켓 주소 구조체 사용
```


IP 주소 변환 함수 (6)

■ 실습 3-2 IP 주소 변환 함수 연습

- IPAddr.cpp
- [윈도우] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter03/IPAddr/IPAddr.cpp>
- [리눅스] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter03/IPAddr.cpp>

04 DNS와 이름 변환 함수



도메인 이름 시스템과 이름 변환 함수 (1)

■ 도메인 이름

- IP 주소처럼 호스트나 라우터의 고유한 식별자이며, IP 주소보다 기억하고 사용하기 쉬움
- 도메인 이름을 반드시 숫자 형태의 IP 주소로 변환해야 함

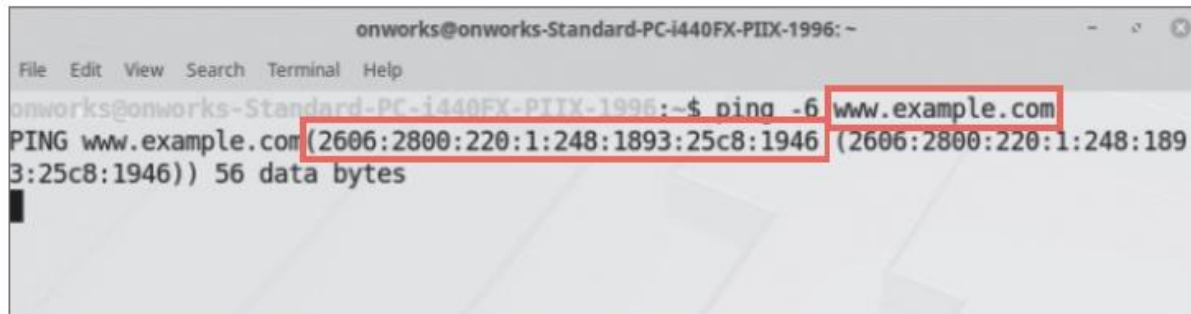


```
C:\Windows\system32\cmd.exe
C:\Users\swkim>ping www.example.com

Ping www.example.com: [93.184.216.34] 32바이트 데이터 사용:
93.184.216.34의 응답: 바이트=32 시간=205ms TTL=53
93.184.216.34의 응답: 바이트=32 시간=205ms TTL=53
93.184.216.34의 응답: 바이트=32 시간=204ms TTL=53
93.184.216.34의 응답: 바이트=32 시간=206ms TTL=53
```

주어진 네임 서버는 다른 도메인 네임 서버와
연결되어 있다.

(a) 윈도우 도메인 이름 → IPv4 주소



```
onworks@onworks-Standard-PC-i440FX-PIIX-1996: ~
File Edit View Search Terminal Help
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ ping -6 www.example.com
PING www.example.com (2606:2800:220:1:248:1893:25c8:1946) (2606:2800:220:1:248:1893:25c8:1946)) 56 data bytes
```

(b) 리눅스 도메인 이름 → IPv6 주소

그림 3-8 도메인 이름의 IP 주소 변환

도메인 이름 시스템과 이름 변환 함수 (2)

■ 도메인 이름 ⇔ IP 주소 변환 함수

```
#include <winsock2.h> 윈도우
#include <netdb.h> 리눅스

/* 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 네임 → IP 주소 } → 신뢰가능

IP 주소 → 도메인 네임

} → 가끔 잘못된 결과를 가져올 수 있다. (IP 주소는 순서로 바뀌기 때문)

↳ 볼록 함수도 같은 함수를 쓰기 때문에 주소 체계와 IP 주소의 길이도 같이 넣는다.

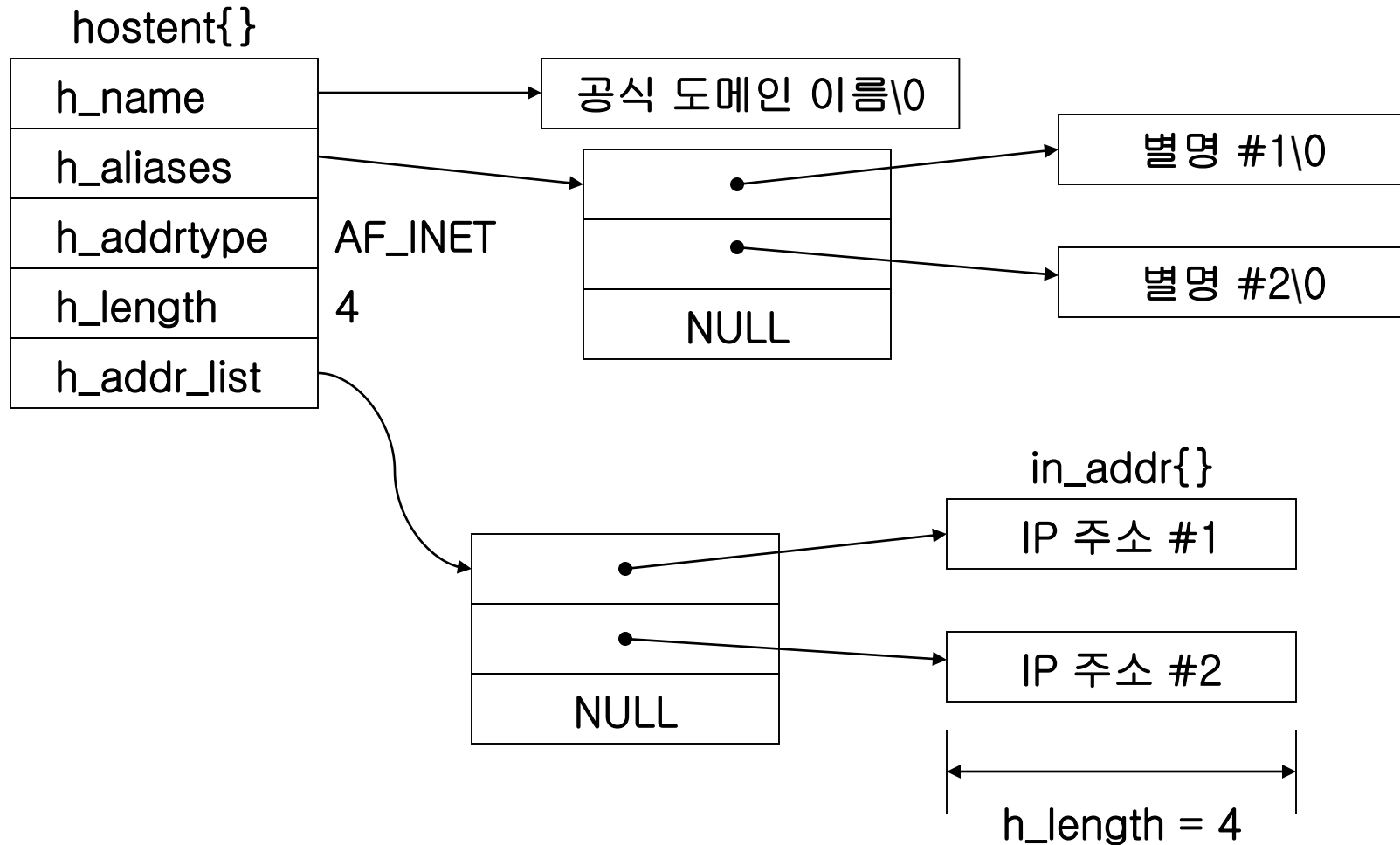
도메인 이름 시스템과 이름 변환 함수 (3)

■ hostent 구조체

```
struct hostent {  
    ① char *h_name; ②도메인 네임  
    ③ char **h_aliases; ④별칭  
    ⑤ short h_addrtype;  
    ④ short h_length;  
    ⑤ char **h_addr_list;  
#define h_addr h_addr_list[0]  
};
```

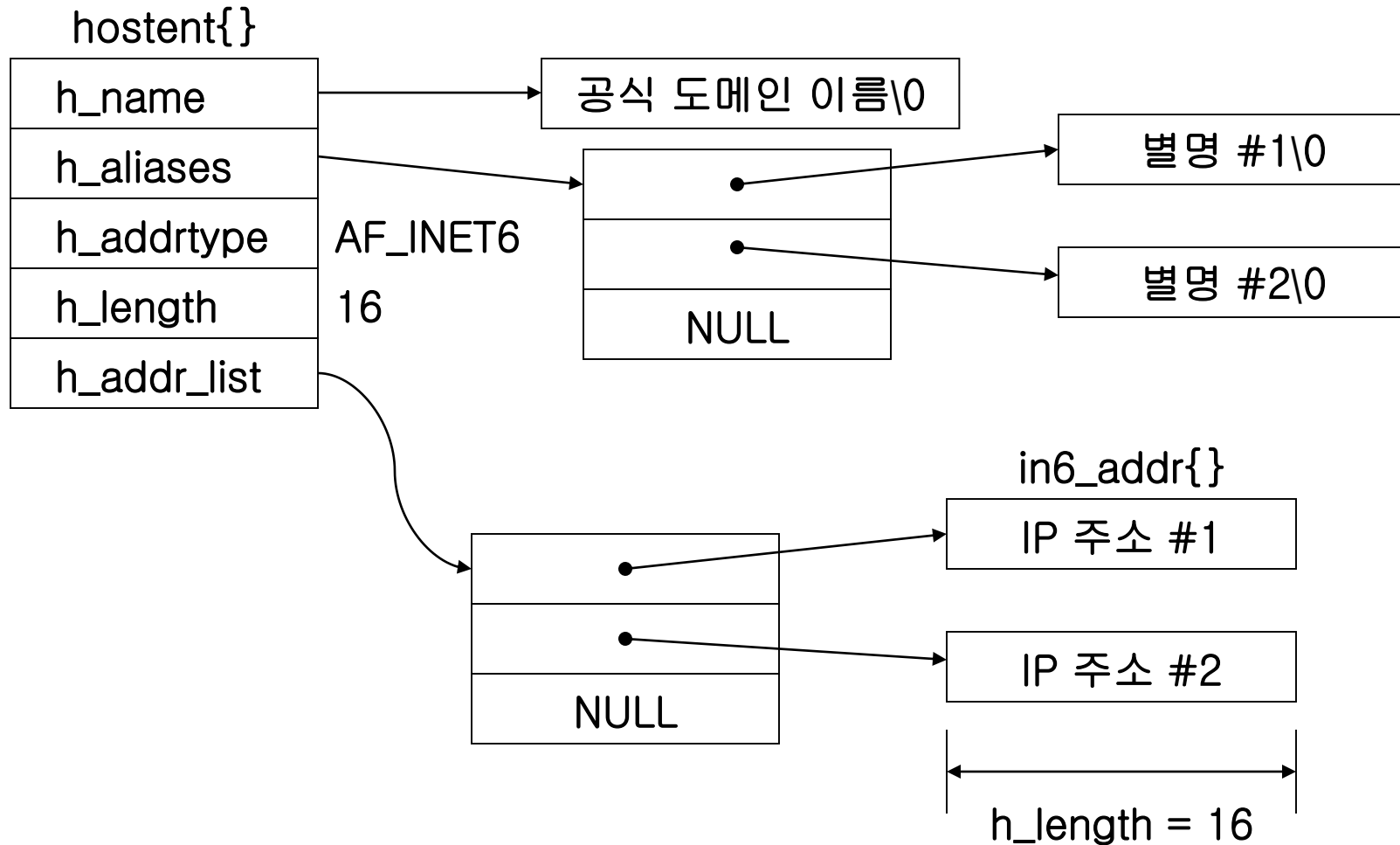
도메인 이름 시스템과 이름 변환 함수 (4)

■ 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

```
1 // 도메인 이름 -> IPv4 주소
2 bool GetIPAddr(const char *name, struct in_addr *addr)
3 {
4     struct hostent *ptr = gethostbyname(name);
5     if (ptr == NULL) {
6         err_display("gethostbyname()");
7         return false;
8     }
9     if (ptr->h_addrtype != AF_INET)
10         return false;
11     memcpy(addr, ptr->h_addr, ptr->h_length);
12     return true;
13 }
```


도메인 이름 시스템과 이름 변환 함수 (7)

■ 사용자 정의 함수 ②

```
1 // IPv4 주소 -> 도메인 이름
2 bool GetDomainName(struct in_addr addr, char *name, int namelen)
3 {
4     struct hostent *ptr = gethostbyaddr((const char *)&addr,
5         sizeof(addr), AF_INET);
6     if (ptr == NULL) {
7         err_display("gethostbyaddr()");
8         return false;
9     }
10    if (ptr->h_addrtype != AF_INET)
11        return false;
12    strncpy(name, ptr->h_name, namelen);
13    return true;
14 }
```

도메인 이름 시스템과 이름 변환 함수 (8)

■ 실습 3-3 이름 변환 함수 연습

- NameResolution.cpp
- [윈도우] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Windows/Chapter03/NameResolution/NameResolution.cpp>
- [리눅스] <https://github.com/promche/TCP-IP-Socket-Prog-Book-2nd/blob/Source/Linux/Chapter03/NameResolution.cpp>