
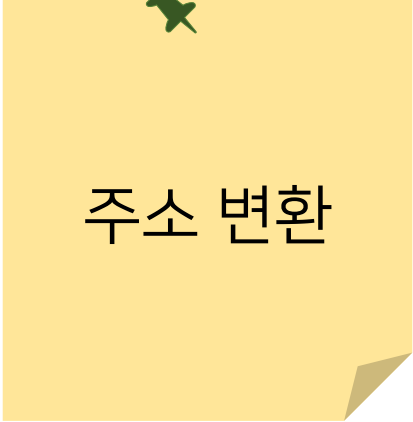


# 인터넷 주소 변환


## 목차



구조체



주소 변환



함수

## IP 주소 정보

IPv4

```
struct sockaddr_in
{
    sa_family_t    sin_family; // 주소 체계
    uint16_t       sin_port;   // port번호(16bit)
    struct in_addr sin_addr;   // IP 주소(32bit)
    char           sin_zero[8]; // 사용 X
}
```

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

## 인터넷 주소 변환

ex) 4바이트 정수 1

00000000 00000000 00000000 00000001

00000001 00000000 00000000 00000000

# 인터넷 주소 변환

Host Byte Order

Big Endian

상위 바이트 값을 **작은**  
번지수에 저장하는 방식

Little Endian

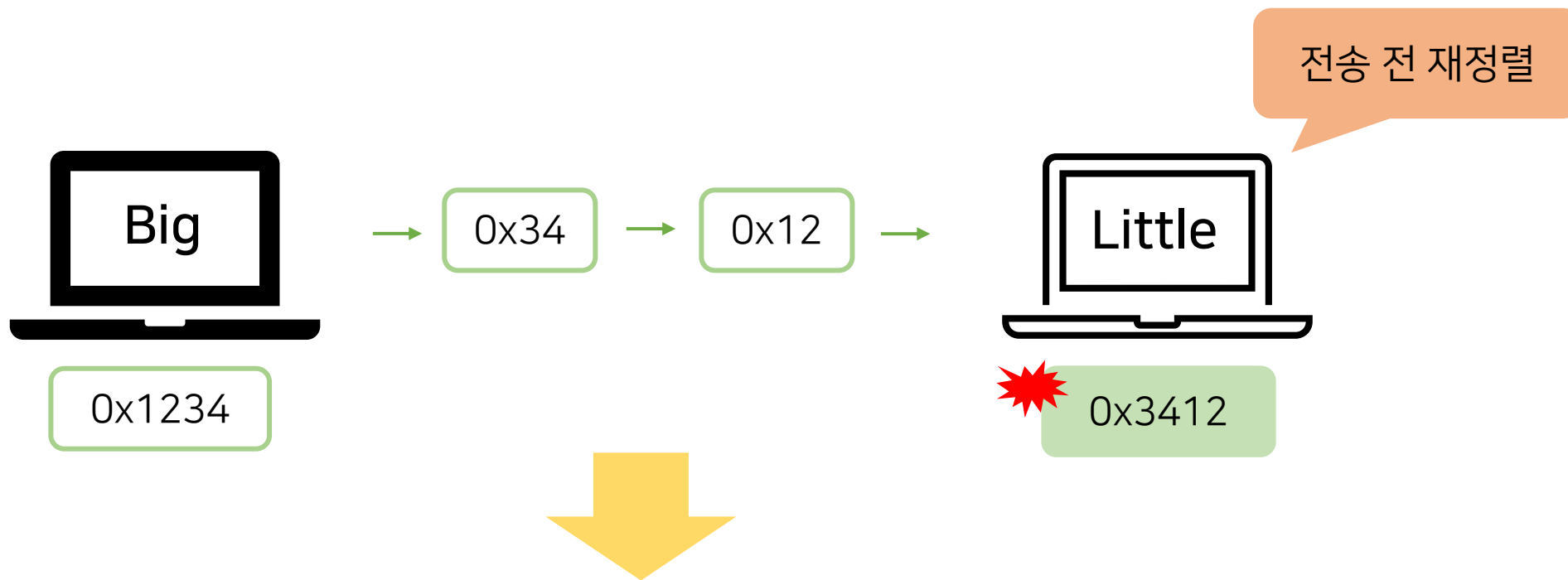
상위 바이트 값을 **큰**  
번지수에 저장하는 방식

0x12345678



	0x20번지	0x21번지	0x22번지	0x23번지
빅 엔디안	0x12	0x34	0x56	0x78
리틀 엔디안	0x78	0x56	0x34	0x12

## 인터넷 주소 변환



네트워크 바이트 순서  
(big endian)

## 정수형 자료형

char  $\begin{cases} \text{signed} \\ \text{unsigned} \end{cases}$   
int  $\vdots$   
short  $\vdots$   
long

singed

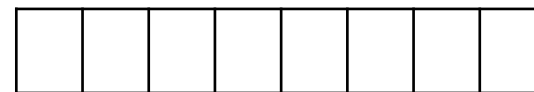
부호(+,-) 있는 정수 표현



$$2^7 = -128 \sim 127$$

unsinged

부호 없는 정수 표현



$$2^8 = 0 \sim 255$$

## 주소 변환 함수



htonl

Long형 데이터를 호스트 바이트 순서에서 네트워크 바이트 순서로 변환

ntohl

Long형 데이터를 네트워크 바이트 순서에서 호스트 바이트 순서로 변환



## 주소 변환 함수

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

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;
}
```

```
(root@kali)-[~]
# ./result
Host ordered port: 0x1234
Network ordered port: 0x3412
Host ordered address: 0x12345678
Network ordered address: 0x78563412
```

## 주소 변환 함수2

127.0.0.1



0x7f000001

`in_addr_t inet_addr(const char * string)`

IP 주소 -> 32비트 정수형  
네트워크 바이트 순서로 변환

성공 : 변환된 32비트 값

실패 : INADDR\_NONE 반환

## 주소 변환 함수2

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

int main(int argc, char *argv[])
{
    char *addr1="4.3.2.1";
    char *addr2="1.2.3.256";

    unsigned long addr=inet_addr(addr1);
    if(addr==INADDR_NONE)
        printf("Error! \n");
    else
        printf("Network addr: %#lx \n", addr);

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

```
(root👤kali)-[~]
# ./inet_addr
Network addr: 0x1020304
Error
```

## 주소 변환 함수3

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

IP 주소 -> 32비트 정수형

네트워크 바이트 순서로 변환

+ 자동으로 in\_addr 구조체 변수에 저장

성공 : true

실패 : false

## 주소 변환 함수2

```
#include <stdio.h>
#include <stdlib.h>
#include <arpa/inet.h>
void error_handling(char *message);

int main(int argc, char *argv[])
{
    char *addr="4.3.2.1";
    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;
}
```

```
(root@kali)-[~]
# ./aton
Network ordered integer addr: 0x1020304
```

## 주소 변환 함수3

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

32비트 정수형 -> IP 주소  
inet\_aton과 반대

성공 : 변환된 주소 값

실패 : -1

### ※ 유의사항

함수 내부적을 메모리 공간을 할당해 변환된 문자열 저장함

-> 재호출되면 그 전 정보 지워짐

## 주소 변환 함수3

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

int main(int argc, char *argv[])
{
    struct sockaddr_in addr1, addr2;
    char *str_ptr;
    char str_arr[20];

    addr1.sin_addr.s_addr=htonl(0x1020304); 0x4030201
    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;
}
```

```
(rootkali)-[~]
# ./ntoa
```

```
Dotted-Decimal notation1: 1.2.3.4
Dotted-Decimal notation2: 1.1.1.1
Dotted-Decimal notation3: 1.2.3.4
```

감사합니다. 😊