7.2 字节序转换 API_物联网 / 嵌入式工程师 - 慕课网

> 慕课网慕课教程 7.2 字节序转换 API 涵盖海量编程基础技术教程，以图文图表的形式，把晦涩难懂的编程专业用语，以通俗易懂的方式呈现给用户。

- 查看 ip 地址
    - windows 系统： ipconfig
      ipconfig/all

      linux 系统： ifconfig

```
linux@ubuntu:~$ ifconfig
ens33     Link encap:Ethernet  HWaddr 00:0c:29:b5:d7:8b
          inet addr:10.226.42.37  Bcast:10.226.43.255  Mask:255.255.254.0
          inet6 addr: fe80::ce4b:d8:e222:887b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:489728 errors:0 dropped:0 overruns:0 frame:0
          TX packets:253394 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:590180509 (590.1 MB)  TX bytes:26181301 (26.1 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:78116 errors:0 dropped:0 overruns:0 frame:0
          TX packets:78116 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:12888799 (12.8 MB)  TX bytes:12888799 (12.8 MB)
```

- 查看网络状态
    - netstat ： 查看当前网络服务和端口情况
      参数:
      -a (all) 显示所有选项，默认不显示LISTEN相关
      -t (tcp)仅显tcp相关选项
      -u (udp)仅显示udp相关选项
      -n 拒绝显示别名，能显示数字的全部转化成数字。
      -l 仅列出有在 Listen (监听) 的服务状态
    - 列出所有端口 (包括监听和未监听的)

```
linux@ubuntu:~$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 ubuntu:domain           *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
udp        0      0 *:56841                 *:*
udp        0      0 localhost:35358         ubuntu:domain           ESTABLISHED
udp        0      0 ubuntu:domain           *:*
udp        0      0 *:bootpc                *:*
udp        0      0 *:42204                 *:*
udp        0      0 *:mdns                  *:*
udp6       0      0 [::]:38750              [::]:*
udp6       0      0 [::]:mdns               [::]:*
raw6       0      0 [::]:ipv6-icmp          [::]:*                  7
```

    - 只列出所有监听 tcp 端口

```
linux@ubuntu:~$ netstat -lt
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 ubuntu:domain           *:*                     LISTEN
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
tcp6       0      0 ip6-localhost:ipp       [::]:*                  LISTEN
linux@ubuntu:~$
```

- 只列出所有监听 udp 端口

```
linux@ubuntu:~$ netstat -lu
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 *:56841                 *:*
udp        0      0 ubuntu:domain           *:*
udp        0      0 *:bootpc                *:*
udp        0      0 *:42204                 *:*
udp        0      0 *:mdns                  *:*
udp6       0      0 [::]:38750              [::]:*
udp6       0      0 [::]:mdns               [::]:*
```

- 主机显示具体的 ip 和端口，不用用户名显示

```
linux@ubuntu:~$ netstat -ntl
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.1.1:53            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
```

1. IP 字符串转换为网络字节序
    1. 函数 API
        1. 方法 1
        2.
```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>


typedef unsigned int  uint32_t;
typedef unsigned int in_addr_t;


in_addr_t inet_addr(const char *cp);
功能:将cp指向的IP字符串转成网络字节序
返回值:
成功返回网络字节序,失败返回INADDR_NONE [0xffffffff]
注意：它不能识别255.255.255.255
```
        3. 方法 2

    2.
```
int inet_aton(const char *cp, struct in_addr *inp);     [addr to network]
功能:将cp指向的IP字符串转成网络字节序inp保存的地址中。
参数:
@cp  IP字符串首地址
@inp 存放网络字节序的地址
返回值:
成功返回非0,失败返回0

  struct in_addr
  {
         unsigned int s_addr;
  };
实质：存储在inp结构体指针中的 s_addr这个变量中。
```
    3. 代码实战
    4.
```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdlib.h>

void ip_convert_first(const char *ip)
{
        in_addr_t net_addr;


        net_addr = inet_addr(ip);
        if(net_addr == INADDR_NONE)
```

```
        {
                perror("Fail to inet_addr");
                exit(EXIT_FAILURE);
        }
        printf("net_addr = %#x\n",net_addr);
        return ;
}

void ip_convert_second(const char *ip)
{
        struct in_addr net_addr;
        int ret;

         ret = inet_aton(ip,&net_addr);
        if(0 == ret)
        {
                perror("Fail to inet_aton");
                exit(EXIT_FAILURE);
        }
        printf("net_addr = %#x\n",net_addr.s_addr);
        return ;
}


int main(int argc, const char *argv[])
{
        if(argc != 2)
        {
                fprintf(stderr,"Usage : %s ip\n",argv[0]);
                exit(EXIT_FAILURE);
        }

        ip_convert_first(argv[1]);
        ip_convert_second(argv[1]);

        return 0;
}
```

-
- 运行结果：

```
linux@ubuntu:~/network$ ./a.out 192.168.1.100
net_addr = 0x6401a8c0
net_addr = 0x6401a8c0

linux@ubuntu:~/network$ ./a.out 255.255.255.255
Fail to inet_addr: Success
```

1. 网络字节序转换为 IP 字符串
    1. 函数 API
    2.
```
char *inet_ntoa(struct in_addr in);   [network to addr]
功能：将IP网络字节序转换成IP字符串
参数：
@in   IP网络字节序

返回值：
成功返回IP字符串首地址，失败返回NULL
```
    3. 代码实战
    4.
```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdlib.h>

void ip_convert(const char *ip)
{
        in_addr_t net_addr;
        struct in_addr addr;


        net_addr = inet_addr(ip);
        printf("network : %#x\n",net_addr);
        if(net_addr == INADDR_NONE)
        {
                perror("Fail to inet_addr");
                exit(EXIT_FAILURE);
        }
```

```
#if 0
        addr.s_addr = net_addr;
        printf("ip = %s\n",inet_ntoa(addr));
#else

        addr = *((struct in_addr *)&net_addr);
        printf("ip = %s\n",inet_ntoa(addr));
#endif

        return ;
}


int main(int argc, const char *argv[])
{
        int ret = 0;
        if(argc < 2)
        {
                fprintf(stderr,"Usage : %s ip\n",argv[0]);
                exit(EXIT_FAILURE);
        }

        ip_convert(argv[1]);

        return 0;
}
```

5. 运行结果

6.
```
linux@ubuntu:~/network$ ./a.out 192.168.0.88
network : 0x5800a8c0
ip = 192.168.0.88
```

2. 主机字节序转换为网络字节序

1.
```
short htons(short data);        [host  to network short ]
功能：将short类型的整数转成网络字节序
参数：
          @data  序号转换的整数
返回值：得到的网络字节序
```

3. 网络字节序转换为十进制数

1.
```
int  atoi(const char *nptr);
功能：把ntpr 所指向的整数字符串转换成整数。
参数：
    @  nptr     字符串
返回值：成功，返回转换后的整数
                失败，返回0
注意：若是只有+，-和整数字符能转换，其他字符返回0

uint32_t ntohs(uint32_t netlong);           [network to host short]
功能：把网络字节序转换为主机端口
参数：
@   netlong     网络字节序
返回值：  返回对应的主机端口
```

2. 代码实战

3.
```
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>

void ip_convert(const char *ip,const char *port)
{
    in_addr_t  net_addr;
    uint16_t   net_port;
    uint16_t   host_port;


    net_addr = inet_addr(ip);

    if(net_addr == INADDR_NONE)
    {
            perror("Fail to inet_addr");
            exit(EXIT_FAILURE);
    }
    printf("net_addr = %#x\n",net_addr);
```

```
    net_port = htons(atoi(port));
    printf("net_port = %#x\n",net_port);


    struct in_addr  addr;
    addr.s_addr = net_addr;
    printf("host_ip = %s\n",inet_ntoa(addr));


    host_port = ntohs(net_port);
    printf("host_port = %d\n",host_port);
}



int main(int argc, const char *argv[])
{
    if(argc != 3)
    {
            fprintf(stderr,"Usage : %s ip port!\n",argv[0]);
            exit(EXIT_FAILURE);
    }

    ip_convert(argv[1],argv[2]);
    return 0;
}
```

4. 运行结果

5.
```
linux@ubuntu:~/network$ gcc ip_port.c
linux@ubuntu:~/network$ ./a.out 127.0.0.1 9090

net_addr : 0x100007f
net_port : 0x8223
========================================
host_ip : 127.0.0.1
host_port : 9090
```

练习：

写一个代码实现以下功能

1. 用户从命令行传递参数 ./a.out 127.0.0.1 9090
2. 利用 inet_aton 和 htons 函数把 ip,port 转换为网络字节序后输出。
3. 利用 inet_ntoa 和 noths 函数把 ip,port 转换为主机字节序后输出。

全文完