<https://blog.csdn.net/qingkongyeyue/article/details/52921530>

**假设在x86平台上,有一个int型变量,在内存中的内部由低到高分别是:0x12,0x34,0x56,0x78当通过网络发送该数据时,正确的发送顺序是**

0x78,0x56,0x34,0x12

* X86 系列 CPU都是 little－endian 的，所以int 型变量值为 0x78563412, 网络发送数据时，采用大段，先发送高位再发送低位

**转自**<http://www.cnblogs.com/myyan/p/5750534.html>

**ntohs =net to host short int 16位  
htons=host to net short int 16位  
ntohl =net to host long int 32位  
htonl=host to net long int 32位**

**网络字节顺序NBO（Network Byte Order）**

按从高到低的顺序存储，在网络上使用同一的网络字节顺序，可避免兼容性问题；

**主机字节顺序HBO(Host Byte Order)**

不同的机器HBO不相同，与CPU的设计有关，数据的顺序是由CPU决定的，而与操作系统无关；

如Intel x86结构下，short型数0x1234表示为34 12，int型数0x12345678表示为78 56 34 12；

如IBM power PC结构下，short型数0x1234表示为 12 34，int型数0x12345678表示为 12 34 56 78.

由于这个原因，不同体系结构的机器之间不能直接通信，所以要转换成一种约定的顺序，也就是网络字节顺序，其实就是如同power pc那样的顺序。在PC开发中有ntohl和htonl函数可以用来进行网络字节和主机字节的转换

**linx系统下，htonl() htons() ntohl() ntohs()的头文件及函数定义如下：**

#include <arpa/inet.h>

uint32\_t htonl(uint32\_t hostlong);

uint16\_t htons(uint16\_t hostshort);

uint32\_t ntohl(uint32\_t netlong);

uint16\_t ntohs(uint16\_t netshort);

**在windows下，htonl() htons() ntohl() ntohs()的使用说明：**

**ntohs()**

简述：

**将一个无符号短整形数从网络字节顺序转换为主机字节顺序。**

#include <winsock.h>  
u\_short PASCAL FAR ntohs( u\_short netshort);  
netshort：一个以网络字节顺序表达的16位数。  
注释：本函数将一个16位数由网络字节顺序转换为主机字节顺序。  
返回值：**ntohs()返回一个以主机字节顺序表达的数。**

**htons()**

　简述：**将主机的无符号短整形数转换成网络字节顺序。**  
　#include <winsock.h>  
　u\_short PASCAL FAR htons( u\_short hostshort);  
　hostshort：主机字节顺序表达的16位数。  
　注释：本函数将一个16位数从主机字节顺序转换成网络字节顺序。  
　返回值：htons()返回一个网络字节顺序的值。

　　这2个函数提供了主机字节顺序与网络字节顺序的转换  
　　比如网络字节 为 00 01  
　　u\_short    a;如何直接对应的话    a=0100; 为什么呢？因为主机是从高字节到低字节的，所以应该转化后a=ntohs(0001); 这样 a=0001;

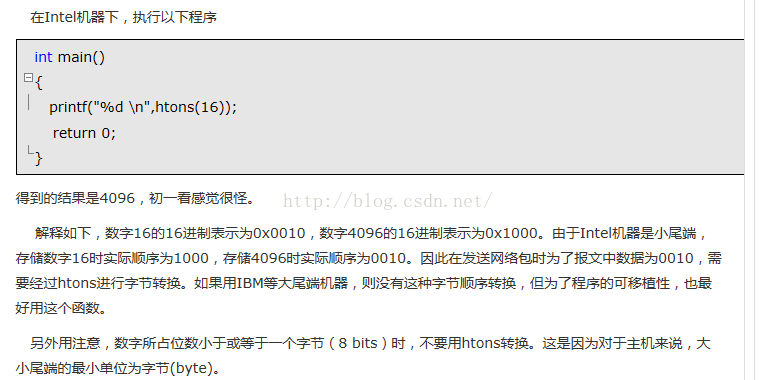
　　htonl()表示将32位的主机字节顺序转化为32位的网络字节顺序 htons()表示将16位的主机字节顺序转化为16位的网络字节顺序（ip地址是32位的端口号是16位的 ）  
  
　　将IP地址转换成长整型：首先，假设你已经有了一个sockaddr\_in结构体ina，你有一个IP地址"132.241.5.10" 要储存在其中，你就要用到函数inet\_addr(),将IP地址从点数格式转换成无符号长整型。使用方法如下：ina.sin\_addr.s\_addr = inet\_addr("132.241.5.10");  
　　注意，inet\_addr()返回的地址已经是网络字节格式，所以你无需再调用函数htonl()。  
　　我们现在发现上面的代码片断不是十分完整的，因为它没有错误检查。显而易见，当inet\_addr()发生错误时返回-1。记住这些二进制数字？(无符号数)-1仅仅和IP地址255.255.255.255相符合！但这可是广播地址！所以，记住要先进行错误检查。

　　怎样将一个in\_addr结构体输出成点数格式？你要用到函数 inet\_ntoa()("ntoa"的含义是"network to ascii")，就像这样：printf("%s",inet\_ntoa(ina.sin\_addr));它将输出IP地址。需要注意的是inet\_ntoa()将结构体in\_addr作为一个参数，不是长整形。同样需要注意的是它返回的是一个指向一个字符的指针。它是一个由inet\_ntoa()控制的静态的固定的指针，所以每次调用 inet\_ntoa()，它就将覆盖上次调用时所得的IP地址。例如：  
char \*a1, \*a2;  
.

# htons和htonl都是把主机字节序转换成网络字节序。那什么时候用htons，什么时候用htonl？？听网上说一个是16位一个是32位，但是如何去判断？假设servaddr.sin\_port = htons(5555);用htonl可以吗？根据什么可以判断？

根据要转换的值是否超过16位来决定，5555转换为2进制为1 0101 1011 0011 ，为13位，所以一般用htons,当然用htonl也可以；

但是如果要转换的数 转换成2进制超过16位，则只能用htonl，此时如果用htons，16位以上的数舍去，造成数据值偏差。



.  
a1 = inet\_ntoa(ina1.sin\_addr);  
a2 = inet\_ntoa(ina2.sin\_addr);  
printf("address 1: %s ",a1);  
printf("address 2: %s ",a2);  
输出如下：  
address 1: 132.241.5.10  
address 2: 132.241.5.10  
假如你需要保存这个IP地址，使用strcopy()函数来指向你自己的字符指针。

　　inet\_ntoa()  
简述：  
　将网络地址转换成“.”点隔的字符串格式。  
  #include　<winsock.h>  
  char FAR\* PASCAL FAR inet\_ntoa( struct in\_addr in);  
  in：一个表示Internet主机地址的结构。  
　注释：  
    本函数将一个用in参数所表示的Internet地址结构转换成以“.” 间隔的诸如“a.b.c.d”的字符串形式。请注意inet\_ntoa()返回的字符串存放在WINDOWS套接口实现所分配的内存中。应用程序不应假设该内存是如何分配的。在同一个线程的下一个WINDOWS套接口调用前，数据将保证是有效。  
　返回值：  
    若无错误发生，inet\_ntoa()返回一个字符指针。否则的话，返回NVLL。其中的数据应在下一个WINDOWS套接口调用前复制出来。  
参见：   inet\_addr().

测试代码如下  
#pragma   comment   (lib,"Ws2\_32.lib")

//noths.obj : error LNK2001: unresolved external ymbol [\_inet\_addr@4](mailto:_inet_addr@4)  
#include <winsock.h>  
#include <iostream.h>  
#include <stdio.h>  
int main(int aargc, char\* argv[])  
{  
         struct in\_addr addr1,addr2;  
         unsigned long l1,l2;  
         l1= inet\_addr("192.168.0.74");  
         l2 = inet\_addr("211.100.21.179");  
         memcpy(&addr1, &l1, 4);  
         memcpy(&addr2, &l2, 4);

         printf("%s : %s \n", inet\_ntoa(addr1), inet\_ntoa(addr2));    //注意这一句的运行结果

         printf("%s \n", inet\_ntoa(addr1));  
         printf("%s \n", inet\_ntoa(addr2));  
         return 0;  
}  
实际运行结果如下：  
192.168.0.74 : 192.168.0.74       //从这里可以看出,printf里的inet\_ntoa只运行了一次。  
192.168.0.74  
211.100.21.179  
　　inet\_ntoa返回一个char \*,而这个char \*的空间是在inet\_ntoa里面静态分配的，所以inet\_ntoa后面的调用会覆盖上一次的调用。第一句printf的结果只能说明在printf里面的可变参数的求值是从右到左的，仅此而已。

 inet\_ntoa  将一个十进制网络字节序转换为点分十进制IP格式的字符串。  
inet\_addr  将一个点分十进制的IP转换成一个长整数型数