<https://blog.csdn.net/zhubosa/article/details/51981294>

typedef unsigned short int uint16;

typedef unsigned long int uint32;

// 短整型大小端互换

#define BigLittleSwap16(A)  ((((uint16)(A) & 0xff00) >> 8) | \

                            (((uint16)(A) & 0x00ff) << 8))

 // 长整型大小端互换

#define BigLittleSwap32(A)  ((((uint32)(A) & 0xff000000) >> 24) | \

                            (((uint32)(A) & 0x00ff0000) >> 8) | \

                            (((uint32)(A) & 0x0000ff00) << 8) | \

                            (((uint32)(A) & 0x000000ff) << 24))

 // 本机大端返回1，小端返回0

int checkCPUendian()

{

       union{

              unsigned long int i;

              unsigned char s[4];

       }c;

       c.i = 0x12345678;

       return (0x12 == c.s[0]);

}

// 模拟htonl函数，本机字节序转网络字节序

unsigned long int t\_htonl(unsigned long int h)

{

       // 若本机为大端，与网络字节序同，直接返回

       // 若本机为小端，转换成大端再返回

       return **checkCPUendian**() ? h : BigLittleSwap32(h);

}

// 模拟ntohl函数，网络字节序转本机字节序

unsigned long int t\_ntohl(unsigned long int n)

{

       // 若本机为大端，与网络字节序同，直接返回

       // 若本机为小端，网络数据转换成小端再返回

       return **checkCPUendian**() ? n : BigLittleSwap32(n);

}

// 模拟htons函数，本机字节序转网络字节序

unsigned short int t\_htons(unsigned short int h)

{

       // 若本机为大端，与网络字节序同，直接返回

       // 若本机为小端，转换成大端再返回

       return **checkCPUendian**() ? h : BigLittleSwap16(h);

}

// 模拟ntohs函数，网络字节序转本机字节序

unsigned short int t\_ntohs(unsigned short int n)

{

       // 若本机为大端，与网络字节序同，直接返回

       // 若本机为小端，网络数据转换成小端再返回

       return **checkCPUendian**() ? n : BigLittleSwap16(n);

}