<https://blog.csdn.net/10km/article/details/49021499>

对于涉及跨平台开发的项目,就可能会遇到数据大小端的问题，其实就是一个数字在内存中的字节序的问题，判断当前系统是大小端有现成的例子，自己实现转换代码也非常方便,网上有好多不用多说。   
但我是个懒人，就算是这么简单的代码，有现成的就不想自己写。   
今天要说的是gcc本身已经提供了大小端的判断和数据转换的函数，真的没必要自己写。

## gcc预定义宏判断大小端(Endian)

先看看这个文章:[查看gcc的默认宏定义命令(点击打开链接)](http://blog.csdn.net/cywosp/article/details/10730931)   
这个文章告诉你如何获取gcc默认的宏定义,在这些宏定义中就有大小端模式的预定义宏\_\_BYTE\_ORDER\_\_   
当\_\_BYTE\_ORDER\_\_==\_\_ORDER\_LITTLE\_ENDIAN\_\_时是小端模式   
当\_\_BYTE\_ORDER\_\_==\_\_ORDER\_BIG\_ENDIAN\_\_时是小端模式   
\_\_BYTE\_ORDER\_\_,\_\_ORDER\_LITTLE\_ENDIAN\_\_,\_\_ORDER\_BIG\_ENDIAN\_\_都是gcc预定义的宏，在代码中可以直接使用。

## gcc内置函数进行大小端转换

gcc提供了不少有用的内置函数(Built-in Functions),这些函数说明可以在gcc的网站上找到

[6.58 Other Built-in Functions Provided by GCC(点击打开链接)](https://gcc.gnu.org/onlinedocs/gcc/Other-Builtins.html)   
这个页面最后面三个函数就是我们需要的:   
— Built-in Function: uint16\_t \_\_builtin\_bswap16 (uint16\_t x)   
Returns x with the order of the bytes reversed; for example, 0xaabb becomes 0xbbaa. Byte here always means exactly 8 bits.//返回x的反序字节,例如：0xaabb变成0xbbaa,下面类同。   
— Built-in Function: uint32\_t \_\_builtin\_bswap32 (uint32\_t x)   
Similar to \_\_builtin\_bswap16, except the argument and return types are 32 bit.   
— Built-in Function: uint64\_t \_\_builtin\_bswap64 (uint64\_t x)   
Similar to \_\_builtin\_bswap32, except the argument and return types are 64 bit.

\_bswap16,\_bswap32,\_bswap64三个函数分别提供了16位，32位，64位数字的字节反转功能，正好可以用来实现16,32,64位数字的大小端转换。

## 样例代码

如下写代码中用到了上面提到的函数，对32位的float和64位的double在大端模式下进行转换

#if \_\_BYTE\_ORDER\_\_ ==\_\_ORDER\_BIG\_ENDIAN\_\_

//当系统为大端时，把face\_code结构体中的以小端模式存储的数据转换为大端

inline face\_code\* \_code\_reverse(face\_code\* code) {

//gcc下用内置函数转换

for(int i=0;i<CODE\_FLOAT\_NUM;i++)

code->element[i]=(float)\_\_builtin\_bswap32((unsigned \_\_int32)code->element[i]);

code->sum=(double)\_\_builtin\_bswap64((unsigned \_\_int64)code->sum);

return code;

}

#define FACE\_CODE\_CONVERT(code) \_code\_reverse(code)

#else//小端模式则什么也不做直接返回

#define FACE\_CODE\_CONVERT(code) code

版权声明：本文为博主原创文章，转载请注明源地址。 https://blog.csdn.net/10km/article/details/49021499