数据在存放到内存里的时候，有两种存放方式，即：Big Endian 和 Little Endian，这两个存取方式决定了内存存放数据的原则是 **高高低低** 原则 还是 **高低低高** 原则。

**高高低低--内存中的高位存放数据的高位，内存中的低位存放数据的低位（**Little Endian **）**

**高低低高--内存中的高位存放数据的低位，内存中的低位存放数据的高位（**Big Endian **）**

谈到字节序的问题，必然牵涉到两大CPU派系。那就是Motorola的PowerPC系列CPU和Intel的x86系列CPU。PowerPC系列采用big endian方式存储数据，而x86系列则采用little endian方式存储数据。那么究竟什么是big endian，什么又是little endian呢？

其实big endian是指低地址存放最高有效字节（MSB），而little endian则是低地址存放最低有效字节（LSB）。

    用文字说明可能比较抽象，下面用图像加以说明。比如数字0x12345678在两种不同字节序CPU中的存储顺序如下所示：

Big Endian

   低地址                                            高地址

 ----------------------------------------------------------------------------->

   |     12     |      34    |     56      |     78    |

Little Endian

   低地址                                            高地址

   ----------------------------------------------------------------------------->

   |     78     |      56    |     34      |     12    |

从上面两图可以看出，采用big endian方式存储数据是符合我们人类的思维习惯的。

     为什么要注意字节序的问题呢？你可能这么问。当然，如果你写的程序只在单机环境下面运行，并且不和别人的程序打交道，那么你完全可以忽略字节序的存在。但是，如果你的程序要跟别人的程序产生交互呢？尤其是当你把你在微机上运算的结果运用到计算机群上去的话。

在这里我想说说两种语言。C/C++语言编写的程序里数据存储顺序是跟编译平台所在的CPU相关的，而JAVA编写的程序则唯一采用big endian方式来存储数据。试想，如果你用C/C++语言在x86平台下编写的程序跟别人的JAVA程序互通时会产生什么结果？就拿上面的0x12345678来说，你的程序传递给别人的一个数据，将指向0x12345678的指针传给了JAVA程序，由于JAVA采取big endian方式存储数据，很自然的它会将你的数据翻译为0x78563412。因此，在你的C程序传给JAVA程序之前有必要进行字节序的转换工作。

所有网络协议也都是采用big endian的方式来传输数据的。所以有时我们也会把big endian方式称之为网络字节序。当两台采用不同字节序的主机通信时，在发送数据之前都必须经过字节序的转换成为网络字节序后再进行传输。ANSI C中提供了下面四个转换字节序的宏。

一道C语言的试题：请写一个C函数，若处理器是Big\_endian的，则返回0；若是Little\_endian的，则返回1。

解答：

int checkCPU()  
{  
　{  
　　union w  
　　{  
　　　int a;  
　　　char b;  
　　} c;  
　　c.a = 1;  
　　return (c.b == 1);  
　}  
}

嵌入式系统开发者应该对Little-endian和Big-endian模式非常了解。采用Little-endian模式的CPU对操作数的存放方式是从低字节到高字节，而Big-endian模式对操作数的存放方式是从高字节到低字节。例如，16bit宽的数0x1234在Little- endian模式CPU内存中的存放方式（假设从地址0x4000开始存放）为：

内存地址

存放内容

0x4000

0x34

0x400

0x12  
　　而在Big-endian模式CPU内存中的存放方式则为：

内存地址

存放内容

0x4000

0x12

0x4001

0x34

　　32bit宽的数0x12345678在Little-endian模式CPU内存中的存放方式（假设从地址0x4000开始存放）为：

内存地址

存放内容

0x4000

0x78

0x4001

0x56

0x4002

0x34

0x4003

0x12

　　而在Big-endian模式CPU内存中的存放方式则为：

内存地址

存放内容

0x4000

0x12

0x4001

0x34

0x4002

0x56

0x4003

0x78

　　联合体union的存放顺序是所有成员都从低地址开始存放，解答利用该特性，轻松地获得了CPU对内存采用Little-endian还是Big-endian模式读写。

分类: [C语言基础知识及数据结构](https://www.cnblogs.com/renyuan/category/420320.html)