先来简单的讲下什么是大小端模式，以及两个模式的区别：所谓大小端模式就是存储数据时，数据的高低位怎么存储在地址的高低位上。（位指的是bit，一个char类型数据有8位）

大端模式：数据的高位，存放在地址的低位。（高位存低位，低位存高位）

小端模式：数据的高位，存放在地址的高位。（高位存高位，低位存低位）

我们的pc机一般都是小端模式，个人感觉这也更符合我们的习惯，在地位置的就是低位数据。下面举个例子：

有int型数据0x12345678存放在地址0x00开始处，则大小端存放方法见下图

| 地址： | 0x03 | 0x02 | 0x01 | 0x00 |
|---|---|---|---|---|
| 小端模式： | 0x12 | 0x34 | 0x56 | 0x78 |
| 大端模式： | 0x78 | 0x56 | 0x34 | 0x12 |



说明：大小端是对多个字节的存储处理，单个字节没影响

以 0x01 为例：

地址： 0x03 0x02 0x01 0x00

小端： 0x00 0x00 0x00 0x01

大端： 0x01 0x00 0x00 0x00

判断大小端模式程序：

第一种：用位移方法（截图时没把下面的return 0 和 }截取到）

```
1 #include<stdio.h>
2 #define BigOrLittle(n) (((n) >> 8)?(printf("Big!\n")):(printf("Little!\n")))
3
4 int main()
5 {
6     unsigned short  a = 1;
7     BigOrLittle(a);
8     //( a >> 8 )? printf("Big!\n"):printf("Little!\n");
9
10
11
```



如果真正理解了大小端模式的原理，本质。用位移来操作，个人感觉是比较简单和比较正确的。因为大小端模式就是位的问题，所以用位移法表明你真正理解了本质。变量右移8（因为a = 1，变成二进制为：0000 0001 所以其实右移一位就可以判断了。）左边不够补0。所以就可以判断了，如果1存放在高位，右移8位后1还是存在的；如果1存放在低位，右移8位后1将被移出数据，所以不存，结果为0。

地址： 高地址 <======= 低地址

小端模式存放：0000 0001 >> 8 == 0000 0000

大端模式存放：1000 0000 >> 8 == 0000 1000

这个可以用上面定义的宏或者用主函数内被注销的第8行代码。用宏时，要注意括号，否则适得其反。运行结果图如下：

```
[root@yzh test]#
[root@yzh test]# gcc BigOrLittle.c
[root@yzh test]# ./a.out
Little!
[root@yzh test]#
```

第二种：数据类型强制转换

```
 1
 2 #include<stdio.h>
 3
 4 int main(void)
 5 {
 6     unsigned short a = 1;
 7     char c = (char)a;
 8 http://blog.csdn.net/YuZhiHui_No1
 9     (c) ? printf("Little!\n"):printf("Big!\n");
10     return 0;
11
12 }
13
```

数据类型强制转换，其实本质还是取低8位数来判断。分析同上：a = 1 ,变成二进制为：0000 0001.强转为char时，保留unsigned short a变量的低8位。所以这样就更加明了了。因为1低位数据，如果char c不为0，表明1存放在低位，即是小端模式；如果char c变量为0，表明1存放在高位，即是大端模式。运行结果图如下：

```
[root@yzh test]#
[root@yzh test]# gcc BigOrLittle.c
[root@yzh test]# ./a.out
Little!
[root@yzh test]#
```

第三种：用共同体

```
 2 #include<stdio.h>
 3
 4 typedef union test{
 5     int  a;
 6     char c;
 7
 8 }T;
 9
10 int main(void)
11 {  http://blog.csdn.net/YuZhiHui_No1
12     T t;
13     t.a = 1;
14
15     (t.c) ? (printf("Little!\n")):(printf("Big!\n"));
16
17     return 0;
18 }
19
```

老实说这是个最差的办法，因为这个程序和第二个强转的是非常非常相似的，或者根本可以说是一样的。但这个程序比较复杂和第二个相比较，所以我是最不想用这个的，但非常不好意思的是我面试时，就是用共用体这种方法来判断的。因为当时是现场出的题，有点紧张，一心只想写出来了就可以，根本没去想用其他更简单的方法。稍微解释下，只要知道共用体就非常好理解了。共用体里面的

成员共用一个内存空间，而且是从低位开始占用，共用体变量的内存空间大小是该变量中某个占用空间最大的那个成员所占的空间。比如上面的结构体变量的空间就是int a占空间的大小，char c是从低位开始占用，也就是占用int a的低8位。这么一解释，我想大家都明白了，这个和上面的其实就是一样的。运行结果如下图：