## 第六十二节： 指针，大小端，化整为零，化零为整。

### 【62.1　内存的大小端。】

C51 编译器的 unsigned int 占 2 字节 RAM（也称为内存），unsigned long 占 4 字节 RAM，这两种数据类型所占的字节数都超过了 1 个字节，而 RAM 内存是每一个地址对应一个字节的 RAM 内存，那么问题就来了，比如像 unsigned long 这种占 4 个字节 RAM 的数据变量，它这 4 个字节在 RAM 中的地址是“连续”的“挨家挨户”的“连号”的，这 4 个字节所存的一个数据，它的数据高低位在地址的排列上，到底是从低到高还是从高到低，到底是“正向”的还是“反向”？这两种不同的排列顺序，在 C 语言里用“大端”和“小端”这两个专业术语来描述。“大端”的方式是将高位存放在低地址，“小端”的方式是将低位存放在低地址。比如：

假设有一个 unsigned long 变量 a 等于 0x12345678，是存放在 RAM 内存中的第 4,5,6,7 这四个“连号”的地址里，现在看看它在“大端”和“小端”的存储方式里的差别。如下：

（1）在“大端”的方式里，将高位存放在低地址。
0x12 存在第 4 个地址，0x34 存在第 5 个地址，0x56 存在第 6 个地址，0x78 存在第 7 个地址。

（2）在“小端”的方式里，将低位存放在低地址。
0x78 存在第 4 个地址，0x56 存在第 5 个地址，0x34 存在第 6 个地址，0x12 存在第 7 个地址。

问题来了，在单片机里，内存到底是“大端”方式还是“小端”方式？答：这个跟 C 编译器有关。比如，在 51 单片机的 C51 编译环境里是“大端”方式，而在 stm32 单片机的 ARM_MDK 编译环境里则是“小端”方式。那么问题又来了？如何知道一个 C 编译器是“大端”还是“小端”？答：有两种方式，一种是看 C 编译器的说明书，另一种是自己编写一个小程序测试一下就知道了（这种方法最简单可靠）。那么问题又来了？讲这个 “大小端”有什么用？答：这个跟指针的使用密切相关。

### 【62.2　化整为零。】

在数据的存储和通信中，往往要先把数据转换成以字节为单位的数组，才能进行数据存储和通信。比如 unsigned long 这种类型的数据，就要先转换成 4 个字节，这种把某个变量转换成 N 个字节的过程，就是“化整为零”。“化整为零”的过程，在代码上，有两种常见的方式，一种是原始的“移位法”，另一种是极具优越性的“指针法”。比如，现在以“大端”方式为例（因为本教程是用 C51 编译器，C51 编译器是“大端”方式），有一个 unsigned long 变量 a 等于 0x12345678，要把这个变量分解成 4 个字节存放在一个数组 Gu8BufferA 中，现在跟大家分享和对比一下这两种方法。

（1）原始的“移位法”。

```
unsigned long a=0x12345678;
unsigned char Gu8BufferA[4];

Gu8BufferA[0]=a>>24;
Gu8BufferA[1]=a>>16;
Gu8BufferA[2]=a>>8;
Gu8BufferA[3]=a;
```

（2）极具优越性的“指针法”。

```
unsigned long a=0x12345678;
unsigned char Gu8BufferA[4];
unsigned long *pu32;   //引入一个指针变量，注意，这里是 unsigned long 类型的指针。

pu32=(unsigned long *)&Gu8BufferA[0];  //指针跟数组“绑定”（也称为“关联”）起来。
*pu32=a;  //这里仅仅 1 行代码就等效于上述（1）“移位”例子中的 4 行代码，所以极具优越性。
```

多说一句，“pu32=(unsigned long *)&Gu8BufferA[0]”这行代码中，其中小括号“(unsigned long *)”是表示数据的强制类型转换，这里表示强制转换成 unsigned long 的指针方式，以后这类代码写多了，就会发现这种书写方法的规律。作为语言来解读先熟悉一下它的表达方式就可以了，暂时不用深究它的含义。

## 【62.3　化零为整。】

从数据存储中提取数据出来，从通讯端接收到一堆数据，这里的“提取”和“接收”都是以字节为单位的数据，所以为了“还原”成原来的类型变量，就涉及“化零为整”的过程。在代码上，有两种常见的方式，一种是原始的“移位法”，另一种是极具优越性的“指针法”。比如，现在以“大端”方式为例（因为本教程是用 C51 编译器，C51 编译器是“大端”方式），有一个数组 Gu8BufferB 存放了 4 个字节数据分别是：0x12,0x34,0x56,0x78。现在要把这 4 个字节数据“合并”成一个 unsigned long 类型的变量 b，这个变量 b 等于 0x12345678。现在跟大家分享和对比一下这两种方法。

（1）原始的“移位法”。

```
unsigned char Gu8BufferB[4]={0x12,0x34,0x56,0x78};
unsigned long b;

b=Gu8BufferB[0];
b=b<<8;
b=b+Gu8BufferB[1];
b=b<<8;
b=b+Gu8BufferB[2];
b=b<<8;
b=b+Gu8BufferB[3];
```

（2）极具优越性的“指针法”。

```
unsigned char Gu8BufferB[4]={0x12,0x34,0x56,0x78};
unsigned long b;
unsigned long *pu32;   //引入一个指针变量，注意，这里是 unsigned long 类型的指针。

pu32=(unsigned long *)&Gu8BufferB[0];  //指针跟数组“绑定”（也称为“关联”）起来。
b=*pu32;  //这里仅仅 1 行代码就等效于上述（1）“移位”例子中的 7 行代码，所以极具优越性。
```

## 【62.4　“指针法”要注意的问题。】

“化整为零”和“化零为整”其实是一个“互逆”的过程，在使用“指针法”的时候，一定要注意“大小端”的问题。“化整为零”和“化零为整”这两个“互逆”过程要么同时为“大端”，要么同时为“小端”，否则会因字节的排列顺序问题而引起数据的严重错误。

## 【62.5　例程练习和分析。】

现在编一个练习程序。

```
/*---C 语言学习区域的开始。-----------------------------------------------*/

    unsigned long a=0x12345678;
    unsigned char Gu8BufferA[4];

    unsigned char Gu8BufferB[4]={0x12,0x34,0x56,0x78};
    unsigned long b;

    unsigned long *pu32;   //引入一个指针变量，注意，这里是 unsigned long 类型的指针。

void main() //主函数
{
    pu32=(unsigned long *)&Gu8BufferA[0];  //指针跟数组“绑定”（也称为“关联”）起来。
    *pu32=a;  //化整为零

    pu32=(unsigned long *)&Gu8BufferB[0];  //指针跟数组“绑定”（也称为“关联”）起来。
    b=*pu32;  //化零为整



    View(Gu8BufferA[0]);  //把第 1 个数 Gu8BufferA[0]发送到电脑端的串口助手软件上观察。
    View(Gu8BufferA[1]);  //把第 2 个数 Gu8BufferA[1]发送到电脑端的串口助手软件上观察。
    View(Gu8BufferA[2]);  //把第 3 个数 Gu8BufferA[2]发送到电脑端的串口助手软件上观察。
    View(Gu8BufferA[3]);  //把第 4 个数 Gu8BufferA[3]发送到电脑端的串口助手软件上观察。

    View(b);              //把第 5 个数 b 发送到电脑端的串口助手软件上观察。

    while(1)
    {
    }
}
/*---C 语言学习区域的结束。-----------------------------------------------*/
```

在电脑串口助手软件上观察到的程序执行现象如下：

```
开始...
```

```
第 1 个数
十进制:18
十六进制:12
二进制:10010

第 2 个数
十进制:52
十六进制:34
二进制:110100

第 3 个数
十进制:86
十六进制:56
二进制:1010110

第 4 个数
十进制:120
十六进制:78
二进制:1111000

第 5 个数
十进制:305419896
十六进制:12345678
二进制:10010001101000101011001111000
```

分析：

Gu8BufferA[0]为 0x12。
Gu8BufferA[1]为 0x34。
Gu8BufferA[2]为 0x56。
Gu8BufferA[3]为 0x78。
b 为 0x12345678。

## 【62.6　如何在单片机上练习本章节 C 语言程序？】

直接复制前面章节中第十一节的模板程序，练习代码时只需要更改“C 语言学习区域”的代码就可以了，其它部分的代码不要动。编译后，把程序下载进带串口的 51 学习板，通过电脑端的串口助手软件就可以观察到不同的变量数值，详细方法请看第十一节内容。