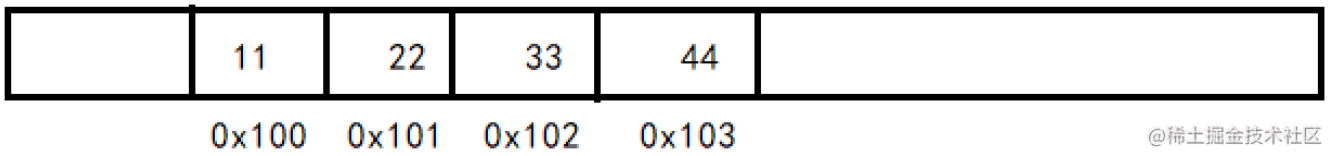
9. 怎么判断大端小端

什么是大端序，小端序？ 其实就是字节的存储顺序，如果数据都是单字节的，那怎么存储无所谓了，但是对于多字节数据，比如int，double等，就要考虑存储的顺序了。 举个例子： 一个32位 int 型变量 0x11223344 占用四个字节，11 占用一个，22占用一个，33占用一个，44占用一个；存储的地址为 0x100 0x101 0x102 0x103；那么问题是，11 占用的是哪个字节呢？ 11 占用 0x100这个字节还是，0x103这个字节呢？这时便有了两种方式排序：

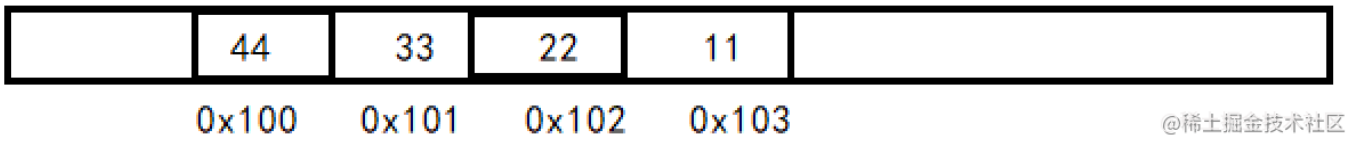
1.大端序：

大端序即数字的高位占用低地址，低位占用高地址，这种也是最符合直觉的



2. 小端序：

小端序即数字的低位占用低地址，高位占用高地址



判断方法1：

实现思想：

1. 定义一个 32 位的 int 型变量，0x11223344
2. 将这个 int 型变量的低地址开始的 8 位存储的值取出来，取出来的方法就是利用强制类型转换
3. 如果这个值是 0x11 那么说明低地址存储了值的高位，所以为大端序
4. 如果这个值是 0x44 那么说明低地址存储了值的低位，所以为小端序

```c
#include "stdio.h"
#include "stdlib.h"

// 判断大端还是小端？？
// 如果是大端序函数返回 1
// 如果是小端序函数返回 0
int Judge_BS(int n) {
    // 如果是大端序，数字 n 的低位存储在高地址中
    // 即 44 存储在高地址中，11 存储在低地址中
    // 地址：    0x100   0x101   0x102   0x103
    // 数字：    11  22  33  44

    // 如果是小端序，数字 n 的低位存储在低地址中
    // 即 11 存储在高地址中，44 存储再低地址中
    // 地址：    0x100   0x101   0x102   0x103
    // 数字：    44      33      22      11

    // 所以我们可以将32位数字 n 的 低 8 位取出来
    // 如果低 8 位是 11 则为大端序
    // 如果低 8 位是 44 则为小端序
```

```
    //此处的地址存储的是低地址
    char* p = &n;
    printf("%x\n", *p);
    printf("%x\n", *(p + 1));
    printf("%x\n", *(p + 2));
    printf("%x\n", *(p + 3));
    // 用 p 获得 32 的低地址，每 8 位打印一次数字，打印结果为
    // 44 33 22 11
    // 低地址对应着低位，是小端序
    char t = *p;
    if (t == 11) {
        return 1;
    }
    return 0;
}
int main() {
    int n = 0x11223344;
    if (Judge_BS(n)) {
        printf("是大端序! ");
    } else {
        printf("是小端序! ");
    }
    system("pause");
    return 0;
}
```

判断方法2：

- 实现思想：思想基本与第一种方法相同，区别在于这次使用了联合体，利用联合体的共用内存的特点实现。

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    union Un {
        int a;
        char b;
    } Un;
    Un.a = 0x11223344;
    if (Un.b == 0x11) {
        printf("大端\n");
    } else {
        printf("小端\n");
    }
    system("pause");
    return 0;
}
```

参考资料：判断 机器是大端还是小端（两种方法）_z7436-CSDN博客_你的机器是大端方式还是小端方式