java字节序、主机字节序和网络字节序扫盲贴

原创 aitangyong 2014-04-08 21:31:54

使用C/C++进行网络编程的程序员，肯定会接触到“字节序”的概念，但是使用java进行网络编程，却根本不会接触到“字节序”。为什么会这样呢？我们先从字节序说起。字节顺序是指占用内存多于一个字节类型的数据在内存中的存放顺序，有小端、大端两种顺序。小端字节序（little endian）：低字节数据存放在内存低地址处，高字节数据存放在内存高地址处；大端字节序（bigendian）：高字节数据存放在低地址处，低字节数据存放在高地址处。

java中一个int型数据占用4个字节，假如有一个16进制的int数，int value = 0x01020304;采用不同的字节序，在内存中的存储情况见下图：

小字节序

内存地址编号	字节内容
0X00001000	04
0X00001001	03
0X00001002	02
0X00001003	01

大字节序

内存地址编号	字节内容
0X00001000	01
0X00001001	02
0X00001002	03
0X00001003	04

显然大字节序，是比较符合人类思维习惯的。

至于计算机到底是BIG-ENDIAN、LITTLE-ENDIAN、跟CPU有关的，一种CPU不是BIG-ENDIAN就是LITTLE-ENDIAN。IA架构(Intel、AMD)的CPU中是Little-Endian，而PowerPC 、SPARC和Motorola处理器是Big-Endian。这其实就是所谓的主机字节序。而网络字节序是指数据在网络上传输时是大头还是小头的，在Internet的网络字节序是BIG-ENDIAN。所谓的JAVA字节序指的是在JAVA虚拟机中多字节类型数据的存放顺序，JAVA字节序也是BIG-ENDIAN。可见网络和JVM都采用的是大字节序，个人感觉就是因为这种字节序比较符合人类的习惯。由于JVM会根据底层的操作系统和CPU自动进行字节序的转换，所以我们使用java进行网络编程，几乎感觉不到字节序的存在。

那么java里面，怎么判断你的计算机是大端存储、还是小端存储呢？JDK为我们提供一个类ByteOrder，通过以下代码就可以知道机器的字节序

```
System.out.println(ByteOrder.nativeOrder());
```

在java.nio包下提供了ByteOrder、ByteBuffer等于字节序相关的类，我们也可以改变JVM中默认的字节序。该例子来源于

http://blog.csdn.net/veryitman/article/details/6819017

代码如下：

```
package net.aty.util;


import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.util.Arrays;

public class JVMEndianTest {

    public static void main(String[] args) {

        int x = 0x01020304;

        ByteBuffer bb = ByteBuffer.wrap(new byte[4]);
        bb.asIntBuffer().put(x);
        String ss_before = Arrays.toString(bb.array());

        System.out.println("默认字节序 " +  bb.order().toString() +  "," +  " 内存数据 " +  ss_before);

        bb.order(ByteOrder.LITTLE_ENDIAN);
        bb.asIntBuffer().put(x);
        String ss_after = Arrays.toString(bb.array());

        System.out.println("修改字节序 " + bb.order().toString() +  "," +  " 内存数据 " +  ss_after);
    }
}
```

执行结果如下：

```
默认字节序 BIG_ENDIAN, 内存数据 [1, 2, 3, 4]
修改字节序 LITTLE_ENDIAN, 内存数据 [4, 3, 2, 1]
```