汉字编码及其相关内容

—CS1904 王翰辉

1. 三种编码及其关联

中国汉字通行的国际标准为我国于1981年制订的“信息交换用汉字编码字符集”,其标准号为GB2312—1980,简称国标码,是我国应用最广泛的汉字编码字符集。其中收录有6763个汉字，可以基本满足拼写需求。而后的一些编码则收录了更多更全的汉字。国标码的每个汉字使用两个字节。

国标码是一个四位的十六进制数(对应两个字节)。而区位码则是一个四位的十进制数，前两个数代表区，后两个数代表位，由于生来对十进制的敏感，在日常使用时我们多选择区位码。例如“保”在17区03位，其区位码就是“1703”。(似乎区位码是从01开始的)

国标码并不等于区位码，它是由区位码稍作转换得到,其转换方法为：先将十进制区码和位码转换为十六进制的区码和位码，这样就得了一个与国标码有一个相对位置差的代码，再将这个代码的第一个字节和第二个字节分别加上(20)h，就得到国标码。如：“保”字的国标码为(3123)h，它是经过下面的转换得到的：(1703)d—>(1103)h—>+(2020)h—>(3123)h。

这里我先忽略一个困扰我挺久的小问题(就是下面的问题1)。

接下来我不给理由地请出机内码，它也是一个四位十六进制数，它是由国标码的每个字节加上(80)h得到。

现在我们遇到了两个小问题：

1. 为什么从区位码到国标码要加(2020)h？

2. 为什么从国标码到机内码要加(8080)h？

或者再加一个，为什么是94？

首先我们明白计算机在开始的大部分标准是由美国佬建立的，他们在最开始并不会考虑加入汉字标准。所以他们制定了供他们使用的美国信息交换标准代码也就是ASCII码，而这个编码的前33位(0至32位)均为控制字符。实际上第127位也是一个控制字符，这样128减去34就是94了，也就是说明除开同样要使用的这些控制字符，我们能够使用的空间就是94\*94的范围。于是我们给每个汉字编码体系中独有的字符的区位码对应的字节加上(20)h，将01区01位编为33区33位，往后以此类推就形成了汇入了“中西两方元素”的国标码。

那么为什么又要有机内码？因为如果直接在计算机内部使用国标码将会和ASCII码产生歧义。还是拿“保”作为例子，它究竟是(3123)h这个整体呢还是(31)h，(23)h两个部分呢。因为ASCII码是0至127，说明其最高位均为0，那么将最高位改为1便可以避免上述混淆，而将两个字节的最高位改为1，在16进制表示下就是分别加上(80)h。这样“保”在计算机内部表示为(B1A3)h，就没有歧义了。

至此似乎解释完了，但又有一个问题“萦绕在我的心头”，国标码存在的意义是啥？这个问题可能需要进一步学习和查阅资料才能解开。

二、 字库与点阵字库中的字模

字库是外文字体、中文字体以及相关字符的电子文字字体集合库，换言之它就是计算机系统中储存标准字形的位置。简而言之就对应这每一种字体(在我看来)。

点阵字库是把每一个字符都分成16×16或24×24个点，然后用每个点的虚实(0为虚，1为实)来表示字符的轮廓，其中每个字形都直接以一组二维像素信息表示。这就导致了一个后果，在对这种类型的字体进行缩放时会出现锯齿边缘的现象(尤其是斜着的笔画)，就会导致失真。

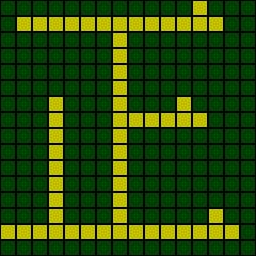
与之对应的有一个叫做矢量字库的东西。它保存的是对每一个汉字的描述信息，比如一个笔划的起始、终止坐标，半径、弧度等等。在显示、打印这一类字库时，要经过一系列的数学运算才能输出结果。这样一来，这一类字库保存的汉字理论上可以被无限地放大，笔划轮廓仍然能保持圆滑。因为所有的参数只是被等比例的进行了放大。打印时使用的字库均为此类字库。

字模在我理解中就是字在点阵上显示时对应的编码。计算机能够处理的是一系列的0，1串，在内部反映电压有无(高低)，而在显示时0对应着暗而1对应着亮。

汉字的点阵字模是从点阵字库文件中提取出来的。例如常用的16×16点阵HZK16文件，文件包括了GB2312字符集中的所有汉字。现在只要弄清汉字点阵在字库文件中的格式，就可以按照自己的意愿去显示汉字了。

如何根据区位码在字库中寻找到汉字对应的字模呢？

计算方法：字模在文件中的位置=(94 × (区号 - 1) + (位号 - 1)) × 表示汉字所需字节数。对于16×16的点阵汉字，其所需字节数就是32。以“正”字为例：

第一字节为—00000000，第二字节为—000010000，将32个字节全部排号，对应亮暗显示就能够实现“正”的显示。

三、 汉字的输入储存及输出

汉字在输入进计算机时，用的是标准的西文键盘，输入码所解决的问题是如何使用西文标准键盘把汉字输入到计算机内。常用的输入码有拼音码，五笔字形码等。而在计算机内部储存的方式则是之前提及的机内码，输出则是通过点阵字模进行显示的。