# 编码集

## 主要几类编码集分析


**张文倩**

# 目录

## 编码集的发展及使用

从头讲讲编码的故事。那么就让我们找个草堆坐下，先抽口烟，看看夜晚天空上的银河，然后想一想要从哪里开始讲起。嗯，也许这样开始比较好……

很久很久以前，有一群人，他们决定用 8 个可以开合的晶体管来组合成不同的状态，以表示世界上的万物。他们看到 8 个开关状态是好的，于是他们把这称为"字节"。 再后来，他们又做了一些可以处理这些字节的机器，机器开动了，可以用字节来组合出很多状态，状态开始变来变去。他们看到这样是好的，于是它们就这机器称为"计算机"。

开始计算机只在美国用。八位的字节一共可以组合出 256(2 的 8 次方)种不同的状态。他们把其中的编号从 0 开始的 32 种状态分别规定了特殊的用途，一但终端、打印机遇上约定好的这些字节被传过来时，就要做一些约定的动作。遇上 00x10, 终端就换行，遇上 0x07, 终端就向人们嘟嘟叫，例好遇上 0x1b, 打印机就打印反白的字，或者终端就用彩色显示字母。他们看到这样很好，于是就把这些 0x20 以下的字节状态称为"控制码"。

他们又把所有的空格、标点符号、数字、大小写字母分别用连续的字节状态表示，一直编到了第 127 号，这样计算机就可以用不同字节来存储英语的文字了。大家看到这样，都感觉很好，于是大家都把这个方案叫做 ANSI 的"Ascii"编码（American Standard Code for Information Interchange，美国信息互换标准代码）。当时世界上所有的计算机都用同样的 ASCII 方案来保存英文文字。

后来，就像建造巴比伦塔一样，世界各地的都开始使用计算机，但是很多国家用的不是英文，他们的字母里有许多是 ASCII 里没有的，为了可以在计算机保存他们的文字，他们决定

采用 127 号之后的空位来表示这些新的字母、符号，还加入了很多画表格时需要用下到的横线、竖线、交叉等形状，一直把序号编到了最后一个状态 255。从 128 到 255 这一页的字符集被称"扩展字符集"。

## GB2312

等中国人得到计算机时，已经没有可以利用的字节状态来表示汉字，况且有6000 多个常用汉字需要保存呢。但是这难不倒智慧的中国人民，我们不客气地把那些 127 号之后的奇异符号们直接取消掉，规定：一个小于 127 的字符的意义与原来相同，但两个大于 127 的字符连在一起时，就表示一个汉字，前面的一个字节（他称之为高字节）从 0xA1 用到 0xF7，后面一个字节（低字节）从 0xA1 到 0xFE，这样我们就可以组合出大约 7000 多个简体汉字了。然后就出现了这个名叫 GB2312 的编码集。GB2312 字符在计算机中存储是以其区位码为基础的，其中汉字的区码和位码分别占一个存储单元，每个汉字占两个存储单元。

## BIG5

因为当时各个国家都像中国这样搞出一套自己的编码标准，结果互相之间谁也不懂谁的编码，谁也不支持别人的编码，连大陆和台湾这样只相隔了 150 海里，使用着同一种语言的兄弟地区，也分别采用了不同的 DBCS 编码方案——当时的中国人想让电脑显示汉字，就必须装上一个"汉字系统"，专门用来处理汉字的显示、输入的问题，但是那个台湾的愚昧封建人士写的算命程序就必须加装另一套支持 BIG5 编码的什么"倚天汉字系统"才可以用，这就是 big5 的诞生原因。其实当中华人民共和国的电邮和转码软件还未普遍之时，在深圳的港商和台商公司亦曾经使用 Big5 系统，以方便与总部的文件交流、以及避免为中国的办公室再写一套不同内码的系统也算他的优点了。大五码是一种繁体中文汉字字符集，其中繁体汉字 13053 个，808 个标点符号、希腊字母及特殊符号。大五码的编码码表直接针对存储而设计，每个字符统一使用两个字节存储表示。

## UNICODE

unicode 编码(统一用 3 个字节)，因为为编码方式各自为政.如果有一种编码，将世界上所有的符号都纳入其中，无论是英文、日文、还是中文等，大家都使用这个编码表，就不会出现编码不匹配现象。每个符号对应一个唯一的编码，乱码问题就不存在了。这就是 Unicode 编码。Unicode 当然是一个很大的集合，现在的规模可以容纳 100 多万个符号。每个符号的编码都不一样，比如，U+0639 表示阿拉伯字母 Ain，U+0041 表示英语的大写字母 A，“汉”这个字的 Unicode 编码是 U+6C49。Unicode 固然统一了编码方式，但是它的效率不高，比如 UCS-4(Unicode 的标准之一)规定用 4 个字节存储一个符号，那么每个英文字母前都必然有三个字节是 0，这对存储和传输来说都很耗资源。

## UTF-8

UTF-8 编码(根据编码的长短来自动确定占用空间),为了提高 Unicode 的编码效率，于是就出现了 UTF-8 编码。UTF-8 可以根据不同的符号自动选择编码的长短。比如英文字母可以只用 1 个字节就够了。UTF-8 的编码是这样得出来的，以”汉”这个字为例：”汉”字的 Unicode 编码是 U+00006C49，然后把 U+00006C49 通过 UTF-8 编码器进行编码，最后输出的 UTF-8 编码是 E6B189

## 以上几种编码集的联系

GB2312,BIG5,UTF8,Unicode 之间是可以相互转换直接或者间接的 GB2312,BIG5,UTF8 之间的转换通过 Unicode 充当切换的桥梁。

```
//例如:GB2312<->Unicode<->BIG5
#define CODEPAGE_GB2312 936
#define CODEPAGE_BIG5 950
#define CODEPAGE_UTF8 65001
//转换成 Unicode
wchar_t*ToWideChar(const char*pStr,int nCodePage)
{int nStrLen = MultiByte To WideChar(nCodePage, 0, pStr, -1, NULL, 0);
wchar_t*pWStr=newwchar_t[nStrLen + 1];
memset(pWStr, 0, nStrLen + 1);
MultiByteToWideChar(nCodePage, 0, pStr, -1, pWStr, nStrLen);
return pWStr;}
return CPTransform(pStrGB2312, CODEPAGE_GB2312, CODEPAGE_UTF8);
}
// UTF8 转 GB2312
char* UTF8ToGB2312(const char* pStrUTF8)
{
return CPTransform(pStrUTF8, CODEPAGE_UTF8, CODEPAGE_GB2312);
}
// GB2312 转 BIG5
char* GB2312ToBIG5(const char* pStrBIG5)
{
return CPTransform(pStrBIG5, CODEPAGE_GB2312, CODEPAGE_BIG5);
}
// BIG5 转 GB2312
char* BIG5ToGB2312(const char* pStrBIG5)
{return CPTransform(pStrBIG5, CODEPAGE_BIG5, CODEPAGE_GB2312);}
```