**四种编码集的相关资料**

（GB2312 big5 UTF-8/16 Unicode）

专业：软件工程

班级：软工11502

姓名：熊政

Unicode

一、起源

Unicode 是为了解决传统的字符编码方案的局限而产生的，例如ISO 8859所定义的字符虽然在不同的国家中广泛地使用，可是在不同国家间却经常出现不兼容的情况。很多传统的[编码方式](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=50014&ss_c=ssc.citiao.link)都有一个共同的问题，即容许电脑处理双语环境（通常使用拉丁字母以及其本地语言），但却无法同时支持多语言环境（指可同时处理多种语言混合的情况）。

Unicode 编码包含了不同写法的字，如“ɑ/a”、“户/户”。然而在汉字方面引起了一字多形的认定争议。

在文字处理方面，统一码为每一个字符而非字形定义唯一的代码（即一个整数）。换句话说，统一码以一种抽象的方式（即数字）来处理字符，并将视觉上的演绎工作（例如字体大小、外观形状、字体形态、文体等）留给其他软件来处理，例如[网页浏览器](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=5217162&ss_c=ssc.citiao.link)或是文字处理器。

几乎所有电脑系统都支持基本拉丁字母，并各自支持不同的其他编码方式。Unicode为了和它们相互兼容，其首256字符保留给ISO 8859-1所定义的字符，使既有的西欧语系文字的转换不需要特别考量；并且把大量相同的字符重复编到不同的字符码中去，使得旧有纷杂的编码方式得以和Unicode编码间互相直接转换，而不会丢失任何信息。举例来说， 全角格式区段包含了主要的拉丁字母的全角格式，在中文、日文、以及[韩文](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=11042205&ss_c=ssc.citiao.link)字形当中，这些字符以全角的方式来呈现，而不以常见的半角形式显示，这对竖排文字和等宽排列文字有重要作用。

在表示一个Unicode的字符时，通常会用“[U+](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=99804179&ss_c=ssc.citiao.link)”然后紧接着一组十六进制的数字来表示这一个字符。在 [基本多文种平面](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=10945487&ss_c=ssc.citiao.link)（英文为 Basic Multilingual Plane，简写 BMP。它又简称为“零号平面”, plane 0）里的所有字符，要用四位[十六进制数](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=53251907&ss_c=ssc.citiao.link)（例如U+4AE0，共支持六万多个字符）；在零号平面以外的字符则需要使用五位或六位十六进制数了。旧版的Unicode标准使用相近的标记方法，但却有些微的差异：在Unicode 3.0里使用“U-”然后紧接着八位数，而“U+”则必须随后紧接着四位数。 [[1]](http://baike.sogou.com/v291611.htm?fromTitle=Unicode#quote1)（维基百科）

二、作用

能够使计算机实现跨语言、 跨平台的文本 转换及处理。

三、层次

Unicode 编码系统，可分为编码方式和实现方式两个层次。

四、方式

Unicode是国际组织制定的可以容纳世界上所有文字和符号的[字符编码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=738663&ss_c=ssc.citiao.link)方案。目前的Unicode字符分为17组编排，0x0000至0xFFFF，每组称为平面（Plane），而每平面拥有65536个码位，共1114112个。然而目前只用了少数平面。[UTF](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=101264050&ss_c=ssc.citiao.link)-8、[UTF-16](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=11006007&ss_c=ssc.citiao.link)、[UTF-32](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=68495805&ss_c=ssc.citiao.link)都是将[数字转换](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=69265523&ss_c=ssc.citiao.link)到程序数据的编码方案。

[通用字符集](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=8804219&ss_c=ssc.citiao.link)（Universal Character Set, UCS）是由ISO制定的ISO 10646（或称ISO/IEC 10646）标准所定义的标准[字符集](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=291629&ss_c=ssc.citiao.link)。UCS-2用两个字节编码，UCS-4用4个字节编码。

历史上存在两个独立的尝试创立单一字符集的组织，即[国际标准化组织](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=264072&ss_c=ssc.citiao.link)（ISO）和多语言软件制造商组成的统一码联盟。前者开发的 ISO/IEC10646 项目，后者开发的 统一码项目。因此最初制定了不同的标准。

1991年前后，两个项目的参与者都认识到，世界不需要两个不兼容的字符集。于是，它们开始合并双方的工作成果，并为创立一个单一编码表而协同工作。从Unicode 2.0开始，Unicode采用了与ISO 10646-1相同的[字库](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=2931979&ss_c=ssc.citiao.link)和字码；ISO也承诺，ISO 10646将不会替超出[U+](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=99804179&ss_c=ssc.citiao.link)10FFFF的UCS-4编码赋值，以使得两者保持一致。两个项目仍都存在，并独立地公布各自的标准。但 统一码联盟和ISO/IEC JTC1/SC2都同意保持两者标准的码表兼容，并紧密地共同调整任何未来的扩展。在发布的时候，Unicode一般都会采用有关字码最常见的字型，但ISO 10646一般都尽可能采用Century字型。

UCS-4根据最高位为0的最高字节分成2 7=128个group。每个group再根据次高字节分为256个平面（plane）。每个平面根据第3个字节分为256行 （row），每行有256个码位（cell）。group 0的平面0被称作BMP（Basic Multilingual Plane）。如果UCS-4的前两个字节为全零，那么将UCS-4的BMP去掉前面的两个零字节就得到了UCS-2。每个平面有2 16=65536个码位。Unicode计划使用了17个平面，一共有17×65536=1114112个码位。在Unicode 5.0.0版本中，已定义的码位只有238605个，分布在平面0、平面1、平面2、平面14、平面15、平面16。其中平面15和平面16上只是定义了两个各占65534个码位的专用区（Private Use Area），分别是0xF0000-0xFFFFD和0x100000-0x10FFFD。所谓专用区，就是保留给大家放自定义字符的区域，可以简写为PUA。

平面0也有一个专用区：0xE000-0xF8FF，有6400个码位。平面0的0xD800-0xDFFF，共2048个码位，是一个被称作代理区（[Surrogate](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=64476050&ss_c=ssc.citiao.link)）的特殊区域。代理区的目的用两个UTF-16字符表示BMP以外的字符。在介绍UTF-16编码时会介绍。

如前所述在Unicode 5.0.0版本中，238605-65534\*2-6400-2048=99089。余下的99089个已定义码位分布在平面0、平面1、平面2和平面14上，它们对应着Unicode定义的99089个字符，其中包括71226个汉字。平面0、平面1、平面2和平面14上分别定义了52080、3419、43253和337个字符。平面2的43253个字符都是汉字。平面0上定义了27973个汉字。

在Unicode中：汉字“字”对应的数字是23383（[十进制](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=3006559&ss_c=ssc.citiao.link)），十六进制表示为5B57。在Unicode中，我们有很多方式将数字23383表示成程序中的数据，包括： UTF-8、 UTF-16、 UTF-32。UTF是“Unicode Transformation Format”的缩写，可以翻译成Unicode字符集转换格式，即怎样将Unicode定义的数字转换成程序数据。

例如，“汉字”对应的数字是0x6c49和0x5b57，而编码的程序数据是：

chardata\_utf8[]={0xE6,0xB1,0x89,0xE5,0xAD,0x97};//UTF-8编码 char16\_tdata\_utf16[]={0x6C49,0x5B57};//UTF-16编码 char32\_tdata\_utf32[]={0x00006C49,0x00005B57};//UTF-32编码

这里用char、char16\_t、char32\_t分别表示无符号8位整数，无符号16位整数和无符号32位整数。UTF-8、UTF-16、UTF-32分别以char、char16\_t、char32\_t作为编码单位。（注： char16\_t 和 char32\_t 是 C++ 11 标准新增的关键字。如果你的[编译器](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=106869&ss_c=ssc.citiao.link)不支持 C++ 11 标准，请改用 unsigned short 和 unsigned long。）“汉字”的UTF-8编码需要6个字节。“汉字”的UTF-16编码需要两个char16\_t，大小是4个字节。“汉字”的UTF-32编码需要两个char32\_t，大小是8个字节。根据[字节序](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=44712328&ss_c=ssc.citiao.link)的不同，UTF-16可以被实现为UTF-16LE或UTF-16BE，UTF-32可以被实现为UTF-32LE或UTF-32BE。 [[1]](http://baike.sogou.com/v291611.htm?fromTitle=Unicode#quote1)（维基百科）

UTF-8/16

**UTF-8**

一、基本介绍

在所有字符集中，最知名的可能要数被称为ASCII的7位字符集了。它是[美国标准信息交换代码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=83398424&ss_c=ssc.citiao.link" \t "_blank)（AmericanStandardCodeforInformationInterchange）的缩写,为美国英语通信所设计。它由128个字符组成，包括大小写字母、数字0-9、标点符号、非打印字符（[换行符](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=72490076&ss_c=ssc.citiao.link" \t "_blank)、[制表符](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=268913&ss_c=ssc.citiao.link)等4个）以及控制字符（退格、响铃等）组成。

但是，由于他是针对英语设计的，当处理带有音调标号（形如汉语的拼音）的欧洲文字时就会出现问题。因此，创建出了一些包括255个字符的由ASCII扩展的字符集。其中有一种通常被称为IBM字符集，它把值为128-255之间的字符用于画图和画线，以及一些特殊的欧洲字符。另一种8位字符集是ISO8859-1Latin1，也简称为ISOLatin-1。它把位于128-255之间的字符用于拉丁[字母表](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=531206&ss_c=ssc.citiao.link" \t "_blank)中特殊语言字符的编码，也因此而得名。欧洲语言不是地球上的唯一语言，因此亚洲和[非洲语言](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=2009781&ss_c=ssc.citiao.link)并不能被8位字符集所支持。仅汉语字母表（或pictograms）就有80000以上个字符。但是把汉语、日语和[越南语](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=480824&ss_c=ssc.citiao.link" \t "_blank)的一些相似的字符结合起来，在不同的语言里，使不同的字符代表不同的字，这样只用2个字节就可以编码地球上几乎所有地区的文字。因此，创建了UNICODE编码。它通过增加一个高字节对ISOLatin-1字符集进行扩展，当这些高字节位为0时，低字节就是ISOLatin-1字符。UNICODE支持欧洲、非洲、中东、亚洲（包括统一标准的东亚象形汉字和韩国象形文字）。但是，UNICODE并没有提供对诸如Braille,Cherokee,Ethiopic,Khmer,Mongolian,Hmong,TaiLu,TaiMau文字的支持。同时它也不支持如Ahom,Akkadian,Aramaic,BabylonianCuneiform,Balti,Brahmi,Etruscan,Hittite,Javanese,Numidian,OldPersianCuneiform,Syrian之类的古老文字。

事实证明，对可以用ASCII表示的字符使用UNICODE并不高效，因为UNICODE比ASCII占用大一倍的空间，而对ASCII来说高字节的0对他毫无用处。为了解决这个问题，就出现了一些中间格式的字符集，他们被称为通用转换格式，即UTF（UniversalTransformationFormat）。常见的UTF格式有：UTF-7,UTF-7.5,UTF-8,[UTF-16](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=11006007&ss_c=ssc.citiao.link),以及[UTF-32](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=68495805&ss_c=ssc.citiao.link" \t "_blank)。 [[2]](http://baike.sogou.com/v291611.htm?fromTitle=Unicode#quote1)（MariusBancila）

二、优点

UTF-8编码可以通过屏蔽位和移位操作快速读写。字符串比较时strcmp()和wcscmp()的返回结果相同，因此使排序变得更加容易。字节FF和FE在UTF-8编码中永远不会出现，因此他们可以用来表明UTF-16或UTF-32文本（见BOM）UTF-8是[字节顺序](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=56845612&ss_c=ssc.citiao.link" \t "_blank)无关的。它的字节顺序在所有系统中都是一样的，因此它实际上并不需要BOM。

三、缺点

你无法从UNICODE字符数判断出UTF-8文本的字节数，因为UTF-8是一种变长编码它需要用2个字节编码那些用[扩展ASCII](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=11005095&ss_c=ssc.citiao.link" \t "_blank)[字符集](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=291629&ss_c=ssc.citiao.link)只需1个字节的字符ISOLatin-1是UNICODE的子集，但不是UTF-8的子集8位字符的UTF-8编码会被email网关过滤，因为internet信息最初设计为7位[ASCII码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=87696&ss_c=ssc.citiao.link" \t "_blank)。因此产生了UTF-7编码。UTF-8在它的表示中使用值100xxxxx的几率超过50%，而现存的实现如ISO2022，4873，6429，和8859系统，会把它错认为是C1控制码。因此产生了UTF-7.5编码。

四、联系

UTF-8以字节为单位对Unicode进行编码。从Unicode到UTF-8的[编码方式](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=50014&ss_c=ssc.citiao.link" \t "_blank)如下：

|  |  |
| --- | --- |
| **Unicode编码(十六进制)** | **UTF-8 字节流(二进制)** |
| 000000-00007F | 0xxxxxxx |
| 000080-0007FF | 110xxxxx 10xxxxxx |
| 000800-00FFFF | 1110xxxx 10xxxxxx 10xxxxxx |
| 010000-10FFFF | 11110xxx10xxxxxx10xxxxxx10xxxxxx |

UTF-8的特点是对不同范围的字符使用不同长度的编码。对于0x00-0x7F之间的字符，UTF-8编码与 [ASCII编码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7770132&ss_c=ssc.citiao.link)完全相同。UTF-8编码的最大长度是6个字节。从上表可以看出，6字节模板有31个x，即可以容纳31位二进制数字。Unicode的最大码位0x7FFFFFFF也只有31位。

例1：“汉”字的Unicode编码是0x6C49。0x6C49在0x0800-0xFFFF之间，使用用3字节模板了：1110xxxx 10xxxxxx 10xxxxxx。将0x6C49写成二进制是：0110 1100 0100 1001， 用这个[比特流](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=4782438&ss_c=ssc.citiao.link)依次代替模板中的x，得到：11100110 10110001 10001001，即E6 B1 89。

例2：Unicode编码0x20C30在0x010000-0x10FFFF之间，使用用4字节模板了：11110xxx 10xxxxxx 10xxxxxx 10xxxxxx。将0x20C30写成21位二进制数字（不足21位就在前面补0）：0 0010 0000 1100 0011 0000，用这个比特流依次代替模板中的x，得到：11110000 10100000 10110000 10110000，即F0 A0 B0 B0。

**UTF-16**

**一、基本内容**

UTF-16是Unicode的其中一个使用方式。 UTF是 Unicode Translation Format，即把Unicode转做某种格式的意思。它定义于ISO/IEC 10646-1的附录Q，而RFC2781也定义了相似的做法。在Unicode基本多文种平面定义的字符（无论是拉丁字母、汉字或其他文字或符号），一律使用2字节储存。而在辅助平面定义的字符，会以代理对（surrogate pair）的形式，以两个2字节的值来储存。　　UTF-16比起UTF-8，好处在于大部分字符都以固定长度的字节 (2字节) 储存，但UTF-16却无法兼容于[ASCII编码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7770132&ss_c=ssc.citiao.link" \t "_blank)。c#中默认的就是UTF-16，所以在处理c#字符串的时候只能是bety,stream等方式去处理

二、有关模式

UTF-16的大尾序和小尾序储存形式都在用。一般来说，以Macintosh制作或储存的文字使用大尾序格式，以Microsoft或Linux制作或储存的文字使用小尾序格式。

为了弄清楚UTF-16文件的大小尾序，在UTF-16文件的开首，都会放置一个U+FEFF字符作为Byte Order Mark（UTF-16LE以FF FE代表，UTF-16BE以FE FF代表），以显示这个文字档案是以UTF-16编码，其中U+FEFF字符在UNICODE中代表的意义是ZERO WIDTH NO-BREAK SPACE，顾名思义，它是个没有宽度也没有断字的空白。

三、**详细讲解**

UTF-16BE(bigendian),比较好理解的,俗称大头比如说char'a',ascii为0x61,那么它的utf-8,则为[0x61],但utf-16是16位的,所以为[0x00,0x61]再说UTF-16LE(littleendian),俗称小头,这个是比较常用的

还是char'a',它的代码却反过来:[0x61,0x00],据说是为了提高速度而迎合CPU的胃口,CPU就是这到倒着吃数据的,这里面有汇编的知识，UTF-16,要从代码里自动判断一个文件到底是UTF-16LE还是BE,对于单纯的英文字符来说还比较好办,但要有[特殊字符](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=269089&ss_c=ssc.citiao.link" \t "_blank),[图形符号](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7954011&ss_c=ssc.citiao.link),汉字,法文,俄语,[火星语](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=25602&ss_c=ssc.citiao.link)之类的话,相信各位都很头痛吧,所以,unicode组织引入了BOM的概念,即byteordermark,顾名思义,就是表名这个文件到底是LE还是BE的,其方法就是,在UTF-16文件的头2个字节里做个标记:LE[0xFF,0xFE],BE[0xFE,0xFF]

理解了这个后,在java里遇到utf-16还是会遇到麻烦,因为要在文件里面单独判断头2个再字节是很不流畅的。

如果这个UTF-16文件里带有BOM的话,charset就用"UTF-16",java会自动根据BOM判断LE还是BE,如果你在这里指定了"UTF-16LE"或"UTF-16BE"的话,猜错了会生成乱七八糟的文件,哪怕猜对了,java也会把头2个字节当成文本输出给你而不会略过去,因为[FFFE]或[FEFF]这2个代码没有内容，另外,UTF-8也有BOM的,[0xEF,0xBB,0xBF],但可有可无,但用windows的notepad另存为时会自动帮你加上这个,而很多非windows平台的UTF8文件又没有这个BOM

四、错误例子

1.文件A,UTF16格式,带BOMLE,

InputStreamReaderreader=newInputStreamReader(fin,"utf-16le")会多输出一个"?"在第一个字节,原因:java没有把头2位当成BOM

2.文件A,UTF16格式,带BOMLE,

InputStreamReaderreader=newInputStreamReader(fin,"utf-16be")

会出乱码,原因:字节的高低位弄反了,'a'在文件里[0x61,0x00],但java以为'a'应该是[0x000x61]

3.文件A,UTF16格式,带BOMBE,

InputStreamReaderreader=newInputStreamReader(fin,"utf-16le")

会出乱码,原因:字节的高低位弄反了,'a'在文件里[0x00,0x61],但java以为'a'应该是[0x610x00]

4.文件A,UTF16格式,带BOMBE,

InputStreamReaderreader=newInputStreamReader(fin,"utf-16be")会多输出一个"?"在第一个字节,原因:java没有把头2位当成BOM

5.文件A,UTF16格式,LE不带BOM,

InputStreamReaderreader=newInputStreamReader(fin,"utf-16")会出乱码,因为utf-16对于java来说,默认为be(1.6JDK,以后的说不准)但windows的notepad打开正常,因为notepad默认为le,--#

五、方式

UTF-16编码以16位[无符号整数](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=74070567&ss_c=ssc.citiao.link" \t "_blank)为单位。我们把Unicode编码记作U。编码规则如下：

如果U<0x10000，U的UTF-16编码就是U对应的16位无符号整数（为书写简便，下文将16位无符号整数记作WORD）。

如果U≥0x10000，我们先计算U'=U-0x10000，然后将U'写成二进制形式：yyyy yyyy yyxx xxxx xxxx，U的UTF-16编码（二进制）就是：110110yyyyyyyyyy 110111xxxxxxxxxx。

为什么U'可以被写成20个二进制位？Unicode的最大码位是0x10ffff，减去0x10000后，U'的最大值是0xfffff，所以肯定可以用20个 二进制位表示。例如：Unicode编码0x20C30，减去0x10000后，得到0x10C30，写成二进制是：0001 0000 1100 0011 0000。用前10位依次替代模板中的y，用后10位依次替代模板中的x，就得到：1101100001000011 1101110000110000，即0xD843 0xDC30。

按照上述规则，Unicode编码0x10000-0x10FFFF的UTF-16编码有两个WORD，第一个WORD的高6位是110110，第二个WORD的高6位是110111。可见，第一个WORD的取值范围（二进制）是11011000 00000000到11011011 11111111，即0xD800-0xDBFF。第二个WORD的取值范围（二进制）是11011100 00000000到11011111 11111111，即0xDC00-0xDFFF。

为了将一个WORD的UTF-16编码与两个WORD的UTF-16编码区分开来，Unicode编码的设计者将0xD800-0xDFFF保留下来，并称为代理区（Surrogate）：

|  |  |  |
| --- | --- | --- |
| D800－DB7F | High Surrogates | 高位替代 |
| DB80－DBFF | High Private Use Surrogates | 高位专用替代 |
| DC00－DFFF | Low Surrogates | 低位替代 |

高位替代就是指这个范围的码位是两个WORD的UTF-16编码的第一个WORD。低位替代就是指这个范围的码位是两个WORD的UTF-16编码的第二个WORD。那么，高位专用替代是什么意思？我们来解答这个问题，顺便看看怎么由UTF-16编码推导Unicode编码。

如果一个字符的UTF-16编码的第一个WORD在0xDB80到0xDBFF之间，那么它的Unicode编码在什么范围内？我们知道第二个WORD的取值范围是0xDC00-0xDFFF，所以这个字符的UTF-16编码范围应该是0xDB80 0xDC00到0xDBFF 0xDFFF。我们将这个范围写成二进制：

1101101110000000 11011100 00000000 - 1101101111111111 1101111111111111

按照编码的相反步骤，取出高低WORD的后10位，并拼在一起，得到

1110 0000 0000 0000 0000 - 1111 1111 1111 1111 1111即0xe0000-0xfffff，按照编码的相反步骤再加上0x10000，得到0xf0000-0x10ffff。这就是UTF-16编码的第一个WORD在0xdb80到0xdbff之间的Unicode编码范围，即平面15和平面16。因为Unicode标准将平面15和平面16都作为专用区，所以0xDB80到0xDBFF之间的保留码位被称作高位专用替代 [[1]](http://baike.sogou.com/v291611.htm?fromTitle=Unicode#quote1)（维基百科） 。

GB2312

一、简介

信息交换用汉字编码字符集和汉字输入编码之间的关系是，根据不同的汉字输入方法，通过必要的设备向计算机输入汉字的编码，计算机接收之后，先转换成信息交换用汉字编码字符，这时计算机就可以识别并进行处理；汉字输出是先把[机内码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7686316&ss_c=ssc.citiao.link)转成汉字编码，再发送到输出设备。

二、收录

GB 2312标准共收录6763个汉字，其中一级汉字3755个，二级汉字3008个；同时，GB 2312收录了包括拉丁字母、希腊字母、日文平假名及[片假名](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=255551&ss_c=ssc.citiao.link" \t "_blank)字母、俄语[西里尔字母](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=424448&ss_c=ssc.citiao.link)在内的682个全角字符。

GB 2312的出现，基本满足了汉字的计算机处理需要，它所收录的汉字已经覆盖中国大陆99.75%的使用频率。

对于人名、古汉语等方面出现的罕用字，GB 2312不能处理，这导致了后来GBK及GB 18030汉字字符集的出现。

三、编码表

GB2312简体中文编码表

code +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F

A1A0 　 、 。 · ˉ ˇ ¨ 〃 々 — ～ ‖ … ‘ ’

A1B0 “ ” 〔 〕 〈 〉 《 》 「 」 『 』 〖 〗 【 】

A1C0 ± × ÷ ∶ ∧ ∨ ∑ ∏ ∪ ∩ ∈ ∷ √ ⊥ ∥ ∠

A1D0 ⌒ ⊙ ∫ ∮ ≡ ≌ ≈ ∽ ∝ ≠ ≮ ≯ ≤ ≥ ∞ ∵

A1E0 ∴ ♂ ♀ ° ′ ″ ℃ $ ¤ ￠ ￡ ‰ § № ☆ ★

A1F0 ○ ● ◎ ◇ ◆ □ ■ △ ▲ ※ → ← ↑ ↓ 〓

code +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F

A2A0 Ⅰ Ⅱ Ⅲ Ⅳ Ⅴ Ⅵ Ⅶ Ⅷ Ⅸ Ⅹ ? ? ? ? ?

A2B0 ? ⒈ ⒉ ⒊ ⒋ ⒌ ⒍ ⒎ ⒏ ⒐ ⒑ ⒒ ⒓ ⒔ ⒕ ⒖

A2C0 ⒗ ⒘ ⒙ ⒚ ⒛ ⑴ ⑵ ⑶ ⑷ ⑸ ⑹ ⑺ ⑻ ⑼ ⑽ ⑾

A2D0 ⑿ ⒀ ⒁ ⒂ ⒃ ⒄ ⒅ ⒆ ⒇ ① ② ③ ④ ⑤ ⑥ ⑦

A2E0 ⑧ ⑨ ⑩ ? ? 一 二 三 四 五 六 七 八 九 十 ?

A2F0 ? Ⅰ Ⅱ Ⅲ Ⅳ Ⅴ Ⅵ Ⅶ Ⅷ Ⅸ Ⅹ Ⅺ Ⅻ ? ?

code +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +A +B +C +D +E +F

A3A0 ！ " # ￥ % & ' （ ） \* + ， － ． /

A3C0 @ A B C D E F G H I J K L M N O

A3D0 P Q R S T U V W X Y Z [ \ ] ^ \_

A3E0 ` a b c d e f g h i j k l m n o

A3F0 p q r s t u v w x y z { | }

……

四、分区表示

GB 2312中对所收汉字进行了“分区”处理，每区含有94个汉字/符号。这种表示方式也称为[区位码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=291660&ss_c=ssc.citiao.link)。

01-09区为特殊符号。

16-55区为一级汉字，按拼音排序。

56-87区为二级汉字，按部首/笔画排序。

10-15区及88-94区则未有编码。

举例来说，“啊”字是GB2312之中的第一个汉字，它的区位码就是1601。

五、字节结构

在使用GB2312的程序中，通常采用EUC储存方法，以便兼容于ASCII。浏览器编码表上的“GB2312”，通常都是指“EUC-CN”表示法。

每个汉字及符号以两个字节来表示。第一个字节称为“高位字节”（也称“区字节）”，第二个字节称为“低位字节”（也称“位字节”）。

“高位字节”使用了0xA1-0xF7(把01-87区的区号加上0xA0)，“低位字节”使用了0xA1-0xFE(把01-94加上 0xA0)。 由于一级汉字从16区起始，汉字区的“高位字节”的范围是0xB0-0xF7，“低位字节”的范围是0xA1-0xFE，占用的码位是 72\*94=6768。其中有5个空位是D7FA-D7FE。

例如“啊”字在大多数程序中，会以两个字节，0xB0（第一个字节） 0xA1（第二个字节）储存。区位码=区字节+位字节（与区位码对比：0xB0=0xA0+16,0xA1=0xA0+1）。

Big5

一、产生背景

在Big5码诞生后，大部分台湾的[电脑软件](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=4319238)都使用了Big5码，加上后来[倚天中文系统](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=4257268)的高度普及，使后来的微软Windows 3.x等亦予以采用。虽然后来台湾还有各种想要取代Big5码，像是倚天中文系统所推行的倚天码、台湾地区中文电脑公会所推动的公会码等，但是由于Big5字码已沿用多年，因此在习惯不易改变的情况下，始终无法成为主流字码。而台湾以后发展的“[国家标准中文交换码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=66243770)”（CNS 11643）由于先天所限，必须使用3字节来表示一个汉字，与现行[英语软件](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=46615122)欠缺相容，所以普及率远远不及Big5码。 [[3]](http://baike.sogou.com/v291611.htm?fromTitle=Unicode#quote1)（朱邦复）

二、初创

“[五大码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=828398)”（Big5)是在1984年由台湾13家厂商与台湾地区[财团法人](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=70802117" \t "_blank)[信息工业](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=74320188)策进会为五大中文套装软件（[宏碁](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=95850)、神通、[佳佳](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=64598952)、零壹、大众）所设计的中文[内码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=8404524&ss_c=ssc.citiao.link" \t "_blank)，所以就称为Big5中文内码，虽然五大套装软件并没有成功，但Big5码却深远地影响中文电脑内码，直至今日。“五大码”的英文名称“Big5”后来被人按英文字序译回中文，以致现在有“五大码”和“[大五码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=828398)”两个中文名称。

Big5码的产生，是因为当时台湾不同厂商各自推出不同的编码，如IBM 5550、[王安](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=64622189" \t "_blank)码等，彼此不能兼容；另一方面，台湾当时尚未推出官方的[汉字编码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=3554862)，而中国内地所推行的GB 2312编码，亦未有收录繁体字。同时，这个计划对于以台湾为核心的亚洲汉字圈也产生了久远的影响。

三、发展

操作系统和应用程式（如[苹果电脑](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=135570" \t "_blank) Mac OS X 和以 Cocoa API 撰写之程式、Microsoft Windows 2000及之后版本、Microsoft Office 2000及之后版本、[Mozilla浏览器](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=63417153&ss_c=ssc.citiao.link" \t "_blank)、Internet Explorer浏览器、Java 语言等等)，已改用Unicode编码。可惜现时仍有一些旧的软件（如[VisualBasic](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=106118936&ss_c=ssc.citiao.link" \t "_blank) 6、部分Telnet或BBS软件），未能支援Unicode编码，故相信Big5缺字的问题仍会困扰用户一段时间，直至所有程式都能改用Unicode为止。

注释：大五码普遍被认为包含13,053字，但在计算0xA259-0xA261的[度量衡](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=166108" \t "_blank)单位用字 (兙兛兞兝兡兣嗧瓩糎） ，再减去重收了两次的“兀”（0xC94A)和“嗀”（0xDDFC)后，应为13,060字。

四、应用延伸

**非官方Big5延伸**

由于Big5码内的万多个字，只是根据台湾地区“教育部”颁布的《常用国字标准字体表》、[《次常用国字标准字体表》](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=64112214)等用字汇编而成，并没有考虑社会上流通的人名、地名用字、方言用字、化学及[生物专业](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=681071)等用字，亦没有放入[日语平假名](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=9043340)及[片假名](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=255551)字母。所以在市面上支援Big5码的软件，有不少都自行在原本的编码外，添加一些符号及用字。

**倚天Big5延伸**

在[倚天中文系统](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=4257268)中，为与IBM5550码相容，他们在Big5码添加了以下的字符：

在0xA3C0-0xA3E0，添加了33个[控制字符](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=10667750" \t "_blank)的图象。

罕用符号区。在0xC6A1-0xC875，添加了圆形1-10、括号1-10、小[罗马字](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=3836330" \t "_blank)i-ix等章节符号、一些部首及笔划结构，日语平假名、片假名及俄语使用的[西里尔字母](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=424448)。

在0xF9D6-0xF9FE，添加了7个倚天扩充字：碁、銹、恒、裏、墻、粧、嫺和34个表格符号。

这个延伸有时被称为Big5-Eten。由于倚天中文系统是Windows 95推出之前[市场占有率](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=742879)最高的中文系统，此延伸是各种非官方延伸当中最重要的一个。

在后期版本的倚天中文系统中，更加入了一些图案和简体中文字，但未被广泛接受。

**Code Page 950**Windows使用的 Code Page 950 (系引用IBM Big 5码的编码页号Code Page 950，简称 CP950) 之中，只添加了上述0xF9D6-0xF9FE的倚天扩充字及表格符号，并没有加入日文假名字母等其他延伸。

在Windows ME之中，微软首度在0xA3E1加入了欧元(?符号，之后所有 Windows 版本的 Code Page 950 也都有这个符号。

**中国海字集**“中国海字集”是中国海公司所出品的繁体汉字造字档。由于它包括了不少社会上常见的用字、日文假名、和字等，加上曾与Office 97中文版一并发售，所以比起其他官方Big5延伸，更被台湾民众所接受。香港部份BBS网络在[香港增补字符集](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=57411577)未出现之前，一度以中国海字集为标准。

**日和字集**“日和字集”乃一香港个人开发的造字档，以兼容香港增补字符集为卖点，为字集中仍没函盖的[日本汉字](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=95615)和日该国字作增补，并附有仓颉、速成等输入法作辅助。

**Unicode补完计划**“Unicode补完计划”前称“BIG5 Extension”，通过修改Microsoft Windows及Mozilla的编码表，从而用户能在网上传递及交换文字。有鉴于“中国海字集”的成功，“Unicode补完计划”第二版采用了“中国海字集”原有的造字，再加上“中国海字集”所欠的部分简体中文字及[香港粤语](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=85416684)用字，建成一个能在Big5及Unicode之间转换的编码表；该计划目前已推出了64位测试版。

**官方Big5延伸**

**台湾“教育部”造字档**

台湾当局“教育部”有它本身的一套造字档，主要给部门内使用，亦有于“教育部”的[网上字典](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=71532324" \t "_blank)使用。

**台湾“农委会”常用中文外字集**

台湾当局“农业委员会”制定了一套有133个汉字的造字档，其中有84个是鱼字部汉字、7个是鸟字部汉字。

**Big5+**

中文数位化技术推广委员会(中推会）在1997年推出Big5+，使用了两万多码位，纳入了Unicode 1.1下所有汉字。由于编码使用到的范围超过原先Big5定义（Big5+使用了高字节0x81-0xFE，低字节0x40-0x7E、0x80-0xFE)，无法安装在Microsoft Windows上，现几乎无人使用。

**Big-5E**

为了使Microsoft Windows使用者可以使用造字档，“[行政院](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=432646)”委托中

BIG-5

[行政单位](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7697078)

**Big5-2003**

鉴于Big5不是一个官方标准，中推会决定编制一个Big5的定义，并把它放到官方编码CNS 11643的附录里，正式成为官方标准的一部分。在Big5-2003之中，收录了所有在1984年Big5编码的所有字符，Big5-2003没有收录[行列输入法](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=76651547)[特殊符号](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=493633)及0xC7F3-0xC875的俄语西里尔字母，理由是以CNS 11643没有这些字符。除此之外，所有倚天延伸全部收录。相对于Big5-2003，最早没有加上任何延伸的Big5则被称为Big5-1984。

**香港增补字符集**

香港增补字符集：是[香港政府](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7688435)基于繁体中文[电脑操作](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=547286)环境中最流行的大五码（Big-5）之上扩展的[字符集](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=291629" \t "_blank)标准，是现时香港的中文资讯交换[内码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=8404524)标准。香港增补字符集以前称为《政府通用[字库](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=2931979&ss_c=ssc.citiao.link)》（GovernmentCommonCharacterSet,简称GCCS），本来只是香港政府内部统一使用的造字档，有三千多字。但由于香港电脑业界不断要求政府迎合本地需要，提出官方的字符集方案，以便与政府进行文件来往，于是香港政府便在1995年把这个内部使用的标准公开。到了1999年，此字集增加到四千多字，并改为现名。此字符集由中文界面咨询委员会管理，仍在不断扩编之中。最新版本为2005年5月推出的HKSCS-2004，收录4,941个字符。

字符分类：在HKSCS-2004版本，汉字字符共4500个，其中3353字可在大型的字典（如[《汉语大字典》](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=102040944)）中查到，包括[简化字](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=81830)、[异体字](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=43018)、[日语汉字](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7882090)等。其余在各大[中文字典](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=75915497)中查不到的字中，有粤语[方言字](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=6669694)（有些可在[方言字典](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=10604634)及学术著作中查到）、人名、公司名、地方名、变形部首、附形、讹字。有些字来自入境事务处、公司注册处、税务局、地政总署。

早期的倚天中文系统、国乔中文系统等对造字缺乏管理，而又没有文字专家的审定，因此当时造字很是混乱，有些甚至可能只是临时使用的“错字”（寻遍各大字典、专书也查不到的字，也作幽灵汉字）；又有同一字有系统区及造字区两个码位，有些[联绵词](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7756387)只收其一不收其二；这个问题带到了政府通用字库和香港增补字符集中，字集因要反向兼容而跳过了一些码位。

各类符号共441个，有汉字笔形、[汉语拼音字母](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7599864)、[国际音标](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=180720)符号、汉字符件、画表符号、日本平假名、片假名等。

香港增补字符集在2005年才有画数、部首、粤音等资料给用户参考，还说明方便检索，而非作为规范标准。（汉字的部首在不同的字典中，归部也不尽相同）

编码和Big-5的关系：香港增补字符集当初因为是补充Big-5的收字不足，使用其外字区而发展的，所以受制于Big-5的编码架构，外字的总数最多只能到6217个（每区块157字，有39区块半）。除去已用码位，剩下千余个码位，其中有部分会保留给用户造字。

Big-5原来的编码，只有汉字、标点、[注音符号](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=729011)等字符及少数图形，后来经过台湾厂商的增收，多了7个“倚天字”（即碁、銹、裏、墻、恒、粧、嫺）及日文的假名，最后这批字符又被香港增补字符集收入。

香港增补字符集所使用的Big-5的外字区分几个区段：

“造字区一”（FA40—FEFE)：早期的GCCS字符集已经填满这一段。

“造字区二”（C6A1—C8FE)：倚天用了这段来放日文假名等符号。这些符号在HKSCS1999年的版本被收纳。

“造字区三”（8140—A0FE)：香港增补字符集把这段开头的（8140—84FE)保留给用户，新增的字符只用其余的码位。“厂商造字区”（F9D6—F9FE)：这段开始的七个码位用来存放裏、恒等“倚天字”，之后的码位被微软的繁体中文Windows用来存放[制表符](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=268913)号。后来HKSCS1999年版本将之全部收纳。

可是一般提及HKSCS的文件，包括来自香港政府的，都没有注明HKSCS以外的一般繁体字编码（即是Big-5本身）使用哪个版本。Big-5在2003年前就只有一个版本，不会造成混淆，但HKSCS-2004的文件仍没有指定Big-5部份是2003年之后还是之前的版本，虽然到目前为止并没有任何系统使用Big5-2003。

五、基本结构

Big5码是一套双字节字符集，使用了双八码储存方法，以两个字节来安放一个字。第一个字节称为“高位字节”，第二个字节称为“低位字节”。“高位字节”使用了0x81-0xFE，“低位字节”使用了0x40-0x7E，及0xA1-0xFE。在Big5的分区中：

0x8140-0xA0FE 保留给使用者自定义字符（造字区）

0xA140-0xA3BF 标点符号、[希腊字母](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=437970)及特殊符号，包括在0xA259-0xA261，安放了[双音节](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7633173)度量衡单位用字：兙兛兞兝兡兣嗧瓩糎。

0xA3C0-0xA3FE 保留。此区没有开放作造字区用。

0xA440-0xC67E [常用汉字](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7801281)，先按笔划再按部首排序。

值得留意的是，Big5重复地收录了两个相同的字：“兀、兀”（0xA461及0xC94A)、“嗀、嗀”(0xDCD1及0xDDFC)。

六、存在问题

由于各厂商及政府推出的Big5延伸，彼此互不兼容，造成[乱码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=262888&ss_c=ssc.citiao.link" \t "_blank)问题。鉴于Unicode能正确地处理七万多个汉字，近年的操作系统和应用程序（如[苹果电脑](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=135570" \t "_blank)Mac OS X和以CocoaAPI撰写之程序、MicrosoftWindows 2000及之后版本、Microsoft [Office2000](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=6988716)及之后版本、[Mozilla浏览器](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=63417153&ss_c=ssc.citiao.link)、Internet Explorer浏览器、Java语言等等），已改用Unicode编码。可惜现时仍有一些旧的软件（如VisualBasic6、部分Telnet或BBS软件），未能支持Unicode编码，故相信Big5缺字的问题仍会困扰用户一段时间，直至所有程序都能改用Unicode为止。

**冲码问题**

因为低位[元字符](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=57356587)中包含了编程语言、[shell](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=267368)、script 中，字串或命令常会用到的[特殊字符](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=269089)，例如0x5C “\”、0x7C “|”等。“\” 在许多用途的字串中是当作转义符号又称为跳脱字符，例如 \n（换行）、\r（归位）、\t（tab）、\\（\本身符号）、\"（[引号](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=461966&ss_c=ssc.citiao.link)）等等。而 “|” 在[UNIX操作系统](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=5436069)中大多当作命令管线的使用，如 "ls -la | more" 等等。如果在字串中有这些特殊的[转义字符](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=262922)，会

BIG-5

被程式或[直译器](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=584648)解释为特殊用途。但是因为是中文的原因，故无法正确解释为上面所述的行为，因此程式可能会忽略此转义符号或是中断执行。若此，就违反了使用者本来要当成中文字符一部份使用的本意。

在[常用字](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7547535)如“功”（0xA55C)、“许”（0xB35C)、“盖”（0xBB5C)、“育”（0xA87C)中时常出现，造成了许多软件无法正确处理以Big5编码的字串或文件。这个问题被戏谑性地人名化，称为“[许功](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=73899658)盖”或“许盖功”（这三个字都有这种问题）。但是额外的困扰是，有些输出功能并不会把“\”当作特殊字符看待，所以有些程式或网页就会错误地常常出现在“许功盖”这些字后面多了“\”。

**私人造字**

在[倚天中文系统](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=4257268)，以及后来的Windows 3.1、95及98中，定义了四个私人造字区范围：0xFA40-0xFEFE、0x8E40-0xA0FE、0x8140-0x8DFE、0xC6A1-0xC8FE。私人造字区的原意，是供使用者加入本来在编码表中缺少的字符，但当每个使用者都在不同的地方加上不同的字符后，当交换资料时，对方便难以知道某一个编码究竟想表达什么字。