**编**

**码**

**集**

学 院： 计算机科学学院

专业班级：软工11502

学生姓名： 许贝

指导老师： 向华

撰写日期：2017年9月10日

对几个编码集的认识

摘 要

本文介绍了几个字符集的名次解释，及部分的发展历史，优缺点，举了少部分的例子说明。

**关键词：**GB2312；big5；UTF\_8；UTF\_16；Unicode

**目录**

[第1章 引言 2](#_Toc492844664)

[第2章 对字符集的认识 2](#_Toc492844665)

[一、GB2312 2](#_Toc492844666)

[（一）名词解释 2](#_Toc492844667)

[（二）字节结构 2](#_Toc492844668)

[（三）字模集 2](#_Toc492844669)

[二、big 5 2](#_Toc492844670)

[（一）名词解释 3](#_Toc492844671)

[（二）产生 3](#_Toc492844672)

[（三）字节结构 3](#_Toc492844673)

[（四）影响与发展 3](#_Toc492844674)

[三、UTF\_8 3](#_Toc492844675)

[（一）名词解释 3](#_Toc492844676)

[（二）UTF-8的特性 3](#_Toc492844677)

[（三）UTF-8编码的优点 4](#_Toc492844678)

[（四）UTF-8编码的缺点 4](#_Toc492844679)

[四、UTF\_16 5](#_Toc492844680)

[（一）名词解释 5](#_Toc492844681)

[（二）UTF-16的编码模式 5](#_Toc492844682)

[（三）UTF-16与UCS-2的关系 5](#_Toc492844683)

[五、Unicode 5](#_Toc492844684)

[（一）名词解释 5](#_Toc492844685)

[（二）起源与发展 5](#_Toc492844686)

[（三）Unicode的编码和实现 6](#_Toc492844687)

[（四）XML和Unicode 6](#_Toc492844688)

[第三章 结论 6](#_Toc492844689)

[参考文献 7](#_Toc492844690)

# 第1章 引言

随着计算机技术的高速发展，各种字符集都慢慢的产生，并被用到各种方面，了解各种字符集的来历，优缺点，相互之间如何转换很重要。

# 第2章 对字符集的认识

## 一、GB2312

### （一）名词解释

GB 2312 或 GB 2312–80 是中华人民共和国国家标准简体中文字符集，全称《信息交换用汉字编码字符集·基本集》，又称GB0，由中国国家标准总局发布，1981年5月1日实施。GB 2312编码通行于中国大陆；新加坡等地也采用此编码。中国大陆几乎所有的中文系统和国际化的软件都支持GB 2312。

### （二）字节结构

在使用GB 2312的程序通常采用EUC储存方法，以便兼容于ASCII。这种格式称为EUC-CN。浏览器编码表上的“GB2312”就是指这种表示法。

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”，第二个字节称为“低位字节”。

“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。

### （三）字模集

GB 5007.1–85《信息交换用汉字 24x24 点阵字模集》

GB 5007.2–85《信息交换用汉字 24x24 点阵字模数据集》

GB 5199.1–85《信息交换用汉字 15x16 点阵字模集》

GB 5199.2–85《信息交换用汉字 15x16 点阵字模数据集》

GB 6345.1–86《信息交换用汉字 32x32 点阵字模集》

GB 6345.2–86《信息交换用汉字 32x32 点阵字模数据集》

GB 12034–89《信息交换用汉字 32x32 点阵仿宋体字模集及数据集》

GB 12035–89《信息交换用汉字 32x32 点阵楷体字模集及数据集》

GB 12036–89《信息交换用汉字 32x32 点阵黑体字模集及数据集》

GB 12037–89《信息交换用汉字 36x36 点阵宋体字模集及数据集》

GB 12038–89《信息交换用汉字 36x36 点阵仿宋体字模集及数据集》

GB 12039–89《信息交换用汉字 36x36 点阵楷体字模集及数据集》

GB 12040–89《信息交换用汉字 36x36 点阵黑体字模集及数据集》

GB 12041–89《信息交换用汉字 48x48 点阵宋体字模集及数据集》

GB 12042–89《信息交换用汉字 48x48 点阵仿宋体字模集及数据集》

GB 12043–89《信息交换用汉字 48x48 点阵楷体字模集及数据集》

GB 12044–89《信息交换用汉字 48x48 点阵黑体字模集及数据集》

GB/T 13443–92《信息交换用汉字 128x128 点阵楷体字模集及数据集》

GB/T 13444–92《信息交换用汉字 128x128 点阵仿宋体字模集及数据集》

GB/T 13445–92《信息交换用汉字 256x256 点阵楷体字模集及数据集》

GB/T 13446–92《信息交换用汉字 256x256 点阵仿宋体字模集及数据集》

GB/T 13844–92《图形信息交换用矢量汉字单线宋体字模集及数据集》

GB/T 13845–92《图形信息交换用矢量汉字宋体字模集及数据集》

GB/T 13846–92《图形信息交换用矢量汉字仿宋体字模集及数据集》

GB/T 13847–92《图形信息交换用矢量汉字楷体字模集及数据集》

GB/T 13848–92《图形信息交换用矢量汉字黑体字模集及数据集》

## 二、big 5

### （一）名词解释

Big5，又称为大五码或五大码，是使用繁体中文（正体中文）社区中最常用的电脑汉字字符集标准，共收录13,060个汉字。

中文码分为内码及交换码两类，Big5属中文内码，知名的中文交换码有CCCII、CNS11643。

Big5虽普及于台湾、香港与澳门等繁体中文通行区，但长期以来并非当地的国家/地区标准或官方标准，而只是业界标准。倚天中文系统、Windows繁体中文版等主要系统的字符集都是以Big5为基准，但厂商又各自增加不同的造字与造字区，派生成多种不同版本。

2003年，Big5被收录到CNS11643中文标准交换码的附录当中，获取了较正式的地位。这个最新版本被称为Big5-2003。

### （二）产生

Big5码的产生，是因为当时个人电脑没有共通的内码，导致厂商推出的中文应用软件无法推广，并且与IBM 5550、王安码等内码，彼此不能兼容；另一方面，台湾当时尚未推出中文编码标准。在这样的时空背景下，为了使台湾早日进入信息时代，所采行的一个项目；同时，这个项目对于以台湾为核心的亚洲繁体汉字圈也产生了久远的影响。

### （三）字节结构

Big5码是一套双字节字符集，使用了双八码存储方法，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”。

值得留意的是，Big5重复收录了两个相同的字：“兀、兀”（0xA461[U+5140]及0xC94A[U+FA0C]）、“嗀、嗀”（0xDCD1[U+55C0]及0xDDFC[U+FA0D]）。此外“十”、“卅”也在符号区又重复了一次，在检索系统中常会造成查询不到字。

### （四）影响与发展

自中文电脑流行后，由于很多日常用字被视为异体字而未收录。很多人，甚至电视台的字幕、报纸的用字习惯都被改变。

由于各厂商及政府推出的Big5延伸，彼此互不兼容，造成乱码问题。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Mac OS X和以Cocoa API撰写之程序、Microsoft Windows 2000及之后版本、Microsoft Office 2000及之后版本、Mozilla浏览器、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Visual Basic 6、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。

## 三、UTF\_8

### （一）名词解释

UTF-8（8-bit Unicode Transformation Format）是一种针对Unicode的可变长度字符编码，也是一种前缀码。它可以用来表示Unicode标准中的任何字符，且其编码中的第一个字节仍与ASCII兼容，这使得原来处理ASCII字符的软件无须或只须做少部分修改，即可继续使用。因此，它逐渐成为电子邮件、网页及其他存储或发送文字的应用中，优先采用的编码。

### （二）UTF-8的特性

UCS字符U+0000到U+007F（ASCII）被编码为字节0x00到0x7F（ASCII兼容），这也意味着只包含7位ASCII字符的文件在ASCII和UTF-8两种编码方式下是一样的。

所有>U+007F的UCS字符被编码为一个多个字节的串，每个字节都有标记位集。因此，ASCII字节（0x00-0x7F）不可能作为任何其他字符的一部分。

表示非ASCII字符的多字节串的第一个字节总是在0xC0到0xFD的范围里，并指出这个字符包含多少个字节。多字节串的其余字节都在0x80到0xBF范围里，这使得重新同步非常容易，并使编码无国界，且很少受丢失字节的影响。

可以编入所有可能的231个UCS代码

UTF-8编码字符理论上可以最多到6个字节长，然而16位BMP字符最多只用到3字节长。

Bigendian UCS-4字节串的排列顺序是预定的。

字节0xFE和0xFF在UTF-8编码中从未用到，同时，UTF-8以字节为编码单元，它的字节顺序在所有系统中都是一様的，没有字节序的问题，也因此它实际上并不需要BOM。

与UTF-16或其他Unicode编码相比，对于不支持Unicode和XML的系统，UTF-8更不容易造成问题。

### （三）UTF-8编码的优点

总体来说，在Unicode字符串中不可能由码点数量决定显示它所需要的长度，或者显示字符串之后在文本缓冲区中光标应该放置的位置；组合字符、变宽字体、不可打印字符和从右至左的文字都是其归因。

所以尽管在UTF-8字符串中字符数量与码点数量的关系比UTF-32更为复杂，在实际中很少会遇到有不同的情形。

更详细的说，UTF-8编码具有以下几点优点：

ASCII是UTF-8的一个子集。因为一个纯ASCII字符串也是一个合法的UTF-8字符串，所以现存的ASCII文本不需要转换。为传统的扩展ASCII字符集设计的软件通常可以不经修改或很少修改就能与UTF-8一起使用。

使用标准的面向字节的排序例程对UTF-8排序将产生与基于Unicode代码点排序相同的结果。（尽管这只有有限的有用性，因为在任何特定语言或文化下都不太可能有仍可接受的文字排列顺序。）

UTF-8和UTF-16都是可扩展标记语言文档的标准编码。所有其它编码都必须通过显式或文本声明来指定。[2]

任何面向字节的字符串搜索算法都可以用于UTF-8的数据（只要输入仅由完整的UTF-8字符组成）。但是，对于包含字符记数的正则表达式或其它结构必须小心。

UTF-8字符串可以由一个简单的算法可靠地识别出来。就是，一个字符串在任何其它编码中表现为合法的UTF-8的可能性很低，并随字符串长度增长而减小。

与UCS-2的比较：ASCII转换成UCS-2，在编码前插入一个0x0。用这些编码，会含括一些控制符，比如"或 '/'，这在UNIX和一些C函数中，将会产生严重错误。因此可以肯定，UCS-2不适合作为Unicode的外部编码，也因此诞生了UTF-8。

### （四）UTF-8编码的缺点

**1、编写不良的解析器**

一份写得很差（并且与当前标准的版本不兼容）的UTF-8解析器可能会接受一些不同的伪UTF-8表示并将它们转换到相同的Unicode输出上。这为设计用于处理八位表示的校验例程提供了一种遗漏信息的方式。

**2、不利于正则表达式检索**

正则表达式可以进行很多英文高级的模糊检索。例如，[a-h]表示a到h间所有字母。

同样GBK编码的中文也可以这样利用正则表达式，比如在只知道一个字的读音而不知道怎么写的情况下，也可用正则表达式检索，因为GBK编码是按读音排序的。只是UTF-8不是按读音排序的，所以会对正则表达式检索造成不利影响。但是这种使用方式并未考虑中文中的破音字，因此影响不大。Unicode是按部首排序的，因此在只知道一个字的部首而不知道如何发音的情况下，UTF-8可用正则表达式检索而GBK不行。

**3、其他**

与其他Unicode编码相比，特别是UTF-16，在UTF-8中ASCII字符占用的空间只有一半，可是在一些字符的UTF-8编码占用的空间就要多出1/3，特别是中文、日文和韩文（CJK）这样的方块文字。

utf8\_unicode\_ci和utf8\_general\_ci区别

在数据库系统MySQL或MariaDB中有多种字符集，其中utf8\_unicode\_ci和utf8\_general\_ci是最常用的，但是utf8\_general\_ci对某些语言的支持有一些小问题，如果可以接受，那最好使用utf8\_general\_ci，因为它速度快。否则，请使用较为精确的utf8\_unicode\_ci，不过速度会慢一些。

## 四、UTF\_16

### （一）名词解释

UTF-16是Unicode字符编码五层次模型的第三层：字符编码表（Character Encoding Form，也称为"storage format"）的一种实现方式。即把Unicode字符集的抽象码位映射为16位长的整数（即码元）的序列，用于数据存储或传递。Unicode字符的码位，需要1个或者2个16位长的码元来表示，因此这是一个变长表示。

UTF是"Unicode/UCS Transformation Format"的首字母缩写，即把Unicode字符转换为某种格式之意。UTF-16正式定义于ISO/IEC 10646-1的附录C，而RFC2781也定义了相似的做法。

### （二）UTF-16的编码模式

UTF-16的大尾序和小尾序存储形式都在用。一般来说，以Macintosh制作或存储的文字使用大尾序格式，以Microsoft或Linux制作或存储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），以显示这个文本文件是以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）UTF-16与UCS-2的关系

UTF-16可看成是UCS-2的父集。在没有辅助平面字符（surrogate code points）前，UTF-16与UCS-2所指的是同一的意思。但当引入辅助平面字符后，就称为UTF-16了。现在若有软件声称自己支持UCS-2编码，那其实是暗指它不能支持在UTF-16中超过2字节的字集。对于小于0x10000的UCS码，UTF-16编码就等于UCS码。

## 五、Unicode

### （一）名词解释

Unicode（中文：万国码、国际码、统一码、单一码）是计算机科学领域里的一项业界标准。它对世界上大部分的文字系统进行了整理、编码，使得电脑可以用更为简单的方式来呈现和处理文字。

Unicode伴随着通用字符集的标准而发展，同时也以书本的形式对外发表。Unicode至今仍在不断增修，每个新版本都加入更多新的字符。目前最新的版本为2017年6月20日公布的10.0.0，已经收录超过十万个字符（第十万个字符在2005年获采纳）。Unicode涵盖的数据除了视觉上的字形、编码方法、标准的字符编码外，还包含了字符特性，如大小写字母。

Unicode发展由非营利机构统一码联盟负责，该机构致力于让Unicode方案取代既有的字符编码方案。因为既有的方案往往空间非常有限，亦不适用于多语环境。

Unicode备受认可，并广泛地应用于电脑软件的国际化与本地化过程。有很多新科技，如可扩展置标语言(Extensible Markup Language，简称：XML)、Java编程语言以及现代的操作系统，都采用Unicode编码。

### （二）起源与发展

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 8859-1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）。

### （三）Unicode的编码和实现

大概来说，Unicode编码系统可分为编码方式和实现方式两个层次。

**10大设计原则**

在《The Unicode Standard Version 6.2 – Core Specification》[27] ，给出了Unicode的十大设计原则：

Universality：提供单一、综合的字符集，编码一切现代与大部分历史文献的字符。

Efficiency：易于处理与分析。

Characters, not glyphs：字符，而不是字形。

Semantics：字符要有良好定义的语义

Plain text：仅限于文本字符

Logical order：默认内存表示是其逻辑序

Unification：把不同语言的同一书写系统（scripts）中相同字符统一起来。

Dynamic composition：附加符号可以动态组合。

Stability：已分配的字符与语义不再改变。

Convertibility：Unicode与其他著名字符集可以精确转换。

**编码方式**

统一码的编码方式与ISO 10646的通用字符集概念相对应。目前实际应用的统一码版本对应于UCS-2，使用16位的编码空间。也就是每个字符占用2个字节。这样理论上一共最多可以表示216（即65536）个字符。基本满足各种语言的使用。实际上当前版本的统一码并未完全使用这16位编码，而是保留了大量空间以作为特殊使用或将来扩展。

**实现方式**

Unicode的实现方式不同于编码方式。一个字符的Unicode编码是确定的。但是在实际传输过程中，由于不同系统平台的设计不一定一致，以及出于节省空间的目的，对Unicode编码的实现方式有所不同。Unicode的实现方式称为Unicode转换格式（Unicode Transformation Format，简称为UTF）

### （四）XML和Unicode

XML及其子集XHTML采用UTF-8作为标准字集，理论上我们可以在各种支持XML标准的浏览器上显示任何地区文字的网页，只要电脑本身安装有合适的字体即可。可以利用&#nnn;的格式显示特定的字符。nnn代表该字符的十进制Unicode代码。如果采用十六进制代码，在编码之前加上x字符即可。但部分旧版本的浏览器可能无法识别十六进制代码。

过去电脑编码的8位标准，使每个国家都只按国家使用的字符而编定各自的编码系统；而对于部分字符系统比较复杂的语言，如越南语，又或者东亚国家的大型字符集，都不能在8位的环境下正常显示。

只是最近才有在文本中对十六进制的支持，那么旧版本的浏览器显示那些字符或许可能有问题-大概首先会遇到的一个问题只是在对于大于8位Unicode字符的显示。解决这个问题的普遍做法仍然是将其中的十六进制码转换成一个十进制码

也有一些字符集标准将一些常用的标志存放在字符内码外面，那么你可能使用象—这样的文本标志来表示一个长划（—）的情况，即使它的字符内码已经被使用，这些标准也不包含那个字符。

# 第三章 结论

本文介绍的5种编码集中GB2312支持中文，GB2312字符集 作用：国家简体中文字符集，兼容ASCII。 位数：使用2个字节表示，能表示7445个符号，包括6763个汉字，几乎覆盖所有高频率汉字。 范围：高字节从A1到F7, 低字节从A1到FE。将高字节和低字节分别加上0XA0即可得到编码。

编码 UTF-8：采用变长字节 (1 ASCII, 2 希腊字母, 3 汉字, 4 平面符号) 表示，网络传输, 即使错了一个字节，不影响其他字节，而双字节只要一个错了，其他也错了，具体如下： 如果只有一个字节则其最高二进制位为0；如果是多字节，其第一个字节从最高位开始，连续的二进制位值为1的个数决定了其编码的字节数，其余各字节均以10开头。UTF-8最多可用到6个字节。UTF-8没有字节顺序问题，所以UTF-8适合传输和通信。

UTF-16：采用2字节，Unicode中不同部分的字符都同样基于现有的标准。这是为了便于转换。Windows NT内核是Unicode（UTF-16），采用UTF-16编码在调用系统API时无需转换，处理速度也比较快。

各种字符集有各自的优缺点，在不同的环境里要按情况选择字符集编码。

# 参考文献

【1】维基百科<https://en.wikipedia.org/wiki/Wiki>

【2】百度文库https://wenku.baidu.com/view/b823dce18e9951e79b8927ed.html