字符集辨析

作者：赵伟龙

**摘要**

全文主要阐述，Unicode，big5，GB2312，UTF-8/16，ASCII编码集的创作者，发明原因，解决问题，构成方式，相互之间的兼容性以及使用的场景。

**关键字**：编码集 Unicode big5 GB2312 UTF-8 UTF-16 ASCI

**简介及发明者**：

**Unicode**：[中文](https://zh.wikipedia.org/wiki/%E4%B8%AD%E6%96%87)：万国码、国际码、统一码、单一码，Unicode发展由非营利机构[统一码联盟](https://zh.wikipedia.org/wiki/%E7%B5%B1%E4%B8%80%E7%A2%BC%E8%81%AF%E7%9B%9F)负责，该机构致力于让Unicode方案取代既有的字符编码方案。

**big5**：又称为大五码或五大码，是使用[繁体中文](https://zh.wikipedia.org/wiki/%E7%B9%81%E4%BD%93%E4%B8%AD%E6%96%87)（正体中文）社区中最常用的电脑[汉字](https://zh.wikipedia.org/wiki/%E6%B1%89%E5%AD%97)[字符集](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6%E9%9B%86)标准，共收录13,060个汉字。“大五码”（Big5）是由台湾[财团法人信息产业策进会](https://zh.wikipedia.org/wiki/%E8%B3%87%E8%A8%8A%E5%B7%A5%E6%A5%AD%E7%AD%96%E9%80%B2%E6%9C%83)为[五大中文套装软件](https://zh.wikipedia.org/wiki/%E4%BA%94%E5%A4%A7%E4%B8%AD%E6%96%87%E5%A5%97%E8%A3%9D%E8%BB%9F%E9%AB%94)所设计的中文共通内码，在1983年12月完成公告，隔年3月，信息产业策进会与[台湾](https://zh.wikipedia.org/wiki/%E8%87%BA%E7%81%A3)13家厂商签定“16位个人电脑套装软件合作开发（BIG-5）项目（[五大中文套装软件](https://zh.wikipedia.org/wiki/%E4%BA%94%E5%A4%A7%E4%B8%AD%E6%96%87%E5%A5%97%E8%A3%9D%E8%BB%9F%E9%AB%94)）”，因为此中文内码是为台湾自行制作开发之“五大中文套装软件”所设计的，所以就称为Big5中文内码。

**GB2312**：：GB 2312 或 GB 2312–80 是[中华人民共和国国家标准](https://zh.wikipedia.org/wiki/%E4%B8%AD%E5%8D%8E%E4%BA%BA%E6%B0%91%E5%85%B1%E5%92%8C%E5%9B%BD%E5%9B%BD%E5%AE%B6%E6%A0%87%E5%87%86)[简体中文](https://zh.wikipedia.org/wiki/%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87)[字符集](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6%E9%9B%86)，全称《信息交换用汉字编码字符集·基本集》，又称[GB0](https://zh.wikipedia.org/wiki/%E5%9B%BD%E5%AE%B6%E6%A0%87%E5%87%86%E4%BB%A3%E7%A0%81)，由[中国国家标准总局](https://zh.wikipedia.org/w/index.php?title=%E4%B8%AD%E5%9B%BD%E5%9B%BD%E5%AE%B6%E6%A0%87%E5%87%86%E6%80%BB%E5%B1%80&action=edit&redlink=1)发布，1981年5月1日实施。

**UTF-8**：UTF-8（8-bit Unicode Transformation Format）是一种针对[Unicode](https://zh.wikipedia.org/wiki/Unicode)的可变长度[字符编码](https://zh.wikipedia.org/wiki/%E5%AD%97%E5%85%83%E7%B7%A8%E7%A2%BC)，也是一种[前缀码](https://zh.wikipedia.org/wiki/%E5%89%8D%E7%BC%80%E7%A0%81)。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](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)仍与[ASCII](https://zh.wikipedia.org/wiki/ASCII)兼容，这使得原来处理ASCII字符的[软件](https://zh.wikipedia.org/wiki/%E8%BB%9F%E9%AB%94)无须或只须做少部分修改。

**UTF-16**: UTF-16是[Unicode](https://zh.wikipedia.org/wiki/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](https://zh.wikipedia.org/wiki/%E7%A0%81%E4%BD%8D)映射为16位长的整数（即[码元](https://zh.wikipedia.org/wiki/%E7%A0%81%E5%85%83)）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。

**ASCII**：American Standard Code for InformationInterchange，美国信息交换标准代码）是基于[拉丁字母](https://zh.wikipedia.org/wiki/%E6%8B%89%E4%B8%81%E5%AD%97%E6%AF%8D)的一套[电脑](https://zh.wikipedia.org/wiki/%E7%94%B5%E8%84%91)[编码](https://zh.wikipedia.org/wiki/%E7%BC%96%E7%A0%81)系统，是由[美国国家标准学会](https://baike.baidu.com/item/%E7%BE%8E%E5%9B%BD%E5%9B%BD%E5%AE%B6%E6%A0%87%E5%87%86%E5%AD%A6%E4%BC%9A)(American National Standard Institute , ANSI )制定的，标准的单[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)字符[编码](https://baike.baidu.com/item/%E7%BC%96%E7%A0%81)方案，用于基于[文本](https://baike.baidu.com/item/%E6%96%87%E6%9C%AC)的数据。

**发明原因：**

**Unicode**：因为既有的方案往往空间非常有限，亦不适用于[多语](https://zh.wikipedia.org/wiki/%E5%A4%9A%E8%AA%9E)环境。Unicode是为了解决传统的[字符编码](https://zh.wikipedia.org/wiki/%E5%AD%97%E5%85%83%E7%B7%A8%E7%A2%BC)方案的局限而产生的，例如[ISO 8859-1](https://zh.wikipedia.org/wiki/ISO/IEC_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](https://zh.wikipedia.org/wiki/%E6%8B%89%E4%B8%81%E5%AD%97%E6%AF%8D)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）。

**Big5**：Big5码的产生，是因为当时[个人电脑](https://zh.wikipedia.org/wiki/%E5%80%8B%E4%BA%BA%E9%9B%BB%E8%85%A6)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](https://zh.wikipedia.org/wiki/IBM_5550)、[王安码](https://zh.wikipedia.org/w/index.php?title=%E7%8E%8B%E5%AE%89%E7%A2%BC&action=edit&redlink=1)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。

**编码方式：**

**Unicode**：目前实际应用的统一码版本对应于[UCS-2](https://zh.wikipedia.org/wiki/UCS-2)，使用16[位](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83)的编码空间。也就是每个字符占用2个[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)。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。

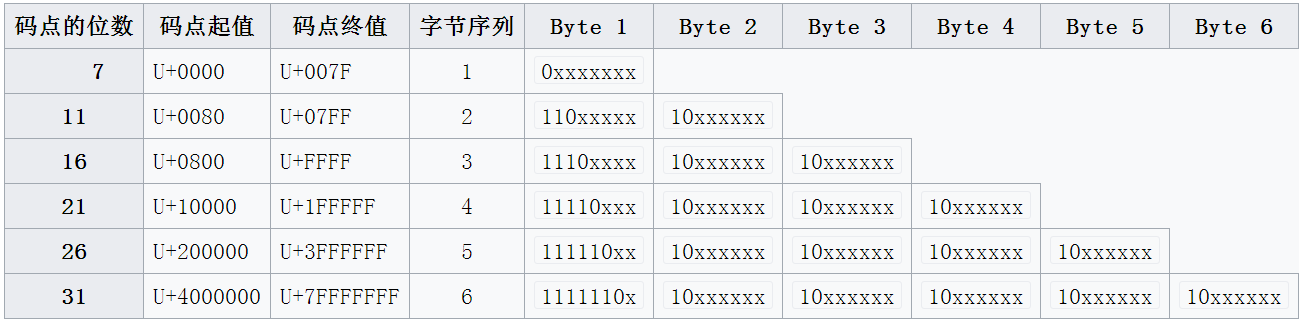
**big5**：Big5码是一套[双字节字符集](https://zh.wikipedia.org/wiki/%E5%8F%8C%E5%AD%97%E8%8A%82%E5%AD%97%E7%AC%A6%E9%9B%86)，使用了双八码存储方法，以两个字节来安放一个字。第一个字节称为“高位字节”，第二个字节称为“低位字节”。“高位字节”使用了0x81-0xFE，“低位字节”使用了0x40-0x7E，及0xA1-0xFE。在Big5的分区中：

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。

**GB2312**：在使用GB 2312的程序通常采用[EUC](https://zh.wikipedia.org/wiki/EUC)储存方法，以便兼容于[ASCII](https://zh.wikipedia.org/wiki/ASCII)。这种格式称为EUC-CN。[浏览器](https://zh.wikipedia.org/wiki/%E7%BD%91%E9%A1%B5%E6%B5%8F%E8%A7%88%E5%99%A8)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](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)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。

**UTF-8**：UTF-8就是以8位为单元对UCS进行编码，而UTF-8不使用大尾序和小尾序的形式，每个使用UTF-8存储的字符，除了第一个字节外，其余字节的头两个比特都是以"10"开始，使文字处理器能够较快地找出每个字符的开始位置。

1. 128个US-ASCII字符只需一个字节编码（Unicode范围由U+0000至U+007F）。
2. 带有[附加符号](https://zh.wikipedia.org/wiki/%E9%99%84%E5%8A%A0%E7%AC%A6%E5%8F%B7)的[拉丁文](https://zh.wikipedia.org/wiki/%E6%8B%89%E4%B8%81%E6%96%87)、[希腊文](https://zh.wikipedia.org/wiki/%E5%B8%8C%E8%87%98%E6%96%87)、[西里尔字母](https://zh.wikipedia.org/wiki/%E8%A5%BF%E9%87%8C%E7%88%BE%E5%AD%97%E6%AF%8D)、[亚美尼亚语](https://zh.wikipedia.org/wiki/%E4%BA%9E%E7%BE%8E%E5%B0%BC%E4%BA%9E%E8%AA%9E)、[希伯来文](https://zh.wikipedia.org/wiki/%E5%B8%8C%E4%BC%AF%E4%BE%86%E6%96%87)、[阿拉伯文](https://zh.wikipedia.org/wiki/%E9%98%BF%E6%8B%89%E4%BC%AF%E6%96%87)、[叙利亚文](https://zh.wikipedia.org/wiki/%E5%8F%99%E5%88%A9%E4%BA%9A%E6%96%87)及[它拿字母](https://zh.wikipedia.org/wiki/%E5%AE%83%E6%8B%BF%E5%AD%97%E6%AF%8D)则需要两个字节编码（Unicode范围由U+0080至U+07FF）。
3. 其他[基本多文种平面](https://zh.wikipedia.org/wiki/%E5%9F%BA%E6%9C%AC%E5%A4%9A%E6%96%87%E7%A8%AE%E5%B9%B3%E9%9D%A2)（BMP）中的字符（这包含了大部分常用字，如大部分的汉字）使用三个字节编码（Unicode范围由U+0800至U+FFFF）。
4. 其他极少使用的Unicode [辅助平面](https://zh.wikipedia.org/wiki/%E8%BC%94%E5%8A%A9%E5%B9%B3%E9%9D%A2)的字符使用四至六字节编码（Unicode范围由U+10000至U+1FFFFF使用四字节，Unicode范围由U+200000至U+3FFFFFF使用五字节，Unicode范围由U+4000000至U+7FFFFFFF使用六字节）。

对上述提及的第四种字符而言，UTF-8使用四至六个字节来编码似乎太耗费资源了。但UTF-8对所有常用的字符都可以用三个字节表示，而且它的另一种选择，[UTF-16编码](https://zh.wikipedia.org/wiki/UTF-16)，对前述的第四种字符同样需要四个字节来编码，所以要决定UTF-8或UTF-16哪种编码比较有效率，还要视所使用的字符的分布范围而定。

**UTF-16**：UTF-16的大尾序和小尾序存储形式都在用。一般来说，以[Macintosh](https://zh.wikipedia.org/wiki/Macintosh)制作或存储的文字使用大尾序格式，以[Microsoft](https://zh.wikipedia.org/wiki/Microsoft)或[Linux](https://zh.wikipedia.org/wiki/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](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83%E7%B5%84%E9%A0%86%E5%BA%8F%E8%A8%98%E8%99%9F)（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。

**ASCII**：ASCII 码使用指定的7 位或8 位[二进制数](https://baike.baidu.com/item/%E4%BA%8C%E8%BF%9B%E5%88%B6%E6%95%B0)组合来表示128 或256 种可能的[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)。标准ASCII 码也叫基础ASCII码，使用7 位[二进制数](https://baike.baidu.com/item/%E4%BA%8C%E8%BF%9B%E5%88%B6%E6%95%B0)（剩下的1位二进制为0）来表示所有的大写和小写字母，数字0 到9、标点符号， 以及在美式英语中使用的特殊[控制字符](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E5%AD%97%E7%AC%A6)。其中：

0～31及127(共33个)是[控制字符](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E5%AD%97%E7%AC%A6)或通信专用字符（其余为可显示字符），如控制符：LF（换行）、CR（[回车](https://baike.baidu.com/item/%E5%9B%9E%E8%BD%A6)）、FF（换页）、DEL（[删除](https://baike.baidu.com/item/%E5%88%A0%E9%99%A4/13020275)）、BS（退格)、BEL（响铃）等；通信专用字符：SOH（文头）、EOT（文尾）、ACK（确认）等；ASCII值为8、9、10 和13 分别转换为[退格](https://baike.baidu.com/item/%E9%80%80%E6%A0%BC)、制表、换行和回车字符。它们并没有特定的图形显示，但会依不同的应用程序，而对[文本](https://baike.baidu.com/item/%E6%96%87%E6%9C%AC)显示有不同的影响。

32～126(共95个)是[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)(32是空格），其中48～57为0到9十个阿拉伯数字。

65～90为26个大写英文字母，97～122号为26个小写英文字母，其余为一些标点符号、运算符号等。

同时还要注意，在标准ASCII中，其最高位(b7)用作[奇偶校验位](https://baike.baidu.com/item/%E5%A5%87%E5%81%B6%E6%A0%A1%E9%AA%8C%E4%BD%8D)。所谓奇偶校验，是指在代码传送过程中用来检验是否出现错误的一种方法，一般分[奇校验](https://baike.baidu.com/item/%E5%A5%87%E6%A0%A1%E9%AA%8C)和偶校验两种。[奇校验](https://baike.baidu.com/item/%E5%A5%87%E6%A0%A1%E9%AA%8C)规定：正确的代码一个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)中1的个数必须是奇数，若非奇数，则在最高位b7添1；偶校验规定：正确的代码一个字节中1的个数必须是[偶数](https://baike.baidu.com/item/%E5%81%B6%E6%95%B0)，若非偶数，则在最高位b7添1。

**兼容哪些哪些编码集：**

**Unicode**：兼容ISO-8859-1

**GB2312**：ASCII

**UTF-8**：ASCII，Unicode

**UTF-16**：Unicode

**应用场景：**

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。

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，例如网页浏览器或是文字处理器。

目前，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。

如果一个仅包含基本7位[ASCII](https://zh.wikipedia.org/wiki/ASCII)字符的Unicode文件，可以使用UTF-8编码，这是一种变长编码，它将基本7位ASCII字符仍用7位编码表示，占用一个字节（首位补0）。而遇到与其他Unicode字符混合的情况，将按一定算法转换，每个字符使用1-3个字节编码，并利用首位为0或1进行识别。这样对以7位ASCII字符为主的西文文档就大幅节省了编码长度。类似的，对未来会出现的需要4个字节的辅助平面字符和其他UCS-4扩充字符，2字节编码的[UTF-16](https://zh.wikipedia.org/wiki/UTF-16)也需要通过一定的算法进行转换。

如果直接使用与Unicode编码一致（仅限于BMP字符）的UTF-16编码，由于每个字符占用了两个字节，在[麦金塔电脑](https://zh.wikipedia.org/wiki/%E9%BA%A5%E9%87%91%E5%A1%94%E9%9B%BB%E8%85%A6)（[Mac](https://zh.wikipedia.org/wiki/Mac)）机和[个人电脑](https://zh.wikipedia.org/wiki/%E5%80%8B%E4%BA%BA%E9%9B%BB%E8%85%A6)上，对字节顺序的理解是不一致的，这时同一字节流可能会被解释为不同内容。UTF-16的编码顺序若不加以人为定义就可能发生混淆，于是在UTF-16编码实现方式中使用了[大端序](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82%E5%BA%8F)（Big-Endian，简写为UTF-16 BE）、[小端序](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82%E5%BA%8F)（Little-Endian，简写为UTF-16 LE）的概念，以及可附加的[字节顺序记号](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83%E7%B5%84%E9%A0%86%E5%BA%8F%E8%A8%98%E8%99%9F)解决方案，目前在PC机上的Windows系统和Linux系统对于UTF-16编码默认使用UTF-16 LE。

此外Unicode的实现方式还包括[UTF-7](https://zh.wikipedia.org/wiki/UTF-7)、[Punycode](https://zh.wikipedia.org/wiki/Punycode)、[CESU-8](https://zh.wikipedia.org/w/index.php?title=CESU-8&action=edit&redlink=1)、[SCSU](https://zh.wikipedia.org/w/index.php?title=SCSU&action=edit&redlink=1)、[UTF-32](https://zh.wikipedia.org/wiki/UTF-32)、[GB18030](https://zh.wikipedia.org/wiki/GB18030)等，这些实现方式有些仅在一定的国家和地区使用，有些则属于未来的规划方式。目前通用的实现方式是UTF-16小端序（LE）、UTF-16大端序（BE）和UTF-8。在微软公司[Windows XP](https://zh.wikipedia.org/wiki/Windows_XP)附带的[记事本](https://zh.wikipedia.org/wiki/%E8%AE%B0%E4%BA%8B%E6%9C%AC)（Notepad）中，“另存为”对话框可以选择的四种编码方式除去非Unicode编码的[ANSI](https://zh.wikipedia.org/wiki/ANSI)（对于英文系统即[ASCII](https://zh.wikipedia.org/wiki/ASCII)编码，中文系统则为[GB2312](https://zh.wikipedia.org/wiki/GB2312)或[Big5](https://zh.wikipedia.org/wiki/Big5)编码）外，其余三种为“Unicode”（对应UTF-16 LE）、“Unicode big endian”（对应UTF-16 BE）和“UTF-8”。 （详情参见下文UTF-8/16编码）。

[XML](https://zh.wikipedia.org/wiki/XML)及其子集[XHTML](https://zh.wikipedia.org/wiki/XHTML)采用[UTF-8](https://zh.wikipedia.org/wiki/UTF-8)作为[标准字集](https://zh.wikipedia.org/w/index.php?title=%E6%A0%87%E5%87%86%E5%AD%97%E9%9B%86&action=edit&redlink=1)，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](https://zh.wikipedia.org/wiki/%E5%8D%81%E8%BF%9B%E5%88%B6)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](https://zh.wikipedia.org/wiki/%E8%B6%8A%E5%8D%97%E8%AF%AD)，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（例如：♠用&#9824;代替&#x2660;）。

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。

Unicode备受认可，并广泛地应用于电脑软件的[国际化与本地化](https://zh.wikipedia.org/wiki/%E5%9C%8B%E9%9A%9B%E5%8C%96%E8%88%87%E6%9C%AC%E5%9C%B0%E5%8C%96)过程。有很多新科技，如[可扩展置标语言](https://zh.wikipedia.org/wiki/%E5%8F%AF%E6%89%A9%E5%B1%95%E7%BD%AE%E6%A0%87%E8%AF%AD%E8%A8%80)(Extensible Markup Language，简称：XML)、[Java编程语言](https://zh.wikipedia.org/wiki/Java)以及现代的[操作系统](https://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1)，都采用Unicode编码。

上述16位统一码字符构成[基本多文种平面](https://zh.wikipedia.org/wiki/%E5%9F%BA%E6%9C%AC%E5%A4%9A%E6%96%87%E7%A8%AE%E5%B9%B3%E9%9D%A2)。最新（但未实际广泛使用）的统一码版本定义了16个[辅助平面](https://zh.wikipedia.org/wiki/%E8%BE%85%E5%8A%A9%E5%B9%B3%E9%9D%A2)，两者合起来至少需要占据21位的编码空间，比3字节略少。但事实上辅助平面字符仍然占用4字节编码空间，与[UCS-4](https://zh.wikipedia.org/wiki/UCS-4)保持一致。未来版本会扩充到ISO 10646-1实现级别3，即涵盖UCS-4的所有字符。UCS-4是一个更大的尚未填充完全的31位字符集，加上恒为0的首位，共需占据32位，即4字节。理论上最多能表示231个字符，完全可以涵盖一切语言所用的符号。

在表示一个Unicode的字符时，通常会用“U+”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](https://zh.wikipedia.org/wiki/%E5%9F%BA%E6%9C%AC%E5%A4%9A%E6%96%87%E7%A8%AE%E5%B9%B3%E9%9D%A2)（英文：Basic Multilingual Plane，简写BMP。又称为“零号平面”、plane 0）里的所有字符，要用四个数字（即两个char,16bit ,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个或六个数字。旧版的Unicode标准使用相近的标记方法，但却有些微小差异：在Unicode 3.0里使用“U-”然后紧接着八个数字，而“U+”则必须随后紧接着四个数字。

Big5编码集: 在1990年代初期，当[中国大陆](https://zh.wikipedia.org/wiki/%E4%B8%AD%E5%9C%8B%E5%A4%A7%E9%99%B8)的[电子邮件](https://zh.wikipedia.org/wiki/%E9%9B%BB%E9%83%B5)和转码软件还未普遍之时，在[深圳](https://zh.wikipedia.org/wiki/%E6%B7%B1%E5%9C%B3)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、以及避免为大陆的办公室再写一套不同内码的系统。使用[简体中文](https://zh.wikipedia.org/wiki/%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87)的社区，最常用的是[GB 2312](https://zh.wikipedia.org/wiki/GB_2312)、[GBK](https://zh.wikipedia.org/wiki/GBK)及其后续的[国标码](https://zh.wikipedia.org/wiki/%E5%9B%BD%E5%AE%B6%E6%A0%87%E5%87%86%E4%BB%A3%E7%A0%81)（[GB 18030](https://zh.wikipedia.org/wiki/GB_18030)）。

除了台湾外，其他使用繁体汉字的地区，如[香港](https://zh.wikipedia.org/wiki/%E9%A6%99%E6%B8%AF)（[香港增补字符集](https://zh.wikipedia.org/wiki/%E9%A6%99%E6%B8%AF%E5%A2%9E%E8%A3%9C%E5%AD%97%E7%AC%A6%E9%9B%86)）、[澳门](https://zh.wikipedia.org/wiki/%E6%BE%B3%E9%96%80)，及使用繁体汉字的海外华人，都曾普遍使用Big5码做为中文内码及交换码。

GB2312编码集: 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。但对于[人名](https://zh.wikipedia.org/wiki/%E4%BA%BA%E5%90%8D)、[古汉语](https://zh.wikipedia.org/wiki/%E5%8F%A4%E6%B1%89%E8%AF%AD)等方面出现的[罕用字](https://zh.wikipedia.org/wiki/%E7%BD%95%E7%94%A8%E5%AD%97)和[繁体字](https://zh.wikipedia.org/wiki/%E7%B9%81%E9%AB%94%E5%AD%97)，GB 2312不能处理，因此后来[GBK](https://zh.wikipedia.org/wiki/GBK)及[GB 18030](https://zh.wikipedia.org/wiki/GB_18030)汉字字符集相继出现以解决这些问题。

UTF-8编码集：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](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)仍与[ASCII](https://zh.wikipedia.org/wiki/ASCII)兼容，这使得原来处理ASCII字符的[软件](https://zh.wikipedia.org/wiki/%E8%BB%9F%E9%AB%94)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](https://zh.wikipedia.org/wiki/%E9%9B%BB%E5%AD%90%E9%83%B5%E4%BB%B6)、[网页](https://zh.wikipedia.org/wiki/%E7%B6%B2%E9%A0%81)及其他[存储](https://zh.wikipedia.org/wiki/%E5%84%B2%E5%AD%98%E8%A3%9D%E7%BD%AE)或发送文字的应用中，优先采用的编码。UTF-8使用一至六个[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)为每个字符编码（尽管如此，2003年11月UTF-8被RFC 3629重新规范，只能使用原来Unicode定义的区域，U+0000到U+10FFFF，也就是说最多四个字节）：

1. 128个US-ASCII字符只需一个字节编码（Unicode范围由U+0000至U+007F）。
2. 带有[附加符号](https://zh.wikipedia.org/wiki/%E9%99%84%E5%8A%A0%E7%AC%A6%E5%8F%B7)的[拉丁文](https://zh.wikipedia.org/wiki/%E6%8B%89%E4%B8%81%E6%96%87)、[希腊文](https://zh.wikipedia.org/wiki/%E5%B8%8C%E8%87%98%E6%96%87)、[西里尔字母](https://zh.wikipedia.org/wiki/%E8%A5%BF%E9%87%8C%E7%88%BE%E5%AD%97%E6%AF%8D)、[亚美尼亚语](https://zh.wikipedia.org/wiki/%E4%BA%9E%E7%BE%8E%E5%B0%BC%E4%BA%9E%E8%AA%9E)、[希伯来文](https://zh.wikipedia.org/wiki/%E5%B8%8C%E4%BC%AF%E4%BE%86%E6%96%87)、[阿拉伯文](https://zh.wikipedia.org/wiki/%E9%98%BF%E6%8B%89%E4%BC%AF%E6%96%87)、[叙利亚文](https://zh.wikipedia.org/wiki/%E5%8F%99%E5%88%A9%E4%BA%9A%E6%96%87)及[它拿字母](https://zh.wikipedia.org/wiki/%E5%AE%83%E6%8B%BF%E5%AD%97%E6%AF%8D)则需要两个字节编码（Unicode范围由U+0080至U+07FF）。
3. 其他[基本多文种平面](https://zh.wikipedia.org/wiki/%E5%9F%BA%E6%9C%AC%E5%A4%9A%E6%96%87%E7%A8%AE%E5%B9%B3%E9%9D%A2)（BMP）中的字符（这包含了大部分常用字，如大部分的汉字）使用三个字节编码（Unicode范围由U+0800至U+FFFF）。
4. 其他极少使用的Unicode [辅助平面](https://zh.wikipedia.org/wiki/%E8%BC%94%E5%8A%A9%E5%B9%B3%E9%9D%A2)的字符使用四至六字节编码（Unicode范围由U+10000至U+1FFFFF使用四字节，Unicode范围由U+200000至U+3FFFFFF使用五字节，Unicode范围由U+4000000至U+7FFFFFFF使用六字节）。

对上述提及的第四种字符而言，UTF-8使用四至六个字节来编码似乎太耗费资源了。但UTF-8对所有常用的字符都可以用三个字节表示，而且它的另一种选择，[UTF-16编码](https://zh.wikipedia.org/wiki/UTF-16)，对前述的第四种字符同样需要四个字节来编码，所以要决定UTF-8或UTF-16哪种编码比较有效率，还要视所使用的字符的分布范围而定。

* 对于UTF-8编码中的任意字节B，如果B的第一位为0，则B独立的表示一个字符(ASCII码)；
* 如果B的第一位为1，第二位为0，则B为一个多字节字符中的一个字节(非ASCII字符)；
* 如果B的前两位为1，第三位为0，则B为两个字节表示的字符中的第一个字节；
* 如果B的前三位为1，第四位为0，则B为三个字节表示的字符中的第一个字节；
* 如果B的前四位为1，第五位为0，则B为四个字节表示的字符中的第一个字节；

因此，对UTF-8编码中的任意字节，根据第一位，可判断是否为ASCII字符；根据前二位，可判断该字节是否为一个字符编码的第一个字节；根据前四位（如果前两位均为1），可确定该字节为字符编码的第一个字节，并且可判断对应的字符由几个字节表示；根据前五位（如果前四位为1），可判断编码是否有错误或数据传输过程中是否有错误。

理由：UTF-8的设计有以下的多字符组序列的特质：

* 单字节字符的[最高有效比特](https://zh.wikipedia.org/wiki/%E6%9C%80%E9%AB%98%E6%9C%89%E6%95%88%E4%BD%8D%E5%85%83)永远为0。
* 多字节序列中的首个字符组的几个[最高有效比特](https://zh.wikipedia.org/wiki/%E6%9C%80%E9%AB%98%E6%9C%89%E6%95%88%E4%BD%8D%E5%85%83)决定了序列的长度。最高有效位为110的是2字节序列，而1110的是三字节序列，如此类推。
* 多字节序列中其余的字节中的首两个最高有效比特为10。

UTF-8的这些特质，保证了一个字符的[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)序列不会包含在另一个字符的[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)序列中。这确保了以字节为基础的部分字符串比对（sub-string match）方法可以适用于在文字中搜索字或词。有些比较旧的可变长度8位编码（如[Shift JIS](https://zh.wikipedia.org/wiki/Shift_JIS)）没有这个特质，故字符串比对的算法变得相当复杂。虽然这增加了UTF-8编码的字符串的[信息冗余](https://zh.wikipedia.org/wiki/%E4%BF%A1%E6%81%AF%E5%86%97%E4%BD%99)，但是利多于弊。另外，[数据压缩](https://zh.wikipedia.org/wiki/%E8%B3%87%E6%96%99%E5%A3%93%E7%B8%AE)并非Unicode的目的，所以不可混为一谈。即使在发送过程中有部分字节因错误或干扰而完全丢失，还是有可能在下一个字符的起点重新同步，令受损范围受到限制。ASCII字母继续使用1字节存储，[重音文字](https://zh.wikipedia.org/w/index.php?title=%E9%87%8D%E9%9F%B3%E6%96%87%E5%AD%97&action=edit&redlink=1)、[希腊字母](https://zh.wikipedia.org/wiki/%E5%B8%8C%E8%87%98%E5%AD%97%E6%AF%8D)或[西里尔字母](https://zh.wikipedia.org/wiki/%E8%A5%BF%E9%87%8C%E7%88%BE%E5%AD%97%E6%AF%8D)等使用2字节来存储，而常用的[汉字](https://zh.wikipedia.org/wiki/%E6%BC%A2%E5%AD%97)就要使用3字节。[辅助平面](https://zh.wikipedia.org/wiki/%E8%BE%85%E5%8A%A9%E5%B9%B3%E9%9D%A2)字符则使用4字节。

在UTF-8文件的开首，很多时都放置一个U+FEFF字符（UTF-8以EF,BB,BF代表），以显示这个文本文件是以UTF-8编码。

另一方面，由于其字节序列设计，如果一个疑似为字符串的序列被验证为UTF-8编码，那么我们可以有把握地说它是UTF-8字符串。一段两字节随机序列碰巧为合法的UTF-8而非ASCII的概率为32分1。对于三字节序列的概率为256分1，对更长的序列的概率就更低了。

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](https://zh.wikipedia.org/wiki/%E5%AD%90%E9%9B%86)。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](https://zh.wikipedia.org/wiki/%E5%8F%AF%E6%89%A9%E5%B1%95%E6%A0%87%E8%AE%B0%E8%AF%AD%E8%A8%80)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]](http://www.w3.org/TR/REC-xml/#charencoding)
* 任何[面向字节](https://zh.wikipedia.org/w/index.php?title=%E9%9D%A2%E5%90%91%E5%AD%97%E8%8A%82&action=edit&redlink=1)的[字符串搜索算法](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6%E4%B8%B2%E6%90%9C%E7%B4%A2%E7%AE%97%E6%B3%95)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。举例说，字符值C0,C1,F5至FF从来没有出现。为了更好的可靠性，可以使用正则表达式来统计非法过长和替代值（可以查看[W3 FAQ: Multilingual Forms](http://www.w3.org/International/questions/qa-forms-utf-8)上的验证UTF-8字符串的正则表达式）。
* 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。

# 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](https://zh.wikipedia.org/wiki/%E8%A7%A3%E6%9E%90%E5%99%A8)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。

# 不利于正则表达式检索

[正则表达式](https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F)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](https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F)，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](https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F)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](https://zh.wikipedia.org/wiki/%E6%AD%A3%E5%88%99%E8%A1%A8%E8%BE%BE%E5%BC%8F)检索而GBK不行。

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](https://zh.wikipedia.org/wiki/CJK)）这样的方块文字。

UTF-16编码集：UTF-16可看成是UCS-2的[父集](https://zh.wikipedia.org/wiki/%E7%88%B6%E9%9B%86)。在没有[辅助平面字符](https://zh.wikipedia.org/wiki/Unicode%E5%AD%97%E7%AC%A6%E5%B9%B3%E9%9D%A2%E6%98%A0%E5%B0%84)（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。

# Microsoft Windows操作系统内核对Unicode的支持

Windows操作系统内核中的字符表示为UTF-16小尾序，可以正确处理、显示以4字节存储的字符。但是Windows API实际上仅能正确处理UCS-2字符，即仅以2字节存储的，码位小于U+FFFF的Unicode字符。其根源是Microsoft C++语言把wchar\_t数据类型定义为16比特的unsigned short，这就与一个wchar\_t型变量对应一个[宽字符](https://zh.wikipedia.org/w/index.php?title=%E5%AE%BD%E5%AD%97%E7%AC%A6&action=edit&redlink=1)，可以存储一个Unicode字符的规定相矛盾。相反，Linux平台的GCC编译器规定一个wchar\_t是4字节长度，可以存储一个UTF-32字符，宁可浪费了很大的存储空间。下例运行于Windows平台的C++程序可说明此点：

ASCII编码集：ASCII是美国标准，所以它不能良好满足其它讲英语国家的需要。1967年，国际标准化组织（ISO：International Standards Organization）推荐一个ASCII的变种，代码0x40、0x5B、0x5C、0x5D、0x7B、0x7C和0x7D“为国家使用保留”，而代码0x5E、0x60和0x7E标为“当国内要求的特殊字符需要8、9或10个空间位置时，可用于其它图形符号”。这显然不是一个最佳的国际解决方案，因为这并不能保证一致性。但这却显示了人们如何想尽办法为不同的语言来编码的。

# 参考文献

[1]维基百科、

[2]百度百科 、

[3]The Unicode Standard 第五版. Addison-Wesley Professional. [ISBN 0321480910](https://zh.wikipedia.org/wiki/Special:%E7%BD%91%E7%BB%9C%E4%B9%A6%E6%BA%90/0321480910).