常见编码集

目录

[一．Unicode 3](#_Toc492815860)

[1.描述 3](#_Toc492815861)

[2.起源 3](#_Toc492815862)

[3.占字节 4](#_Toc492815863)

[4.兼容的字符集 4](#_Toc492815864)

[5.优点 6](#_Toc492815865)

[6.缺点 6](#_Toc492815866)

[7.编码方式 6](#_Toc492815867)

[8.相互转换 9](#_Toc492815868)

[9.解决的问题 9](#_Toc492815869)

[二．Big5 10](#_Toc492815870)

[1.描述 10](#_Toc492815871)

[2.来源 10](#_Toc492815872)

[3.占字节 11](#_Toc492815873)

[4.优点 11](#_Toc492815874)

[5．缺点 11](#_Toc492815875)

[6.编码方式 11](#_Toc492815876)

[7.兼容 11](#_Toc492815877)

[三．GB2312 12](#_Toc492815878)

[1.描述 12](#_Toc492815879)

[2.解决的问题 12](#_Toc492815880)

[3.优点 12](#_Toc492815881)

[4.缺点 12](#_Toc492815882)

[5.字节结构 12](#_Toc492815883)

[6.兼容性 13](#_Toc492815884)

[四．UTF-8 13](#_Toc492815885)

[1.描述 13](#_Toc492815886)

[2.来源 14](#_Toc492815887)

[3.产生原因 15](#_Toc492815888)

[4.编码方式 16](#_Toc492815889)

[5.优点 16](#_Toc492815890)

[6.缺点 17](#_Toc492815891)

[7.兼容的字符集 17](#_Toc492815892)

[8.占字节 17](#_Toc492815893)

[五．UTF-16 20](#_Toc492815894)

[1.描述 20](#_Toc492815895)

[2.来源 20](#_Toc492815896)

[3.编码方式 21](#_Toc492815897)

[4．优点 21](#_Toc492815898)

[5.缺点 21](#_Toc492815899)

[6.占字节 21](#_Toc492815900)

[六．引用资料 21](#_Toc492815901)

# 一．Unicode

1.描述：**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）是[计算机科学](https://zh.wikipedia.org/wiki/%E9%9B%BB%E8%85%A6%E7%A7%91%E5%AD%B8" \o "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](https://zh.wikipedia.org/wiki/%E6%96%87%E5%AD%97%E7%B3%BB%E7%B5%B1)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。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](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6" \o "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](https://zh.wikipedia.org/wiki/%E5%AD%97%E7%AC%A6%E7%BC%96%E7%A0%81" \o "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](https://zh.wikipedia.org/wiki/%E7%B5%B1%E4%B8%80%E7%A2%BC%E8%81%AF%E7%9B%9F" \o "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](https://zh.wikipedia.org/wiki/%E5%A4%9A%E8%AA%9E" \o "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](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" \o "国际化与本地化)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" \o "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。

2.起源：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" \o "字符编码)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" \o "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）。

3.占字节：目前的用于实用的 Unicode 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。

4.兼容的字符集：[XML](https://zh.wikipedia.org/wiki/XML)及其子集[XHTML](https://zh.wikipedia.org/wiki/XHTML" \o "XHTML)采用[UTF-8](https://zh.wikipedia.org/wiki/UTF-8" \o "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" \o "标准字集（页面不存在）)，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" \o "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](https://zh.wikipedia.org/wiki/%E8%B6%8A%E5%8D%97%E8%AF%AD" \o "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（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。

然而部分由于Unicode版本发展原因，很多浏览器只能显示UCS-2完整字符集，也即现在使用的Unicode版本中的一个小子集。下表可以检验您的浏览器如何显示各种Unicode代码：

|  |  |  |
| --- | --- | --- |
| **代码** | **字符标准名称（英语）** | **在浏览器上的显示** |
| &#x0041; | 大写[拉丁字母](https://zh.wikipedia.org/wiki/%E6%8B%89%E4%B8%81%E5%AD%97%E6%AF%8D" \o "拉丁字母)“A” | A |
| &#x00DF; | 小写拉丁字母“Sharp S” | ß |
| &#x00FE; | 小写拉丁字母“Thorn” | þ |
| &#x0394; | 大写[希腊字母](https://zh.wikipedia.org/wiki/%E5%B8%8C%E8%85%8A%E5%AD%97%E6%AF%8D" \o "希腊字母)“Delta” | Δ |
| &#x0419; | 大写[斯拉夫字母](https://zh.wikipedia.org/wiki/%E6%96%AF%E6%8B%89%E5%A4%AB%E5%AD%97%E6%AF%8D" \o "斯拉夫字母)“Short I” | Й |
| &#x05E7; | [希伯来字母](https://zh.wikipedia.org/wiki/%E5%B8%8C%E4%BC%AF%E6%9D%A5%E5%AD%97%E6%AF%8D)“Qof” | ק |
| &#x0645; | [阿拉伯字母](https://zh.wikipedia.org/wiki/%E9%98%BF%E6%8B%89%E4%BC%AF%E5%AD%97%E6%AF%8D)“Meem” | م |
| &#x0E57; | [泰文](https://zh.wikipedia.org/wiki/%E6%B3%B0%E8%AF%AD)[数字](https://zh.wikipedia.org/wiki/%E6%95%B0%E5%AD%97)7 | ๗ |
| &#x1250; | [埃塞俄比亚音节文字](https://zh.wikipedia.org/wiki/%E9%98%BF%E5%A7%86%E5%93%88%E6%8B%89%E8%AF%AD)“Qha” | ቐ |
| &#x3042; | 日语[平假名](https://zh.wikipedia.org/wiki/%E5%B9%B3%E5%81%87%E5%90%8D" \o "平假名)“A” | あ |
| &#x30A2; | 日语[片假名](https://zh.wikipedia.org/wiki/%E7%89%87%E5%81%87%E5%90%8D" \o "片假名)“A” | ア |
| &#x53F6; | [简体](https://zh.wikipedia.org/wiki/%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87)[汉字](https://zh.wikipedia.org/wiki/%E6%B1%89%E5%AD%97)“叶” | 叶 |
| &#x8449; | [繁体](https://zh.wikipedia.org/wiki/%E7%B9%81%E4%BD%93%E4%B8%AD%E6%96%87)汉字“葉” | 葉 |
| &#xC5FD; | [韩国音节文字](https://zh.wikipedia.org/wiki/%E8%B0%9A%E6%96%87)“Yeop” | 엽 |

一些多语言支持的网页浏览器，比如[微软](https://zh.wikipedia.org/wiki/%E5%BE%AE%E8%BD%AF)[Windows](https://zh.wikipedia.org/wiki/Windows)系统的[Internet Explorer](https://zh.wikipedia.org/wiki/Internet_Explorer" \o "Internet Explorer) 5.5及以上版本，以及跨平台的浏览器[Mozilla](https://zh.wikipedia.org/wiki/Mozilla" \o "Mozilla)／[Netscape](https://zh.wikipedia.org/wiki/Netscape" \o "Netscape) 6，可以在安装时根据需要动态地使用相应的字符集，预先安装了合适的语言包，就可以同时显示页面上的各种Unicode字符。[Internet Explorer](https://zh.wikipedia.org/wiki/Internet_Explorer" \o "Internet Explorer) 5.5还提出用户可以在需要新字体时，即装即用。另外的浏览器如[Netscape Navigator](https://zh.wikipedia.org/wiki/Netscape_Navigator" \o "Netscape Navigator) 4.77，则只能显示跟页面编码相应字符集中的文字。当你使用后一种浏览器时，你不大可能预先安装所有的字体，即使有了字体，浏览器也不一定能将这些字体完全应用起来。可能遇到的情况是，这种浏览器只能够显示部分文字，因为它们是按照标准进行编码，尽管理论上在兼容的系统中，只要有了相应的[Code2000](https://zh.wikipedia.org/wiki/Code2000)字体，就可以正确显示。一种变通的办法，是将某些少见的字符，通过“名称实体引用”的方式来使用。

5.优点：Unicode能够表示世界上绝大多数语言。

6.缺点：Unicode使用两个字节表示一个字符，空间占用翻倍了，网络传输的数据量也增大了。

7.编码方式：统一码的编码方式与[ISO 10646](https://zh.wikipedia.org/wiki/ISO_10646)的[通用字符集](https://zh.wikipedia.org/wiki/%E9%80%9A%E7%94%A8%E5%AD%97%E7%AC%A6%E9%9B%86" \o "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](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" \o "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。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" \o "基本多文种平面)。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" \o "辅助平面)，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。基本多文种平面的字符的编码为U+hhhh，其中每个*h*代表一个[十六进制](https://zh.wikipedia.org/wiki/%E5%8D%81%E5%85%AD%E8%BF%9B%E5%88%B6" \o "十六进制)数字，与UCS-2编码完全相同。而其对应的4字节UCS-4编码后两个字节一致，前两个字节则所有位均为0。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](https://zh.wikipedia.org/wiki/%E7%B3%BB%E7%BB%9F%E5%B9%B3%E5%8F%B0" \o "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）例如，如果一个仅包含基本7位[ASCII](https://zh.wikipedia.org/wiki/ASCII" \o "ASCII)字符的Unicode文件，如果每个字符都使用2字节的原Unicode编码传输，其第一字节的8位始终为0。这就造成了比较大的浪费。对于这种情况，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（具体方案参见*[UTF-8](https://zh.wikipedia.org/wiki/UTF-8" \o "UTF-8)*）。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" \o "UTF-16)也需要通过一定的算法进行转换。再如，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" \o "麦金塔电脑)（[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" \o "个人电脑)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，如某字符为十六进制编码4E59，按两个字节拆分为4E和59，在Mac上读取时是从低字节开始，那么在Mac OS会认为此4E59编码为594E，找到的字符为“奎”，而在Windows上从高字节开始读取，则编码为U+4E59的字符为“乙”。就是说在Windows下以UTF-16编码保存一个字符“乙”，在Mac OS环境下打开会显示成“奎”。此类情况说明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" \o "字节序)（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" \o "字节序)（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" \o "字节顺序记号)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。（具体方案参见*[UTF-16](https://zh.wikipedia.org/wiki/UTF-16" \o "UTF-16)*）此外Unicode的实现方式还包括[UTF-7](https://zh.wikipedia.org/wiki/UTF-7" \o "UTF-7)、[Punycode](https://zh.wikipedia.org/wiki/Punycode" \o "Punycode)、[CESU-8](https://zh.wikipedia.org/w/index.php?title=CESU-8&action=edit&redlink=1" \o "CESU-8（页面不存在）)、[SCSU](https://zh.wikipedia.org/w/index.php?title=SCSU&action=edit&redlink=1" \o "SCSU（页面不存在）)、[UTF-32](https://zh.wikipedia.org/wiki/UTF-32" \o "UTF-32)、[GB18030](https://zh.wikipedia.org/wiki/GB18030" \o "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" \o "Windows XP)附带的[记事本](https://zh.wikipedia.org/wiki/%E8%AE%B0%E4%BA%8B%E6%9C%AC" \o "记事本)（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" \o "ANSI)（对于英文系统即[ASCII](https://zh.wikipedia.org/wiki/ASCII" \o "ASCII)编码，中文系统则为[GB2312](https://zh.wikipedia.org/wiki/GB2312" \o "GB2312)或[Big5](https://zh.wikipedia.org/wiki/Big5" \o "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”。目前辅助平面的工作主要集中在第二和第三平面的[中日韩统一表意文字](https://zh.wikipedia.org/wiki/%E4%B8%AD%E6%97%A5%E9%9F%A9%E7%BB%9F%E4%B8%80%E8%A1%A8%E6%84%8F%E6%96%87%E5%AD%97)中，因此包括[GBK](https://zh.wikipedia.org/wiki/GBK)、[GB18030](https://zh.wikipedia.org/wiki/GB18030" \o "GB18030)、[Big5](https://zh.wikipedia.org/wiki/Big5" \o "Big5)等[简体中文](https://zh.wikipedia.org/wiki/%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87" \o "简体中文)、[繁体中文](https://zh.wikipedia.org/wiki/%E7%B9%81%E4%BD%93%E4%B8%AD%E6%96%87)、[日文](https://zh.wikipedia.org/wiki/%E6%97%A5%E6%96%87)、[韩文](https://zh.wikipedia.org/wiki/%E9%9F%A9%E6%96%87)以及越南[喃字](https://zh.wikipedia.org/wiki/%E5%96%83%E5%AD%97)的各种编码与Unicode的协调性被重点关注。考虑到Unicode最终要涵盖所有的字符。从某种意义而言，这些编码方式也可视作Unicode的出现于其之前的*既成事实*的实现方式，如同[ASCII](https://zh.wikipedia.org/wiki/ASCII)及其扩展[Latin-1](https://zh.wikipedia.org/wiki/Latin-1" \o "Latin-1)一样，后两者的字符在16位Unicode编码空间中的编码第一字节各位全为0，第二字节编码与原编码完全一致。但上述东亚语言编码与Unicode编码的对应关系要复杂得多。

8.相互转换：Unicode与其他著名字符集可以精确转换。

9.解决的问题：Unicode编码包含了不同写法的字，如“ɑ／a”、“強／强”、“戶／户／戸”。然而在[汉字](https://zh.wikipedia.org/wiki/%E6%B1%89%E5%AD%97" \o "汉字)方面引起了一字多形的认定争议（详见[中日韩统一表意文字](https://zh.wikipedia.org/wiki/%E4%B8%AD%E6%97%A5%E9%9F%93%E7%B5%B1%E4%B8%80%E8%A1%A8%E6%84%8F%E6%96%87%E5%AD%97" \o "中日韩统一表意文字)主题）。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。举例来说，[全角](https://zh.wikipedia.org/wiki/%E5%85%A8%E5%BD%A2" \o "全角)格式区块包含了主要的拉丁字母的全角格式，在中文、日文、以及韩文字形当中，这些字符以全角的方式来呈现，而不以常见的半角。

# 二．Big5

1.描述：**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" \o "汉字)[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。中文码分为[内码](https://zh.wikipedia.org/wiki/%E5%85%A7%E7%A2%BC" \o "内码)及[交换码](https://zh.wikipedia.org/wiki/%E4%BA%A4%E6%8F%9B%E7%A2%BC)两类，Big5属中文内码，知名的中文交换码有[CCCII](https://zh.wikipedia.org/wiki/CCCII" \o "CCCII)、[CNS11643](https://zh.wikipedia.org/wiki/CNS11643" \o "CNS11643)。

2.来源：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" \o "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](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" \o "王安码（页面不存在）)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。Big5产生前，研发[中文电脑](https://zh.wikipedia.org/wiki/%E4%B8%AD%E6%96%87%E9%9B%BB%E8%85%A6" \o "中文电脑)的[朱邦复](https://zh.wikipedia.org/wiki/%E6%9C%B1%E9%82%A6%E5%BE%A9)认为内码字集应该广纳所有的正异体字，以顾及如户政等应用上的需要，故在当时的内码会议中，建议希望采用他的五万多字的字库。工程师认为虽其技术可行，但是三个字节（超过两个字节）长度的内码却会造成英文显示屏画面映射成中文画面会发生文字无法对齐的问题，因为当时盛行之倚天中文系统画面系以两个字节文字宽度映射成一个中文字图样，英文软件中只要以两个英文字宽度去显示一个中文字，画面就不会乱掉，造成中文系统业者偏爱二个字节长度的内码[[9]](https://zh.wikipedia.org/wiki/%E5%A4%A7%E4%BA%94%E7%A2%BC#cite_note-9)；此外以仓颉输入码压缩成的内码不具排序等功能，因此未被采用。1983年有人诬指朱邦复为[共产党](https://zh.wikipedia.org/wiki/%E5%85%B1%E7%94%A2%E9%BB%A8" \o "共产党)，其研究成果更不可能获采用。

3.占字节：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" \o "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”。“高位字节”使用了0x81-0xFE，“低位字节”使用了0x40-0x7E，及0xA1-0xFE。

4.优点：提供了一种中文编码标准，使得厂商推出的中文应用软件得以推广。

5．缺点：尽管[Big5码](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7703959&ss_c=ssc.citiao.link)内包含一万多个字符，但是没有考虑社会上流通的人名、地名用字、方言用字、化学及生物科等用字，没有包含日文[平假名](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=215422&ss_c=ssc.citiao.link)及[片假名](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=255551&ss_c=ssc.citiao.link)字母。例如台湾视“着”为“著”的异体字，故没有收录“着”字。康熙字典中的一些部首用字(如“亠”、“疒”、“辵”、“癶”等)、常见的人名用字(如“堃”、“煊”、“栢”、“喆”等)也没有收录到Big5之中。

6.编码方式：各编码范围对应的字符类型如下：0xA140-0xA3BF为标点符号、[希腊字母](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=437970&ss_c=ssc.citiao.link" \t "_blank)及[特殊符号](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=493633&ss_c=ssc.citiao.link)，另外于0xA259-0xA261，存放了双音节[度量衡](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=166108&ss_c=ssc.citiao.link" \t "_blank)单位用字：兙兛兞兝兡兣嗧瓩糎；0xA440-0xC67E为常用汉字，先按笔划再按部首排序；0xC940-0xF9D5为次常用汉字，亦是先按笔划再按部首排序。

7.兼容：与画线字符相冲，Big5码字符的首字节会与DOS[代码页437](https://zh.wikipedia.org/wiki/%E4%BB%A3%E7%A2%BC%E9%A0%81437)的[画线字符](https://zh.wikipedia.org/wiki/%E6%96%B9%E6%A1%86%E7%BB%98%E5%88%B6%E5%AD%97%E7%AC%A6" \o "方框绘制字符)相冲而产生乱码。

# 三．GB2312

1.描述：**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" \o "中华人民共和国国家标准)[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" \o "中国国家标准总局（页面不存在）)发布，1981年5月1日实施。GB 2312编码通行于中国大陆；[新加坡](https://zh.wikipedia.org/wiki/%E6%96%B0%E5%8A%A0%E5%9D%A1" \o "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。GB 2312标准共收录6763个[汉字](https://zh.wikipedia.org/wiki/%E6%B1%89%E5%AD%97" \o "汉字)，其中[一级汉字](https://zh.wikipedia.org/wiki/%E5%B8%B8%E7%94%A8%E5%AD%97)3755个，[二级汉字](https://zh.wikipedia.org/wiki/%E6%AC%A1%E5%B8%B8%E7%94%A8%E5%AD%97" \o "次常用字)3008个；同时收录了包括[拉丁字母](https://zh.wikipedia.org/wiki/%E6%8B%89%E4%B8%81%E5%AD%97%E6%AF%8D" \o "拉丁字母)、[希腊字母](https://zh.wikipedia.org/wiki/%E5%B8%8C%E8%85%8A%E5%AD%97%E6%AF%8D)、[日文](https://zh.wikipedia.org/wiki/%E6%97%A5%E8%AF%AD)[平假名](https://zh.wikipedia.org/wiki/%E5%B9%B3%E5%81%87%E5%90%8D)及[片假名](https://zh.wikipedia.org/wiki/%E7%89%87%E5%81%87%E5%90%8D)字母、[俄语](https://zh.wikipedia.org/wiki/%E4%BF%84%E8%AF%AD)[西里尔字母](https://zh.wikipedia.org/wiki/%E6%96%AF%E6%8B%89%E5%A4%AB%E5%AD%97%E6%AF%8D)在内的682个字符。

2.解决的问题：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。

3.优点：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，应用范围广，使用频率高。

4.缺点：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" \o "GBK)及[GB 18030](https://zh.wikipedia.org/wiki/GB_18030" \o "GB 18030)汉字字符集相继出现以解决这些问题。

5.字节结构：在使用GB 2312的程序通常采用[EUC](https://zh.wikipedia.org/wiki/EUC" \o "EUC)储存方法，以便兼容于[ASCII](https://zh.wikipedia.org/wiki/ASCII" \o "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" \o "网页浏览器)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](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。例如“啊”字在大多数程序中，会以两个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82" \t "_blank)，0xB0（第一个字节） 0xA1（第二个字节）储存。[区位码](https://baike.baidu.com/item/%E5%8C%BA%E4%BD%8D%E7%A0%81" \t "_blank)=区字节+位字节（与区位码对比：0xB0=0xA0+16,0xA1=0xA0+1）。

6.兼容性：GBK子集与GBK/GB 18030兼容，GB2312.TXT则不兼容。后者基于ftp.unicode.org曾经提供的GB2312.TXT实现，于2011年由官方弃用，2016年9月时已无原文件踪迹。此外还有很多种厂商实现。截至2015年，微软.NET使用的是“GBK子集”实现。[ICU](https://zh.wikipedia.org/w/index.php?title=%E4%B8%87%E5%9B%BD%E7%A0%81%E5%9B%BD%E9%99%85%E5%8C%96%E7%BB%84%E4%BB%B6&action=edit&redlink=1)、libiconv-1.14、php-5.6、ActivePerl-5.20、Java 1.7、Python 3.4都使用“GB2312.TXT”实现。Ruby 2.2兼容两者编码，但内部使用“GBK子集”实现。W3C的编码技术指南规定，应将gb2312字节流视为GBK编码，与GB18030一并使用同一解码器解码。在使用GB 2312的程序通常采用[EUC](https://zh.wikipedia.org/wiki/EUC" \o "EUC)储存方法，以便兼容于[ASCII](https://zh.wikipedia.org/wiki/ASCII" \o "ASCII)。

# 四．UTF-8

1.描述：**UTF-8**（**8-bit Unicode Transformation Format**）是一种针对[Unicode](https://zh.wikipedia.org/wiki/Unicode" \o "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" \o "字符编码)，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" \o "字节)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" \o "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](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。

2.来源：1992年初，为创建良好的[字节串编码系统](https://zh.wikipedia.org/w/index.php?title=%E4%BD%8D%E5%85%83%E7%B5%84%E4%B8%B2%E7%B7%A8%E7%A2%BC%E7%B3%BB%E7%B5%B1&action=edit&redlink=1" \o "字节串编码系统（页面不存在）)以供多[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82" \o "字节)字符集使用，开始了一个正式的研究。[ISO/IEC 10646](https://zh.wikipedia.org/wiki/%E9%80%9A%E7%94%A8%E5%AD%97%E7%AC%A6%E9%9B%86)的初稿中有一个非必须的附录，名为UTF。当中包含了一个供32[比特](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83)的[字符](https://zh.wikipedia.org/wiki/UCS)使用的字节串编码系统。这个编码方式的性能并不令人满意，但它提出了将0-127的范围保留给ASCII以兼容旧系统的概念。1992年7月，[X/Open](https://zh.wikipedia.org/wiki/X/Open" \o "X/Open)委员会XoJIG开始寻求一个较佳的编码系统。[Unix系统实验室](https://zh.wikipedia.org/wiki/Unix%E7%B3%BB%E7%BB%9F%E5%AE%9E%E9%AA%8C%E5%AE%A4" \o "Unix系统实验室)（USL）的Dave Prosser为此提出了一个编码系统的建议。它具备可更快速实现的特性，并引入一项新的改进。其中，7[比特](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83)的[ASCII](https://zh.wikipedia.org/wiki/ASCII)符号只代表原来的意思，所有多字节序列则会包含第8[比特](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83)的符号，也就是所谓的[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)。1992年8月，这个建议由[IBM](https://zh.wikipedia.org/wiki/IBM)[X/Open](https://zh.wikipedia.org/wiki/X/Open)的代表流传到一些感兴趣的团体。与此同时，[贝尔实验室](https://zh.wikipedia.org/wiki/%E8%B2%9D%E7%88%BE%E5%AF%A6%E9%A9%97%E5%AE%A4)[九号项目](https://zh.wikipedia.org/wiki/%E8%B2%9D%E7%88%BE%E5%AF%A6%E9%A9%97%E5%AE%A4%E4%B9%9D%E8%99%9F%E8%A8%88%E7%95%AB)[操作系统](https://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1)工作小组的[肯·汤普逊](https://zh.wikipedia.org/wiki/%E8%82%AF%C2%B7%E6%B1%A4%E6%99%AE%E9%80%8A)对这编码系统作出重大的修改，让编码可以自我同步，使得不必从字符串的开首读取，也能找出字符间的分界。1992年9月2日，[肯·汤普逊](https://zh.wikipedia.org/wiki/%E8%82%AF%C2%B7%E6%B1%A4%E6%99%AE%E9%80%8A)和[罗勃·派克](https://zh.wikipedia.org/wiki/%E7%BE%85%E5%8B%83%C2%B7%E6%B4%BE%E5%85%8B)一起在[美国](https://zh.wikipedia.org/wiki/%E7%BE%8E%E5%9C%8B)[新泽西州](https://zh.wikipedia.org/wiki/%E6%96%B0%E6%BE%A4%E8%A5%BF%E5%B7%9E)一架餐车的餐桌垫上描绘出此设计的要点。接下来的日子，Pike及汤普逊将它实现，并将这编码系统完全应用在[九号项目](https://zh.wikipedia.org/wiki/%E8%B2%9D%E7%88%BE%E5%AF%A6%E9%A9%97%E5%AE%A4%E4%B9%9D%E8%99%9F%E8%A8%88%E7%95%AB)当中，及后他将有关成果回馈X/Open。1993年1月25-29日的在[圣地牙哥](https://zh.wikipedia.org/wiki/%E8%81%96%E5%9C%B0%E7%89%99%E5%93%A5)举行的[USENIX](https://zh.wikipedia.org/wiki/USENIX)会议首次正式介绍UTF-8。自1996年起，[微软](https://zh.wikipedia.org/wiki/%E5%BE%AE%E8%BB%9F" \o "微软)的[CAB](https://zh.wikipedia.org/wiki/CAB)（MS Cabinet）规格在UTF-8标准正式落实前就明确容许在任何地方使用UTF-8编码系统。但有关的编码器实际上从来没有实现这方面的规格。

3.产生原因：**UTF-8**（**8-bit Unicode Transformation Format**）是一种针对[Unicode](https://zh.wikipedia.org/wiki/Unicode" \o "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" \o "字符编码)，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" \o "字节)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" \o "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](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" \o "字节)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" \o "附加符号)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" \o "基本多文种平面)（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）。

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）。

4.编码方式：UTF-8是UNICODE的一种变长度的编码表达方式《一般UNICODE为双位元组（指UCS2）》，它由[肯·汤普逊](https://zh.wikipedia.org/wiki/%E8%82%AF%C2%B7%E6%B1%A4%E6%99%AE%E9%80%8A)（Ken Thompson）于1992年创建，现在已经标准化为RFC 3629。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。但为了与以前的ASCII码兼容（ASCII为一个字节），因此UTF-8选择了使用可变长度[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)来存储Unicode。

5.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。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。

6.缺点：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。

7.兼容的字符集：ASCII

8.占字节：UTF-8是UNICODE的一种变长度的编码表达方式《一般UNICODE为双位元组（指UCS2）》，它由[肯·汤普逊](https://zh.wikipedia.org/wiki/%E8%82%AF%C2%B7%E6%B1%A4%E6%99%AE%E9%80%8A" \o "肯·汤普逊)（Ken Thompson）于1992年创建，现在已经标准化为RFC 3629。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。

但为了与以前的ASCII码兼容（ASCII为一个字节），因此UTF-8选择了使用可变长度[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82" \o "字节)来存储Unicode：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Unicode 和 UTF-8 之间的转换关系表 ( x 字符表示码点占据的位 )** | | | | | | | | | |
| **码点的位数** | **码点起值** | **码点终值** | **字节序列** | **Byte 1** | **Byte 2** | **Byte 3** | **Byte 4** | **Byte 5** | **Byte 6** |
| **7** | U+0000 | U+007F | 1 | 0xxxxxxx |  |  |  |  |  |
| **11** | U+0080 | U+07FF | 2 | 110xxxxx | 10xxxxxx |  |  |  |  |
| **16** | U+0800 | U+FFFF | 3 | 1110xxxx | 10xxxxxx | 10xxxxxx |  |  |  |
| **21** | U+10000 | U+1FFFFF | 4 | 11110xxx | 10xxxxxx | 10xxxxxx | 10xxxxxx |  |  |
| **26** | U+200000 | U+3FFFFFF | 5 | 111110xx | 10xxxxxx | 10xxxxxx | 10xxxxxx | 10xxxxxx |  |
| **31** | U+4000000 | U+7FFFFFFF | 6 | 1111110x | 10xxxxxx | 10xxxxxx | 10xxxxxx | 10xxxxxx | 10xxxxxx |

* 在ASCII码的范围，用一个字节表示，超出ASCII码的范围就用字节表示，这就形成了我们上面看到的UTF-8的表示方法，这样的好处是当UNICODE文件中只有ASCII码时，存储的文件都为一个字节，所以就是普通的ASCII文件无异，读取的时候也是如此，所以能与以前的ASCII文件兼容。
* 大于ASCII码的，就会由上面的第一字节的前几位表示该unicode字符的长度，比如110xxxxx前三位的二进制表示告诉我们这是个2BYTE的UNICODE字符；1110xxxx是个三位的UNICODE字符，依此类推；xxx的位置由字符编码数的二进制表示的位填入。越靠右的x具有越少的特殊意义。只用最短的那个足够表达一个字符编码数的多字节串。注意在多字节串中，第一个字节的开头"1"的数目就是整个串中字节的数目。

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

# 五．UTF-16

1.描述：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。Unicode的编码空间从U+0000到U+10FFFF，共有1,112,064个码位（code point）可用来映射字符. Unicode的编码空间可以划分为17个平面（plane），每个平面包含216（65,536）个码位。17个平面的码位可表示为从U+xx0000到U+xxFFFF，其中xx表示十六进制值从0016到1016，共计17个平面。第一个平面称为基本多语言平面（Basic Multilingual Plane, BMP），或称第零平面（Plane 0）。其他平面称为辅助平面（Supplementary Planes）。基本多语言平面内，从U+D800到U+DFFF之间的码位区块是永久保留不映射到Unicode字符。UTF-16就利用保留下来的0xD800-0xDFFF区段的码位来对辅助平面的字符的码位进行编码。

2.来源：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](https://zh.wikipedia.org/wiki/ISO_10646)的附录C，而[RFC](https://zh.wikipedia.org/wiki/RFC)2781也定义了相似的做法。

3.编码方式：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。以下的例子有四个字符：“朱”（U+6731）、半角逗号（U+002C）、“聿”（U+807F）、“𪚥”（U+2A6A5）。

4．优点：可以快速定位字符，对于string.charAt(index)方法有着较好的支持。

5.缺点：英文字母也要四个字节来存储，空间浪费太大了。

6.占字节：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。

# 六．引用资料

https://en.wikipedia.org/wiki/GB\_2312

<https://en.wikipedia.org/wiki/Big5>

<https://en.wikipedia.org/wiki/UTF-8>

<https://en.wikipedia.org/wiki/UTF-16>

<https://en.wikipedia.org/wiki/Unicode>