

**学术论文报告**

**系 （院）：** 计算机科学学院 .

**专业班级：** 软工1501 .

**姓 名：** 余逗 .

**学 号：** 201503421 .

**目 录**

一、GB2312

二、big5

三、UTF-8

四、UTF-16

五、Unicode

1. **GB2312**

**1.来历**

**GB2312**是一个关键的正式注册的互联网名称[字符集](https://en.wikipedia.org/wiki/Character_set" \o "字符集)的的[中国人民共和国](https://en.wikipedia.org/wiki/People's_Republic_of_China" \o "中华人民共和国)，用于[简化](https://en.wikipedia.org/wiki/Simplified_chinese" \o "简体中文)中国字。**GB国际**标准简称国家标准，即*国家标准*。GB2312（1980）已被[GBK](https://en.wikipedia.org/wiki/GBK" \o "GBK)和[GB18030](https://en.wikipedia.org/wiki/GB18030" \o "GB18030)取代，其中包括附加字符，但GB2312仍然被广泛使用。

**2.产生原因**

虽然GB2312占地面积当代使用的字符的99％，[[1]](https://en.wikipedia.org/wiki/GB_2312" \l "cite_note-1)历史教科书和许多名字仍然超出范围。GB2312包括6763个中国字符（在两个层面：第一是通过读取布置，所述第二通过[自由基](https://en.wikipedia.org/wiki/Radical_(Chinese_characters)" \o "激进（汉字）)然后笔画的数目）与符号和标点，日语，沿着[假名](https://en.wikipedia.org/wiki/Kana" \o "假名)，在[希腊](https://en.wikipedia.org/wiki/Greek_alphabet" \o "希腊字母表)和[西里尔字母](https://en.wikipedia.org/wiki/Cyrillic_alphabets" \o "西里尔字母)，[注音](https://en.wikipedia.org/wiki/Zhuyin" \o "注音)，和双字节集的[拼音](https://en.wikipedia.org/wiki/Pinyin" \o "拼音)字母带声调的。所有网页的0.7％在2017年4月使用GB2312，比2010年1月的3.5％下降。[[2]](https://en.wikipedia.org/wiki/GB_2312" \l "cite_note-2)

## GB2312中的字符排列在94x94网格（如[ISO 2022中](https://en.wikipedia.org/wiki/ISO_2022" \o "ISO 2022)）中，每个字符的两字节编码点以kuten（或quwei）格式表示，该格式指定一行（ku或qu），并且行内的字符（单元格，十或六）。

行（编号从1到94）包含以下字符：

* 01-09，包括标点符号和其他特殊字符; 还有[平假名](https://en.wikipedia.org/wiki/Hiragana" \o "平假名)，[片假名](https://en.wikipedia.org/wiki/Katakana" \o "片假名)，[希腊文](https://en.wikipedia.org/wiki/Greek_alphabet" \o "希腊字母表)，[西里尔文](https://en.wikipedia.org/wiki/Cyrillic" \o "西里尔)，[拼音](https://en.wikipedia.org/wiki/Pinyin" \o "拼音)，[Bopomofo](https://en.wikipedia.org/wiki/Bopomofo" \o "汉语拼音)
* 16-55，第一架[汉字](https://en.wikipedia.org/wiki/Chinese_characters" \o "中国文字)，按[拼音](https://en.wikipedia.org/wiki/Pinyin" \o "拼音)排列。（3755个字符）。
* 56-87，汉字的第二个平面，根据激进和笔画排列。（3008个字符）。
* 88-89，进一步汉字。（103个字符）。仅限于GB / T 12345，而不是GB 2312。

行10-15和90-94未分配。

GB2312-80包含682个符号和6763个汉字。

**3、解决问题、字节数及描述**

在处理GB2312的程序中，[EUC-CN](https://en.wikipedia.org/wiki/EUC-CN" \o "EUC-CN)通常用作[字符编码](https://en.wikipedia.org/wiki/Character_encoding" \o "字符编码)（即外部存储），从而保持与[ASCII的](https://en.wikipedia.org/wiki/ASCII" \o "ASCII)兼容性。两个[字节](https://en.wikipedia.org/wiki/Bytes" \o "字节)用于表示[ASCII中](https://en.wikipedia.org/wiki/ASCII" \o "ASCII)未找到的每个字符。第一个字节的值为0xA1-0xF7（161-247），第二个字节的值为0xA1-0xFE（161-254）。由于所有这些范围都超出ASCII，如UTF-8，当使用EUC-CN时，可以检查一个字节是否是多字节构造的一部分，但是如果字节是第一个或最后一个字节，则不能。

与[UTF-8](https://en.wikipedia.org/wiki/UTF-8" \o "UTF-8)相比，GB2312（无论是本机还是在EUC-CN中编码）都具有更高的存储效率：而[UTF-8](https://en.wikipedia.org/wiki/UTF-8" \o "UTF-8)每个[CJK表意文字](https://en.wikipedia.org/wiki/CJK_Unified_Ideographs_(Unicode_block)" \o "CJK统一表意符号（Unicode块）)使用三个字节[[a]](https://en.wikipedia.org/wiki/GB_2312" \l "cite_note-4)，GB2312只使用两个。然而，GB2312并没有涵盖像Unicode那样多的表意文字。和GB2312是较少的能量效率，它需要[除法](https://en.wikipedia.org/wiki/Division_algorithm" \o "分算法)由94是比较昂贵的，而UTF-8使用一个便宜的[变速操作](https://en.wikipedia.org/wiki/Bitwise_operation" \o "按位操作)。

要将代码点映射到字节，将158（0x98）添加到代码点的行号以形成高字节，并添加158列代码点以形成低字节。行号是代码点整数除以94，该列的代码点[模数为](https://en.wikipedia.org/wiki/Modulo_operation" \o "模运算) 94。

例如，如果您的GB2312代码点为4566（“外”，表示外部），则高字节将为4566/94 + 158 = 206 = 0xCE，低字节将来自4566％94 + 158 = 212 = 0xD4。所以，完整的编码是0xCED4 = 52948。

### **4、兼容性**

[HZ](https://en.wikipedia.org/wiki/HZ_(character_encoding)" \o "HZ（字符编码）)是GB2312的另一个编码，主要用于[Usenet](https://en.wikipedia.org/wiki/Usenet" \o "新闻组)发布。的GBK / GB18030子集是与两个兼容[GBK](https://en.wikipedia.org/wiki/GBK" \o "GBK)和[GB18030](https://en.wikipedia.org/wiki/GB18030" \o "GB18030) ; GB2312.TXT是ftp.unicode.org [[5]](https://en.wikipedia.org/wiki/GB_2312" \l "cite_note-many-mappings-6)的官方实现，自2011年8月以来已经过时[[6]，](https://en.wikipedia.org/wiki/GB_2312" \l "cite_note-7)并且在2016年9月失踪。更多的供应商映射存在。[[5]](https://en.wikipedia.org/wiki/GB_2312" \l "cite_note-many-mappings-6)

截至2015年，Microsoft .NET Framework正在使用*子集*。[ICU](https://en.wikipedia.org/wiki/International_Components_for_Unicode" \o "Unicode的国际组件)，[[7]](https://en.wikipedia.org/wiki/GB_2312" \l "cite_note-8) iconv-1.14，[[8]](https://en.wikipedia.org/wiki/GB_2312" \l "cite_note-9) php-5.6，ActivePerl-5.20，Java 1.7和Python 3.4 [[9]](https://en.wikipedia.org/wiki/GB_2312" \l "cite_note-10)正在使用*GB2312.TXT*。Ruby 2.2与两个实现兼容; 它将冲突的字符内部转换*为子集*。W3C的技术建议规定了一个*GBK*编码被推断用于标注的流gb2312，而后者又使用了*GB18030*解码器。[[10]](https://en.wikipedia.org/wiki/GB_2312" \l "cite_note-11)

## 参考文献[ [编辑](https://en.wikipedia.org/w/index.php?title=GB_2312&action=edit&section=7" \o "编辑部分：参考)]

1. Hannas，William C.（1997）。*[亚洲正义困境](https://books.google.com/books?id=aJfv8Iyd2m4C&pg=PA264)*。夏威夷大学出版社。页。264。
2. [“使用字符编码的网站的历史趋势”](http://w3techs.com/technologies/history_overview/character_encoding)。*W3Techs*。
3. [“GB / T 12345”](http://examples.oreilly.com/9781565922242/AppF/gbt12345.pdf)（PDF）。
4. [“GB 2312-1980：信息技术 - 信息交换中文表意文字编码字符集（基本集）”](https://archive.org/details/GB2312-1980)。检索于 2016 年10月2日。
5. 海尔，布鲁诺。[“GB2312（转换表）”](http://haible.de/bruno/charsets/conversion-tables/GB2312.html)。检索年9月29日到2016年。
6. [“自述 - MAPPINGS / OBSOLETE / EASTASIA”](http://ftp.unicode.org/Public/MAPPINGS/OBSOLETE/EASTASIA/ReadMe.txt)。2001年8月9日。检索年9月29日到2016年。
7. [“java-EUC\_CN-1.3\_P.ucm”](https://ssl.icu-project.org/trac/browser/data/trunk/charset/data/ucm/java-EUC_CN-1.3_P.ucm)。检索年9月29日到2016年。
8. [“libiconv的：LIB / gb2312.h”](http://git.savannah.gnu.org/cgit/libiconv.git/tree/lib/gb2312.h" \l "n26)。*GNU萨凡纳*。检索年9月29日到2016年。
9. [“问题24036”](http://bugs.python.org/issue24036)。*Python Bug Tracker*。
10. [“编码§名称和标签”](https://www.w3.org/TR/encoding/" \l "names-and-labels)。W3C。检索年9月29日到2016年。

**二、Big5**

## **1、来历**

无力[ASCII](https://en.wikipedia.org/wiki/ASCII" \o "ASCII)如用于中国支持大字符集，日语和韩语导致政府和工业界寻找创造性的解决方案，使他们的语言将计算机上呈现。各种特设和通常专有的输入法导致了开发标准系统的努力。其结果是，BIG5编码被定义[财团法人资讯工业](https://en.wikipedia.org/wiki/Institute_for_Information_Industry" \o "信息产业研究所)台湾1984年“中文”这个名字是在认识到标准，从台湾最大的IT公司五个合作出现了：[宏碁](https://en.wikipedia.org/wiki/Acer_Inc." \o "宏碁公司)（[宏碁](https://zh.wikipedia.org/wiki/%E5%AE%8F%E7%A2%81" \o "ZH：宏碁)）; [神达](https://en.wikipedia.org/wiki/MiTAC" \o "神达)（神通）; 佳佳（ZERO ONE Technology）（零壹或[01tech](http://www.01tech.com/)）; 佳佳 和[第一国际电脑（FIC）](https://en.wikipedia.org/wiki/First_International_Computer" \o "第一届国际电脑)（大众）。

Big5在台湾和全球范围内迅速普及，通过采用几种商业软件包，特别是[E-TEN](https://en.wikipedia.org/wiki/E-TEN" \o "E-TEN)中文[DOS](https://en.wikipedia.org/wiki/DOS" \o "DOS)输入系统（[ETen中文系统](https://en.wikipedia.org/wiki/ETen_Chinese_System" \o "ETen中文系统)），采用传统汉字。在[中华民国](https://en.wikipedia.org/wiki/Republic_of_China" \o "中国共和国)政府宣布**中文**作为他们在20世纪80年代的标准，因为它是，届时，*事实上*电脑上使用传统的中国标准。

**2、产生原因及字节**

Big5字符集的某些变体，例如[HKSCS](https://en.wikipedia.org/wiki/HKSCS" \o "香港增补字符集)，使用扩展范围的引导字节，包括0x81到0xA0范围内的值（类似于Shift JIS）。

如果第二个字节不在正确的范围内，[行为是未定义的](https://en.wikipedia.org/wiki/Undefined_behaviour" \o "未定义的行为)（即，从系统到系统不同）。

单个Big5代码的数值经常被给出为4位十六进制数，它描述了构成Big5代码的两个字节，就像这两个字节是16位数字的[大端](https://en.wikipedia.org/wiki/Big_endian" \o "大端)表示一样。例如，全角空格的Big5代码，即0xa1 0x40的字节通常写为0xa140或仅写入A140。

严格来说，Big5编码只包含DBCS字符。然而，在实践中，Big5代码总是与未指定的系统相关的单字节字符集（[ASCII](https://en.wikipedia.org/wiki/ASCII" \o "ASCII)或8位字符集，如[代码页437](https://en.wikipedia.org/wiki/Code_page_437" \o "代码页437)）一起使用，以便您可以找到DBCS的混合Big5编码文本中的字符和单字节字符。不属于双字节字符的0x00至0x7f范围内的字节假定为单字节字符。（有关此问题的更详细说明，请参阅下文“匹配SBCS”的讨论。）

不符合双字节字符的允许值之外的非ASCII单字节的含义因系统而异。在旧的基于MSDOS的系统中，它们可能显示为8位字符; 在现代系统中，它们可能会产生不可预测的结果或产生错误。

在原来的Big5中，编码分为不同的区域：

|  |  |
| --- | --- |
| 0x8140到0xa0fe | 保留为用户定义的字符造字 |
| 0xa140到0xa3bf | “图形字符”图形码 |
| 0xa3c0到0xa3fe | 保留，*不是*用户定义的字符 |
| 0xa440到0xc67e | 常用字常用字 |
| 0xc6a1到0xc8fe | 保留用于用户定义的字符 |
| 0xc940到0xf9d5 | 不常用的字符次常用字 |
| 0xf9d6到0xfefe | 保留用于用户定义的字符 |

“图形字符”实际上包括标点符号，部分标点符号（例如，半条痕迹的一半，省略号的一半;见下文），[标记](https://en.wikipedia.org/wiki/Dingbat" \o "白带)，外来字符和其他特殊字符（例如，表示“全宽”形式，数字对于[苏州码子](https://en.wikipedia.org/wiki/Suzhou_numerals" \o "苏州数字)，[注音符号](https://en.wikipedia.org/wiki/Zhuyin" \o "注音)等）

在大多数供应商扩展中，扩展字符被放置在为用户定义字符保留的各个区域中，每个区域通常被认为与前一个区域相关联。例如，额外的“图形字符”（例如，标点符号）将被预期放置在0xa3c0-0xa3fe范围内，并且额外的标志将被放置在0xc6a1-0xc8fe或0xf9d6-0xfefe范围内。有时，这是不可能的，因为大量的扩展字符被添加; 例如，[西里尔](https://en.wikipedia.org/wiki/Cyrillic" \o "西里尔)字母和日本[假名](https://en.wikipedia.org/wiki/Kana" \o "假名)已被放置在与“常用字符”相关联的区域中。

1. **解决问题**

一个单独的Big5代码并不总是代表一个完整的语义单元。标志的Big5代码总是标识符，但“图形字符”部分中的代码并不总是完整的“图形字符”。Big5编码是字符或部分字符的特殊图形表示，恰好适合两个等宽ASCII字符占用的空间。这是CJK（中文，日文和韩文）通常使用的双字节字符集的属性，而不是Big5的独特问题。

（以上可能需要一些解释，因为它在*理论上*是不正确的：返回当文本模式个人计算仍然是常态时，字符通常表示为单字节，每个字符在屏幕上一个位置，有因此，坚持双字节字符必须在屏幕上占据两个位置的实际原因，即现有的美制软件将可在基于DBCS的系统中进行修改，如果一个字符可以任意数量的屏幕位置，假定一个*字节的*软件的文本需要一个屏幕位置会产生不正确的输出。当然，如果一台电脑从来不用处理这个文字屏幕，制造商就不会执行这个人为的限制; 苹果Macintosh就是一个例子。然而，编码本身必须设计成使其在基于文本屏幕的系统上正常工作。）

为了说明这一点，请考虑Big5代码0xa14b（...）。对于英语的人来说，这看起来像省略号，而Unicode标准就是这样标识的; 然而，在中文中，省略号由两个汉字（......）的空格组成的六个点组成，实际上中文省略号并没有Big5代码，而Big5代码0xa14b只是中文省略号的一半。它只代表省略号的一半，因为整个省略号应该占用两个汉字的空间，而在许多DBCS系统中，一个DBCS字符必须占据一个汉字的空间。

在Big5中编码的字符并不总是表示可以在纯文本文件中使用的东西; 一个例子是“引用标记”（0xa1ca，﹋），当使用时，需要在文学作品的标题下排版。另一个例子是[苏州数字](https://en.wikipedia.org/wiki/Suzhou_numerals" \o "苏州数字)，这是一种[科学符号](https://en.wikipedia.org/wiki/Scientific_notation" \o "科学计数法)的形式，需要以至少两行的二维形式进行编排。

1. **兼容性**

实际上，如果没有匹配的[单字节字符集（SBCS）](https://en.wikipedia.org/wiki/SBCS" \o "SBCS)，Big5不能使用; 这主要是与兼容性原因。但是，与其他CJK DBCS字符集的情况一样，从未指定要使用的SBCS。Big5一直被定义为DBCS，尽管在使用时它必须与合适的*未指定的* SBCS 配对，因此被用作某些人称之为[MBCS](https://en.wikipedia.org/wiki/Variable-width_encoding" \o "可变宽度编码) ; 然而，如Big3所定义的那样，它绝对是一个DBCS。

SBCS使用未指定意味着所使用的SBCS在理论上可以随系统而变化。如今，ASCII是唯一可能使用的SBCS。然而，在旧的基于[DOS](https://en.wikipedia.org/wiki/MS-DOS" \o "MS-DOS)的系统中，[代码Page 437](https://en.wikipedia.org/wiki/Code_Page_437" \o "代码页437) - 其控制代码区域中包含位置127的额外特殊符号更为常见。然而，在具有中文语言套件的Macintosh系统上，或者在运行cxterm终端仿真器的Unix系统上，与Big5配对的SBCS不会是Code Page 437。

在Big5的有效范围之外，旧的基于DOS的系统将根据与该系统上的Big5配对的SBCS进行常规的解释。在这样的系统中，例如字符127到160很可能不会被避免，因为它们会产生无效的Big5，而是使用它们，因为它们在代码页437中是有效的。

因此，Big5作为由Big5的DBCS加上ASCII的SBCS组成的MBCS的现代表征因此历史上是不正确的，并且具有潜在的缺陷，因为匹配的SBCS的选择在理论上仍然与使用Big5的风味完全无关。

## 参考文献

1. [中文mac字符集](http://chinesemac.org/pages/character_sets.html)
2. [使用字符编码的历史趋势，2016年12月](http://w3techs.com/technologies/history_overview/character_encoding)
3. [内的常见问题解答](http://w3techs.com/faq)
4. [狗爷语录»博客存档»什么是代码页951（CP951）？](http://me.abelcheung.org/2006/09/12/what-is-cp951/)
5. 黄国书 [“Chinasea 1.0中国海字集”](http://ftp.isu.edu.tw/pub/Windows/Chinese/font/chinasea/csw10_exp.txt)。ISU FTP。检索2016-12-05。

* Lunde，Ken（1999）。*CJKV信息处理*（第一版）。O'Reilly and Associates，Inc. [ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number" \o "国际标准书号) [1-56592-224-7](https://en.wikipedia.org/wiki/Special:BookSources/1-56592-224-7" \o "特别声明：BookSources / 1-56592-224-7)。

## 外部链接

* [Mozilla和Big5编码系列](http://moztw.org/docs/big5/)：对Big5编码的概述，其中包含每个扩展的代码图和相关的Firefox错误。（繁体中文）
* [Big5字符代码表](http://ash.jp/code/cn/big5tbl.htm)
* [中文字符代码：](http://kura.hanazono.ac.jp/paper/codes.html) Christian Wittern [的更新](http://kura.hanazono.ac.jp/paper/codes.html)
* [CNS 11643官方网站](http://www.cns11643.gov.tw/AIDB/welcome.do)在“中国信息代码”部分中有关于Big5e字符集（Big5的扩展版本）的信息
* [Big5简介](http://www.cns11643.gov.tw/web/big5/)包含扩展名之间的差异。
* [BigC的图形视图在ICU的转换器资源管理器](http://demo.icu-project.org/icu-bin/convexp?conv=Big5)
* [教育部标准字体](http://depart.moe.edu.tw/ED2400/News_Content.aspx?n=8940E5C0456177C3&sms=893AAA1CBFE149DE&s=DFBE7BE3EE0DB6AE)下载页面的台湾教育部字体
* [文献处理实验室](http://www.sinica.edu.tw/~cdp/)下载CDP字体的页面
* [香港补充字符集信息可](http://www.ogcio.gov.hk/en/business/tech_promotion/ccli/hkscs/)下载的HKSCS文件及字体
* [香港参考宋体](http://glyph.iso10646hk.net/chinese/download_001.jsp) Dynalab（华康科技有限公司）的HKSCS字体的下载页面。
* [微软Windows Codepage 950](http://www.microsoft.com/globaldev/reference/dbcs/950.mspx)（繁体中文Big5）
* [on.cc](http://on.cc/orimain/orisunfaq/hkfonts_bottom.html) OPG字体的下载页面
* [中国海字集视窗版（v3.0）下载网页](http://www.mimosapudica.org/ChinaSea/index_c.html) ChinaSea字体下载页面
* [Big5编码概述](http://www.b-t.asia/chinese/big5.php)

1. **UTF-8/16**

**1、来历**

**UTF-8**是一种[字符编码，](https://en.wikipedia.org/wiki/Character_encoding" \o "字符编码)可以使用一到四个8位[字节](https://en.wikipedia.org/wiki/Byte" \o "Byte)对[Unicode中的](https://en.wikipedia.org/wiki/Unicode" \o "Unicode)所有1,112,064 [[1]个](https://en.wikipedia.org/wiki/UTF-8" \l "cite_note-1)有效[代码点](https://en.wikipedia.org/wiki/Code_point" \o "代码点)进行编码。[[2]](https://en.wikipedia.org/wiki/UTF-8" \l "cite_note-2)编码由Unicode标准定义，最初由[Ken Thompson](https://en.wikipedia.org/wiki/Ken_Thompson" \o "Ken Thompson)和[Rob Pike设计](https://en.wikipedia.org/wiki/Rob_Pike" \o "Rob Pike)。[[3]](https://en.wikipedia.org/wiki/UTF-8" \l "cite_note-:0-3)[[4]](https://en.wikipedia.org/wiki/UTF-8" \l "cite_note-4)该名称源自*Unicode*（或*通用编码字符集*）*转换格式 - 8位*。[[5]](https://en.wikipedia.org/wiki/UTF-8" \l "cite_note-5)

**2、兼容性**

它被设计为与[ASCII的](https://en.wikipedia.org/wiki/ASCII" \o "ASCII)[向后兼容性](https://en.wikipedia.org/wiki/Backward_compatibility" \o "向后兼容性)。较低数值的代码点往往更频繁地出现，使用较少的字节进行编码。Unicode的前128个字符与ASCII对应一对一，使用与ASCII相同的二进制值的单个八位字节进行编码，因此有效的ASCII文本也是有效的UTF-8编码Unicode。由于在将非ASCII码代码编码为UTF-8时不会出现ASCII字节，所以UTF-8在大多数以特殊方式解释某些ASCII字符的编程和文档语言中是安全的，例如文件名中的“ [/](https://en.wikipedia.org/wiki/Slash_(punctuation)" \o "Slash (punctuation)) ” [\](https://en.wikipedia.org/wiki/Backslash" \o "Backslash) “ [转义序列](https://en.wikipedia.org/wiki/String_literal" \l "Escape_sequence" \o "String literal)，[printf中的](https://en.wikipedia.org/wiki/Printf" \o "Printf) ”％“ 。

由于在2003年将Unicode代码空间限制为21位值，因此UTF-8被定义为以一到四个字节的方式对代码点进行编码，具体取决于代码点数值中的有效位数。下表显示了编码的结构。该x字符由代码点的位所取代。如果有效位数不超过7，则第一行适用; 如果不超过11​​＆nps;位，第二行适用，等等。

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 数 字节 | 代码点的位 | 第一个 代码点 | 最后的 代码点 | 字节1 | 字节2 | 字节3 | 字节4 |
| 1 | 7 | U + 0000 | U + 007F | 0xxxxxxx |  |  |  |
| 2 | 11 | U + 0080 | U + 07FF | 110xxxxx | 10xxxxxx |  |  |
| 3 | 16 | U + 0800 | U + FFFF | 1110xxxx | 10xxxxxx | 10xxxxxx |  |
| 4 | 21 | U + 10000 | U + 10FFFF | 11110xxx | 10xxxxxx | 10xxxxxx | 10xxxxxx |

前128个字符（US-ASCII）需要一个字节。接下来的1,920个字符需要两个字节进行编码，其中包括几乎所有[拉丁字母字母](https://en.wikipedia.org/wiki/Latin-script_alphabet" \o "拉丁字母字母)的剩余部分，还有[希腊语](https://en.wikipedia.org/wiki/Greek_alphabet" \o "希腊字母表)，[西里尔语](https://en.wikipedia.org/wiki/Cyrillic_script" \o "西里尔文脚本)，[科普特语](https://en.wikipedia.org/wiki/Coptic_alphabet" \o "科普特字母表)，[亚美尼亚语](https://en.wikipedia.org/wiki/Armenian_alphabet" \o "亚美尼亚字母表)，[希伯来语](https://en.wikipedia.org/wiki/Hebrew_alphabet" \o "希伯来文字母)，[阿拉伯语](https://en.wikipedia.org/wiki/Arabic_alphabet" \o "阿拉伯字母表)，[叙利亚语](https://en.wikipedia.org/wiki/Syriac_alphabet" \o "叙利亚字母表)，[Thaana](https://en.wikipedia.org/wiki/Thaana" \o "塔纳文)和[N'Ko](https://en.wikipedia.org/wiki/N'Ko_alphabet" \o "N'Ko字母表)字母表以及[组合变音符号标志](https://en.wikipedia.org/wiki/Combining_Diacritical_Marks" \o "组合变音标记)。[基本多语言平面](https://en.wikipedia.org/wiki/Basic_Multilingual_Plane" \o "基本多语言平面)的其余部分需要三个字节，其中包含几乎所有常用的字符[[11]，](https://en.wikipedia.org/wiki/UTF-8" \l "cite_note-unicode-ch02-bmp-11)其中包括大多数[中文，日文和韩文](https://en.wikipedia.org/wiki/CJK_characters" \o "CJK字符)。[Unicode](https://en.wikipedia.org/wiki/Plane_(Unicode)" \o "平面（Unicode）)的[其他平面中的字符](https://en.wikipedia.org/wiki/Plane_(Unicode)" \o "平面（Unicode）)需要四个字节，其中包括较不常见的[CJK字符](https://en.wikipedia.org/wiki/CJK_characters" \o "CJK字符)，各种历史脚本，数学符号和[表情符号](https://en.wikipedia.org/wiki/Emoji" \o "表情符号)（象形符号）。

这种编码的一些重要特征如下：

* 向后兼容性：与ASCII的向后兼容性和设计用于处理ASCII编码文本的大量软件是UTF-8设计背后的主要动力。在UTF-8中，值范围为0到127的单字节直接映射到ASCII范围内的Unicode代码点。这个范围内的单字节表示字符，就像ASCII一样。此外，7位字节（最高有效位为0的字节）不会出现在多字节序列中，没有有效的多字节序列解码为ASCII码点。7位字节的序列都是有效的ASCII和有效的UTF-8，在任一解释下都代表相同的字符序列。因此，UTF-8流中的7位字节表示流中的所有且仅有ASCII字符。因此，许多文本处理器，解析器，协议，文件格式，文本显示程序等，使用ASCII字符进行格式化和控制的目的将继续按照预期的方式处理UTF-8字节流作为单字节字符序列，而不解码多字节序列。处理转换的ASCII字符（如标点符号，空格和控制字符）永远不会被编码为多字节序列。因此，这样的处理器可以安全地忽略或传递多字节序列，而不对其进行解码。例如，可以使用ASCII空格 并且控制字符将永远不会被编码为多字节序列。因此，这样的处理器可以安全地忽略或传递多字节序列，而不对其进行解码。例如，可以使用ASCII空格 并且控制字符将永远不会被编码为多字节序列。因此，这样的处理器可以安全地忽略或传递多字节序列，而不对其进行解码。例如，可以使用ASCII空格将UTF-8流[标记](https://en.wikipedia.org/wiki/Tokenize" \o "令牌化)为单词; 可以使用ASCII换行来将UTF-8流分割成行; 和ASCII NUL字符可用于将UTF-8编码的数据分割为空终止的字符串。类似地，库函数（如“printf”）使用的许多格式字符串将正确地处理UTF-8编码的输入参数。
* 回退和自动检测： UTF-8提供7位ASCII的向后兼容性，但是许多软件和数据使用8位[扩展ASCII](https://en.wikipedia.org/wiki/Extended_ASCII" \o "扩展ASCII)编码，在采用Unicode之前设计来表示欧洲语言的字符集。UTF-8的一部分受欢迎程度是因为它为这些提供了一种向后兼容的形式。错误地接收[扩展ASCII](https://en.wikipedia.org/wiki/Extended_ASCII" \o "扩展ASCII)文件作为输入的UTF-8处理器可以使用Unicode [Latin-1补](https://en.wikipedia.org/wiki/Latin-1_Supplement" \o "拉丁文1补编)码中的相应代码点“退回”或替换8位字节当8位字节出现在有效的多字节序列之外时，则阻塞。尽管如此，扩展ASCII编码中的8位字符通常不具有UTF-8多字节序列的正确形式。这是因为在UTF-8中引入多字节序列的8位字节在通用扩展ASCII编码中主要是重音字母（大部分是元音），UTF-8连续字节是标点符号和符号字符。要显示为有效的UTF-8多字节序列，一系列2到4个扩展ASCII 8位字符必须是符号和重音字母的不寻常组合。简而言之，扩展的ASCII字符序列看起来像有效的UTF-8多字节序列是不太可能的。回退错误将是假阴性，这些将是罕见的。此外，在许多应用中，例如文本显示，错误回退的后果通常很轻微。只有可读性受到影响，而不是显着。这两件事情使后备可行，如果有些不完美。实际上，如下面进一步讨论的那样[HTML5](https://en.wikipedia.org/wiki/HTML5" \o "HTML5)标准要求假设UTF-8数据中的错误字节被替换，假定它们是[Windows-1252](https://en.wikipedia.org/wiki/Windows-1252" \o "Windows的1252)字符。有效多字节序列之外的无效8位字符的存在也可以用于“自动检测”编码实际上是扩展ASCII编码而不是UTF-8，并相应地对其进行解码。UTF-8流可能只是包含错误，导致自动检测方案产生误报; 但在大多数情况下，自动检测是成功的，尤其是较长的文本，并被广泛使用。
* [前缀代码](https://en.wikipedia.org/wiki/Prefix_code" \o "前缀代码)：第一个字节表示序列中的字节数。从流中读取可以立即解码每个单独的完全接收的序列，而无需等待下一个序列的第一个字节或流末尾指示。多字节序列的长度很容易由人确定，因为它只是前导字节中高1的数目。如果流结束中间序列，则不正确的字符将不被解码。
* [自同步](https://en.wikipedia.org/wiki/Self-synchronizing_code" \o "自同步代码)：前导字节和连续字节不共享值（连续字节从10单字节开始，0长字节从头开始11）开始。这意味着搜索不会意外地从另一个字符的中间找到一个字符的序列。这也意味着可以通过备份最多3个字节从一个随机位置找到一个字符的开始来找到前导字节。如果流中间序列启动，则不正确的字符将不会被解码，较短的序列将不会出现在较长的序列中。
* 排序顺序：前导字节的选定值和连续字节具有高位位的事实首先意味着可以通过对相应字节序列进行排序，以代码顺序排列UTF-8字符串列表。

## 参考文献[ [编辑](https://en.wikipedia.org/w/index.php?title=UTF-8&action=edit&section=19" \o "编辑部分：参考)]

1. 17×216= 1,114,112个代码点减去2,048个技术无效代理代码点
2. 一组8位在Unicode标准中称为[八位字节](https://en.wikipedia.org/wiki/Octet_(computing)" \o "八位字节（计算）)。
3. 来自：“Rob”Commander'Pike“，日期：2003年4月30日，*... UTF-8设计在我的眼前在9月份的一个晚上，在新泽西州的餐厅里放了一个placemat ... 1992年...那个晚上，Ken写了包装和打包代码，我开始撕裂到C和图形库。第二天所有的代码都完成了...*
4. 派克，罗布 汤普森，肯（1993）。“Hello World orΚαλημέρακόσμε或こんにちは世界”。*[1993年冬季会议论文集](https://www.cl.cam.ac.uk/~mgk25/ucs/UTF-8-Plan9-paper.pdf)* （PDF）。
5. “第2章总体结构”。*[Unicode标准](http://www.unicode.org/versions/Unicode6.0.0/)*（6.0版）。Mountain View，加州，美国：[Unicode联盟](https://en.wikipedia.org/wiki/Unicode_Consortium" \o "Unicode联盟)。[ISBN](https://en.wikipedia.org/wiki/International_Standard_Book_Number" \o "国际标准书号) [978-1-936213-01-6](https://en.wikipedia.org/wiki/Special:BookSources/978-1-936213-01-6" \o "特别声明：BookSources / 978-1-936213-01-6)。
6. [戴维斯，马克](https://en.wikipedia.org/wiki/Mark_Davis_(Unicode)" \o "马克·戴维斯（Unicode）)（2010-01-28）。[“Unicode接近网路的50％”](http://googleblog.blogspot.com/2010/01/unicode-nearing-50-of-web.html)。*谷歌官方博客*。[Google](https://en.wikipedia.org/wiki/Google" \o "谷歌)。检索2010-12-05。
7. [“使用字符编码的历史趋势”](http://w3techs.com/technologies/history_overview/character_encoding)。检索2017-09-09。
8. [“UTF-8使用统计”](http://trends.builtwith.com/encoding/UTF-8)。BuiltWith。检索2011-03-28。
9. [“在互联网邮件中使用国际字符”](http://www.imc.org/mail-i18n.html)。互联网邮件联盟。1998-08-01。检索2007-11-08。
10. “指定文档的字符编码”，*[HTML5](http://www.w3.org/TR/html5/document-metadata.html" \l "charset)*，[万维网联盟](https://en.wikipedia.org/wiki/World_Wide_Web_Consortium" \o "万维网联盟)，2014-06-17，检索2014-07-30
11. 艾伦，朱莉D。安德森，德博拉 贝克尔 库克，理查德，编辑。（2012年）。“Unicode标准版本6.1”。Mountain View，California：Unicode Consortium。基本多语言平面（BMP或平面0）包含世界上所有现代脚本的常用字符以及许多历史和罕见字符。到目前为止，几乎所有文本数据的大多数Unicode字符都可以在BMP中找到。
12. 马林，马文（2000-10-17）。[“Web服务器文件夹遍历MS00-078”](http://www.sans.org/resources/malwarefaq/wnt-unicode.php)。
13. [“国家漏洞数据库 - CVE-2008-2938摘要”](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-2938)。
14. Yergeau，F.（2003）。[“RFC 3629 - UTF-8，ISO 10646的转换格式”](https://tools.ietf.org/html/rfc3629)。[互联网工程任务组](https://en.wikipedia.org/wiki/Internet_Engineering_Task_Force" \o "互联网工程任务组)。检索2015-02-03。
15. [Java的DataInput IO接口](https://docs.oracle.com/javase/8/docs/api/java/io/DataInput.html)
16. [“系统字符接口中的不可解码字节”](https://www.python.org/dev/peps/pep-0383/)。*python.org*。2009-04-22。检索2014-08-13。
17. [库恩，马库斯](https://en.wikipedia.org/wiki/Markus_Kuhn_(computer_scientist)" \o "马库斯·库恩（计算机科学家）)（2000-07-23）。[“在解码器中替换畸形的UTF-8序列”](https://archive.fo/20150315055005/http://permalink.gmane.org/gmane.comp.internationalization.linux/920)。于2015-03-15从[原来](http://permalink.gmane.org/gmane.comp.internationalization.linux/920)存档。检索2014-09-25。
18. Sittler，B（2006-04-02）。[“二进制与UTF-8，为什么它不需要”](https://web.archive.org/web/20140723192908/http://bsittler.livejournal.com/10381.html)。从[原来](http://bsittler.livejournal.com/10381.html)的2014-07-23存档。检索2014-09-25。
19. [“字符集”](http://www.iana.org/assignments/character-sets)。[互联网号码分配机构](https://en.wikipedia.org/wiki/Internet_Assigned_Numbers_Authority" \o "互联网号码分配机构)。2013-01-23 。检索2013-02-08。
20. Dürst，马丁。[“设置HTTP字符集参数”](http://www.w3.org/International/O-HTTP-charset)。[W3C](https://en.wikipedia.org/wiki/W3C" \o "W3C)。检索2013-02-08。
21. [“BOM - suikawik​​i”](http://suika.fam.cx/~wakaba/wiki/sw/n/BOM)（日文）。检索2013-04-26。
22. [戴维斯，马克](https://en.wikipedia.org/wiki/Mark_Davis_(Unicode)" \o "马克·戴维斯（Unicode）)。[“Unicode格式”](https://web.archive.org/web/20050506211548/http://www-128.ibm.com/developerworks/library/utfencodingforms/index.html)。[IBM](https://en.wikipedia.org/wiki/IBM" \o "IBM)。从存档的[原始](http://www-128.ibm.com/developerworks/library/utfencodingforms/index.html)上2005-05-06。检索2013-09-18。
23. [PCL符号集](http://pclhelp.com/pcl-symbol-sets/)
24. [“接口java.io.DataInput的Java SE文档，修改的UTF-8上的小节”](http://docs.oracle.com/javase/8/docs/api/java/io/DataInput.html" \l "modified-utf-8)。[Oracle公司](https://en.wikipedia.org/wiki/Oracle_Corporation" \o "甲骨文公司)。2015年。检索2015-10-16。
25. [“Java虚拟机规范，第4.4.7节：”CONSTANT\_Utf8\_info结构“ ”](http://docs.oracle.com/javase/specs/jvms/se8/html/jvms-4.html" \l "jvms-4.4.7)。[Oracle公司](https://en.wikipedia.org/wiki/Oracle_Corporation" \o "甲骨文公司)。2015年。检索2015-10-16。Java虚拟机UTF-8字符串从未有嵌入的空值。
26. [“的Java对象序列化规范，第6章：对象序列化流协议，第2节：流元素”](http://docs.oracle.com/javase/8/docs/platform/serialization/spec/protocol.html" \l "a8299)。[Oracle公司](https://en.wikipedia.org/wiki/Oracle_Corporation" \o "甲骨文公司)。2010年。检索2015-10-16。[...]以修改的UTF-8编码。
27. [“Java本机接口规范，第3章：JNI类型和数据结构，部分：修改的UTF-8字符串”](https://docs.oracle.com/javase/8/docs/technotes/guides/jni/spec/types.html" \l "modified_utf_8_strings)。[Oracle公司](https://en.wikipedia.org/wiki/Oracle_Corporation" \o "甲骨文公司)。2015年。检索2015-10-16。JNI使用修改的UTF-8字符串来表示各种字符串类型。
28. [“Java虚拟机规范，第4.4.7节：”CONSTANT\_Utf8\_info结构“ ”](http://docs.oracle.com/javase/specs/jvms/se8/html/jvms-4.html" \l "jvms-4.4.7)。[Oracle公司](https://en.wikipedia.org/wiki/Oracle_Corporation" \o "甲骨文公司)。2015年。检索2015-10-16。[...]此格式与“标准”UTF-8格式之间的差异。
29. [“艺术与Dalvik的”](http://source.android.com/tech/dalvik/dex-format.html)。*Android开源项目*。检索2013-04-09。他的dex格式以事实上的标准修改的UTF-8格式（以下称为MUTF-8）对其字符串数据进行编码。
30. [“Tcler的维基：UTF-8逐位（修订版6）”](http://wiki.tcl.tk/_/revision?N=1211&V=6)。2009-04-25。检索2009-05-22。在正统UTF-8中，NUL字节（\ x00）由NUL字节表示。[...]但是[...]我们[...]需要[...]字符串中的NUL字节[...]
31. Sapin，Simon（2016-03-11）[2014-09-25] [“WTF-8编码”](https://simonsapin.github.io/wtf-8/)。[存档](https://web.archive.org/web/20160524180037/https://simonsapin.github.io/wtf-8/)于原来的第二十四号。检索2016年5月24日。
32. [“WTF-8.com”](http://wtf-8.com/)。2006-05-18。检索2016年6月21日。
33. Speer，Rob（2015-05-21）。[“ftfy（修正你的文本）4.0：改变更少和修复更多”](https://blog.luminoso.com/2015/05/21/ftfy-fixes-text-for-you-4-0-changing-less-and-fixing-more/)。检索2016年6月21日。
34. [“WTF-8，代码页1252的转换格式”](http://www-uxsup.csx.cam.ac.uk/~fanf2/hermes/doc/qsmtp/draft-fanf-wtf8.html)。*www-uxsup.csx.cam.ac.uk*。已回覆2016-10-12。
35. [“Unicode标准 - 第2章”](http://www.unicode.org/versions/Unicode6.0.0/ch02.pdf)（PDF）。页。30。
36. [“附录F. FSS-UTF /文件系统安全UCS转换格式”](http://www.unicode.org/versions/Unicode1.1.0/appF.pdf)（PDF）。*Unicode标准1.1*。从原来的06-06-07[存档](https://web.archive.org/web/20160607215950/http://www.unicode.org/versions/Unicode1.1.0/appF.pdf) （PDF）。检索2016年6月7日。
37. 惠斯勒，肯尼斯（2001-06-12）。[“FSS-UTF，UTF-2，UTF-8和UTF-16”](http://unicode.org/mail-arch/unicode-ml/y2001-m06/0318.html)。从原来的06-06-07[存档](https://web.archive.org/web/20160607220249/http://unicode.org/mail-arch/unicode-ml/y2001-m06/0318.html)。检索2006-06-07。
38. [派克，罗布](https://en.wikipedia.org/wiki/Rob_Pike" \o "罗伯·派克)（2003-04-30）。[“UTF-8历史”](http://www.cl.cam.ac.uk/~mgk25/ucs/utf-8-history.txt)。检索2012-09-07。
39. [派克，罗布](https://en.wikipedia.org/wiki/Rob_Pike" \o "罗伯·派克)（2012-09-06）。[“UTF-8昨天变了20岁”](https://plus.google.com/u/0/101960720994009339267/posts/Rz1udTvtiMg)。检索2012-09-07。
40. [戴维斯，马克](https://en.wikipedia.org/wiki/Mark_Davis_(Unicode)" \o "马克·戴维斯（Unicode）)（2008-05-05）。[“移动到Unicode 5.1”](http://googleblog.blogspot.com/2008/05/moving-to-unicode-51.html)。检索2013-03-01。
41. **Unicode**

**1、来历**

Unicode具有超越传统字符编码的限制的明确目标，例如[ISO 8859](https://en.wikipedia.org/wiki/ISO_8859" \o "ISO 8859)标准定义的限制，在世界各国广泛使用，但在很大程度上彼此不兼容。许多传统字符编码有一个共同的问题，它们允许双语计算机处理（通常使用[拉丁字符](https://en.wikipedia.org/wiki/Latin_character" \o "拉丁字符)和本地脚本），但不是多语言计算机处理（任意脚本的计算机处理彼此混合）。

Unicode在意图中编码底层字符 - [字形](https://en.wikipedia.org/wiki/Grapheme" \o "字形)和类似字形的单元，而不是这些字符的变体[字形](https://en.wikipedia.org/wiki/Glyph" \o "雕文)（渲染）。在[汉字](https://en.wikipedia.org/wiki/Chinese_character" \o "汉字)的情况下，这有时会导致将基础字符与其变体字形区分开来的争议（见[韩统一](https://en.wikipedia.org/wiki/Han_unification" \o "汉统一)）。

在文本处理中，Unicode承担着为每个字符提供唯一的*代码点* - [数字](https://en.wikipedia.org/wiki/Number" \o "数)，而不是字形。换句话说，Unicode以抽象的方式表示一个字符，并将可视化渲染（大小，形状，[字体](https://en.wikipedia.org/wiki/Font" \o "字形)或样式）留给其他软件，如[Web浏览器](https://en.wikipedia.org/wiki/Web_browser" \o "网页浏览器)或[文字处理器](https://en.wikipedia.org/wiki/Word_processor" \o "文字处理器)。然而，这个简单的目标变得复杂，因为Unicode设计师的让步，希望能够更快速地采用Unicode。

前256个代码点与[ISO-8859-1](https://en.wikipedia.org/wiki/ISO-8859-1" \o "ISO-8859-1)的内容相同，从而使现有的西方文本变得微不足道。许多本质上相同的字符在不同的代码点被多次编码，以保留传统编码所使用的区别，因此允许从那些编码转换为Unicode（并返回），而不会丢失任何信息。例如，代码点的“ [全宽形式](https://en.wikipedia.org/wiki/Fullwidth_form" \o "全宽形式) ”部分包含与拉丁字母主要部分分离的完整拉丁字母，因为中文，日文和韩文（[CJK](https://en.wikipedia.org/wiki/CJK" \o "CJK)）字体，这些拉丁字符的渲染与CJK的宽度相同[表意文字](https://en.wikipedia.org/wiki/Ideograph" \o "表意文字)，而不是宽度的一半。

根据与经验[施乐字符代码标准](https://en.wikipedia.org/wiki/Xerox_Character_Code_Standard" \o "施乐字符代码标准)（的Xccs）自1980年以来，[[2]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-unicode-88-2)的Unicode日期的起源到1987年，当[乔·贝克尔](https://en.wikipedia.org/wiki/Joe_Becker_(Unicode)" \o "乔贝克（Unicode）)从[施乐公司](https://en.wikipedia.org/wiki/Xerox" \o "复印)和[李·科林斯](https://en.wikipedia.org/wiki/Lee_Collins_(software_engineer)" \o "李柯林斯（软件工程师）)和[马克戴维斯](https://en.wikipedia.org/wiki/Mark_Davis_(Unicode)" \o "马克·戴维斯（Unicode）)从[苹果](https://en.wikipedia.org/wiki/Apple_Inc." \o "苹果公司。)开始调查创建一个通用字符集的实用性。[[3]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-3)由Peter Fenwick和Dave Opstad提供的额外投入，[[2]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-unicode-88-2) Joe Becker在1988年8月发表了一项“国际/多语言文本字符编码系统”草案，暂定为Unicode。他解释说，“他的名字”Unicode“旨在表明一种独特的统一的通用编码。[[2]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-unicode-88-2)

在本文中，标题为*Unicode 88*，Becker概述了一个16位字符模型：[[2]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-unicode-88-2)

Unicode旨在解决对可行，可靠的世界文本编码的需求。Unicode可以粗略地描述为“宽身体ASCII”，已经被扩展到16位以包含世界上所有生活语言的字符。在正确设计的设计中，每个字符16位是足够的。

他的原始16位设计是基于这样的假设，即只需要对现代使用中的脚本和字符进行编码：[[2]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-unicode-88-2)

Unicode比保留过去的古物更为优先，以确保未来的效用。Unicode首先在现代文字中发表的角色（例如1988年在世界上所有报纸和杂志的联合体中），其数量无疑远低于2 14 = 16,384。除了这些现代用途的角色，所有其他人都可能被定义为过时或罕见; 这些是比私人使用注册更好的候选人，而不是拥挤一般有用的Unicodes的公开列表。

1989年初，Unicode工作组扩大到包括康培的Ken Whistler和Mike Kernaghan，[RLG的](https://en.wikipedia.org/wiki/Research_Libraries_Group" \o "研究图书馆小组) Karen Smith-Yoshimura和Joan Aliprand ，[Sun Microsystems的](https://en.wikipedia.org/wiki/Sun_Microsystems" \o "太阳微系统公司) Glenn Wright ，以及1990年，来自[微软的](https://en.wikipedia.org/wiki/Microsoft" \o "微软) Michel Suignard和Asmus Freytag 以及Rick McGowan的[NeXT](https://en.wikipedia.org/wiki/NeXT" \o "下一个)加入了集团。到1990年底，大多数绘制现有字符编码标准的工作已经完成，Unicode的最终审查草案已经准备好了。

在[Unicode协会](https://en.wikipedia.org/wiki/Unicode_Consortium" \o "Unicode联盟)在加利福尼亚州注册成立于1991年1月3日，[[4]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-4)和1991年10月，Unicode标准的第一册出版。第二卷涵盖汉字表意，1992年6月出版。

在1996年，Unicode 2.0中实现了代理角色机制，因此Unicode不再局限于16位。这将Unicode代码空间增加到超过一百万个代码点，这允许编码许多历史性脚本（例如，[埃及象形文字](https://en.wikipedia.org/wiki/Egyptian_hieroglyph" \o "埃及象形文字)）以及数千个未被使用或过时的字符，这些字符未被预期为需要编码。不是原来用于Unicode的字符很少使用汉字或汉字，其中许多是个人和地名的一部分，使得它们很少被使用，但比原始的Unicode体系结构更加重要。[[5]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-unicoderevisited-5)

来自1992年的Microsoft TrueType规范版本1.0 在命名表中使用了*Apple Unicode*而不是*Unicode*作为平台ID。

2产生原因

Unicode已经成为文本内部处理和存储的主流方案。虽然大量文本仍然存储在传统编码中，但Unicode几乎专门用于构建新的信息处理系统。早期采用者倾向于使用[UCS-2](https://en.wikipedia.org/wiki/UCS-2" \o "UCS-2)（UTF-16的固定宽度两字节前缀），之后转移到[UTF-16](https://en.wikipedia.org/wiki/UTF-16" \o "UTF-16)（可变宽度当前标准），因为这是添加非支持性的最小破坏性方式-BMP字符 最知名的系统是[Windows NT](https://en.wikipedia.org/wiki/Windows_NT" \o "Windows NT)（及其后代，[Windows 2000](https://en.wikipedia.org/wiki/Windows_2000" \o "Windows 2000)，[Windows XP](https://en.wikipedia.org/wiki/Windows_XP" \o "Windows XP)，[Windows Vista](https://en.wikipedia.org/wiki/Windows_Vista" \o "Windows Vista)和[Windows 7](https://en.wikipedia.org/wiki/Windows_7" \o "Windows 7的)），它使用UTF-16作为唯一的内部字符编码。在[Java的](https://en.wikipedia.org/wiki/Java_virtual_machine" \o "Java虚拟机)和[.NET](https://en.wikipedia.org/wiki/.NET_Framework" \o ".NET Framework)字节码环境，[Mac OS X](https://en.wikipedia.org/wiki/Mac_OS_X" \o "Mac OS X)和[KDE](https://en.wikipedia.org/wiki/KDE" \o "KDE)也用于内部表示。Unicode在[Windows 95上](https://en.wikipedia.org/wiki/Windows_95" \o "Windows 95)通过[Microsoft Layer for Unicode](https://en.wikipedia.org/wiki/Microsoft_Layer_for_Unicode" \o "Microsoft Layer for Unicode)以及其后代[Windows 98](https://en.wikipedia.org/wiki/Windows_98" \o "Windows 98)和[Windows ME提供](https://en.wikipedia.org/wiki/Windows_ME" \o "Windows ME)。

[UTF-8](https://en.wikipedia.org/wiki/UTF-8" \o "UTF-8)（最初为[Plan 9](https://en.wikipedia.org/wiki/Plan_9_from_Bell_Labs" \o "计划9从贝尔实验室)开发）[[51]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-56)已经成为大多数[类Unix](https://en.wikipedia.org/wiki/Unix-like" \o "类Unix)操作系统的主要存储编码（虽然其他库也被使用），因为它是传统[扩展ASCII](https://en.wikipedia.org/wiki/Extended_ASCII" \o "扩展ASCII)字符集的替代品。UTF-8也是[万维网](https://en.wikipedia.org/wiki/World_Wide_Web" \o "万维网)上[HTML](https://en.wikipedia.org/wiki/HTML" \o "HTML)文档中最常用的Unicode编码。

使用Unicode的多语言文字呈现引擎包括用于Microsoft Windows的[Uniscribe](https://en.wikipedia.org/wiki/Uniscribe" \o "的Uniscribe)和[DirectWrite](https://en.wikipedia.org/wiki/DirectWrite" \o "DirectWrite的)，适用于Mac OS X的[ATSUI](https://en.wikipedia.org/wiki/ATSUI" \o "ATSUI)和[Core Text](https://en.wikipedia.org/wiki/Core_Text" \o "核心文本)，以及用于[GTK +](https://en.wikipedia.org/wiki/GTK+" \o "GTK +)和[GNOME](https://en.wikipedia.org/wiki/GNOME" \o "GNOME)桌面的[Pango](https://en.wikipedia.org/wiki/Pango" \o "攀高)

3、解决问题

由于键盘布局对于所有字符都不能具有简单的组合键，因此几个操作系统提供了允许访问整个曲目的替代输入法。

[ISO 14755](https://en.wikipedia.org/wiki/ISO_14755" \o "ISO 14755)，[[52]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-57)，其标准化了用于从他们的代码分进入Unicode字符的方法，指定了几个方法。有*Basic方法*，其中*开始序列*后面是代码点和*结束序列*的十六进制表示。还有一个指定的*屏幕选择条目方法*，其中字符列在屏幕中的表格中，例如使用字符映射程序。

### **电子邮件**

*主要文章：[Unicode和电子邮件](https://en.wikipedia.org/wiki/Unicode_and_email" \o "Unicode和电子邮件)*

[MIME](https://en.wikipedia.org/wiki/MIME" \o "哑剧)定义了两种不同的机制，用于在[电子邮件中](https://en.wikipedia.org/wiki/Email" \o "电子邮件)编码非ASCII字符，具体取决于字符是否在电子邮件头中（例如“主题：”）或消息的文本正文中; 在这两种情况下，识别原始字符集以及传输编码。对于Unicode的电子邮件传输，建议使用[UTF-8](https://en.wikipedia.org/wiki/UTF-8" \o "UTF-8)字符集和[Base64](https://en.wikipedia.org/wiki/Base64" \o "Base64编码)或[Quoted-printable](https://en.wikipedia.org/wiki/Quoted-printable" \o "引号的可打印)传输编码，具体取决于大部分消息是否包含[ASCII](https://en.wikipedia.org/wiki/ASCII" \o "ASCII)字符。两种不同机制的细节在MIME标准中有所规定，一般都是电子邮件软件的用户隐藏的。

在电子邮件中采用Unicode非常慢。一些东亚文本仍然以编码如[ISO-2022](https://en.wikipedia.org/wiki/ISO-2022" \o "ISO-2022)编码，而某些设备（如手机）仍然无法正确处理Unicode数据。不过，支持一直在改善。许多主要的免费邮件提供商，如[雅虎](https://en.wikipedia.org/wiki/Yahoo" \o "雅虎)，[谷歌](https://en.wikipedia.org/wiki/Google" \o "谷歌)（[Gmail](https://en.wikipedia.org/wiki/Gmail" \o "Gmail的)）和[微软](https://en.wikipedia.org/wiki/Microsoft" \o "微软)（[Outlook.com](https://en.wikipedia.org/wiki/Outlook.com" \o "Outlook.com)）都支持它。

### **网**

*主要文章：[Unicode和HTML](https://en.wikipedia.org/wiki/Unicode_and_HTML" \o "Unicode和HTML)*

所有的[W3C](https://en.wikipedia.org/wiki/W3C" \o "W3C)建议都使用Unicode作为自HTML 4.0以来的*文档字符集*。[Web浏览器](https://en.wikipedia.org/wiki/Web_browser" \o "网页浏览器)多年来一直支持Unicode，尤其是UTF-8。以前是显示问题，主要是[字体](https://en.wikipedia.org/wiki/Typeface" \o "字体)相关的问题; 例如v6及更早版本的Microsoft [Internet Explorer](https://en.wikipedia.org/wiki/Internet_Explorer" \o "IE浏览器)没有呈现许多代码点，除非明确告知要使用包含它们的字体。[[53]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-58)

虽然语法规则可能会影响允许显示字符的顺序，但根据定义，[XML](https://en.wikipedia.org/wiki/XML" \o "XML)（包括[XHTML](https://en.wikipedia.org/wiki/XHTML" \o "XHTML)）文档[[54]](https://en.wikipedia.org/wiki/Unicode" \l "cite_note-59)包含大多数Unicode代码点的字符，但以下情况除外：

* 大多数[C0控制代码](https://en.wikipedia.org/wiki/C0_and_C1_control_codes" \o "C0和C1控制代码)
* 永久未分配的代码点D800-DFFF
* FFFE或FFFF

HTML字符根据文档的编码直接显示为[字节](https://en.wikipedia.org/wiki/Byte" \o "字节)，如果编码支持它们，或者用户可以根据字符的Unicode代码点将它们写入数字字符引用。例如，参考文献&#916;，&#1049;，&#1511;，&#1605;，&#3671;，&#12354;，&#21494;，&#33865;，和&#47568;（或以十六进制表示，以相同的数值&#x作为前缀）应显示在所有浏览器为Δ，Й，ק，م，7，あ，叶，叶，和말。

当指定[的URI](https://en.wikipedia.org/wiki/Uniform_Resource_Identifier" \o "统一资源标识符)，例如，如[网址](https://en.wikipedia.org/wiki/Uniform_Resource_Locator" \o "统一资源定位器)在[HTTP](https://en.wikipedia.org/wiki/HTTP" \o "HTTP)请求，非ASCII字符必须是[百分比编码](https://en.wikipedia.org/wiki/Percent_encoding" \o "百分比编码)。

**字体**

基于Unicode的免费和零售[字体](https://en.wikipedia.org/wiki/Font" \o "字形)广泛可用，因为[TrueType](https://en.wikipedia.org/wiki/TrueType" \o "TrueType字体)和[OpenType](https://en.wikipedia.org/wiki/OpenType" \o "OpenType字体)支持Unicode。这些字体格式将Unicode代码点映射到字形。

市场上存在[数千种字体](https://en.wikipedia.org/wiki/List_of_typefaces" \o "字体列表)，但少于十几种字体（有时称为“泛Unicode”字体），试图支持大多数Unicode的字符库。相反，基于Unicode的[字体](https://en.wikipedia.org/wiki/List_of_Unicode_fonts" \o "Unicode字体列表)通常侧重于仅支持基本的ASCII和特定的脚本或一组字符或符号。几种理由证明了这种方法：应用程序和文档很少需要从多于一个或两个写入系统中呈现字符; 字体倾向于在计算环境中需要资源; 并且操作系统和应用程序在根据需要从单独的字体文件获取字形信息（即，[字体替换）](https://en.wikipedia.org/wiki/Font_substitution" \o "字体替换)方面显示增加的智能。此外，为数万个字形设计一套一致的渲染指令构成了巨大的任务; 这样一个冒险就是通过大多数字体的[收益递减](https://en.wikipedia.org/wiki/Diminishing_returns" \o "收益递减)。

### **新线**

Unicode部分地解决了尝试在不同平台上读取文本文件时出现的[换行](https://en.wikipedia.org/wiki/Newline" \o "新队)问题。Unicode定义了符合应用程序应识别为行终止符的大量[字符](https://en.wikipedia.org/wiki/Newline" \l "Unicode" \o "新队)。

在换行方面，Unicode引入了U + 2028 LINE SEPARATOR和U + 2029 PARAGRAPH SEPARATOR。这是一种尝试提供Unicode解决方案来语义地编码段落和行，可能替代所有各种平台解决方案。在这样做的过程中，Unicode确实提供了依赖历史平台的解决方案。尽管如此，Unicode解决方案几乎没有采用这些Unicode行和段落分隔符作为唯一的规范行结束字符。然而，解决这个问题的一个常见方法是通过换行规范化。这是通过Mac OS X中的Cocoa文本系统以及W3C XML和HTML建议实现的。在这种方法中，每个可能的换行符都被内部转换为一个通用的换行符（由于它是一个用于渲染的内部操作，所以它不是很重要）。换句话说，文本系统可以正确地将字符视为换行符，

## 参考文献

1. [“Unicode标准：技术介绍”](http://www.unicode.org/standard/principles.html)。检索2010-03-16。
2. [Becker，Joseph D.](https://en.wikipedia.org/wiki/Joseph_D._Becker" \o "约瑟夫·贝克尔)（1998-09-10）[1988-08-29]。[“Unicode 88”](http://unicode.org/history/unicode88.pdf)（PDF）。*unicode.org*（10周年转载）。[Unicode联盟](https://en.wikipedia.org/wiki/Unicode_Consortium" \o "Unicode联盟)。[档案](https://web.archive.org/web/20161125224409/http://unicode.org/history/unicode88.pdf) （PDF）从原来的2016-11-25 。检索2016-10-25。1978年，一组“通用标志”的初步建议是由制作[鲍勃·贝尔维尔](https://en.wikipedia.org/wiki/Bob_Belleville" \o "鲍勃·贝勒维尔)在[施乐PARC](https://en.wikipedia.org/wiki/Xerox_PARC" \o "施乐PARC)。许多人为开发新的编码设计提出了想法。从1980年开始，这些努力发展成为[施乐字符代码标准](https://en.wikipedia.org/wiki/Xerox_Character_Code_Standard" \o "施乐字符代码标准)（XCCS）作为一种多语言编码，由施乐公司自1982年以来一直由埃尔·斯穆拉（Ed Smura），罗恩·佩拉（Ron Pellar）等人的努力保持为内部公司标准。  
   Unicode是由于XCCS八年的工作经验而诞生的。它与XCCS的根本区别是由Peter Fenwick和Dave Opstad（纯16位代码）和[Lee Collins](https://en.wikipedia.org/wiki/Lee_Collins_(Unicode)" \o "李柯林斯（Unicode）)（表意字符统一）提出。Unicode保留了XCCS的许多功能，其实用程序多年来在国际通信多语言系统产品线中得到证明。
3. [“总结叙事”](http://www.unicode.org/history/summary.html)。检索2010-03-15。
4. [统一发行和出版日期的历史](http://unicode.org/history/publicationdates.html)上*unicode.org。*检举于二零一七年二月二十八日。
5. Searle，Stephen J.[“Unicode Revisited”](http://tronweb.super-nova.co.jp/unicoderevisited.html)。检索2013-01-18。
6. [“Unicode术语词汇”](http://unicode.org/glossary/)。检索2010-03-16。
7. [“附录A：符号公约”](http://www.unicode.org/versions/Unicode10.0.0/appA.pdf)（PDF）。*Unicode标准*。Unicode联盟。2017年6月
8. [“字的Unicode编码稳定政策”](http://unicode.org/policies/stability_policy.html)。检索2010-03-16。
9. [“属性”](http://www.unicode.org/versions/Unicode5.0.0/ch03.pdf" \l "G43463)（PDF）。检索2010-03-16。
10. [“Unicode字符编码模型”](http://unicode.org/reports/tr17/)。检索2010-03-16。
11. [“Unicode命名序列”](http://unicode.org/Public/UNIDATA/NamedSequences.txt)。检索2010-03-16。
12. [“Unicode名称别名”](http://unicode.org/Public/UNIDATA/NameAliases.txt)。检索2010-03-16。
13. [“Unicode Consortium Members”](http://www.unicode.org/consortium/memblogo.html)。检索2010-03-16。
14. [“的Unicode 6.1平装本”](http://www.unicode.org/mail-arch/unicode-ml/y2012-m05/0240.html)。*announcements\_at\_unicode.org*。检索2012-05-30。
15. [“枚举版本的Unicode标准”](http://www.unicode.org/versions/enumeratedversions.html)。检索2016年6月21日。
16. [“Unicode数据1.0.0”](http://www.unicode.org/Public/reconstructed/1.0.0/UnicodeData.txt)。检索2010-03-16。
17. [“Unicode数据1.0.1”](http://www.unicode.org/Public/reconstructed/1.0.1/UnicodeData.txt)。检索2010-03-16。
18. [“Unicode数据1995”](http://www.unicode.org/Public/1.1-Update/UnicodeData-1.1.5.txt)。检索2010-03-16。
19. [“Unicode数据2.0.14”](http://www.unicode.org/Public/2.0-Update/UnicodeData-2.0.14.txt)。检索2010-03-16。
20. [“Unicode Data-2.1.2”](http://www.unicode.org/Public/2.1-Update/UnicodeData-2.1.2.txt)。检索2010-03-16。
21. [“Unicode Data-3.0.0”](http://www.unicode.org/Public/3.0-Update/UnicodeData-3.0.0.txt)。检索2010-03-16。
22. [“Unicode数据-3.1.0”](http://www.unicode.org/Public/3.1-Update/UnicodeData-3.1.0.txt)。检索2010-03-16。
23. [“Unicode Data-3.2.0”](http://www.unicode.org/Public/3.2-Update/UnicodeData-3.2.0.txt)。检索2010-03-16。
24. [“Unicode Data-4.0.0”](http://www.unicode.org/Public/4.0-Update/UnicodeData-4.0.0.txt)。检索2010-03-16。
25. [“Unicode数据”](http://www.unicode.org/Public/4.1.0/ucd/UnicodeData.txt)。检索2010-03-16。
26. [“Unicode数据5.0.0”](http://www.unicode.org/Public/5.0.0/ucd/UnicodeData.txt)。检索2010-03-17。
27. [“Unicode数据5.1.0”](http://www.unicode.org/Public/5.1.0/ucd/UnicodeData.txt)。检索2010-03-17。
28. [“Unicode数据5.2.0”](http://www.unicode.org/Public/5.2.0/ucd/UnicodeData.txt)。检索2010-03-17。
29. [“Unicode数据6.0.0”](http://www.unicode.org/Public/6.0.0/ucd/UnicodeData.txt)。检索2010-10-11。
30. [“Unicode数据6.1.0”](http://www.unicode.org/Public/6.1.0/ucd/UnicodeData.txt)。检索2012-01-31。