

**关于GB 2312、Big5、Unicode、UTF-8和UTF-16的分析**

**姓 名：郝玉坤**

**指导教师：向华**

**所在院系：计算机科学学院软件工程系**

目录

[1 GB 2312 1](#_Toc492739704)

[1.1 GB 2312概述 1](#_Toc492739705)

[1.2 GB 2312的分区表示 1](#_Toc492739706)

[1.3 GB 2312的字节结构 2](#_Toc492739707)

[2 Big5 3](#_Toc492739708)

[2.1 Big5的起源与发展 3](#_Toc492739709)

[2.2 Big5的字节结构 4](#_Toc492739710)

[3 Unicode 5](#_Toc492739711)

[3.1 Unicode的起源与发展 5](#_Toc492739712)

[3.2 Unicode的编码和实现 5](#_Toc492739713)

[4 UTF-8 8](#_Toc492739714)

[4.1 UTF-8的起源与发展 8](#_Toc492739715)

[4.2 UTF-8编码字节含义 9](#_Toc492739716)

[4.3 UTF-8的字节结构 9](#_Toc492739717)

[5 UTF-16 10](#_Toc492739718)

[5.1 UTF-16的描述 10](#_Toc492739719)

[5.2 UTF-16的编码模式 12](#_Toc492739720)

[参考文献 12](#_Toc492739721)

## 1 GB 2312

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。

## 1.1 GB 2312概述

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。

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" \o "人名)、[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。

## 1.2 GB 2312的分区表示

GB 2312中对所收汉字进行了“分区”处理，每区含有94个汉字／符号。这种表示方式也称为[区位码](https://zh.wikipedia.org/wiki/ISO/IEC_2022" \o "ISO/IEC 2022)。

01–09区为特殊符号。

16–55区为一级汉字，按[拼音](https://zh.wikipedia.org/wiki/%E6%8B%BC%E9%9F%B3)排序。

56–87区为二级汉字，按[部首](https://zh.wikipedia.org/wiki/%E9%83%A8%E9%A6%96" \o "部首)／[笔画](https://zh.wikipedia.org/wiki/%E7%AC%94%E7%94%BB)排序。

举例来说，“啊”字是GB 2312之中的第一个汉字，它的区位码就是1601。10–15区及88–94区则未有编码。但在附录3，则在第10区推荐作为 [GB 1988–80](https://zh.wikipedia.org/w/index.php?title=GB_1988&action=edit&redlink=1) 中的94个图形字符区域（即第3区字符之[半形](https://zh.wikipedia.org/wiki/%E5%8D%8A%E5%BD%A2" \o "半形)版本）。

## 1.3 GB 2312的字节结构

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" \o "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。

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

## 2 Big5

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[[1]](https://zh.wikipedia.org/wiki/%E5%A4%A7%E4%BA%94%E7%A2%BC" \l "cite_note-1)。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)。

## 2.1 Big5的起源与发展

“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" \o "信息产业策进会)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[[2]](https://zh.wikipedia.org/wiki/%E5%A4%A7%E4%BA%94%E7%A2%BC" \l "cite_note-2)[[3]](https://zh.wikipedia.org/wiki/%E5%A4%A7%E4%BA%94%E7%A2%BC#cite_note-3)，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" \o "台湾)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" \o "五大中文套装软件)）”[[4]](https://zh.wikipedia.org/wiki/%E5%A4%A7%E4%BA%94%E7%A2%BC#cite_note-4)，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[[5]](https://zh.wikipedia.org/wiki/%E5%A4%A7%E4%BA%94%E7%A2%BC#cite_note-5)[[6]](https://zh.wikipedia.org/wiki/%E5%A4%A7%E4%BA%94%E7%A2%BC#cite_note-6)[[7]](https://zh.wikipedia.org/wiki/%E5%A4%A7%E4%BA%94%E7%A2%BC#cite_note-7)。五大中文套装软件虽然并没有如预期的取代国外的套装软件，但随着采用Big5码的[国乔中文系统](https://zh.wikipedia.org/w/index.php?title=%E5%9C%8B%E5%96%AC%E4%B8%AD%E6%96%87%E7%B3%BB%E7%B5%B1&action=edit&redlink=1" \o "国乔中文系统（页面不存在）)及[倚天中文系统](https://zh.wikipedia.org/wiki/%E5%80%9A%E5%A4%A9%E4%B8%AD%E6%96%87%E7%B3%BB%E7%B5%B1)先后在台湾市场获得成功，使得Big5码深远地影响繁体中文电脑[内码](https://zh.wikipedia.org/wiki/%E5%85%A7%E7%A2%BC" \o "内码)，直至今日。“五大码”的英文名称“Big5”后来被人按英文字序译回中文，以致现在有“五大码”和“大五码”两个中文名称。

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虽普及于[台湾](https://zh.wikipedia.org/wiki/%E5%8F%B0%E7%81%A3" \o "台湾)、[香港](https://zh.wikipedia.org/wiki/%E9%A6%99%E6%B8%AF)与[澳门](https://zh.wikipedia.org/wiki/%E6%BE%B3%E9%96%80)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](https://zh.wikipedia.org/wiki/De_facto" \o "De facto)。[倚天中文系统](https://zh.wikipedia.org/wiki/%E5%80%9A%E5%A4%A9%E4%B8%AD%E6%96%87%E7%B3%BB%E7%B5%B1)、[Windows](https://zh.wikipedia.org/wiki/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](https://zh.wikipedia.org/w/index.php?title=Big5-2003&action=edit&redlink=1" \o "Big5-2003（页面不存在）)。

## 2.2 Big5的字节结构

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。在Big5的分区中：

|  |  |
| --- | --- |
| 0x8140-0xA0FE | 保留给用户自定义字符（[造字](https://zh.wikipedia.org/wiki/%E9%80%A0%E5%AD%97" \o "造字)区） |
| 0xA140-0xA3BF | [标点符号](https://zh.wikipedia.org/wiki/%E6%A8%99%E9%BB%9E%E7%AC%A6%E8%99%9F)、[希腊字母](https://zh.wikipedia.org/wiki/%E5%B8%8C%E8%85%8A%E5%AD%97%E6%AF%8D)及特殊符号，包括在0xA259-0xA261，安放了九个[计量用汉字](https://zh.wikipedia.org/wiki/%E8%A8%88%E9%87%8F%E7%94%A8%E6%BC%A2%E5%AD%97)：兙兛兞兝兡兣嗧瓩糎。 |
| 0xA3C0-0xA3FE | 保留。此区没有开放作造字区用。 |
| 0xA440-0xC67E | [常用汉字](https://zh.wikipedia.org/wiki/%E5%B8%B8%E7%94%A8%E6%BC%A2%E5%AD%97)，先按[笔划](https://zh.wikipedia.org/wiki/%E7%AD%86%E5%8A%83)再按[部首](https://zh.wikipedia.org/wiki/%E9%83%A8%E9%A6%96)排序。 |
| 0xC6A1-0xC8FE | 保留给用户自定义字符（造字区） |
| 0xC940-0xF9D5 | [次常用汉字](https://zh.wikipedia.org/wiki/%E6%AC%A1%E5%B8%B8%E7%94%A8%E5%AD%97)，亦是先按笔划再按部首排序。 |
| 0xF9D6-0xFEFE | 保留给用户自定义字符（造字区） |

## 3 Unicode

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。

## 3.1 Unicode的起源与发展

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)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" \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）。

Unicode伴随着[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)的标准而发展，同时也以书本的形式[[8]](https://zh.wikipedia.org/wiki/Unicode#cite_note-1)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[[9]](https://zh.wikipedia.org/wiki/Unicode#cite_note-Unicode10.0-2)，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。

## 3.2 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。

### 3.2.1 编码方式

统一码的编码方式与[ISO 10646](https://zh.wikipedia.org/wiki/ISO_10646" \o "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" \o "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。

**3.2.2 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](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”。

## 4 UTF-8

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。

## 4.1 UTF-8的起源与发展

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" \o "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" \o "贝尔实验室)[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" \o "肯·汤普逊)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" \o "贝尔实验室九号项目)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" \o "圣地牙哥)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。

## UTF-8编码字节含义

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。

## 4.3 UTF-8的字节结构

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）：128个US-ASCII字符只需一个字节编码（Unicode范围由U+0000至U+007F）；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）；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)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）。

## 5 UTF-16

UTF-16是[Unicode](https://zh.wikipedia.org/wiki/Unicode" \o "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" \o "码位)映射为16位长的整数（即[码元](https://zh.wikipedia.org/wiki/%E7%A0%81%E5%85%83" \o "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。

## 5.1 UTF-16的描述

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。

### 5.1.1 从U+0000至U+D7FF以及从U+E000至U+FFFF的码位

第一个Unicode平面（码位从U+0000至U+FFFF）包含了最常用的字符。该平面被称为基本多语言平面，缩写为*BMP*（Basic Multilingual Plane, BMP）。UTF-16与[UCS-2](https://zh.wikipedia.org/wiki/UCS-2)编码这个范围内的码位为16比特长的单个码元，数值等价于对应的码位. BMP中的这些码位是仅有的可以在UCS-2中表示的码位。

### 5.1.2 从U+10000到U+10FFFF的码位

辅助平面（Supplementary Planes）中的码位，在UTF-16中被编码为一对16比特长的码元（即32bit,4Bytes），称作*代理对*（surrogate pair），具体方法是：

码位减去0x10000,得到的值的范围为20比特长的0..0xFFFFF.

高位的10比特的值（值的范围为0..0x3FF）被加上0xD800得到第一个码元或称作高位代理（high surrogate），值的范围是0xD800..0xDBFF.由于高位代理比低位代理的值要小，所以为了避免混淆使用，Unicode标准现在称高位代理为前导代理（lead surrogates）。

低位的10比特的值（值的范围也是0..0x3FF）被加上0xDC00得到第二个码元或称作低位代理（low surrogate），现在值的范围是0xDC00..0xDFFF.由于低位代理比高位代理的值要大，所以为了避免混淆使用，Unicode标准现在称低位代理为后尾代理（trail surrogates）。

上述算法可理解为：辅助平面中的码位从U+10000到U+10FFFF，共计FFFFF个，即220=1,048,576个，需要20位来表示。如果用两个16位长的整数组成的序列来表示，第一个整数（称为前导代理）要容纳上述20位的前10位，第二个整数（称为后尾代理）容纳上述20位的后10位。还要能根据16位整数的值直接判明属于前导整数代理的值的范围（210=1024)，还是后尾整数代理的值的范围（也是210=1024）。因此，需要在基本多语言平面中保留不对应于Unicode字符的2048个码位，就足以容纳前导代理与后尾代理所需要的编码空间。这对于基本多语言平面总计65536个码位来说，仅占3.125%.

由于前导代理、后尾代理、BMP中的有效字符的码位，三者互不重叠，搜索是简单的：一个字符编码的一部分不可能与另一个字符编码的不同部分相重叠。这意味着UTF-16是自同步（self-synchronizing）:可以通过仅检查一个码元就可以判定给定字符的下一个字符的起始码元. [UTF-8](https://zh.wikipedia.org/wiki/UTF-8)也有类似优点，但许多早期的编码模式就不是这样，必须从头开始分析文本才能确定不同字符的码元的边界。

由于最常有的字符都在基本多文种平面中，许多软件的处理代理对的部分往往得不到充分的测试。这导致了一些长期的bug与潜在安全漏洞，甚至在广为流行得到良好评价的应用软件[[10]](https://zh.wikipedia.org/wiki/UTF-16#cite_note-1).

### 5.1.3 从U+D800到U+DFFF的码位

Unicode标准规定U+D800..U+DFFF的值不对应于任何字符。但是在使用UCS-2的时代，U+D800..U+DFFF内的值被占用，用于某些字符的映射。但只要不构成代理对，许多UTF-16编码解码还是能把这些不匹配Unicode标准的字符映射正确的辨识、转换成合规的码元[[11]](https://zh.wikipedia.org/wiki/UTF-16#cite_note-2).按照Unicode标准，这种码元序列本来应算作编码错误。

## 5.2 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](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。

## 参考文献

1. 普遍认为大五码包含13,053字，但在计算0xA259-0xA261的九个[度量衡单位用字](https://zh.wikipedia.org/wiki/%E8%A8%88%E9%87%8F%E7%94%A8%E6%BC%A2%E5%AD%97)（兙兛兞兝兡兣嗧瓩糎），再减去重收了两次的“兀”（0xC94A）和“嗀”（0xDDFC）后，应为13,060字。
2. 来源参考：（中文）[资策会大事纪要，1983年](http://www.iii.org.tw/about/history_1983.html) [互联网档案馆](https://zh.wikipedia.org/wiki/Wayback_Machine)的[存档](https://web.archive.org/web/20101104215940/http:/www.iii.org.tw/about/history_1983.html)，存档日期2010-11-0
3. 来源参考：（中文）[资策会大事纪要](http://www.iii.org.tw/about/1_6.asp) [互联网档案馆](https://zh.wikipedia.org/wiki/Wayback_Machine)的[存档](https://web.archive.org/web/20101104200611/http:/www.iii.org.tw/about/1_6.asp)，存档日期2010-11-04.
4. 来源参考：（中文）[资策会大事纪要，1984年](http://www.iii.org.tw/about/history_1984.html) [互联网档案馆](https://zh.wikipedia.org/wiki/Wayback_Machine)的[存档](https://web.archive.org/web/20101104215950/http:/www.iii.org.tw/about/history_1984.html)，存档日期2010-11-04.
5. 来源参考：（中文）[行政院主计处电子处理数据中心中文全字库——中文码介绍-BIG-5码介绍](http://www.cns11643.gov.tw/AIDB/encodings.do#encode4)
6. 来源参考：（中文）[数字杂谈 - 中文的电脑或电脑的中文？，谌家雄](http://www.libertytimes.com.tw/2001/new/mar/11/today-i1.htm) [互联网档案馆](https://zh.wikipedia.org/wiki/Wayback_Machine)的[存档](https://web.archive.org/web/20070222072342/http:/www.libertytimes.com.tw/2001/new/mar/11/today-i1.htm)，存档日期2007-02-22.
7. 来源参考：（中文）[CMEX财团法人中文数字化技术推广基金会 - 认识中文码：九、Big5和Big5E](http://www.cmex.org.tw/page.do?path=service_1-2.html)
8. 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).
9. [Unicode 10.0.0](http://www.unicode.org/versions/Unicode10.0.0/). Unicode Consortium. [2017-06-20].
10. [Code in Apache Xalan 2.7.0 which can fail on surrogate pairs](http://www.google.com/codesearch/p?hl=en&sa=N&cd=3&ct=rc#WiHOcNHN_BU/xalan-j_2_7_0/src/org/apache/xalan/lib/ExsltStrings.java&q=file:ExsltStrings%5C.java%20align). Apache Foundation. The code wrongly assumes it is safe to use substring on the input
11. Python 2.6 decode of UTF16 does this on Linux, and it correctly handles surrogate pairs. All "CESU" decoders do it too, though they also mistranslate correct surrogate pairs into 2 characters