**XML作业**

**GB2312**

**来历：**GB2312又称为GB2312-80字符集，全称为《信息交换用汉字编码字符集·基本集》，由原中国国家标准总局发布，1981年5月1日实施。

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。

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" \o "拼音)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。

**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。

**两种不同的GB2312实现**

有两种不同的GB 2312实现，在它们之间存在少量的差别，其中至少有一个是错误的。

|  |  |  |  |
| --- | --- | --- | --- |
| **字节序** | **GBK子集** | **GB2312.TXT** | **字符名称** |
| A1A4 | U+00B7 “·” MIDDLE DOT | U+30FB “・” KATAKANA MIDDLE DOT | [间隔点](https://zh.wikipedia.org/wiki/%E9%97%B4%E9%9A%94%E7%82%B9) |
| A1AA | U+2014 “—” EM DASH | U+2015 “―” HORIZONTAL BAR | [破折号](https://zh.wikipedia.org/wiki/%E7%A0%B4%E6%8A%98%E5%8F%B7) |

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。

**big5**

**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)。

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)。[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）)。

**来历：**“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，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)）”，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。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。

**产生原因：**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；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。

**解决的问题：**在Big5码诞生后，大部分台湾的电脑软件都使用了Big5码，加上后来[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" \o "倚天中文系统)的高度普及，使后来的[微软](https://zh.wikipedia.org/wiki/%E5%BE%AE%E8%BD%AF)[Windows 3.x](https://zh.wikipedia.org/wiki/Windows_3.x)等亦予以采用。虽然后来台湾还有各种想要取代Big5码，像是倚天中文系统所推行的[倚天码](https://zh.wikipedia.org/wiki/%E5%80%9A%E5%A4%A9%E7%A2%BC" \o "倚天码)、台北市电脑公会所推动的[公会码](https://zh.wikipedia.org/w/index.php?title=%E5%85%AC%E6%9C%83%E7%A2%BC&action=edit&redlink=1)等，但是由于Big5字码已沿用多年，因此在习惯不易改变的情况下，始终无法成为主流字码。而台湾后来发展的[国家标准CNS 11643中文标准交换码](https://zh.wikipedia.org/wiki/CNS11643" \o "CNS11643)由于非一般的内码系统，是以交换使用为目的，受先天所限，必须使用至少三个字节来表示一个汉字，所以普及率远远不及Big5码。

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" \o "中国大陆)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" \o "简体中文)的社区，最常用的是[GB2312](https://zh.wikipedia.org/wiki/GB_2312)、[GBK](https://zh.wikipedia.org/wiki/GBK" \o "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" \o "国家标准代码)（[GB 18030](https://zh.wikipedia.org/wiki/GB_18030)）。

## 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)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" \o "计量用汉字)：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） |

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。

**兼容问题：**由于各厂商及政府推出的Big5延伸，彼此互不兼容，造成[乱码](https://zh.wikipedia.org/wiki/%E4%BA%82%E7%A2%BC" \o "乱码)问题。鉴于[Unicode](https://zh.wikipedia.org/wiki/Unicode)能正确地处理七万多个汉字，近年的[操作系统](https://zh.wikipedia.org/wiki/%E4%BD%9C%E6%A5%AD%E7%B3%BB%E7%B5%B1" \o "操作系统)和应用程序（如[苹果电脑](https://zh.wikipedia.org/wiki/%E8%98%8B%E6%9E%9C%E9%9B%BB%E8%85%A6" \o "苹果电脑)[Mac OS X](https://zh.wikipedia.org/wiki/Mac_OS_X)和以[Cocoa](https://zh.wikipedia.org/wiki/Cocoa" \o "Cocoa) API撰写之程序、[Microsoft](https://zh.wikipedia.org/wiki/Microsoft" \o "Microsoft) [Windows 2000](https://zh.wikipedia.org/wiki/Windows_2000)及之后版本、[Microsoft Office](https://zh.wikipedia.org/wiki/Microsoft_Office" \o "Microsoft Office) 2000及之后版本、[Mozilla](https://zh.wikipedia.org/wiki/Mozilla" \o "Mozilla)浏览器、[Internet Explorer](https://zh.wikipedia.org/wiki/Internet_Explorer" \o "Internet Explorer)浏览器、[Java](https://zh.wikipedia.org/wiki/Java" \o "Java)语言等等），已改用Unicode编码。可惜现时仍有一些旧的软件（如[Visual Basic](https://zh.wikipedia.org/wiki/Visual_Basic" \o "Visual Basic) 6、部分[Telnet](https://zh.wikipedia.org/wiki/Telnet" \o "Telnet)或[BBS](https://zh.wikipedia.org/wiki/BBS" \o "BBS)软件），未能支持Unicode编码，故相信Big5缺字的问题仍会困扰用户一段时间，直至所有程序都能改用Unicode为止。

**UTF8**

**描述：**目前有好几份关于UTF-8详细规格的文件，但这些文件在定义上有些许的不同：

* [RFC 3629](https://tools.ietf.org/html/rfc3629) / STD 63（2003），这份文件制定了UTF-8是标准的[互联网](https://zh.wikipedia.org/wiki/%E7%B6%B2%E9%9A%9B%E7%B6%B2%E8%B7%AF" \o "互联网)[协议](https://zh.wikipedia.org/wiki/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE)元素
* *第四版，The Unicode Standard*，§3.9－§3.10（2003）
* ISO/IEC 10646-1:2000附加文件D（2000）

它们取代了以下那些被淘汰的定义：

* ISO/IEC 10646-1:1993修正案2／附加文件R（1996）
* *第二版，The Unicode Standard*，附录A（1996）
* [RFC 2044](https://tools.ietf.org/html/rfc2044)（1996）
* [RFC 2279](https://tools.ietf.org/html/rfc2279)（1998）
* *第三版，The Unicode Standard*，§2.3（2000）及勘误表#1：UTF-8 Shortest Form（2000）
* *Unicode Standard附加文件#27: Unicode 3.1*（2001）

事实上，所有定义的基本原理都是相同的，它们之间最主要的不同是支持的字符范围及无效输入的处理方法。

Unicode[字符](https://zh.wikipedia.org/wiki/UCS)的[比特](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83)被分区为数个部分，并分配到UTF-8的[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82" \o "字节)串中较低的[比特](https://zh.wikipedia.org/wiki/%E4%BD%8D%E5%85%83)的位置。在U+0080的以下[字符](https://zh.wikipedia.org/wiki/UCS" \o "UCS)都使用内含其字符的单[字节](https://zh.wikipedia.org/wiki/%E5%AD%97%E8%8A%82)编码。这些编码正好对应7比特的ASCII字符。在其他情况，有可能需要多达4个字符组来表示一个字符。这些多字节的[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" \o "最高有效比特)会设置成1，以防止与7比特的ASCII字符混淆，并保持标准的字节主导字符串运作顺利。

|  |  |  |  |
| --- | --- | --- | --- |
| **代码范围** [**十六进制**](https://zh.wikipedia.org/wiki/%E5%8D%81%E5%85%AD%E9%80%B2%E5%88%B6) | **标量值（scalar value）** [**二进制**](https://zh.wikipedia.org/wiki/%E4%BA%8C%E9%80%B2%E5%88%B6) | **UTF-8** [**二进制**](https://zh.wikipedia.org/wiki/%E4%BA%8C%E9%80%B2%E5%88%B6)**／**[**十六进制**](https://zh.wikipedia.org/wiki/%E5%8D%81%E5%85%AD%E9%80%B2%E5%88%B6) | **注释** |
| 000000 - 00007F 128个代码 | 00000000 00000000 0zzzzzzz | 0zzzzzzz（00-7F） | ASCII字符范围，字节由零开始 |
| 七个z | 七个z |
| 000080 - 0007FF 1920个代码 | 00000000 00000yyy yyzzzzzz | 110yyyyy（C0-DF) 10zzzzzz(80-BF） | 第一个字节由110开始，接着的字节由10开始 |
| 三个y；二个y；六个z | 五个y；六个z |
| 000800 - 00D7FF 00E000 - 00FFFF 61440个代码 [[Note 1]](https://zh.wikipedia.org/wiki/UTF-8#endnote_D800Note_1) | 00000000 xxxxyyyy yyzzzzzz | 1110xxxx(E0-EF) 10yyyyyy 10zzzzzz | 第一个字节由1110开始，接着的字节由10开始 |
| 四个x；四个y；二个y；六个z | 四个x；六个y；六个z |
| 010000 - 10FFFF 1048576个代码 | 000wwwxx xxxxyyyy yyzzzzzz | 11110www(F0-F7) 10xxxxxx 10yyyyyy 10zzzzzz | 将由11110开始，接着的字节由10开始 |
| 三个w；二个x；四个x；四个y；二个y；六个z | 三个w；六个x；六个y；六个z |

[**Note 1**](https://zh.wikipedia.org/wiki/UTF-8#ref_D800Note_1) Unicode在范围D800-DFFF中不存在任何字符，[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)中约定了这个范围用于UTF-16扩展标识[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)（两个UTF-16表示一个[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)字符）。当然，任何编码都是可以被转换到这个范围，但在unicode中他们并不代表任何合法的值。

例如，希伯来语字母aleph（א）的Unicode代码是U+05D0，按照以下方法改成UTF-8：

* 它属于U+0080到U+07FF区域，这个表说明它使用双字节，*110*yyyyy *10*zzzzzz.
* [十六进制](https://zh.wikipedia.org/wiki/%E5%8D%81%E5%85%AD%E8%BF%9B%E5%88%B6)的0x05D0换算成[二进制](https://zh.wikipedia.org/wiki/%E4%BA%8C%E8%BF%9B%E5%88%B6" \o "二进制)就是101-1101-0000.
* 这11位数按顺序放入"y"部分和"z"部分：110**10111** 10**010000**.
* 最后结果就是双字节，用十六进制写起来就是0xD7 0x90，这就是这个字符aleph（א）的UTF-8编码。

所以开始的128个字符（US-ASCII）只需一字节，接下来的1920个字符需要双字节编码，包括带[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[拉丁字母](https://zh.wikipedia.org/wiki/%E6%8B%89%E4%B8%81%E5%AD%97%E6%AF%8D)，[希腊字母](https://zh.wikipedia.org/wiki/%E5%B8%8C%E8%85%8A%E5%AD%97%E6%AF%8D)，[西里尔字母](https://zh.wikipedia.org/wiki/%E8%A5%BF%E9%87%8C%E5%B0%94%E5%AD%97%E6%AF%8D)，[科普特语](https://zh.wikipedia.org/wiki/%E7%A7%91%E6%99%AE%E7%89%B9%E8%AF%AD)字母，[亚美尼亚语](https://zh.wikipedia.org/wiki/%E4%BA%9A%E7%BE%8E%E5%B0%BC%E4%BA%9A%E8%AF%AD)字母，[希伯来文](https://zh.wikipedia.org/wiki/%E5%B8%8C%E4%BC%AF%E6%9D%A5%E6%96%87)字母和[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)的字符。[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)中其余的字符使用三个字节，剩余字符使用四个字节。

根据这种方式可以处理更大数量的字符。原来的规范允许长达6字节的序列，可以覆盖到31位（[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)原来的极限）。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。根据这些规范，以下字节值将无法出现在合法UTF-8序列中：

|  |  |  |
| --- | --- | --- |
| **编码（**[**二进制**](https://zh.wikipedia.org/wiki/%E4%BA%8C%E8%BF%9B%E5%88%B6)**）** | **编码（**[**十六进制**](https://zh.wikipedia.org/wiki/%E5%8D%81%E5%85%AD%E8%BF%9B%E5%88%B6)**）** | **注释** |
| 1100000x | C0, C1 | 过长编码：双字节序列的头字节，但码点 <= 127 |
| 1111111x | FE, FF | 无法达到：7或8字节序列的头字节 |
| 111110xx 1111110x | F8, F9, FA, FB, FC, FD | 被RFC 3629规范：5或6字节序列的头字节 |
| 11110101 1111011x | F5, F6, F7 | 被RFC 3629规范：码点超过10FFFF的头字节 |

**来历：**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)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

**字节：**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。不过，如果使用一些传统的压缩系统，比如[DEFLATE](https://zh.wikipedia.org/wiki/DEFLATE)，则这些不同编码系统间的的差异就变得微不足道了。若顾及传统压缩算法在压缩较短文字上的效果不大，可以考虑使用[Unicode标准压缩格式](https://zh.wikipedia.org/w/index.php?title=Unicode%E6%A8%99%E6%BA%96%E5%A3%93%E7%B8%AE%E6%A0%BC%E5%BC%8F&action=edit&redlink=1)（SCSU）。

**兼容问题：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，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。

**Unicode**

**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)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。

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)的标准而发展，同时也以书本的形式[[1]](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[[2]](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)（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)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)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。

**来历：**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）。

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)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)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。

**兼容问题：**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)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，而不以常见的半角形式显示，这对竖排文字和等宽排列文字有重要作用。

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。

[gbk, gb2312,big5,unicode,utf-8,utf-16的区别](http://blog.csdn.net/a351945755/article/details/37689303)

0、big endian和little endian   
big endian和little endian是CPU处理多字节数的不同方式。例如“汉”字的Unicode编码是6C49。那么写到文件里时，究竟是将6C写在前面，还是将49写在前面？如果将6C写在前面，就是big endian。还是将49写在前面，就是little endian。

“endian”这个词出自《格列佛游记》。小人国的内战就源于吃鸡蛋时是究竟从大头(Big-Endian)敲开还是从小头(Little-Endian)敲开，由此曾发生过六次叛乱，其中一个皇帝送了命，另一个丢了王位。

我们一般将endian翻译成“字节序”，将big endian和little endian称作“大尾”和“小尾”。

1、字符编码、内码，顺带介绍汉字编码   
字符必须编码后才能被计算机处理。计算机使用的缺省编码方式就是计算机的内码。早期的计算机使用7位的ASCII编码，为了处理汉字，程序员设计了用于简体中文的GB2312和用于繁体中文的big5。

GB2312(1980年)一共收录了7445个字符，包括6763个汉字和682个其它符号。汉字区的内码范围高字节从B0-F7，低字节从A1-FE，占用的码位是72\*94=6768。其中有5个空位是D7FA-D7FE。

GB2312支持的汉字太少。1995年的汉字扩展规范GBK1.0收录了21886个符号，它分为汉字区和图形符号区。汉字区包括21003个字符。2000年的GB18030是取代GBK1.0的正式国家标准。该标准收录了27484个汉字，同时还收录了藏文、蒙文、维吾尔文等主要的少数民族文字。现在的PC平台必须支持GB18030，对嵌入式产品暂不作要求。所以手机、MP3一般只支持GB2312。

从ASCII、GB2312、GBK到GB18030，这些编码方法是向下兼容的，即同一个字符在这些方案中总是有相同的编码，后面的标准支持更多的字符。在这些编码中，英文和中文可以统一地处理。区分中文编码的方法是高字节的最高位不为0。按照程序员的称呼，GB2312、GBK到GB18030都属于双字节字符集 (DBCS)。

有的中文Windows的缺省内码还是GBK，可以通过GB18030升级包升级到GB18030。不过GB18030相对GBK增加的字符，普通人是很难用到的，通常我们还是用GBK指代中文Windows内码。

这里还有一些细节：

GB2312的原文还是区位码，从区位码到内码，需要在高字节和低字节上分别加上A0。

在DBCS中，GB内码的存储格式始终是big endian，即高位在前。

GB2312的两个字节的最高位都是1。但符合这个条件的码位只有128\*128=16384个。所以GBK和GB18030的低字节最高位都可能不是1。不过这不影响DBCS字符流的解析：在读取DBCS字符流时，只要遇到高位为1的字节，就可以将下两个字节作为一个双字节编码，而不用管低字节的高位是什么。

2、Unicode、UCS和UTF   
前面提到从ASCII、GB2312、GBK到GB18030的编码方法是向下兼容的。而Unicode只与ASCII兼容（更准确地说，是与ISO-8859-1兼容），与GB码不兼容。例如“汉”字的Unicode编码是6C49，而GB码是BABA。

Unicode也是一种字符编码方法，不过它是由国际组织设计，可以容纳全世界所有语言文字的编码方案。Unicode的学名是"Universal Multiple-Octet Coded Character Set"，简称为UCS。UCS可以看作是"Unicode Character Set"的缩写。

根据维基百科全书(<http://zh.wikipedia.org/wiki/>)的记载：历史上存在两个试图独立设计Unicode的组织，即国际标准化组织（ISO）和一个软件制造商的协会（unicode.org）。ISO开发了ISO 10646项目，Unicode协会开发了Unicode项目。

在1991年前后，双方都认识到世界不需要两个不兼容的字符集。于是它们开始合并双方的工作成果，并为创立一个单一编码表而协同工作。从Unicode2.0开始，Unicode项目采用了与ISO 10646-1相同的字库和字码。

目前两个项目仍都存在，并独立地公布各自的标准。Unicode协会现在的最新版本是2005年的Unicode 4.1.0。ISO的最新标准是10646-3:2003。

UCS规定了怎么用多个字节表示各种文字。怎样传输这些编码，是由UTF(UCS Transformation Format)规范规定的，常见的UTF规范包括UTF-8、UTF-7、UTF-16。

IETF的RFC2781和RFC3629以RFC的一贯风格，清晰、明快又不失严谨地描述了UTF-16和UTF-8的编码方法。我总是记不得IETF是Internet Engineering Task Force的缩写。但IETF负责维护的RFC是Internet上一切规范的基础。

3、UCS-2、UCS-4、BMP

UCS有两种格式：UCS-2和UCS-4。顾名思义，UCS-2就是用两个字节编码，UCS-4就是用4个字节（实际上只用了31位，最高位必须为0）编码。下面让我们做一些简单的数学游戏：

UCS-2有2^16=65536个码位，UCS-4有2^31=2147483648个码位。

UCS-4根据最高位为0的最高字节分成2^7=128个group。每个group再根据次高字节分为256个plane。每个plane根据第3个字节分为256行 (rows)，每行包含256个cells。当然同一行的cells只是最后一个字节不同，其余都相同。

group 0的plane 0被称作Basic Multilingual Plane, 即BMP。或者说UCS-4中，高两个字节为0的码位被称作BMP。

将UCS-4的BMP去掉前面的两个零字节就得到了UCS-2。在UCS-2的两个字节前加上两个零字节，就得到了UCS-4的BMP。而目前的UCS-4规范中还没有任何字符被分配在BMP之外（也就是说现在的字符的unicode编码都是两个字节的，因为UCS-4的前两个字节为0）。

4、UTF编码

UTF-8就是以8位为单元对UCS进行编码。从UCS-2到UTF-8的编码方式如下：

UCS-2编码(16进制) UTF-8 字节流(二进制)   
0000 - 007F 0xxxxxxx   
0080 - 07FF 110xxxxx 10xxxxxx   
0800 - FFFF 1110xxxx 10xxxxxx 10xxxxxx

例如“汉”字的Unicode编码是6C49。6C49在0800-FFFF之间，所以肯定要用3字节模板了：1110xxxx 10xxxxxx 10xxxxxx。将6C49写成二进制是：0110 110001 001001， 用这个比特流依次代替模板中的x，得到：11100110 10110001 10001001，即E6 B1 89。

读者可以用记事本测试一下我们的编码是否正确。

UTF-16以16位为单元对UCS进行编码。对于小于0x10000的UCS码，UTF-16编码就等于UCS码对应的16位无符号整数。对于不小于0x10000的UCS码，定义了一个算法。不过由于实际使用的UCS2，或者UCS4的BMP必然小于0x10000，所以就目前而言，可以认为UTF-16和UCS-2基本相同。但UCS-2只是一个编码方案，UTF-16却要用于实际的传输，所以就不得不考虑字节序的问题。

5、UTF的字节序和BOM   
UTF-8以字节为编码单元，没有字节序的问题。UTF-16以两个字节为编码单元，在解释一个UTF-16文本前，首先要弄清楚每个编码单元的字节序。例如收到一个“奎”的Unicode编码是594E，“乙”的Unicode编码是4E59。如果我们收到UTF-16字节流“594E”，那么这是“奎”还是“乙”？

Unicode规范中推荐的标记字节顺序的方法是BOM。BOM不是“Bill Of Material”的BOM表，而是Byte Order Mark。BOM是一个有点小聪明的想法：

在UCS编码中有一个叫做"ZERO WIDTH NO-BREAK SPACE"的字符，它的编码是FEFF。而FFFE在UCS中是不存在的字符，所以不应该出现在实际传输中。UCS规范建议我们在传输字节流前，先传输字符"ZERO WIDTH NO-BREAK SPACE"。

这样如果接收者收到FEFF，就表明这个字节流是Big-Endian的；如果收到FFFE，就表明这个字节流是Little-Endian的。因此字符"ZERO WIDTH NO-BREAK SPACE"又被称作BOM。

UTF-8不需要BOM来表明字节顺序，但可以用BOM来表明编码方式。字符"ZERO WIDTH NO-BREAK SPACE"的UTF-8编码是EF BB BF（读者可以用我们前面介绍的编码方法验证一下）。所以如果接收者收到以EF BB BF开头的字节流，就知道这是UTF-8编码了。

Windows就是使用BOM来标记文本文件的编码方式的。