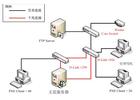
# 字符编码

字符编码（英语：Character encoding）也称字集码，是把[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)中的[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)[编码](https://baike.baidu.com/item/%E7%BC%96%E7%A0%81)为指定[集合](https://baike.baidu.com/item/%E9%9B%86%E5%90%88)中某一[对象](https://baike.baidu.com/item/%E5%AF%B9%E8%B1%A1)（例如：[比特](https://baike.baidu.com/item/%E6%AF%94%E7%89%B9)模式、[自然数](https://baike.baidu.com/item/%E8%87%AA%E7%84%B6%E6%95%B0)[序列](https://baike.baidu.com/item/%E5%BA%8F%E5%88%97)、8位组或者[电脉冲](https://baike.baidu.com/item/%E7%94%B5%E8%84%89%E5%86%B2)），以便[文本](https://baike.baidu.com/item/%E6%96%87%E6%9C%AC)在[计算机](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA)中存储和通过[通信](https://baike.baidu.com/item/%E9%80%9A%E4%BF%A1)[网络](https://baike.baidu.com/item/%E7%BD%91%E7%BB%9C)的传递。常见的例子包括将[拉丁字母表](https://baike.baidu.com/item/%E6%8B%89%E4%B8%81%E5%AD%97%E6%AF%8D%E8%A1%A8)编码成[摩斯电码](https://baike.baidu.com/item/%E6%91%A9%E6%96%AF%E7%94%B5%E7%A0%81)和[ASCII](https://baike.baidu.com/item/ASCII)。其中，[ASCII](https://baike.baidu.com/item/ASCII)将字母、数字和其它符号[编号](https://baike.baidu.com/item/%E7%BC%96%E5%8F%B7)，并用7[比特](https://baike.baidu.com/item/%E6%AF%94%E7%89%B9)的[二进制](https://baike.baidu.com/item/%E4%BA%8C%E8%BF%9B%E5%88%B6)来表示这个整数。通常会额外使用一个扩充的比特，以便于以1个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)的方式存储。

在计算机技术发展的早期，如[ASCII](https://baike.baidu.com/item/ASCII)（1963年）和[EBCDIC](https://baike.baidu.com/item/EBCDIC)（1964年）这样的字符集逐渐成为标准。但这些字符集的局限很快就变得明显，于是人们开发了许多方法来扩展它们。对于支持包括东亚[CJK](https://baike.baidu.com/item/CJK)字符家族在内的[写作系统](https://baike.baidu.com/item/%E5%86%99%E4%BD%9C%E7%B3%BB%E7%BB%9F)的要求能支持更大量的字符，并且需要一种系统而不是临时的方法实现这些字符的编码。

编码

在显示器上看见的文字、图片等信息在电脑里面

[](https://baike.baidu.com/pic/%E5%AD%97%E7%AC%A6%E7%BC%96%E7%A0%81/8446880/0/faacb56410d9b5c8f636546b?fr=lemma&ct=single)EBCDIC

其实并不是我们看见的样子，即使你知道所有信息都存储在硬盘里，把它拆开也看不见里面有任何东西，只有些盘片。假设，你用显微镜把盘片放大，会看见盘片表面凹凸不平，凸起的地方被磁化，凹的地方是没有被磁化；凸起的地方代表数字1，凹的地方代表数字0。硬盘只能用0和1来表示所有文字、图片等信息。那么字母”A”在硬盘上是如何存储的呢？可能小张计算机存储字母”A”是1100001，而小王存储字母”A”是11000010，这样双方交换信息时就会误解。比如小张把1100001发送给小王，小王并不认为1100001是字母”A”，可能认为这是字母”X”，于是小王在用[记事本](https://baike.baidu.com/item/%E8%AE%B0%E4%BA%8B%E6%9C%AC)访问存储在硬盘上的1100001时，在[屏幕](https://baike.baidu.com/item/%E5%B1%8F%E5%B9%95)上显示的就是字母”X”。也就是说，小张和小王使用了不同的编码表。小张用的编码表是ASCII，ASCII编码表把26个字母都一一的对应到2进制1和0上；小王用的编码表可能是EBCDIC,只不过EBCDIC编码与ASCII编码中的字母和01的对应关系不同。一般地说，开放的[操作系统](https://baike.baidu.com/item/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F)（LINUX 、WINDOWS等）采用ASCII 编码，而大型[主机系统](https://baike.baidu.com/item/%E4%B8%BB%E6%9C%BA%E7%B3%BB%E7%BB%9F)（MVS 、OS/390等）采用EBCDIC 编码。在发送数据给对方前，需要事先告知对方自己所使用的编码，或者通过[转码](https://baike.baidu.com/item/%E8%BD%AC%E7%A0%81)，使不同编码方案的两个系统可沟通自如。

ASCII码使用7位2进制数表示一个字符，7位2进制数可以表示出2的7次方个字符，共128个字符。EBCDIC码使用8位，可以表示出2的8次方个字符，256个字符。

无论是ASCII码还是EBCDIC码，都无法对拥有几万个的汉字进行编码。因为上面已经提过，7位2进制数最多对应上128个字符，8位最多对应上256个字符。

0~31及127（共33个）是控制字符或通信专用字符（其余为可显示字符），如控制符：LF（换行）、CR（回车）、FF（换页）、DEL（删除）、BS（退格）、BEL（振铃）等；通信专用字符：SOH（文头）、EOT（文尾）、ACK（确认）等；ASCII值为8、9、10和13分别转换为退格、制表、换行和回车字符。它们并没有特定的图形显示，但会依不同的应用程序而对文本显示有不同的影响。

32~126（共95个）是字符（32sp是空格），其中48~57为0到9十个阿拉伯数字，65~90为26个大写英文字母，97~122为26个小写字母，其余为一些标点符号、运算符号等。

## ASCII

美国(国家)信息交换标准(代)码，一种使用7个或8个二进制位进行编码的方案，最多可以给256个字符

(包括字母、数字、标点符号、[控制字符](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E5%AD%97%E7%AC%A6)及其他符号)分配(或指定)数值。

ASCII码于1961年提出，用于在不同[计算机硬件](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA%E7%A1%AC%E4%BB%B6)和[软件系统](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E7%B3%BB%E7%BB%9F)中实现数据传输标准化，在大多数的[小型机](https://baike.baidu.com/item/%E5%B0%8F%E5%9E%8B%E6%9C%BA)和全部的[个人计算机](https://baike.baidu.com/item/%E4%B8%AA%E4%BA%BA%E8%AE%A1%E7%AE%97%E6%9C%BA)都使用此码。ASCII码划分为两个集合：128个字符的标准ASCII码和附加的128个字符的扩充和ASCII码。比较EBCDIC。其中95个字符可以显示。另外33个不可以显示。 标准ASCII码为7位，扩充为8位。

目前使用最广泛的西文[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)及其编码是 ASCII 字符集和 ASCII 码（ ASCII 是 American Standard Code for Information Interchange 的缩写），它同时也被[国际标准化组织](https://baike.baidu.com/item/%E5%9B%BD%E9%99%85%E6%A0%87%E5%87%86%E5%8C%96%E7%BB%84%E7%BB%87)（ International Organization for Standardization, ISO ）批准为国际标准。

基本的 ASCII [字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)共有 128 个字符，其中有 96 个可打印字符，包括常用的字母、数字、标点符号等，另外还有 32 个[控制字符](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E5%AD%97%E7%AC%A6)。标准 ASCII 码使用 7 个二进位对字符进行编码，对应的 ISO 标准为 ISO646 标准。下表展示了基本 ASCII [字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)及其编码：

字母和数字的 ASCII 码的记忆是非常简单的。我们只要记住了一个字母或数字的 ASCII 码（例如记住 A 为 65 ， 0 的 ASCII 码为 48 ），知道相应的大小写字母之间差 32 ，就可以推算出其余字母、数字的 ASCII 码。

虽然标准 ASCII 码是 7 位编码，但由于计算机基本处理单位为[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)（ 1byte = 8bit ），所以一般仍以一个字节来存放一个 ASCII 字符。每一个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)中多余出来的一位（最高位）在计算机内部通常保持为 0 （在数据传输时可用作[奇偶校验位](https://baike.baidu.com/item/%E5%A5%87%E5%81%B6%E6%A0%A1%E9%AA%8C%E4%BD%8D)）。

由于标准 ASCII [字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)字符数目有限，在实际应用中往往无法满足要求。为此，[国际标准化组织](https://baike.baidu.com/item/%E5%9B%BD%E9%99%85%E6%A0%87%E5%87%86%E5%8C%96%E7%BB%84%E7%BB%87)又制定了 ISO2022 标准，它规定了在保持与 ISO646 兼容的前提下将 ASCII [字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)扩充为 8 位代码的统一方法。 ISO 陆续制定了一批适用于不同地区的扩充 ASCII [字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)，每种扩充 ASCII 字符集分别可以扩充 128 个字符，这些扩充字符的编码均为高位为 1 的 8 位代码（即十进制数 128~255 ），称为扩展 ASCII 码。

通过了解字符的存储编码，可以解决很多由编码不匹配引起的问题，比如网页[乱码](https://baike.baidu.com/item/%E4%B9%B1%E7%A0%81)、邮件乱码，本文简单扼要地阐明了[ASCII编码](https://baike.baidu.com/item/ASCII%E7%BC%96%E7%A0%81)、EBCDIC编码、GB2312编码、Unicode编码、[UTF-8](https://baike.baidu.com/item/UTF-8)编码、以及Base64编码。

## ISO-8859-1

[编辑](javascript:;)

ISO-8859-1编码是单[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)编码，向下兼容ASCII，其编码范围是0x00-0xFF，0x00-0x7F之间完全和ASCII一致，0x80-0x9F之间是[控制字符](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E5%AD%97%E7%AC%A6)，0xA0-0xFF之间是文字符号。

此字符集支持部分于欧洲使用的语言，包括[阿尔巴尼亚语](https://baike.baidu.com/item/%E9%98%BF%E5%B0%94%E5%B7%B4%E5%B0%BC%E4%BA%9A%E8%AF%AD)、[巴斯克语](https://baike.baidu.com/item/%E5%B7%B4%E6%96%AF%E5%85%8B%E8%AF%AD)、[布列塔尼语](https://baike.baidu.com/item/%E5%B8%83%E5%88%97%E5%A1%94%E5%B0%BC%E8%AF%AD)、加泰罗尼亚语、丹麦语、[荷兰语](https://baike.baidu.com/item/%E8%8D%B7%E5%85%B0%E8%AF%AD)、[法罗语](https://baike.baidu.com/item/%E6%B3%95%E7%BD%97%E8%AF%AD)、[弗里西语](https://baike.baidu.com/item/%E5%BC%97%E9%87%8C%E8%A5%BF%E8%AF%AD)、[加利西亚语](https://baike.baidu.com/item/%E5%8A%A0%E5%88%A9%E8%A5%BF%E4%BA%9A%E8%AF%AD)、德语、[格陵兰语](https://baike.baidu.com/item/%E6%A0%BC%E9%99%B5%E5%85%B0%E8%AF%AD)、[冰岛语](https://baike.baidu.com/item/%E5%86%B0%E5%B2%9B%E8%AF%AD)、[爱尔兰盖尔语](https://baike.baidu.com/item/%E7%88%B1%E5%B0%94%E5%85%B0%E7%9B%96%E5%B0%94%E8%AF%AD)、意大利语、拉丁语、[卢森堡语](https://baike.baidu.com/item/%E5%8D%A2%E6%A3%AE%E5%A0%A1%E8%AF%AD)、[挪威语](https://baike.baidu.com/item/%E6%8C%AA%E5%A8%81%E8%AF%AD)、葡萄牙语、里托罗曼斯语、[苏格兰盖尔语](https://baike.baidu.com/item/%E8%8B%8F%E6%A0%BC%E5%85%B0%E7%9B%96%E5%B0%94%E8%AF%AD)、西班牙语及瑞典语。

英语虽然没有[重音](https://baike.baidu.com/item/%E9%87%8D%E9%9F%B3)字母，但仍会标明为ISO/IEC 8859-1编码。除此之外，欧洲以外的部分语言，如[南非荷兰语](https://baike.baidu.com/item/%E5%8D%97%E9%9D%9E%E8%8D%B7%E5%85%B0%E8%AF%AD)、[斯瓦希里语](https://baike.baidu.com/item/%E6%96%AF%E7%93%A6%E5%B8%8C%E9%87%8C%E8%AF%AD)、[印尼语](https://baike.baidu.com/item/%E5%8D%B0%E5%B0%BC%E8%AF%AD)及[马来语](https://baike.baidu.com/item/%E9%A9%AC%E6%9D%A5%E8%AF%AD)、菲律宾他加洛语等也可使用ISO/IEC 8859-1编码。

法语及芬兰语本来也使用ISO/IEC 8859-1来表示。但因它没有法语使用的 [œ](https://baike.baidu.com/item/%C5%93)、[Œ](https://baike.baidu.com/item/%C5%92)、Ÿ 三个字母及芬兰语使用的 Š、š、Ž、ž ，故于1998年被ISO/IEC 8859-15所取代。（ISO 8859-15同时加入了欧元符号）

### ISO-8859-1简介

ISO-8859-1收录的[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)除ASCII收录的字符外，还包括西欧语言、希腊语、泰语、阿拉伯语、希伯来语对应的文字符号。[欧元符号](https://baike.baidu.com/item/%E6%AC%A7%E5%85%83%E7%AC%A6%E5%8F%B7)出现的比较晚，没有被收录在ISO-8859-1当中。

因为ISO-8859-1编码范围使用了单[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)内的所有空间，在支持ISO-8859-1的系统中传输和存储其他任何编码的[字节流](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82%E6%B5%81)都不会被抛弃。换言之，把其他任何编码的字节流当作ISO-8859-1编码看待都没有问题。这是个很重要的特性，[MySQL数据库](https://baike.baidu.com/item/MySQL%E6%95%B0%E6%8D%AE%E5%BA%93)默认编码是Latin1就是利用了这个特性。[ASCII编码](https://baike.baidu.com/item/ASCII%E7%BC%96%E7%A0%81)是一个7位的容器，ISO-8859-1编码是一个8位的容器。

HTML 4.01 支持 ISO 8859-1 (Latin-1) 字符集。[1]

ISO-8859-1 的较低部分（从 1 到 127 之间的代码）是最初的 7 比特 ASCII。

ISO-8859-1 的较高部分（从 160 到 255 之间的代码）全都有实体名称。

这些符号中的大多数都可以在不进行实体引用的情况下使用，但是实体名称或实体编号为那些不容易通过键盘键入的符号提供了表达的方法。

注释：实体名称对大小写敏感。

### Latin1与ISO-8859-1的关系

[编辑](javascript:;)

[Latin1](https://baike.baidu.com/item/Latin1)是ISO-8859-1的别名，有些环境下写作Latin-1。

## MBCS

为了扩充[ASCII编码](https://baike.baidu.com/item/ASCII%E7%BC%96%E7%A0%81)，以用于显示本国的语言，不同的国家和地区制定了不同的标准，由此产生了 GB2312, BIG5, JIS 等各自的编码标准。这些使用 2 个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)来代表一个字符的各种汉字延伸编码方式，称为 ANSI 编码，又称为"MBCS（Muilti-Bytes Charecter Set，多字节字符集）"。在简体中文系统下，ANSI 编码代表 GB2312 编码，在日文[操作系统](https://baike.baidu.com/item/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F)下，ANSI 编码代表 JIS 编码，所以在中文 windows下要转码成gb2312,gbk只需要把文本保存为ANSI 编码即可。 不同 ANSI 编码之间互不兼容，当信息在国际间交流时，无法将属于两种语言的文字，存储在同一段 ANSI 编码的文本中。一个很大的缺点是，同一个编码值，在不同的编码体系里代表着不同的字。这样就容易造成混乱。导致了unicode码的诞生。

其中每个语言下的ANSI编码，都有一套一对一的编码转换器，Unicode变成所有编码转换的中间介质。所有的编码都有一个转换器可以转换到Unicode，而Unicode也可以转换到其他所有的编码。

## GB2312

GB2312 也是ANSI编码里的一种，对ANSI编码最初始的[ASCII编码](https://baike.baidu.com/item/ASCII%E7%BC%96%E7%A0%81)进行扩充，为了满足国内在计算机中使用汉字的需要，中国国家标准总局发布了一系列的汉字[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)国家标准编码，统称为GB码，或国标码。其中最有影响的是于1980年发布的《信息交换用汉字编码[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86) 基本集》，标准号为GB 2312-1980,因其使用非常普遍，也常被通称为国标码。GB2312编码通行于我国内地；新加坡等地也采用此编码。几乎所有的中文系统和国际化的软件都支持GB 2312。

GB 2312是一个[简体中文](https://baike.baidu.com/item/%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87)[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)，由6763个常用汉字和682个[全角](https://baike.baidu.com/item/%E5%85%A8%E8%A7%92)的非汉字字符组成。其中汉字根据使用的频率分为两级。一级汉字3755个，二级汉字3008个。由于字符数量比较大，GB2312采用了二维矩阵编码法对所有字符进行编码。首先构造一个94行94列的方阵，对每一行称为一个“区”，每一列称为一个“位”，然后将所有字符依照下表的规律填写到方阵中。这样所有的字符在方阵中都有一个唯一的位置，这个位置可以用区号、位号合成表示，称为字符的区位码。如第一个汉字“啊”出现在第16区的第1位上，其区位码为1601。因为区位码同字符的位置是完全对应的，因此区位码同字符之间也是一一对应的。这样所有的字符都可通过其区位码转换为数字编码信息。GB2312字符的排列分布情况见表1－4。

表1－4 GB2312 字符编码分布表

|  |  |
| --- | --- |
| 分区范围 | 符号类型 |
| 第01区 | 中文标点、数学符号以及一些特殊字符 |
| 第02区 | 各种各样的数学序号 |
| 第03区 | 全角西文字符 |
| 第04区 | 日文平假名 |
| 第05区 | 日文片假名 |
| 第06区 | 希腊字母表 |
| 第07区 | 俄文字母表 |
| 第08区 | 中文拼音字母表 |
| 第09区 | 制表符号 |
| 第10－15区 | 无字符 |
| 第16－55区 | 一级汉字（以拼音字母排序） |
| 第56－87区 | 二级汉字（以部首笔画排序） |
| 第88－94区 | 无字符 |

GB2312字符在计算机中存储是以其区位码为基础的，其中汉字的区码和位码分别占一个存储单元，每个汉字占两个存储单元。由于区码和位码的取值范围都是在1－94之间，这样的范围同西文的存储表示冲突。例如汉字‘珀’在GB2312中的区位码为7174，其两[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)表示形式为71，74；而两个西文字符‘GJ’的存储码也是71,74。这种冲突将导致在解释编码时到底表示的是一个汉字还是两个西文字符将无法判断。

为避免同西文的存储发生冲突，GB2312字符在进行存储时，通过将原来的每个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)第8bit设置为1同西文加以区别，如果第8bit为0，则表示西文字符，否则表示GB2312中的字符。实际存储时，采用了将区位码的每个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)分别加上A0H（160）的方法转换为存储码，计算机存储规则是此编码的补码，而且是位码在前，区码在后。例如汉字‘啊’的区位码为1601，其存储码为B0A1H，其转换过程为：

|  |  |  |  |
| --- | --- | --- | --- |
| 区位码 | 区码转换 | 位码转换 | 存储码 |
| 1001H | 10H+A0H=B0H | 01H+A0H=A1H | B0A1H |

GB2312编码用两个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)(8位2进制)表示一个汉字，所以理论上最多可以表示256×256=65536个汉字。但这种编码方式也仅仅在中国行得通，如果您的网页使用的GB2312编码，那么很多外国人在浏览你的网页时就可能无法正常显示，因为其[浏览器](https://baike.baidu.com/item/%E6%B5%8F%E8%A7%88%E5%99%A8)不支持GB2312编码。当然，中国人在浏览外国网页(比如日文)时，也会出现[乱码](https://baike.baidu.com/item/%E4%B9%B1%E7%A0%81)或无法打开的情况，因为我们的[浏览器](https://baike.baidu.com/item/%E6%B5%8F%E8%A7%88%E5%99%A8)没有安装日文的编码表。

## GBK

GBK全称《汉字内码扩展规范》（GBK即“国标”、“扩展”汉语拼音的第一个字母，英文名称：Chinese Internal Code Specification） ，[中华人民共和国](https://baike.baidu.com/item/%E4%B8%AD%E5%8D%8E%E4%BA%BA%E6%B0%91%E5%85%B1%E5%92%8C%E5%9B%BD)全国信息技术标准化技术委员会1995年12月1日制订，[国家技术监督局](https://baike.baidu.com/item/%E5%9B%BD%E5%AE%B6%E6%8A%80%E6%9C%AF%E7%9B%91%E7%9D%A3%E5%B1%80)标准化司、电子工业部科技与[质量监督](https://baike.baidu.com/item/%E8%B4%A8%E9%87%8F%E7%9B%91%E7%9D%A3)司1995年12月15日联合以技监标函1995 229号文件的形式，将它确定为技术规范指导性文件。这一版的GBK规范为1.0版

### 基本含义

GBK 向下与 GB 2312 编码兼容，向上支持 ISO 10646.1[国际标准](https://baike.baidu.com/item/%E5%9B%BD%E9%99%85%E6%A0%87%E5%87%86)，是前者向后者过渡过程中的一个承上启下的产物。ISO 10646 是[国际标准化组织](https://baike.baidu.com/item/%E5%9B%BD%E9%99%85%E6%A0%87%E5%87%86%E5%8C%96%E7%BB%84%E7%BB%87) ISO 公布的一个编码标准，即 Universal Multilpe-Octet Coded Character Set（简称UCS），大陆译为《通用多八位编码字符集》，台湾译为《广用多八位元编码字元集》，它与 Unicode 组织的 Unicode 编码完全兼容。ISO 10646.1 是该标准的第一部分《体系结构与基本多文种平面》。我国 1993 年以 GB 13000.1 国家标准的形式予以认可（即 GB 13000.1 等同于 ISO 10646.1）。

GBK编码，是在[GB2312-80](https://baike.baidu.com/item/GB2312-80)标准基础上的[内码](https://baike.baidu.com/item/%E5%86%85%E7%A0%81)扩展规范，使用了双[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)编码方案，其编码范围从8140至FEFE（剔除xx7F），共23940个码位，共收录了21003个汉字，完全兼容[GB2312-80](https://baike.baidu.com/item/GB2312-80)标准，支持国际标准ISO/IEC10646-1和国家标准GB13000-1中的全部中日韩汉字，并包含了BIG5编码中的所有汉字。GBK编码方案于1995年10月制定， 1995年12月正式发布，目前中文版的[WIN95](https://baike.baidu.com/item/WIN95)、WIN98、[WINDOWS NT](https://baike.baidu.com/item/WINDOWS%20NT)以及WINDOWS 2000、WINDOWS XP、WIN 7等都支持GBK编码方案。

### 代码与缩写

[编辑](javascript:;)

1、原gb2312 HTML编码标签代码如下：

<metahttp-equiv="Content-Type"content="text/html;charset=gb2312"/>

2、可以缩写为<meta charset="gb2312"/>或<meta charset="gbk"/>[1]

3、gbk简写截图（右图）



4、缩写兼容性：所有浏览器均兼容，无论新旧版本IE还是不同品牌浏览器均兼容。至于GBK编码简写时候编码填写为gb2312还是填写为gbk，DIVCSS认为没有什么区别，均可。为了符合大家都使用gbk字符编码，大家可以写为“gb2312”。

## Big5

在台湾、香港与澳门地区，使用的是繁体中文[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)。而1980年发布的GB2312面向[简体中文](https://baike.baidu.com/item/%E7%AE%80%E4%BD%93%E4%B8%AD%E6%96%87)[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)，并不支持繁体汉字。在这些使用繁体中文[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)的地区，一度出现过很多不同厂商提出的字符集编码，这些编码彼此互不兼容，造成了信息交流的困难。为统一繁体[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)编码，1984年，台湾五大厂商宏碁、神通、佳佳、零壹以及大众一同制定了一种繁体中文编码方案，因其来源被称为五大码，英文写作Big5，后来按英文翻译回汉字后，普遍被称为[大五码](https://baike.baidu.com/item/%E5%A4%A7%E4%BA%94%E7%A0%81)。

[大五码](https://baike.baidu.com/item/%E5%A4%A7%E4%BA%94%E7%A0%81)是一种繁体中文汉字[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)，其中繁体汉字13053个，808个标点符号、希腊字母及特殊符号。[大五码](https://baike.baidu.com/item/%E5%A4%A7%E4%BA%94%E7%A0%81)的编码码表直接针对[存储](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8)而设计，每个字符统一使用两个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)存储表示。第1[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)范围81H－FEH，避开了同ASCII码的冲突，第2字节范围是40H－7EH和A1H－FEH。因为Big5的字符编码范围同GB2312字符的存储码范围存在冲突，所以在同一正文不能对两种[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)的字符同时支持。

Big5编码的分布如表1－5所示，Big5字符主要部分集中在三个段内：标点符号、希腊字母及特殊符号；常用汉字；非常用汉字。其余部分保留给其他厂商支持。

表1－5　Big5字符编码分布表

|  |  |
| --- | --- |
| 编码范围 | 符号类别 |
| 8140H－A0FEH | 保留（用作造字区） |
| A140H－A3BFH | 标点符号、希腊字母及特殊符号 |
| A3C0H－A3FEH | 保留（未开放用于造字区） |
| A440H－C67EH | 常用汉字（先按笔划，再按部首排序） |
| C6A1H－C8FEH | 保留（用作造字区） |
| C940H－F9D5H | 非常用汉字（先按笔划，再按部首排序） |
| F9D6H－FEFEH | 保留（用作造字区） |

Big5编码推出后，得到了繁体中文软件厂商的广泛支持，在使用繁体汉字的地区迅速普及使用。目前，Big5编码在台湾、香港、澳门及其他海外华人中普遍使用，成为了繁体中文编码的事实标准。在互联网中检索繁体中文网站，所打开的网页中，大多都是通过Big5编码产生的文档。

## Unicode

如上ANSI编码条例中所述，世界上存在着多种编码方式，在ANSi编码下，同一个编码值，在不同的编码体系里代表着不同的字。在简体中文系统下，ANSI 编码代表 GB2312 编码，在日文[操作系统](https://baike.baidu.com/item/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F)下，ANSI 编码代表 JIS 编码，可能最终显示的是中文，也可能显示的是日文。在ANSI编码体系下，要想打开一个文本文件，不但要知道它的编码方式，还要安装有对应编码表，否则就可能无法读取或出现[乱码](https://baike.baidu.com/item/%E4%B9%B1%E7%A0%81)。为什么[电子邮件](https://baike.baidu.com/item/%E7%94%B5%E5%AD%90%E9%82%AE%E4%BB%B6)和网页都经常会出现[乱码](https://baike.baidu.com/item/%E4%B9%B1%E7%A0%81)，就是因为信息的提供者可能是日文的ANSI编码体系和信息的读取者可能是中文的编码体系，他们对同一个二进制编码值进行显示，采用了不同的编码，导致乱码。这个问题促使了unicode码的诞生。

如果有一种编码，将世界上所有的符号都纳入其中，无论是英文、日文、还是中文等，大家都使用这个[编码表](https://baike.baidu.com/item/%E7%BC%96%E7%A0%81%E8%A1%A8)，就不会出现编码不匹配现象。每个符号对应一个唯一的编码，[乱码](https://baike.baidu.com/item/%E4%B9%B1%E7%A0%81)问题就不存在了。这就是Unicode编码。

Unicode当然是一个很大的集合，现在的规模可以容纳100多万个符号。每个符号的编码都不一样，比如，U+0639表示阿拉伯字母Ain，U+0041表示英语的大写字母A，“汉”这个字的Unicode编码是U+6C49。

Unicode固然统一了编码方式，但是它的效率不高，比如UCS-4(Unicode的标准之一)规定用4个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)[存储](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8)一个符号，那么每个英文字母前都必然有三个字节是0，这对存储和传输来说都很耗资源。

## UTF-8

UTF-8（8-bit Unicode Transformation Format）是一种针对Unicode的可变长度字符编码，又称万国码，由Ken Thompson于1992年创建。现在已经标准化为RFC 3629。UTF-8用1到6个字节编码Unicode字符。用在网页上可以统一页面显示中文简体繁体及其它语言（如英文，日文，韩文）。

### 发展历程

在所有[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)中，最知名的可能要数被称为[ASCII](https://baike.baidu.com/item/ASCII)的7位字符集了。它是[美国标准信息交换代码](https://baike.baidu.com/item/%E7%BE%8E%E5%9B%BD%E6%A0%87%E5%87%86%E4%BF%A1%E6%81%AF%E4%BA%A4%E6%8D%A2%E4%BB%A3%E7%A0%81)（American Standard Code for Information Interchange）的缩写, 为美国英语通信所设计。它由128个[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)组成，包括大小写字母、数字0-9、标点符号、非[打印](https://baike.baidu.com/item/%E6%89%93%E5%8D%B0)字符（换行符、[制表符](https://baike.baidu.com/item/%E5%88%B6%E8%A1%A8%E7%AC%A6)等4个）以及[控制字符](https://baike.baidu.com/item/%E6%8E%A7%E5%88%B6%E5%AD%97%E7%AC%A6)（退格、响铃等）组成。

但是，由于他是针对英语设计的，当处理带有音调标号（形如汉语的拼音）的亚洲文字时就会出现问题。因此，创建出了一些包括255个字符的由ASCII扩展的字符集。其中有一种通常被称为IBM字符集，它把值为128-255之间的字符用于画图和画线，以及一些特殊的欧洲字符。另一种8位字符集是[ISO 8859-1](https://baike.baidu.com/item/ISO%208859-1)Latin 1，也简称为ISOLatin-1。它把位于128-255之间的[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)用于拉丁字母表中特殊语言字符的编码，也因此而得名。欧洲语言不是地球上的唯一语言，因此亚洲和非洲语言并不能被8位[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)所支持。仅汉语字母表（或pictograms）就有80000以上个[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)。但是把汉语、日语和越南语的一些相似的字符结合起来，在不同的语言里，使不同的字符代表不同的字，这样只用2个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)就可以编码地球上几乎所有地区的文字。因此，创建了[UNICODE](https://baike.baidu.com/item/UNICODE)编码。它通过增加一个高字节对ISO Latin-1字符集进行扩展，当这些高字节位为0时，低字节就是ISO Latin-1字符。UNICODE支持欧洲、非洲、中东、亚洲（包括统一标准的东亚象形汉字和韩国表音文字）。但是，UNICODE并没有提供对诸如Braille,[Cherokee](https://baike.baidu.com/item/Cherokee/61025), Ethiopic, Khmer, Mongolian, Hmong, Tai Lu, Tai Mau文字的支持。同时它也不支持如Ahom, Akkadian, Aramaic, [Babylonian](https://baike.baidu.com/item/Babylonian)Cuneiform, Balti, Brahmi, Etruscan, Hittite, Javanese, Numidian, Old Persian Cuneiform, Syrian之类的古老文字。

事实证明，对可以用[ASCII](https://baike.baidu.com/item/ASCII)表示的[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)使用UNICODE并不高效，因为UNICODE比ASCII占用大一倍的空间，而对ASCII来说高[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)的0对他毫无用处。为了解决这个问题，就出现了一些中间格式的[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)，他们被称为通用转换格式，即[UTF](https://baike.baidu.com/item/UTF)（Unicode Transformation Format）。常见的UTF格式有：UTF-7, UTF-7.5, UTF-8,[UTF-16](https://baike.baidu.com/item/UTF-16), 以及 [UTF-32](https://baike.baidu.com/item/UTF-32)。

### 字符集

如果UNICODE[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)由2个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)表示，则编码成UTF-8很可能需要3个字节。而如果UNICODE字符由4个字节表示，则编码成UTF-8可能需要6个字节。用4个或6个字节去编码一个UNICODE字符可能太多了，但很少会遇到那样的UNICODE字符。

UTF-8编码规则：如果只有一个字节则其最高二进制位为0；如果是多字节，其第一个字节从最高位开始，连续的二进制位值为1的个数决定了其编码的字节数，其余各字节均以10开头。UTF-8转换表表示如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Unicode/UCS-4** | bit数 | **UTF-8** | byte数 | 备注 |
| 0000 ~  007F | 0~7 | **0**XXX XXXX | 1 |  |
| 0080 ~  07FF | 8~11 | **110**X XXXX  **10**XX XXXX | 2 |  |
| 0800 ~  FFFF | 12~16 | **1110**XXXX  **10**XX XXXX  **10**XX XXXX | 3 | 基本定义范围：0~FFFF |
| 1 0000 ~  1F FFFF | 17~21 | **1111 0**XXX  **10**XX XXXX  **10**XX XXXX  **10**XX XXXX | 4 | Unicode6.1定义范围：0~10 FFFF |
| 20 0000 ~  3FF FFFF | 22~26 | **1111 10**XX  **10**XX XXXX  **10**XX XXXX  **10**XX XXXX  **10**XX XXXX | 5 | 说明：此非unicode编码范围，属于UCS-4 编码  早期的规范UTF-8可以到达6字节序列，可以覆盖到31位元（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。根据规范，这些字节值将无法出现在合法 UTF-8序列中 |
| 400 0000 ~  7FFF FFFF | 27~31 | **1111 110**X  **10**XX XXXX  **10**XX XXXX  **10**XX XXXX  **10**XX XXXX  **10**XX XXXX | 6 |

实际表示ASCII[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)的UNICODE字符，将会编码成1个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)，并且UTF-8表示与ASCII字符表示是一样的。所有其他的UNICODE字符转化成UTF-8将需要至少2个字节。每个字节由一个[换码序列](https://baike.baidu.com/item/%E6%8D%A2%E7%A0%81%E5%BA%8F%E5%88%97)(转义字符)开始。第一个字节由唯一的换码序列，由n位连续的1加一位0组成, 首字节连续的1的个数表示[字符编码](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E7%BC%96%E7%A0%81)所需的字节数

Unicode转换为UTF-8时，可以将Unicode二进制从低位往高位取出二进制数字，每次取6位，如上述的二进制就可以分别取出为如下示例所示的格式，前面按格式填补，不足8位用0填补。

注：Unicode转换为UTF-8需要的字节数可以根据这个规则计算：如果Unicode小于0X80（Ascii字符），则转换后为1个字节。否则转换后的字节数为Unicode二进制位数减1再除以5。

### 优缺点

#### 优点

UTF-8编码可以通过屏蔽位和[移位](https://baike.baidu.com/item/%E7%A7%BB%E4%BD%8D)操作快速读写。[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)串比较时strcmp()和[wcscmp](https://baike.baidu.com/item/wcscmp)()的返回结果相同，因此使排序变得更加容易。[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)FF和FE在UTF-8编码中永远不会出现，因此他们可以用来表明UTF-16或UTF-32文本（见BOM） UTF-8 是[字节顺序](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82%E9%A1%BA%E5%BA%8F)无关的。它的字节顺序在所有系统中都是一样的，因此它实际上并不需要BOM。

#### 缺点

你无法从UNICODE[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)数判断出UTF-8文本的[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)数，因为UTF-8是一种[变长编码](https://baike.baidu.com/item/%E5%8F%98%E9%95%BF%E7%BC%96%E7%A0%81)它需要用2个字节编码那些用[扩展ASCII](https://baike.baidu.com/item/%E6%89%A9%E5%B1%95ASCII)[字符集](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E9%9B%86)只需1个字节的字符 ISO Latin-1 是UNICODE的[子集](https://baike.baidu.com/item/%E5%AD%90%E9%9B%86)，但不是UTF-8的子集 8位字符的UTF-8编码会被email网关过滤，因为internet信息最初设计为7位ASCII码。因此产生了UTF-7编码。 UTF-8 在它的表示中使用值100xxxxx的几率超过50%， 而现存的实现如ISO 2022， 4873， 6429， 和8859系统，会把它错认为是C1 控制码。因此产生了UTF-7.5编码。

### 修正更新

[java](https://baike.baidu.com/item/java/85979)使用UTF-16表示内部文本，并支持用于[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)串[串行化](https://baike.baidu.com/item/%E4%B8%B2%E8%A1%8C%E5%8C%96)的非标准的修正UTF-8编码。

[](https://baike.baidu.com/pic/UTF-8/481798/0/ae10edde6a81136c94ee37ec?fr=lemma&ct=single)UTF-8保存使用

标准UTF-8和修正的UTF-8有两点不同：

修正的UTF-8中，null[字符编码](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6%E7%BC%96%E7%A0%81)成2个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)（1100000010000000）而不是标准的1个字节（00000000），这样作可以保证编码后的字符串中不会嵌入null字符。因此如果在[类C语言](https://baike.baidu.com/item/%E7%B1%BBC%E8%AF%AD%E8%A8%80)中处理[字符](https://baike.baidu.com/item/%E5%AD%97%E7%AC%A6)串，文本不会在第一个null字符时截断（C字符串以'\0'结尾）。

在标准UTF-8编码中，超出基本多语言范围（BMP-Basic Multilingual Plane）的字符被编码为4字节格式，但是在修正的UTF-8编码中，他们由[代理](https://baike.baidu.com/item/%E4%BB%A3%E7%90%86)编码对（surrogatepairs）表示，然后这些代理编码对在序列中分别重新编码。结果标准UTF-8编码中需要4个[字节](https://baike.baidu.com/item/%E5%AD%97%E8%8A%82)的字符，在修正后的UTF-8编码中将需要6个字节。

https://en.wikipedia.org/wiki/Byte\_order\_mark

BOM的UTF-8表示是（十六进制）的字节序列：0xEF,0xBB,0xBF。

Unicode标准允许在UTF-8中使用BOM，但是不要求，也不推荐使用。字节顺序（BOM）在UTF-8中没有意义，因此它使用在UTF-8中标识在文本流开头仅仅是为了表明文本流是以UTF-8编码的，或者文本流是从包含可选择的BOM一个流中转换来的。当这个BOM在文本中存在的时候，Unicode标准也不推荐将其去除，这样编码之间的转换不会丢失信息， 并且对于依赖于此BOM的代码也能够继续去工作。IETF建议，如果协议(a)总是使用UTF-8，或者(b)有其他方法来指示正在使用的编码，那么它“应该禁止使用U+FEFF作为签名”。

不使用BOM，这样允许文本向后兼容一些不支持unicode的软件。例如那些允许在文本中包含非ASCII字节，这些字节并不是在文件的开头的编程语言。

试探性的分析能够充分确定是否使用的UTF-8带有BOM，因为有大量的字节在UTF-8中是无效的。

Microsoft编译器和解释器，以及Microsoft Windows上的许多软件(如Notepad)都将BOM视为必需的神奇数字，而不是试探性的使用。这些工具在将文本保存为UTF-8时添加一个BOM，并且除非有BOM或者文件只包含ASCII，否则不能够解析UTF-8。当将文本文档转为纯文本文件时，谷歌文档也添加了BOM

BOM不受欢迎主要是在UNIX环境下，因为很多UNIX程序不鸟BOM。主要问题出在UNIX那个所有脚本语言通行的首行#!标示，这东西依赖于shell解析，而很多shell出于兼容的考虑不检测BOM，所以加进BOM时shell会把它解释为某个普通字符输入导致破坏#!标示，这就麻烦了。其实很多现代脚本语言，比如Python，其解释器本身都是能处理BOM的，但是shell卡在这里，没办法，只能躺着也中枪。说起来这也不能怪shell，因为BOM本身违反了一个UNIX设计的常见原则，就是文档中存在的数据必须可见。BOM不能作为可见字符被文本编辑器编辑，就这一条很多UNIX开发者就不满意。

顺便说一句，即使脚本语言能处理BOM，随处使用BOM也不是推荐的办法。各个脚本语言对Unicode的处理都有自己的一套，Python的 # -\*- coding: utf-8 -\*-，Perl的use utf8，都比BOM简单而且可靠。另一个好消息是，即使是必须在Windows和UNIX之间切换的朋友也不会悲催。幸亏在UNIX环境下我们还有VIM这种神器，即使遇到BOM挡道，我们也可以通过 set nobomb; set fileencoding=utf8; w 三条命令解决问题

总结：

ANSI也称为MBCS，为了扩充[ASCII编码](https://baike.baidu.com/item/ASCII%E7%BC%96%E7%A0%81)，以用于显示本国的语言，不同的国家和地区制定了不同的标准，由此产生了 GB2312, BIG5, JIS 等各自的编码标准。

ISO8859-1欧洲编码，包含拉丁字符，向下兼容ASCII码（一个字节的7位或者8位表示一个字符）。用一个字节表示一个字符。

GBK汉字内码扩展码，向下兼容GB2312,比GB2312包含更多的字符（如古汉字，不常见的汉字）,用两个字节表示一个字符

Big5台湾香港等使用繁体字的地区使用

Unicode可以编码所有的字符