**Unicode**

Unicode（中文：万国码、国际码、统一码、单一码）是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伴随着通用字符集的标准而发展，同时也以书本的形式[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]，已经收录超过十万个字符（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过程。有很多新科技，如可扩展置标语言(Extensible Markup Language，简称：XML)、Java编程语言以及现代的操作系统，都采用Unicode编码。

**来历**

Unicode是为了解决传统的字符编码方案的局限而产生的，例如ISO 8859-1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）。

**解决问题**

解决传统的字符编码方案的局限而产生的，例如ISO 8859-1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）。

**占字节数**

2

**兼容语言**

几乎所有语言

**UTF-8（8-bit Unicode Transformation Format）**

UTF-8（8-bit Unicode Transformation Format）是一种针对Unicode的可变长度字符编码，也是一种前缀码。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仍与ASCII兼容，这使得原来处理ASCII字符的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、网页及其他存储或发送文字的应用中，优先采用的编码。

UTF-8使用一至六个字节为每个字符编码（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. 带有附加符号的拉丁文、希腊文、西里尔字母、亚美尼亚语、希伯来文、阿拉伯文、叙利亚文及它拿字母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（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，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，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（SCSU）。

互联网工程工作小组（IETF）要求所有互联网协议都必须支持UTF-8编码[1]。互联网邮件联盟（IMC）建议所有电子邮件软件都支持UTF-8编码。[1]

**来历**

1992年初，为创建良好的字节串编码系统以供多字节字符集使用，开始了一个正式的研究。ISO/IEC 10646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比特的字符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委员会XoJIG开始寻求一个较佳的编码系统。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比特的ASCII符号只代表原来的意思，所有多字节序列则会包含第8比特的符号，也就是所谓的最高有效比特。

1992年8月，这个建议由IBMX/Open的代表流传到一些感兴趣的团体。与此同时，贝尔实验室九号项目操作系统工作小组的肯·汤普逊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·汤普逊和罗勃·派克一起在美国新泽西州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日的在圣地牙哥举行的USENIX会议首次正式介绍UTF-8。

自1996年起，微软的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。

**解决问题**

UTF-8的设计有以下的多字符组序列的特质：

* 单字节字符的最高有效比特永远为0。
* 多字节序列中的首个字符组的几个最高有效比特决定了序列的长度。最高有效位为110的是2字节序列，而1110的是三字节序列，如此类推。
* 多字节序列中其余的字节中的首两个最高有效比特为10。

UTF-8的这些特质，保证了一个字符的字节序列不会包含在另一个字符的字节序列中。这确保了以字节为基础的部分字符串比对（sub-string match）方法可以适用于在文字中搜索字或词。有些比较旧的可变长度8位编码（如Shift JIS）没有这个特质，故字符串比对的算法变得相当复杂。虽然这增加了UTF-8编码的字符串的信息冗余，但是利多于弊。另外，数据压缩并非Unicode的目的，所以不可混为一谈。即使在发送过程中有部分字节因错误或干扰而完全丢失，还是有可能在下一个字符的起点重新同步，令受损范围受到限制。

另一方面，由于其字节序列设计，如果一个疑似为字符串的序列被验证为UTF-8编码，那么我们可以有把握地说它是UTF-8字符串。一段两字节随机序列碰巧为合法的UTF-8而非ASCII的概率为32分1。对于三字节序列的概率为256分1，对更长的序列的概率就更低了。

**占字节数**

每个字符最小字节数 1

每个字符最大字节数 4

**兼容语言**

几乎所有语言

**GB 2312**

GB 2312或 GB 2312–80 是中华人民共和国国家标准简体中文字符集，全称《信息交换用汉字编码字符集·基本集》，又称GB0，由中国国家标准总局发布，1981年5月1日实施。GB 2312编码通行于中国大陆；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。

来历：随着我国汉字信息处理技术的发展,计算机的应用范围不断扩大,使用汉字字数较多的部门迫切需要在《基本集》的基础上继续制定信息交换用汉字编码字符集各辅助集的国家标准。

**解决问题**

GB 2312的出现，基本满足了汉字的计算机处理需要，它所收录的汉字已经覆盖中国大陆99.75%的使用频率。但对于人名、古汉语等方面出现的罕用字和繁体字，GB 2312不能处理，因此后来GBK及GB 18030汉字字符集相继出现以解决这些问题。

**占字节数**

每个汉字及符号以两个字节来表示。第一个字节称为“高位字节”，第二个字节称为“低位字节”。、

**兼容语言**

收录6763个汉字，其中一级汉字3755个，二级汉字3008个；同时收录了包括拉丁字母、希腊字母、日文平假名及片假名字母、俄语西里尔字母在内的682个字符。

**Big5**

又称为大五码或五大码，是使用繁体中文（正体中文）社区中最常用的电脑汉字字符集标准，共收录13,060个汉字[1]。

中文码分为内码及交换码两类，Big5属中文内码，知名的中文交换码有CCCII、CNS11643。

Big5虽普及于台湾、香港与澳门等繁体中文通行区，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。倚天中文系统、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。

**来历**

大五码”（Big5）是由台湾财团法人信息产业策进会为五大中文套装软件所设计的中文共通内码，在1983年12月完成公告[2][3]，隔年3月，信息产业策进会与台湾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）”[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][6][7][8]。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码的国乔中文系统及倚天中文系统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，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、王安码等内码，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，研发中文电脑的朱邦复认为内码字集应该广纳所有的正异体字，以顾及如户政等应用上的需要，故在当时的内码会议中，建议希望采用他的五万多字的字库。工程师认为虽其技术可行，但是三个字节（超过两个字节）长度的内码却会造成英文显示屏画面映射成中文画面会发生文字无法对齐的问题，因为当时盛行之倚天中文系统画面系以两个字节文字宽度映射成一个中文字图样，英文软件中只要以两个英文字宽度去显示一个中文字，画面就不会乱掉，造成中文系统业者偏爱二个字节长度的内码[9]；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，其研究成果更不可能获采用。[10]

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的高度普及，使后来的微软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、台北市电脑公会所推动的公会码等，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，当中国大陆的电子邮件和转码软件还未普遍之时，在深圳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。使用简体中文的社区，最常用的是GB 2312、GBK及其后续的国标码（GB 18030）。

除了台湾外，其他使用繁体汉字的地区，如香港（香港增补字符集）、澳门，及使用繁体汉字的海外华人，都曾普遍使用Big5码做为中文内码及交换码。

**解决问题**

推出统一编码，使台湾早日进入资讯时代

**占字节数**

每个汉字及符号以两个字节来表示。第一个字节称为“高位字节”，第二个字节称为“低位字节”。

**兼容语言**

繁体中文 包括香港地名、人名用汉字、粤语用字（包括粗言秽语在内，这是应警方及法庭需要记录口供的需要）、异体字、小部分简体字、平假名、片假名及俄语西里尔字母。

**UTF-16**

UTF-16是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字符集的抽象码位映射为16位长的整数（即码元）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。

UTF是"Unicode/UCS Transformation Format"的首字母缩写，即把Unicode字符转换为某种格式之意。UTF-16正式定义于ISO/IEC 10646-1的附录C，而RFC2781也定义了相似的做法。

**来历**

在Unicode基本多文种平面定义的字符（无论是拉丁字母、汉字或其他文字或符号），一律使用2字节储存+。而在辅助平面定义的字符，会以代理对（surrogate pair）的形式，以两个2字节的值来储存。

**解决问题**

UTF-16比起UTF-8，好处在于大部分字符都以固定长度的字节 (2字节) 储存，但UTF-16却无法兼容于ASCII编码。

**占字节数**

2

**兼容语言**

几乎所有语言

**字符集之间的转换**

UTF-8，UTF16，Unicode之间任意二者可以直接转换。

Gb2312需要通过查表转换为UTF-8，UTF16，Unicode.

Big5需要通过查表转换为UTF-8，UTF16，Unicode.

Gb2312和Big5之间的转换需要通过转化为Unicode间接转换

**参考资料**

https://zh.wikipedia.org/wiki/UTF-16

https://zh.wikipedia.org/wiki/%E5%A4%A7%E4%BA%94%E7%A2%BC

https://zh.wikipedia.org/wiki/UTF-8

https://zh.wikipedia.org/wiki/Unicode

https://zh.wikipedia.org/wiki/Gb\_2312

https://baike.baidu.com/item/%E4%BF%A1%E6%81%AF%E4%BA%A4%E6%8D%A2%E7%94%A8%E6%B1%89%E5%AD%97%E7%BC%96%E7%A0%81%E5%AD%97%E7%AC%A6%E9%9B%86/8074272?fr=aladdin&fromid=1682930&fromtitle=GB+2312

https://stackoverflow.com/questions/17211780/how-do-i-convert-gbk-to-utf8-with-pure-javascript