Unicode（统一码） 字符编码 编码 UTF-8

关注者 2,201 被浏览 279,735

Unicode 和 UTF-8 有何区别？

面试时被问到了，网上看了一下，实在是看不懂。
最好几句话概括一下，简介就好

关注问题 写回答 2 条评论 分享 邀请回答 ···

47 个回答 默认排序

邱昊宇
C++ 话题的优秀回答者

收录于编辑推荐 · 415 人赞同了该回答

简单来说：

- Unicode 是「字符集」
- UTF-8 是「编码规则」

其中：

- 字符集：为每一个「字符」分配一个唯一的 ID（学名为码位 / 码点 / Code Point）
- 编码规则：将「码位」转换为字节序列的规则（编码/解码 可以理解为 加密/解密 的过程）

广义的 Unicode 是一个标准，定义了一个字符集以及一系列的编码规则，即 Unicode 字符集和 UTF-8、UTF-16、UTF-32 等等编码……

Unicode 字符集为每一个字符分配一个码位，例如「知」的码位是 30693，记作 U+77E5（30693 的十六进制为 0x77E5）。 展开阅读全文

▲ 415 ▼ 26 条评论 分享 收藏 感谢

于洋
主页封面是冲绳古宇利岛

2,481 人赞同了该回答

很久很久以前，有一群人，他们决定用8个可以开合的晶体管来组合成不同的状态，以表示世界上的万物。他们看到8个开关状态是好的，于是他们把这称为”**字节**“。再后来，他们又做了一些可以处理这些字节的机器，机器开动了，可以用字节来组合出很多状态，状态开始变来变去。他们看到这样是好的，于是它们就这机器称为”**计算机**“。

开始计算机只在美国用。八位的字节一共可以组合出256(2的8次方)种不同的状态。
他们把其中的编号从0开始的32种状态分别规定了特殊的用途，一但终端、打印机遇上约定好的这些字节被传过来时，就要做一些约定的动作：

遇上0×10, 终端就换行；

遇上0×07, 终端就向人们嘟嘟叫；

遇上0x1b, 打印机就打印反白的字，或者终端就用彩色显示字母。

他们看到这样很好，于是就把这些0×20以下的字节状态称为”控制码“。他们又把所有的空格、标点符号、数字、大小写字母分别用连续的字节状态表示，一直编到了第127号，这样计算机就可以用不同字节来存储英语的文字了。大家看到这样，都感觉
很好，于是大家都把这个方案叫做 **ANSI** 的”Ascii”编码（American Standard Code for Information Interchange，美国信息互换标准代码）。当时世界上所有的计算机都用同样的**ASCII**方案来保存英文文字。

相关问题

为什么 Unicode 中会存在「凉」和「凉」这样两个极其相像的字符？ 4 个回答

在计算机中为何不直接使用UTF8编码进行存储，而要使用Unicode再转换成UTF8？ 26 个回答

Python2.7 中文字符编码，使用Unicode时，选择什么编码格式？ 5 个回答

Windows 记事本的 ANSI、Unicode、UTF-8 这三种编码模式有什么区别？ 13 个回答

截止到 2017年，C++ 对于 Unicode 支持情况如何？ 10 个回答

相关推荐

后来，就像建造巴比伦塔一样，世界各地都开始使用计算机，但是很多国家用的不是英文，他们的字母里有许多是ASCII里没有的，为了可以在计算机保存他们的文字，他们决定采用127号之后的空位来表示这些新的字母、符号，还加入了很多画表格时需要用下到的横线、竖线、交叉等形状，一直把序号编到了最后一个状态255。从128 到255这一页的字符集被称”**扩展字符集**“。从此之后，贪婪的人类再没有新的状态可以用了，美帝国主义可能没有想到还有第三世界国家的人们也希望可以用到计算机吧！

等中国人们得到计算机时，已经没有可以利用的字节状态来表示汉字，况且有6000多个常用汉字需要保存呢。但是这难不倒智慧的中国人民，我们不客气地把那些127号之后的奇异符号们直接取消掉, 规定：一个小于127的字符的意义与原来相同，但两个大于127的字符连在一起时，就表示一个汉字，前面的一个字节（他称之为高字节）从0xA1用到0xF7，后面一个字节（低字节）从0xA1到0xFE，这样我们就可以组合出大约7000多个简体汉字了。在这些编码里，我们还把数学符号、罗马希腊的字母、日文的假名们都编进去了，连在 ASCII 里本来就有的数字、标点、字母都统统重新编了两个字节长的编码，这就是常说的”全角”字符，而原来在127号以下的那些就叫”半角”字符了。中国人民看到这样很不错，于是就把这种汉字方案叫做 “**GB2312**“。GB2312 是对 ASCII 的中文扩展。

但是中国的汉字太多了，我们很快就就发现有许多人的人名没有办法在这里打出来，特别是某些很会麻烦别人的国家领导人。于是我们不得不继续把GB2312 没有用到的码位找出来老实不客气地用上。后来还是不够用，于是干脆不再要求低字节一定是127号之后的内码，只要第一个字节是大于127就固定表示这是一个汉字的开始，不管后面跟的是不是扩展字符集里的内容。结果扩展之后的编码方案被称为 **GBK** 标准，GBK包括了GB2312 的所有内容，同时又增加了近20000个新的汉字（包括繁体字）和符号。 后来少数民族也要用电脑了，于是我们再扩展，又加了几千个新的少数民族的字，GBK扩成了 **GB18030**。从此之后，中华民族的文化就可以在计算机时代中传承了。 中国的程序员们看到这一系列汉字编码的标准是好的，于是通称他们叫做 “**DBCS**“（Double Byte Charecter Set 双字节字符集）。在DBCS系列标准里，最大的特点是两字节长的汉字字符和一字节长的英文字符并存于同一套编码方案里，因此他们写的程序为了支持中文处理，必须要注意字串里的每一个字节的值，如果这个值是大于127的，那么就认为一个双字节字符集里的字符出现了。那时候凡是受过加持，会编程的计算机僧侣们都要每天念下面这个咒语数百遍： “一个汉字算两个英文字符！一个汉字算两个英文字符……”

因为当时各个国家都像中国这样搞出一套自己的编码标准，结果互相之间谁也不懂谁的编码，谁也不支持别人的编码，连大陆和台湾这样只相隔了150海里，使用着同一种语言的兄弟地区，也分别采用了不同的 DBCS 编码方案——当时的中国人想让电脑显示汉字，就必须装上一个”汉字系统”，专门用来处理汉字的显示、输入的问题，像是那个台湾的愚昧封建人士写的算命程序就必须加装另一套支持 BIG5 编码的什么”倚天汉字系统”才可以用，装错了字符系统，显示就会乱了套！这怎么办？而且世界民族之林中还有那些一时用不上电脑的穷苦人民，他们的文字又怎么办？真是计算机的巴比伦塔命题啊！

正在这时，大天使加百列及时出现了——一个叫 **ISO**（国际标谁化组织）的国际组织决定着手解决这个问题。他们采用的方法很简单：废了所有的地区性编码方案，重新搞一个包括了地球上所有文化、所有字母和符号 的编码！他们打算叫它”Universal Multiple-Octet Coded Character Set”，简称 **UCS**, 俗称 “**unicode**“。

unicode开始制订时，计算机的存储器容量极大地发展了，空间再也不成为问题了。于是 ISO 就直接规定必须用两个字节，也就是16位来统一表示所有的字符，对于ASCII里的那些“半角”字符，unicode包持其原编码不变，只是将其长度由原来的8位扩展为16位，而其他文化和语言的字符则全部重新统一编码。由于”半角”英文符号只需要用到低8位，所以其高8位永远是0，因此这种大气的方案在保存英文文本时会多浪费一倍的空间。

这时候，从旧社会里走过来的程序员开始发现一个奇怪的现象：他们的 *strlen* 函数靠不住了，一个汉字不再是相当于两个字符了，而是一个！是的，从unicode开始，无论是半角的英文字母，还是全角的汉字，它们都是统一的”**一个字符**“！同时，也都是统一的”**两个字节**“，请注意”字符”和”字节”两个术语的不同，“**字节**”是一个8位的物理存贮单元，而“**字符**”则是一个文化相

关的符号。在unicode中，一个字符就是两个字节。一个汉字算两个英文字符的时代已经快过去了。

unicode同样也不完美，这里就有两个的问题，一个是，如何才能区别unicode和ascii？计算机怎么知道三个字节表示一个符号，而不是分别表示三个符号呢？第二个问题是，我们已经知道，英文字母只用一个字节表示就够了，如果unicode统一规定，每个符号用三个或四个字节表示，那么每个英文字母前都必然有二到三个字节是0，这对于存储空间来说是极大的浪费，文本文件的大小会因此大出二三倍，这是难以接受的。

unicode在很长一段时间内无法推广，直到互联网的出现，为解决unicode如何在网络上传输的问题，于是面向传输的众多 **UTF**（UCS Transfer Format）标准出现了，顾名思义，**UTF-8**就是每次8个位传输数据，而**UTF-16**就是每次16个位。UTF-8就是在互联网上使用最广的一种unicode的实现方式，这是为传输而设计的编码，并使编码无国界，这样就可以显示全世界上所有文化的字符了。UTF-8最大的一个特点，就是它是一种变长的编码方式。它可以使用1~4个字节表示一个符号，根据不同的符号而变化字节长度，当字符在ASCII码的范围时，就用一个字节表示，保留了ASCII字符一个字节的编码做为它的一部分，注意的是unicode一个中文字符占2个字节，而UTF-8一个中文字符占3个字节）。从unicode到utf-8并不是直接的对应，而是要过一些算法和规则来转换。

Unicode符号范围 | UTF-8编码方式

(十六进制) | （二进制）

———————————————————————

0000 0000-0000 007F | 0xxxxxxx

0000 0080-0000 07FF | 110xxxxx 10xxxxxx

0000 0800-0000 FFFF | 1110xxxx 10xxxxxx 10xxxxxx

0001 0000-0010 FFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

最后简单总结一下：

- 中国人民通过对 ASCII 编码的中文扩充改造，产生了 GB2312 编码，可以表示6000多个常用汉字。
- 汉字实在是太多了，包括繁体和各种字符，于是产生了 GBK 编码，它包括了 GB2312 中的编码，同时扩充了很多。
- 中国是个多民族国家，各个民族几乎都有自己独立的语言系统，为了表示那些字符，继续把 GBK 编码扩充为 GB18030 编码。
- 每个国家都像中国一样，把自己的语言编码，于是出现了各种各样的编码，如果你不安装相应的编码，就无法解释相应编码想表达的内容。
- 终于，有个叫 ISO 的组织看不下去了。他们一起创造了一种编码 UNICODE ，这种编码非常大，大到可以容纳世界上任何一个文字和标志。所以只要电脑上有 UNICODE 这种编码系统，无论是全球哪种文字，只需要保存文件的时候，保存成 UNICODE 编码就可以被其他电脑正常解释。
- UNICODE 在网络传输中，出现了两个标准 UTF-8 和 UTF-16，分别每次传输 8个位和 16个位。于是就会有人产生疑问，UTF-8 既然能保存那么多文字、符号，为什么国内还有这么多使用 GBK 等编码的人？因为 UTF-8 等编码体积比较大，占电脑空间比较多，如果面向的使用人群绝大部分都是中国人，用 GBK 等编码也可以。

来源于网络，出处实在无法考证，无法署名，有删减修改，如有侵权请直接联系。

可能的原文：unicode,ansi,utf-8,unicode big endian编码的区别，网页编码就是那点事

编辑于 2018-01-04