开源中国
首页 开源项目 问答 动弹 博客 翻译 资讯 专题 城市 我的空间 添加
推荐加入 上海城市圈 x
阿里云 云服务器就选阿里云 数据可靠性99.999% 230万用户信赖
微信分享

返回博客列表

原 荐 字符集与编码（四）——Unicode

国栋

发布时间: 2014/09/05 00:52 阅读: 7572 收藏: 328 点赞: 58 评论: 20

摘要

本文系统介绍了Unicode方面的一些重要知识，如码点，平面，代理区，代理对以及UTF，用具体的例子讲解了码点到UTF-8及UTF-16的转换原理与过程。文中还顺便鸟瞰了一下BMP字符集，以此获取更加直观的印象。

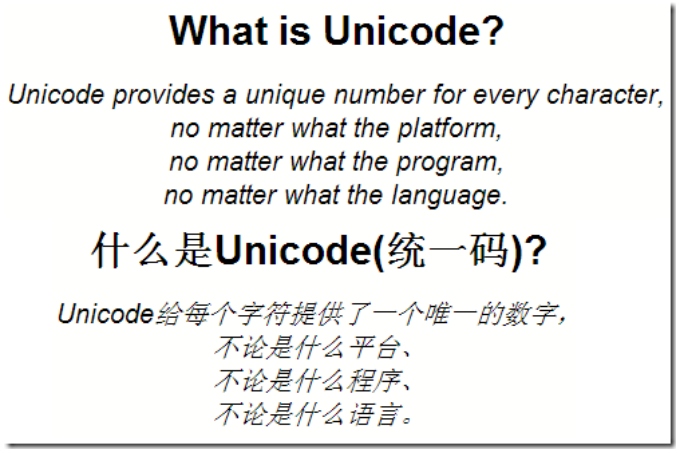
前面谈到不少的Unicode，但一直没有系统地谈及Unicode的方方面面，所以本篇文章专门谈谈Unicode，当然了，Unicode是一个庞大的主题，这里也是拣些重要的方面谈谈而已，免不了挂一漏万。

什么是Unicode？

按Unicode官方的说法，Unicode是Unicode Standard（Unicode标准）的简写，所以Unicode即是指Unicode标准。

按wiki的说法，它是一个计算机工业标准（a computing industry standard）。

下图来自http://www.unicode.org/standard/WhatIsUnicode.html中的截图，在这里我把中文和英文的合在一起



这样一个所谓的一个唯一的数字在Unicode中就叫做**码点**。

Unicode中的码点是什么？

字符集通常又叫"编码字符集"（coded charset），这里的coded与"字符集编码"（charset encoding）中的encoding是不同的。

> “
> 一个是code，一个是encode，翻译时都可以译成"编码"，但把coded charset译成"编号字符集"也许更不易引发误解。
> ”

码点（Code Point）即是这里的code，表示的是一种抽象的数字编号。UTF-X则是最终的encoding。

> “
> 这点如不明白，仍请参见字符集与编码（二）--编号 vs 编码。
> ”

码点的表示形式与范围是？

U+[XX]XXXX是码点的表示形式，X代表一个十六制数字，可以有4-6位，不足4位前补0补足4位，超过则按是几位就是几位。以下是码点的一些具体示例：U+0048，U+4F60，U+1D11E。最后一个是5位的码点。

> “
>
> 有人可能以为码点只有4位，并常常将它与UTF-16的编码搞混，这些都是对码点的误解。
>
> ”

它的范围目前是U+0000~U+10FFFF，理论大小为10FFFF+1=110000_{16}。后一个1代表是65536，因为是16进制，所以前一个1是后一个1的16倍，所以总共有1×16+1=17个的65536的大小，粗略估算为17×6万=102万，所以这是一个百万级别的数。

> “
>
> 准确的值是1114112，一般记为111万左右即可。
>
> ”

110000写成二进制是10001000000000000000，是一个21位的二进制数，我们知道2^{10}=K，2^{20}=K×K=M，即百万级别，所以2^{21}理论上限是两百万左右。10001000000000000000大小基本上由第一个1决定，所以也就一百万左右，从这里也可印证前面的估算。

> “
>
> 按照Unicode官方的说法，码点范围就这些了，以后也不会再扩充了。
>
> ”

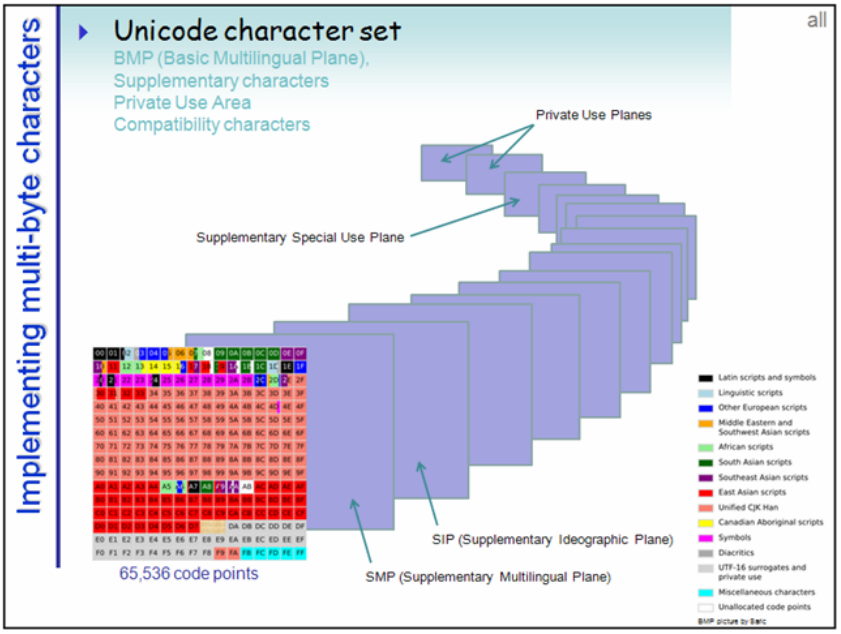
为了更好分类管理如此庞大的码点数，把每65536个码点作为一个**平面**，总共17个平面。

平面，BMP，SP

什么是平面？

由前面可知，码点的全部范围可以均分成17个65536大小的部分，这里面的每一个部分就是一个**平面（Plane）**。编号从0开始，第一个平面称为Plane 0.

下图来自http://rishida.net/docs/unicode-tutorial/part2



什么是BMP？

第一个平面即是**BMP（Basic Multilingual Plane 基本多语言平面）**，也叫Plane 0，它的码点范围是U+0000~U+FFFF。这也是我们最常用的平面，日常用到的字符绝大多数都落在这个平面内。

> “
>
> 上图中第一个花花绿绿的平面就是BMP。
>
> ”

UTF-16只需要用两字节编码此平面内的字符。

> “
>
> 很多人错误地把UTF-16当成定长两字节看待，但只要处理的字符都在这一平面内，一般也不会遇到什么问题。
>
> ”

什么是增补平面？

后续的16个平面称为**SP（Supplementary Planes）**。显然，这些码点已经是超过U+FFFF的了，所以已经超过了16位空间的理论上限，对于这些平面内的字符，UTF-16采用了四字节编码。

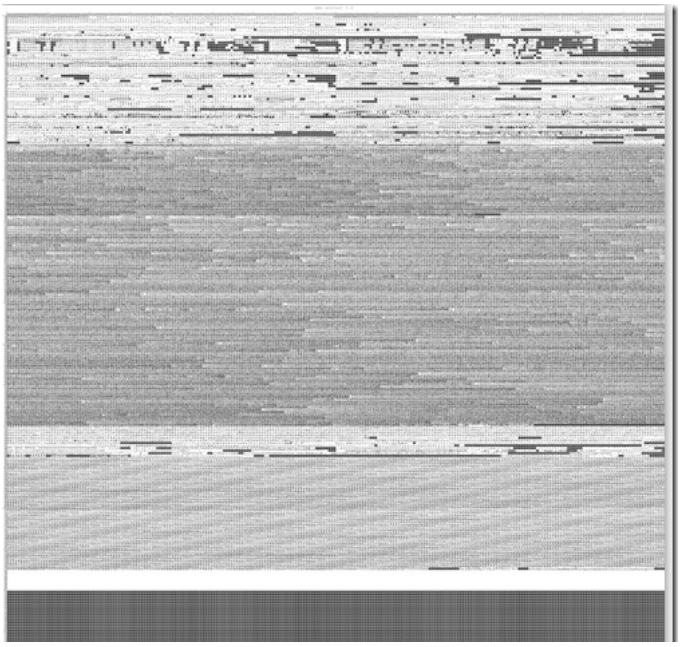
> “
>
> 注：其中很多平面还是空的，还没有分配任何字符，只是先规划了这么多。
>
> 另：有些还属于私有的，如上图中的最后两个Private Use Planes，在此可自定义字符。
>
> ”

鸟瞰BMP字符集

Unicode的字符如此之多，即使是最常用的BMP，它的码点空间也有6万多，如果把这些字符都放到一张图片上，会是什么情况呢？GNU Unifont就制作了一张这样的图片。见http://unifoundry.com/pub/unifont-7.0.03/unifont-7.0.03.bmp

> “
>
> 提示：打开它需要一点时间，它的像素是4000×4000这个级别！
>
> ”

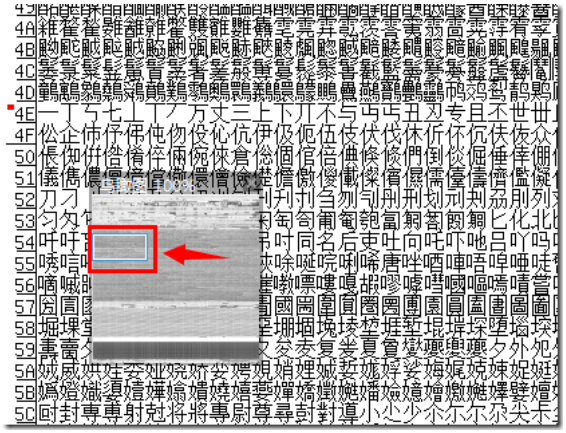
下图是它的一个缩略版本。



这是一个256×256=65536的表格，横向纵向都是从00~FF。

CJK统一汉字

你可能已经注意到上图中间一大片的区域，没错，它就是我们的汉字，在Unicode中，称为CJK统一汉字（CJK：Chinese, Japanese, and Korean，中日韩）。我们可以局部放大看一下。

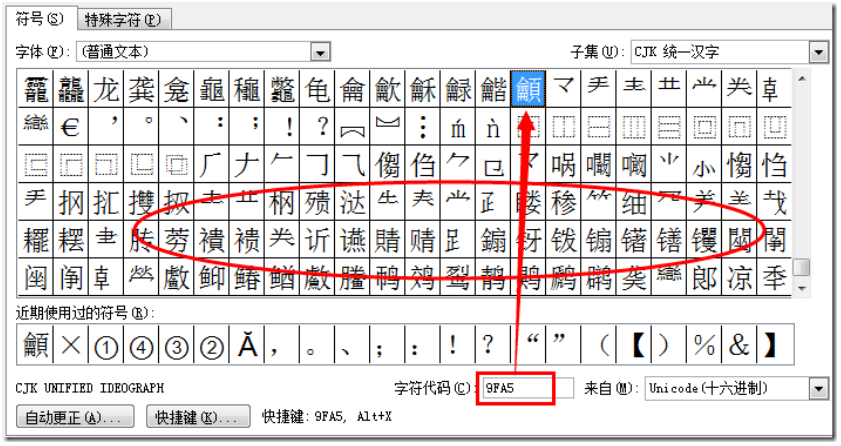


正则表达式[\u4E00-\u9FA5]来匹配中文的问题在哪？

你可能在不少地方见过这种写法，严格来说这只是Unicode最主要的一段中文区域。

> “
>
> 你只要稍加计算就可知这一段大小不过是两万多一点，\u4E00-\u9FA5（19968-40869），中文怎么可能只有这两万多字呢？
>
> ”

这里的“天字第一号”字4E00是哪个字呢？请看上面的图，它就是“一“字，我们还可以看到它上面还有不少的汉字，这就是后来增补的汉字了。所以严格来说，这个上限是不准确的。那么它的下限又是否准确呢？下面是Word的一个插入符号功能的一个截图



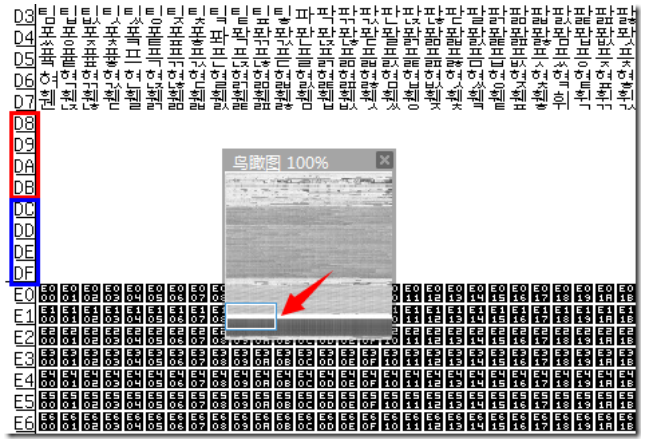
可以看到9FA5后面也还有不少的汉字，它们中间又还夹杂着一些符号，所以想正确地表示Unicode中的汉字还是个不小的挑战。

> “
>
> 应该说，Unicode处在不断发展中，它有一百多万的空间，目前也只是定义了十万左右的字符，还会不断增加，汉字自然也有可能增加，所以汉字的范围实际上是动态的，变化的。当然了，常用的基本落在了这一范围内，而事实上已经包含了许多的不常用汉字，毕竟连只有6千多字的GB2312中都含有大量的不常用汉字。在要求不那么严格的应用中，按以上范围去判断基本也OK，而“汉字”这一概念实际上也没有准确定义，比方说上图中一些“偏旁部首”，这些是“汉字”吗？

”

代理区

你可能还注意到前面的BMP缩略图中有一片空白，这白花花一片亮瞎了我们的猿眼的是啥呢？正如标题已经告诉你的，这就是所谓的**代理区（Surrogate Area）**了。



可以看到这段空白从D8~DF。其中前面的红色部分D800–DBFF属于高代理区（High Surrogate Area），后面的蓝色部分DC00–DFFF属于低代理区（Low Surrogate Area），各自的大小均为4×256=1024。

> 关于代理区的相关用途，我们在讲到UTF-16编码时再说。

还可以看到在它之前是韩文的区域，之后E0开始到F8的则是属于私有的（private），可以在这里定义自己专用的字符。

至此我们对Unicode的码点，平面都有了一定的了解，但我们还没有触及一个重要的方面，那就是码点到最终编码的转换，在Unicode中，这称为UTF。

什么是UTF？

UTF即是**Unicode转换格式**（Unicode (or UCS) Transformation Format）

> 关于UCS：Universal Character Set（统一字符集），也称ISO/IEC 10646标准，不那么严格的情况下，可以认为它和”Unicode字符集“这一概念是等价的。如有兴趣的可以自行搜索了解。

码点如何转换成UTF的几种形式呢？我想这是大家很关心的问题，再发一次前面的一个图

	H	你	𝄞	字符层面
码点(Code Point) 括号内为十进制	U+0048(72)	U+4F60 (20320)	U+1D11E (119070)	抽象编码层面（编号）
UTF	UTF-8	UTF-16	UTF-32	
H U+0048	48	00 48	00 00 00 48	具体编码层面
你 U+4F60	E4 BD A0	4F 60	00 00 4F 60	
𝄞 U+1D11E	F0 9D 84 9E	D8 34 DD 1E	00 01 D1 1E	
	变长 1-4 字节	变长 2 或 4 字节	定长 4 字节	

让我们先从最简单的UTF-32说起

UTF-32

我们说码点最大的10FFFF也就21位，而UTF-32采用的定长四字节则是32位，所以它表示所有的码点不但毫无压力，反而绰绰有余，所以只要把码点的表示形式以前补0的形式补够32位即可。这种表示的最大缺点是占用空间太大。

再来看稍复杂一点的UTF-8。

UTF-8

UTF-8是变长的编码方案，可以有1，2，3，4四种字节组合。在前面的定长与变长篇章我们提到UTF-8采用了高位保留方式来区别不同变长，如下：

```
0XXXXXXX
110XXXXX10XXXXXX
1110XXXX10XXXXXX10XXXXXX
11110XXX10XXXXXX10XXXXXX10XXXXXX
```

> “
>
> 如上，彩色的表示是保留的固定位，X表示是有效编码位。
>
> 单字节最高位都是0，多字节的最高位都是1.
>
> 多字节方面，更具体的讲，N字节模式，首字节以“N个1再加0”打头，后跟“N-1”个以“10”打头的字节。
>
> ”

码点与字节如何对应？

哪些码点用哪种变长呢？可以先把码点变成二进制，看它有多少有效位（去掉前导0）就可以确定了。

1. 一字节有效编码位有7位，2^7=128，码点U+0000~U+007F（0~127）使用一字节。

> “
>
> 一字节留给了ASCII，所以UTF-8兼容ASCII。
>
> ”

2. 二字节有效编码位只有5+6=11位，最多只有2^{11}=2048个编码空间，所以数量众多的汉字是无法容身于此的了。码点U+0080~U+07FF（128~2047）使用二字节。

> “
>
> 注意：这里码点从128~2047，因为去掉了一字节的码点，所以不会占满2048个编码空间，是有冗余的，但你不能把适用于一字节的码点放到这里来编码。下同。
>
> ”

3. 三字节模式可看到光是保留位就达到4+2+2=8位，相当一字节，所以只剩下两字节16位有效编码位，它的容量实际也只有65536。码点U+0800~U+FFFF（2048~65535）使用三字节编码。

> “
>
> 我们前面说到，一些汉字字典收录的汉字达到了惊人的10万级别。基本上，常用的汉字都落在了这三字节的空间里，这就是我们常说的汉字在UTF-8里用三字节表示。当然了，这么说并不严谨，如果这10万的汉字都被收录进来的话，那些偏门的汉字自然只能被挤到四字节空间上去了。
>
> ”

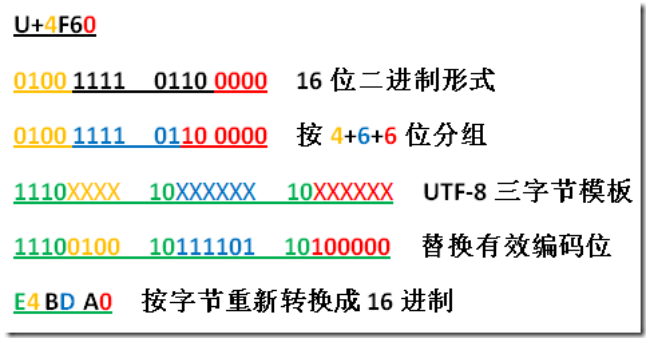
4. 四字节的可以看到它的有效位是3+6+6+6=21位，前面说到最大的码点10FFFF也是21位，U+FFFF以上的增

补平面的字符都在这里来表示。

> “
>
> 按照UTF-8的模式，它还可以扩展到5字节，乃至6字节变长，但Unicode说了码点就到10FFFF，不扩充了，所以UTF-8最多到四字节就足够了。
>
> ”

码点到UTF-8如何转换？

那么具体是如何转换呢，其实不难，来看一个汉字”你“（U+4F60）的转换示意，如下图所示：



上图显示了一有效位为15位的码点到三字节转换的一个基本原理，我们还可看到原来4F60中的一头一尾的两个4和0在转换后还存在于最终的三字节结果中。UTF-8三字节模式固定了1110的开头模式，所以多数汉字总是以1110开头，换成16进制形式，1110就是字母E。

> “
>
> 如果看到一串的16进制有如下的形式：EX XX XX EX XX XX...每三个三个字节前面都是E打头，那么它很可能就是一串汉字的UTF-8编码了。
>
> ”

其它变长字节转换道理也类似，其中分组从低位开始，高位如不足则补零。这里就不再示例了。

最后来看最复杂的UTF-16，在此之前我们先要理解代理区与代理对等概念。

UTF-16

UTF-16是一种变长的2或4字节编码模式。对于BMP内的字符使用2字节编码，其它的则使用4字节组成所谓的代理对来编码。

什么是代理区？

在前面的鸟瞰图中，我们看到了一片空白的区域，这就是所谓的**代理区（Surrogate Area）**了，代理区是UTF-16为了编码增补平面中的字符而保留的，总共有2048个位置，均分为**高代理区**（D800–DBFF）和**低代理区**（DC00–DFFF）两部分，各1024，这两个区组成一个二维的表格，共有

$1024\times1024=2^{10}\times2^{10}=2^4\times2^{16}=16\times65536$，所以它恰好可以表示增补的16个平面中的所有字符。

> “
>
> 当然了，说恰好是不对的，显然代理区就是冲着表示增补平面来设计的，或者至少它们是一起考虑的。
>
> ”

下面的图片来自wiki

UTF-16 decoder

Lead \ Trail	DC00	DC01	···	DFFF
D800	010000	010001	···	0103FF
D801	010400	010401	···	[illegible]

D801	010400	010401	…	0 F
⋮	⋮	⋮	⋱	⋮
DBFF	10FC00	10FC01	…	10FFFF

什么是代理对？

一个高代理区（即上图中的Lead（头），行）的加一个低代理区（即上图中的Trail（尾），列）的编码组成一对即是一个**代理对（Surrogate Pair）**，必须是这种先高后低的顺序，如果出现两个高，两个低，或者先低后高，都是非法的。

> “
>
> 在图中可以看到一些转换的例子，如
>
> （D8 00 DC 00）—>U+10000，左上角，第一个增补字符
>
> （DB FF DF FF）—>U+10FFFF，右下角，最后一个增补字符
>
> ”

码点到UTF-16如何转换？

分成两部分：

1. BMP中直接对应，无须做任何转换；

2. 增补平面SP中，则需要做相应的计算。其实由上图中的表也可看出，码点就是从上到下，从左到右排列过去的，所以只需做个简单的除法，拿到除数和余数即可确定行与列。

拿到一个码点，先减去10000_{16}，再除以400_{16}（$=1024_{10}$）就是所在行了，余数就是所在列了，再加上行与列所在的起始值，就得到了代理对了。

> “
>
> Lead = (码点 - 10000_{16}) ÷ 400_{16} + D800
>
> Trail = (码点 - 10000_{16}) % 400_{16} + DC00
>
> ”

下面以前面的U+1D11E具体示例了代理对的计算：

> “
>
> Lead = (1D11E - 10000_{16}) ÷ 400_{16} + DB00 = D11E ÷ 400_{16} + D800 = 34 + D800 = D834
>
> Trail = (1D11E - 10000_{16}) % 400_{16} + DC00 = D11E % 400_{16} + DC00 = 11E + DC00 = DD1E
>
> ”

所以，码点U+1D11E对应的代理对即是 D834 DD1E。

> “
>
> 注意：以上计算方式仅用于说明转换原理，不代表实际采用的计算方式。一个码点减去10000_{16}后实际最多只有20位，再除以400_{16}（$=2^{10}=10000000000_2$），这个除数实际是一个二进制整数，相当于十进制中整十整百的数。所以结果实际上低10位上的就是余数，而高10位（或者不到10位）上就是商，可以通过更为快速的移位操作实现。举个十进制的例子，就好比是“1234÷100=12……34”，你都不需要拿笔去算。应该说，代理区的设计是有效率上的考虑的，如果我们要做转换，应该考虑是否有系统API可供调用，而不要自行去实现。
>
> ”

关于Unicode的基本知识，就讲到这里。

© 著作权归作者所有

分类：字符集编码系列 字数：3879 标签： unicode UTF 码点 平面 代理区

点赞 (58) 收藏 (328) 分享

国栋 关注此人

粉丝: 242 博客数: 44 共码了 105576 字

评论（20）

东厢里的一只喵

1楼 2014/09/05 08:42

那个，在那可以下载到u的全符字符字体呢？
u码中有没有包括蒙，藏文等等，为什么韩文还有一大堆，是不是同汉字一样还有扩展呢？

Howard.L.Huang

2楼 2014/09/05 09:03

good job

Kevin19701

3楼 2014/09/05 09:07

涨姿势

walkskyer

4楼 2014/09/05 09:14

mark

sevk

5楼 2014/09/05 09:16

果然牛

Slayer

6楼 2014/09/05 10:21

很好 楼主辛苦了。。

哪一天

7楼 2014/09/05 11:33

very good

jetmeng

8楼 2014/09/05 12:54

多谢楼主，这回彻底搞清楚了！

rainmanqqst

9楼 2014/09/05 14:30

太牛了

naughty

10楼 2014/09/05 15:48

很赞

tinyhare

11楼 2014/09/05 18:19

snowdream

12楼 2014/09/06 07:26

这篇文章，太赞了

小Q先生

13楼 2014/09/06 09:34

支持

zhuqm

14楼 2014/09/06 15:43

果然牛

国栋

15楼 2014/09/07 20:11

> 引用来自"东厢里的一只喵"的评论
> 那个，在那可以下载到u的全符字符字体呢？
> u码中有没有包括蒙，藏文等等，为什么韩文还有一大堆，是不是同汉字一样还有扩展呢？

全部字符字体这个我也不清楚哪里有，微软中有Arial Unicode MS，BMP内的应该都包括了。之外的在文中提到那个缩略图的网站中有，http://www.unifoundry.com/unifont.html，下载那个Unifont Upper，但这也仅仅是增补平面的一部分而已，而且制作得比较粗糙。蒙藏应该都有包括，不过具体是在BMP中还是在之外我也不清楚，你可以下载那个缩略图看看呗，我也不清楚蒙藏文长啥样。韩文有一大堆有啥好惊讶的呢？Unicode就是要包含所有的字符，至于它有无扩展这点我也不清楚。

潘孙友

16楼 2014/09/10 08:16

写得真心好赞！

Ouyang_Yifan

17楼 2014/09/17 09:16

内容详实，文品风趣有味，真是用心人。

过马路的蚂蚁

18楼 2014/12/03 15:58

非常感谢楼主，一直有个问题困扰我。
理论上utf8 表达中文的总量应该最多，但在实践中有些偏僻字、少数民族的字，utf8不能完全表达。
gbk可以正常显示。
而且gbk有明确的汉字个数，utf8没有明确汉字的数量？为什么

国栋

19楼 2014/12/05 17:22

> 引用来自"过马路的蚂蚁"的评论
> 非常感谢楼主，一直有个问题困扰我。
> 理论上utf8 表达中文的总量应该最多，但在实践中有些偏僻字、少数民族的字，utf8不能完全表达。
> gbk可以正常显示。
> 而且gbk有明确的汉字个数，utf8没有明确汉字的数量？为什么

utf8 表达中文的总量应该最多？我没听过这种说法。我也不太清楚你说的"utf8不能完全表达"是什么意思。你是可以自己定义一些字符的，所谓的"造字"，另外你确定是否用的是GB18030编码呢？这一编码应该能支持很多少数民族的文字。随着整理工作的深入，unicode中收录的汉字会增加，这是有可能的，但就某个特定时期而言，里面包含的汉字数量肯定是确定的，我不清楚你说的utf8没有明确汉字的数量是什么意思。

talan1314

20楼 2016/03/25 00:04

这篇还要再看 有的地方还是不是特别明白 楼主牛逼

插入： 表情 开源软件

发表评论

© 开源中国(OSChina.NET) | 关于我们 | 广告联系 | @新浪微博 | 开源中国手机版 | 粤ICP备12009483号-3
开源中国社区(OSChina.net)是工信部 开源软件推进联盟 指定的官方社区
开源中国手机客