转 中文字符集编码Unicode ,gb2312 , cp936 ,GBK，GB18030

2017年11月07日 14:09:14 xbsoul 阅读数：291

中文字符集编码Unicode ,gb2312 , cp936 ,GBK，GB18030

转自：http://hi.baidu.com/okptqdwpfrbosuq/item/0fc063f8b65f0516d6ff8c03
中文字符集编码Unicode ,gb2312 , cp936 ,GBK，GB18030
转自： http://www.blog.edu.cn/user3/flyingcs/archives/2006/1418577.shtml 概要：UTF-8的一个特别的好处是它与ISO- 8859-1完全兼容，可以表示世界上所有的字符，汉字通常用３个字节来表示。GB2312的code page是CP20936。GBK的code page是CP936 。GB18030支持的字符数更多。GB2312、GBK、GB18030均为双字节。
这是一篇程序员写给程序员的趣味读物。所谓趣味是指可以比较轻松地了解一些原来不清
楚的概念，增进知识，类似于打RPG游戏的升级。整理这篇文章的动机是两个问题：
问题一： 使用Windows记事本的“另存为”，可以在GBK、Unicode、Unicode big endian和UTF-8这 几种编码方式间相互转换。同样是txt文件，Windows是怎样识别编码方式的呢？
我很早前就发现Unicode、Unicode big endian和UTF-8编码的txt文件的开头会多出几个字 节，分别是FF、FE（Unicode）,FE、FF（Unicode big endian）,EF、BB、BF（UTF-8）。 但这些标记是基于什么标准呢？
问题二： 最近在网上看到一个ConvertUTF.c，实现了UTF-32、UTF-16和UTF-8这三种编码方式的相互 转换。对于Unicode(UCS2)、GBK、UTF-8这些编码方式，我原来就了解。但这个程序让我有 些糊涂，想不起来UTF-16和UCS2有什么关系。
查了查相关资料，总算将这些问题弄清楚了，顺带也了解了一些Unicode的细节。作者写成 一篇文章，送给有过类似疑问的朋友。本文在写作时尽量做到通俗易懂，但要求读者知道 什么是字节，什么是十六进制。
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。如果将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，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。

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GB18 030相对GBK增加的字符，普通人是很难用到的，通常[illegible]indows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GB K和GB18030的低字节最高位都可能不是1。不过这不影响DBCS字[illegible]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ASCI I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ISO 10646-3:2003。
UCS只是规定如何编码，并没有规定如何传输、保存这个编码。例如“汉”字的UCS编码是 6C49，我可以用4个ascii数字来传输、保存这个编码；也可以用utf-8编码:3个连续的字节 E6 B1 89来表示它。关键在于通信双方都要认可。UTF-8、UTF-7、UTF-16都是被广泛接受 的方案。UTF-8的一个特别的好处是它与ISO- 8859-1完全兼容。UTF是“UCS Transformat ion Format”的缩写。
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。
2.1、内码和code page
目前Windows的内核已经采用Unicode编码，这样在内核上可以支持全世界所有的语言文字 。但是由于现有的大量程序和文档都采用了某种特定语言的编码，例如GBK，Windows不可 能不支持现有的编码，而全部改用Unicode。
Windows使用代码页(code page)来适应各个国家和地区。code page可以被理解为前面提到 的内码。GBK对应的code page是CP936。
微软也为GB18030定义了code page：CP54936。但是由于GB18030有一部分4字节编码，而W indows的代码页只支持单字节和双字节编码，所以这个code page是无法真正使用的。
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cell[illegible]是最后一个字节不同，其余都相同。
group 0的plane 0被称作[illegible]，即BMP。或者说[illegible]的码位被称作BMP[illegible]
对UCS-4的BMP空间[illegible]UCS-2，在UCS-2[illegible]字节，就得到了UCS-4的BMP。而目前的UCS-4规范中是没有给字符被分配在BMP之外。

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 110xxx xx 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 ，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。需要注意，UltraEdit在打开utf-8编码 的文本文件时会自动转换为UTF-16，可能产生混淆。你可以在设置中关掉 项。更好 的工具是Hex Workshop。

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 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 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“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-BR EAK 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。

6、进一步的参考资料

本文主要参考的资料是 "Short overview of ISO-IEC 10646 and Unicode" (http://ww w.nada.kth.se/i18n/ucs/unicode-iso10646-oview.html)。

我还找了两篇看上去不错的资料，不过因为我开始的疑问都找到了答案，所以就没有看：

"Understanding Unicode A general introduction to the Unicode Standard" (http: //scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id=IWS-Chapter04a) " Character set encoding basics Understanding character set encodings and legacy encodings" (http://scripts.sil.org/cms/scripts/page.php?site_id=nrsi&item_id =IWS-Chapter03) 我写过UTF-8、UCS-2、GBK相互转换的软件包，包括使用Windows API和 不使用Windows API的版本。以后有时间的话，我会整理一下放到我的个人主页上(http: //fmddlmyy.home4u.china.com)。

附录1 再说说区位码、GB2312、内码和代码页

有的朋友对文章中这句话还有疑问： “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。”

我再详细解释一下：

“GB2312的原文”是指国家1980年的一个标准《中华人民共和国国家标准 信息交换用汉字 编码字符集 基本集 GB 2312-80》。这个标准用两个数来编码汉字和中文符号。第一个数 称为“区”，第二个数称为“位”。所以也称为区位码。1-9区是中文符号，16-55 区是一 级汉字，56-87区是二级汉字。现在Windows也还有区位输入法，例如输入1601得到“啊” 。

内码是指操作系统内部的字符编码。早期操作系统的内码是与语言相关的.现在的Windows 在内部统一使用Unicode，然后用代码页适应各种语言,“内码”的概念就比较模糊了。微 软一般将能 代码页指定的编码说成是内码，在特殊的场合也会说自己的内码是Unicode， 例如在 GB18030问题的处理上。

所谓代码页(code page)就是针对一种语言文字的字符编码。例如GBK的code page是CP936 ， BIG5的code page是CP950，GB2312的code page是CP20936。

也可能找到错误的字符。所谓“错误”是指与文本作者的本意不符，这时就产生了 乱码。
答案是Windows按照当前的缺省代码页去解释文本文件里的字节流。缺省代码页可以通过控 制面板的区域选项设置。记事本的另存为中有一项ANSI，[illegible]码 方法保存。
Windows的内码是Unicode，它在技术上可以同时支持多个代码页。只要文件能说明自己使 用什么编码，用户又安装了对应的代码页，Windows就能正确显示，[illegible]HTML文件中就可 以指定charset。
有的HTML文件作者，特别是英文作者，认为世界上所有人都使用英文，在文件中不指定ch arset。如果他使用了0x80-0xff之间的字符，中文Windows又按照缺省[illegible]去解释，就会 出现乱码。这时只要在这个html文件中加上指定charset的语句，例如：如果原作者使用的 代码页和ISO8859-1兼容，就不会出现乱码了。
再说区位码，啊的区位码是1601，写成16进制是0x10,0x01。这和计算机广泛使用的ASCII 编码冲突。为了兼容00-7f的 ASCII编码，我们在区位码的高、低字节[illegible]加上A0。这样 “啊”的编码就成为B0A1。我们将加过两个A0的编码也称为GB2312编码，虽然 GB2312的原 文根本没提到这一点。