文章
Qiao Peng · 一月 17, 2022 阅读大约需 8 分钟

精华文章--多语言字符集系列文章--第一篇 多语言字符集和相关标准简史

各大技术社区常年充斥着关于字符集支持、乱码的问题。Cache’/Ensemble/HealthConnect/IRIS的用户也经常遇到这类问题。为何文字乱码在信息化发展这么久后还会困扰我们？因为字符集、多语言实在有点复杂。

我计划写三篇：第一篇花点时间回顾一下多语言字符集的简史，第二篇介绍一下各种技术对于字符集和字符编码的使用声明，最后一篇会介绍常见的ISC技术和工具的乱码、尤其是中文乱码的现象和解决办法。

第一篇 多语言字符集和相关标准简史

如果您已经了解多语言字符集和相关标准，请绕道此章。

相关概念

要理解多语言字符集，先了解一下相关概念。

- 字符(char)：每个语言都有一系列特有的字符，例如英文26个字母、加减乘除等各种符号、中文汉字。
- 字形(glyphs)：同样一个字符，有不同的写法、不同的风格和设计，就是不同的字形。
- 字体(font)：字体算是计算机术语，是针对字符集的电子化的字符展现形式。
- 字符集(character set)：通常是按语言归集的一个字符集合，用于记录每个字符和对应的代码。
- 字符编码(encoding)：针对字符集的特定编码格式。

字符集发展简史

电子字符集的发展历史很长，再加上不同语言字符集的复杂性，所以现在我们面对的字符集和字符编码是比较复杂的，而且相同字符集还有很多别名，在不同环境下看到的别名还不一样。下面仅介绍我们常见的字符集的发展历史，字符集全景远比这复杂。

1.1 单字节编码

1.1.1 ASCII(American Standard Code for Information Interchange)：

它最初是Bell在1960年代基于电报码提出的，用于电传打印机。后成为ANSI标准（ANSIX3.4-1968）。它使用7位2进制编码，表达128个字符：英文字母、阿拉伯数字和符号。这些符号包括可显示的符号，如加减乘除，也包含32个不能显示的控制符，如换行符（ASCII码为10进制数13）。

因为包含不能显示的符号，严格意义上，它不算字符集，而是一种字符编码。

128个字符码位，对于美国够用了，但随着计算机技术应用的扩展，其他拉丁语系国家的字符也需要表达，例如西欧语言里的变音字符，显然7位的ASCII不够用了。所以很多国家也基于7位编码，类似于ASCII，搞出了自己的字符集。这些字符集里，用ASCII的码位表达不同的字符。所以它们并不能与ASCII兼容。

1.1.2 ISO/IEC 8859、ANSI:

既然7位编码（128个）不够表达足够多的字符，最简单的办法是增加1位，就是8位2进制编码，这样就可以表达256个字符。而8位2进制数正好就是一个字节，用前面的128个编码位兼容ASCII，用后面增加的128个编码位表达其它拉丁字符，扩展基本没有难度。

最初ANSI发布了基于8位的标准，它就是这么做的，从而对各种欧洲国家的语言提供支持。所以8位的ASCII扩展字符集也叫做ANSI。后来ISO采用并扩展了该标准，就是ISO/IEC 8859系列字符集，例如ISO 8859-1 (Latin-1, 西欧语言) 、ISO 8859-2 (Latin-2, 中东欧语言)。这些字符集的低128个字符与ASCII一致，而高128个字符按不同的语言来安排不同的字符。所以，这些字符集并不互相兼容，差异就在这高128个字符。

注：要区别这些高128个字符到底是什么，微软公司使用代码页（code page）来确定当前使用什么字符集。Windows中可以用chcp来查看当前加载的代码页。另外，微软还将那些看不见的控制字符，替换成了可以显示的符号，例如笑脸符号。代码页应用到了今天，甚至包括了多字节编码，例如中文GBK的代码页是936。

1.2 双字节编码

1.2.1 GB2312、GBK、BIG5：

8位的编码基本可以解决拉丁语系的字符编码问题。可是以中日韩为代表的东亚文字字符数量远远超过了256个编码位。东亚国家开始考虑创建自己的数据集，即然一个8位的字节不足以编码这么多文字，最初大家就使用双字节来表达，从而创造出了一系列双字节字符集（Double Byte Character Set/DBCS），例如中文的GB2312、BIG5和日本的Shift JIS。

中国在1980年发布了GB2312这个中文字符集，思路与ANSI类似，但它使用2个8位字节编码，包含6763个汉字，同时它兼容ASCII。之后不久中国台湾地区发布了繁体字的BIG5（大五码）。

GB2312 里的汉字覆盖面不够，尤其是用于人名、地名的字符。后续又出了一些列补充标准，如GB/T 7589 - 87、GB/T 12345 – 90，以及兼容ISO/IEC 10646-2003的GB13000。综合这些扩展，1995年发布了《汉字内码扩展规范》，也就是GBK，K就是扩展的意思。GBK包含 21003个汉字，大大提高了对生僻字的适用性。

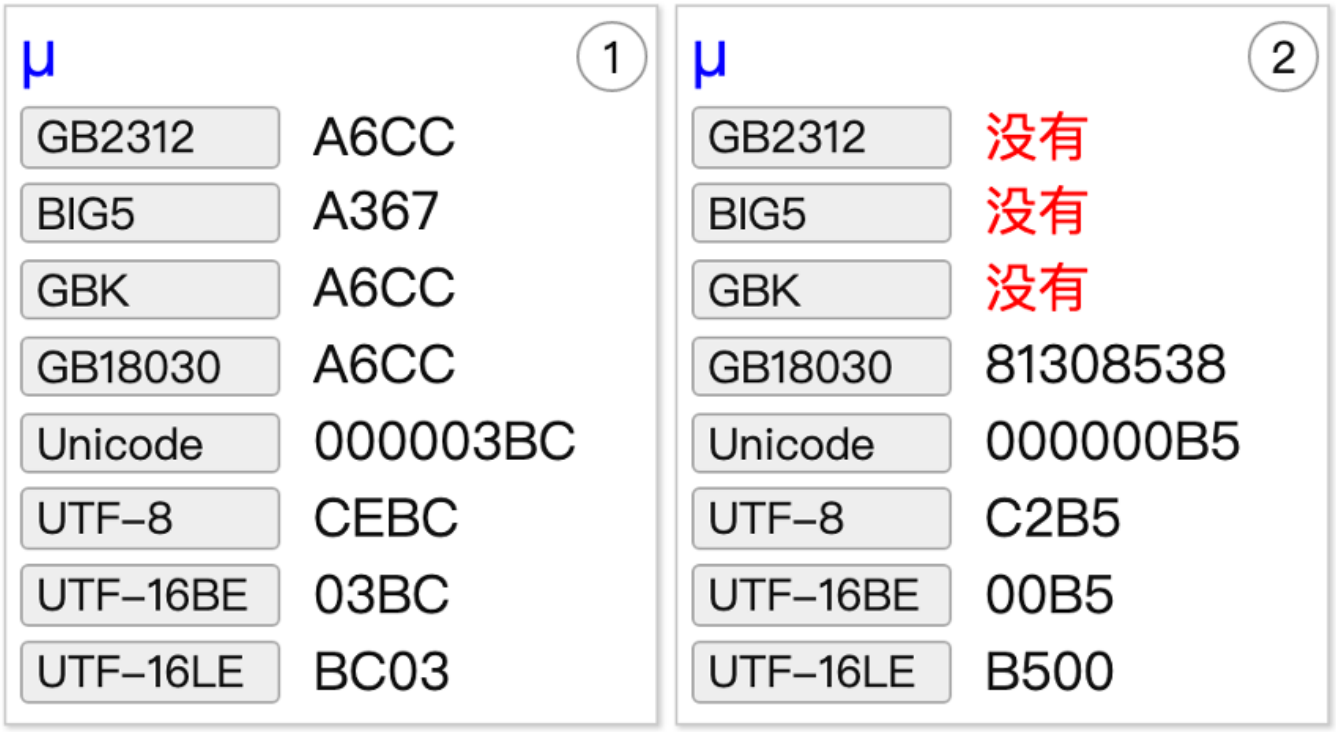
1.3 多字节编码

在单字节、双字节编码的字符集时代，没有任何一个字符集能涵盖所有语言，这对于想在全球拓展业务的IT厂商，例如操作系统厂商、打印机厂商，是一个挑战。因此施乐、苹果等厂商于1988年组成统一码联盟，开发Unicode字符集。同时，ISO也开始开发ISO/IEC 10646 Universal Character Set (UCS)，想统一字符集。很快双方就注意到对方在做相同的事情，并决定让正在开发的两个字符集标准兼容。ISO/IEC 10646定义了128组*256个平面*256行*256单元的编码空间，其中00组、00平面被称之为基本多语言平面（Basic Multilingual Plane，BMP），01到0F这15个平面称之为辅助多语言平面。理论可以编码20亿个字符，但由于有保留使用的编码范围，它实际可以编码679,477,248个字符。

1.3.1 Unicode（ISO/IEC 10646）：

Unicode早期（1991-1995）采用固定的2字节（16位）设计，理论可以编码65536个字符。1996年的Unicode 2.0开始，突破了这个限制，目前用到了21位（3个字节）的编码空间，未来计划扩展到31位（4个字节）。不过计算机表达时，统一用4个字节的表达形式：U+[4字节数]，例如“ 中 ”字的Unicode码为：U+4E2D。它的目标是包含所有国家的文字，由于一些码位作为特殊用途保留，目前有超过1百万的可编码的码位。Unicode持续进化，截止2021年9月，已经发展到版本14，包含超过14万个字符。

它保留了前256个代码给ISO 8859-1，因此和ISO 8859-1、ASCII相兼容。注意，出于对之前字符集标准兼容的需求，有些相同的字符在Unicode中有多个编码，例如：



Unicode有UTF-8、UTF-16、UTF-32这3种编码方式。UTF是Unicode Transformation Forma

，UTF-32是固定4字节编码并与Unicode码一致。

为何不直接使用Unicode，而是要使用UTF编码方式呢？原因挺多：要兼容之前的应用和代码、要降低编码长度从而节省磁盘、内存…

UTF-8：
在互联网世界最为流行，它以8位2进制作为一个字节，使用1个字节到4个字节来表达特定字符。对于兼容ASCII的字符，UTF-8使用1个字节就够了，而且编码与ASCII完全一致，因此既节省了内存和磁盘的存储空间，又兼容了之前的应用和代码。对于其它字符，UTF-8可以使用更多字节表达，并可以表达所有Unicode字符。

如何保证使用2个字节表达的UTF-8码不会与1个字节表达的UTF-8码产生混淆？UTF-8设计了一套规范，通过每个UTF-8开始字节的规律确保不会发生混淆，例如单字节的UTF-8码的字节第一位是0（2进制），而双字节的UTF-8码的首字节由110开始，三字节的UTF-8码的首字节由1110开始，四节字的UTF-8码的首字节由11110开始。下图是UTF-8的编码空间和编码规律。

代码范围 十六进制	标量值（scalar value） 二进制	UTF-8 二进制 / 十六进制	注释
000000 – 00007F 128个代码	00000000 00000000 0zzzzzzz	0zzzzzzz（00–7F）	ASCII字符范围，字节由零开始
	七个z	七个z	
000080 – 0007FF 1920个代码	00000000 00000yyy yyzzzzzz	110yyyyy（C0–DF） 10zzzzzz（80–BF）	第一个字节由110开始，接着的字节由10开始
	三个y；二个y；六个z	五个y；六个z	
000800 – 00D7FF 00E000 – 00FFFF 61440个代码 [Note 1]	00000000 xxxxyyyy yyzzzzzz	1110xxxx（E0–EF） 10yyyyyy 10zzzzzz	第一个字节由1110开始，接着的字节由10开始
	四个x；四个y；二个y；六个z	四个x；六个y；六个z	
010000 – 10FFFF 1048576个代码	000wwwxx xxxxyyyy yyzzzzzz	11110www（F0–F7） 10xxxxxx 10yyyyyy 10zzzzzz	将由11110开始，接着的字节由10开始
	三个w；二个x；四个x；四个y；二个y；六个z	三个w；六个x；六个y；六个z	

UTF-16: 也是非常流行的Unicode编码方案，例如Windows就是用UTF-16编码，在很多Windows应用中所说的Unicode编码，其实就是UTF-16。它采用1个或2个16位编码单位。UTF-8以8位字节为单位，多字节UTF-8码严格按字节顺序表达即可。但UTF-16编码单位为16位（2个8位字节），那么就会产生这2个字节哪个放在前面的问题：大端序（Big-Endian）还是小端序（Little-Endian）*。不同的操作系统使用不同的顺序，例如Linux和Windows使用小端序，而Unix使用大端序。

既然要区分不同的端序，那就给文件/字符流的头部增加一个标识符吧，这就是BOM（Byte Order Mark），标识大端码时用0xFE 0xFF、标识小端码是用0xFF 0xFE。而FFFE（FEFF）不是任何Unicode字符的编码，从而不会造成误解。

*注：大小端序来自小说《格列佛游记》，故事里2个小人国为从大端敲开鸡蛋还是小端敲开鸡蛋发生战争。

*注：UTF-8并不需要BOM，因为不会产生类似的混淆。虽然不推荐，但的确有为UTF-8的BOM，它是0xEF 0xBB 0xBF，通常被用来标记这是UTF-8的文件。有时它会造成显示问题。

最初UTF-16用1个16位编码，可以编码65536个字符，涵盖常见中文字符没有问题，这个编码方案也被称为UCS-2，它定义的字符范围就是基本多语言平面（Basic Multilingual Plane，BMP）的字符范围。但随着2006年GB18030-2005的强制标准实施，需要包含GB18030里的7万多个汉字，因此UCS-2方案不够用了，需要动用15个辅助多语言平面。

1.3.2 GB18030：

随着ISO开发ISO/IEC 10646，中国也开始了GB13000的开发，它和ISO/IEC 10646一致，将中文字符编码进ISO/IEC 10646编码空间。1993年发布了GB13000-1993《信息技术 通 用多 八位编码 字符集（UCS）第 一部分：体系结构与基本多文种平面 》，不过它和之前已经广泛使用的GB2312编码不兼容，所以并没有大规模采用。

随后，在2000年推出了GB18030，并在2005年修订。它的特点是兼容ASCII和GB2312码、基本兼容GBK、并包含Unicode中的所有中文字符。这是一个强制标准，是目前执行的中文字符集标准。它包含70244个汉字字符，采用单字节、双字节、四字节变长编码。单字节部分就是ASCII码，中文字符在双字节和四字节编码空间。

表 1　码位范围分配图

字节数	码　位　空　间				码位数目
单字节	0x00～0x7F				128 个码位
双字节	第一字节		第二字节		23940 个码位
	0x81～0xFE		0x40～0x7E， 0x80～0xFE		
四字节	第一字节	第二字节	第三字节	第四字节	1587600 个码位
	0x81～0xFE	0x30～0x39	0x81～0xFE	0x30～0x39	

那么GB18030和Unicode到底是什么对应关系？GB18030兼容GB2312的部分，由于GB2312的开发早于Unicode，因此这部分GB18030（GB2312）编码与Unicode码不连续相关，因此是没办法通过算法来做二者编码的转换，只能通过对照表来关联和转换，如下图：

GB18030值：B0A1

Unicode值：554A

双字节的这一区域的GB码与Unicode码连续性不一致！

双字节 2 区

B0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		啊 554A	阿 963F	埃 57C3	挨 6328	哎 54CE	唉 5509	哀 54C0	皑 7691	癌 764C	蔼 853C	矮 77EE	艾 827E	碍 788D	爱 7231	隘 9698
B	鞍 978D	氨 6C28	安 5B89	俺 4FFA	按 6309	暗 6697	岸 5CB8	胺 80FA	案 6848	肮 80AE	昂 6602	盎 76CE	凹 51F9	敖 6556	熬 71AC	翱 7FF1
C	袄 8884	傲 50B2	奥 5965	懊 61CA	澳 6FB3	芭 82AD	捌 634C	扒 6252	叭 53ED	吧 5427	笆 7B06	八 516B	疤 75A4	巴 5DF4	拔 62D4	跋 8DCB
D	靶 9776	把 628A	耙 8019	坝 575D	霸 9738	罢 7F62	爸 7238	白 767D	柏 67CF	百 767E	摆 6446	佰 4F70	败 8D25	拜 62DC	稗 7A17	斑 6591
E	班 73ED	搬 642C	扳 6273	般 822C	颁 9881	板 677F	版 7248	扮 626E	拌 62CC	伴 4F34	瓣 74E3	半 534A	办 529E	绊 7ECA	邦 90A6	帮 5E2E
F	梆 6886	榜 699C	膀 8180	绑 7ED1	棒 68D2	磅 78C5	蚌 868C	镑 9551	傍 508D	谤 8C24	苞 82DE	胞 80DE	包 5305	褒 8912	剥 5265	

而另一部分的GB18030编码是在Unicode标准之后开发的，这部分与Unicode连续相关，也就是通过算法就能做二者间的转码，如下图：

4字节这一区域的GB码与Unicode码连续性一致！

8230

	30	31	32	33	34	35	36	37	38	39
81	俨 34A3	俩 34A4	俪 34A5	俫 34A6	俬 34A7	俭 34A8	修 34A9	俯 34AA	俰 34AB	俱 34AC
82	倪 34AD	倬 34AE	倭 34AF	倮 34B0	倯 34B1	倰 34B2	倱 34B3	倲 34B4	倳 34B5	倴 34B6
83	倵 34B7	倶 34B8	倷 34B9	倸 34BA	倹 34BB	偀 34BC	偁 34BD	偂 34BE	偃 34BF	偄 34C0
84	偅 34C1	冠 34C2	冡 34C3	冢 34C4	冣 34C5	冷 34C6	冸 34C7	冹 34C8	冺 34C9	冻 34CA
85	冼 34CB	冽 34CC	冾 34CD	冿 34CE	凁 34CF	凂 34D0	凃 34D1	凄 34D2	凅 34D3	准 34D4
86	凇 34D5	凈 34D6	凉 34D7	凊 34D8	凋 34D9	凌 34DA	凍 34DB	减 34DC	凑 34DD	凒 34DE

这也是为什么有些文献说GB18030也是一种Unicode编码。

通过上面的字符集简史能看出，对于我们可能遇到各种字符集和不同的编码：ASCII、ANSI、Unicode、UTF-8、UTF-16、GB2312、GB18030…
我们怎么知道在数据交换中对方用的是什么字符集和字符编码？

下一个章节，我们将汇总不同的技术对于字符集/字符编码使用的声明。

#本地化 #语言 #Caché #InterSystems IRIS #InterSystems IRIS for Health #其他

源 URL: https://cn.community.intersystems.com/post/%E7%B2%BE%E5%8D%8E%E6%96%87%E7%AB%A0-%E5%A4%9A%E8%AF%AD%E8%A8%80%E5%AD%97%E7%AC%A6%E9%9B%86%E7%B3%BB%E5%88%97%E6%96%87%E7%AB%A0-%E7%AC%AC%E4%B8%80%E7%AF%87-%E5%A4%9A%E8%AF%AD%E8%A8%80%E5%AD%97%E7%AC%A6%E9%9B%86%E5%92%8C%E7%9B%B8%E5%85%B3%E6%A0%87%E5%87%86%E7%AE%80%E5%8F%B2