[【字符编码】彻底理解字符编码](https://www.cnblogs.com/leesf456/p/5317574.html)

**二、字符编码**

ASCII

这套编码规则是由美国定制，一共规定了128个字符的编码，只占用了一个字节（8 bit）的后面7位，最前面的1位统一规定为0。总共才有128个字符编码，一个字节都没有用完，于是乎，就开始压榨最高位，对其为1时也进行编码，利用最高位进行编码的方式就称为非**ASCII**编码，如**ISO-8859-1**编码。

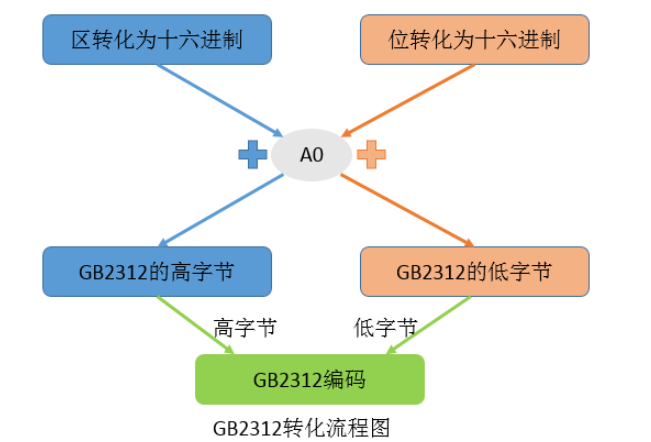
ISO-8859-1

ISO-8859-1编码也是单字节编码，最多能够表示256个字符，ISO-8859-1编码也是单字节编码，最多能够表示256个字符。而且在很多协议上，默认使用该编码。比如，虽然"中文"两个字不存在ISO8859-1编码，以GB2312编码为例，应该是D6D0 CEC4两个字符，使用ISO8859-1编码的时候则将它拆开为4个字节来表示：D6D0 CEC4

GB2312

GB2312其对所收录字符进行了"分区"处理，共94个区，区从1（十进制）开始，一直到94（十进制），每区含有94个位，位从1（十进制）开始，一直到94（十进制），共8836（94 \* 94）个码位，这种表示方式也称为区位码，GB2312是双字节编码，其中高字节表示区，低字节表示位。

1. 01-09区收录除汉字外的682个字符，有164个空位（9 \* 94 - 682）。
2. 10-15区为空白区，没有使用。
3. 16-55区收录3755个一级汉字（简体），按拼音排序。
4. 56-87区收录3008个二级汉字（简体），按部首/笔画排序。
5. 88-94区为空白区，没有使用。



BIG5

BIG5采用双字节编码，使用两个字节来表示一个字符。

GBK

GBK编码扩展了GB2312，完全兼容GB2312编码

在GBK之后又出现了GB18030编码，但是没有形成主流，故不做介绍，至此，中文编码的问题已经讲解完成。那么问题又来了，大陆网民与在海峡两岸网民交流时，若都使用GBK编码，则没有问题，若一方使用GBK编码，一方使用BIG5编码，那么就会出现乱码问题，这是在海峡两岸网民交流，如果漂洋过海进行交流呢？那就更容易出现乱码问题，**这时候我们可能想，要是有一套全世界都通用的编码就好了，不要担心，这样的编码确实是存在的，那就是Unicode。**

Unicode

有两个独立的, 创立单一字符集的尝试. 一个是国际标准化组织(ISO)的 ISO 10646 项目, 另一个是由多语言软件制造商组成的协会组织的 Unicode 项目。

Unicode 和 ISO 10646 标准的码表兼容。

Unicode是指一张表，里面包含了可能出现的所有字符，每个字符对应一个数字，这个数字称为码点(Code Point)，如字符'H'的码点为72（十进制），字符'李'的码点为26446（十进制）。

Unicode表包含了1114112个码点，即从000000（十六进制） - 10FFFF（十六进制）。地球上所有字符都可以在Unicode表中找到对应的唯一码点。

Unicode将码空间划分为17个平面，从00 - 10（十六进制，最高两位），即从0 - 16（十进制），每个平面有65536个码点（2^16），其中最重要的是第一个Unicode平面(码位从0000 - FFFF)，包含了最常用的字符，该平面被称为基本多语言平面（Basic Multilingual Plane），缩写为BMP，其他平面称为辅助平面(Supplementary Planes)，在基本多文种平面內， 从D800到DFFF之间的码位区段是永久保留不映射到字符的， 因此**UTF-16**编码巧妙的利用了这保留下来的码位来对辅助平面内的字符进行编码。

***Unicode只是一个符号集，只规定的字符所对应的码点，并没有指定如何存储，如何进行存储出现了不同的编码方案。***

UCS

UCS全称为"Universal Character Set"，在UCS中主要有UCS-2和UCS-4。

1. UCS-2

　　UCS-2是定长字节的，固定使用2个字节进行编码，从0000（十六进制）- FFFF（十六进制）的码位范围，对应第一个Unicode平面。采用BOM(Byte Order Mark)机制，该机制作用如下：1. 确定字节流采用的是大端序还是小端序。2. 确定字节流的Unicode编码方案。

2. UCS-4

　　UCS-4是定长字节的，固定使用4个字节进行编码。也采用了BOM机制。

UTF

　UTF全称为**"Unicode Transformation Format"**，在UTF中主要有UTF-8，UTF-16和UTF-32。

1. **UTF-8**

UTF-8是一种变长编码方式，使用1-4个字节进行编码。UTF-8完全兼容ASCII，对于ASCII中的字符，UTF-8采用的编码值跟ASCII完全一致。UTF-8是Unicode一种具体的编码实现。UTF-8是在互联网上使用最广的一种Unicode的编码规则，因为这种编码有利于节约网络流量（因为变长编码，而非统一长度编码）。关于Unicode码点如何转化为UTF-8编码，可以参照如下规则：

　　① 对于单字节的符号，字节的第一位设为0，后面7位为这个符号的unicode码。因此对于英语字母，UTF-8编码和ASCII码是相同的。

　　② 对于n字节的符号（n>1），第一个字节的前n位都设为1，第n+1位设为0，后面字节的前两位一律设为10。剩下的没有提及的二进制位，全部为这个符号的unicode码。

总结的编码规则如下：

1. Unicode符号范围　　                 | 　　UTF-8编码方式
2. (十六进制) (十进制)　　　　        | 　　（二进制）
3. ----------------------------------------------------------------------------------------------------
4. 0000 0000-0000 007F (0-127)           | 　　 0xxxxxxx
5. 0000 0080-0000 07FF (128-2047)        | 　　 110xxxxx 10xxxxxx
6. 0000 0800-0000 FFFF (2048-65535)      | 　   1110xxxx 10xxxxxx 10xxxxxx
7. 0001 0000-0010 FFFF (65536-1114111)   | 　　 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
8. 说明：字符'A'的Unicode码点为65（十进制），根据上表，在第一行范围，则字符'A'的UTF-8编码为01000001，中文字符'李'的Unicode码点为26446（十进制），二进制为01100111 01001110，十六进制为674E。根据上表，在第三行范围，则将'李'二进制代码从低位到高位依次填入x中，不足的填入0。得到UTF-8编码为11100110 10011101 10001110，即E69D8E（十六进制）。
9. 由上述编码规则可知，0000 0000 - 0000 FFFF（第一行到第三行）为Unicode第一个平面（基本多语言平面），而0001 0000 - 10 FFFF（第四行）为Unicode其他平面（辅助平面）。在基本多语言平面对应了绝大多数常用的字符。对于大于65535（十进制）的码点，即在辅助平面上的码点，需要使用4个字节来进行UTF-8编码
10. UTF-16

UTF-8是不定长的编码，使用1、2、3、4个字节编码，而UTF-16则只使用2或4个字节编码。UTF-16也是Unicode一种具体的编码实现。关于Unicode如何转化为UTf-16编码规则如下

1. 若Unicode码点在第一平面（BPM）中，则使用2个字节进行编码。

② 若Unicode码点在其他平面（辅助平面），则使用4个字节进行编码。

**四、字符编码区别**

　4.1 UCS-2 与 UTF-16区别

　　从上面的分析知道，UCS-2采用的两个字节进行编码。在0000到FFFF的码位范围内，它和UTF-16基本一致，为什么说基本一致，因为在UTF-16中从U+D800到U+DFFF的码位不对应于任何字符，而在使用UCS-2的时代，U+D800到U+DFFF内的值被占用。

　　UCS-2只能表示BMP内的码点（只采用2个字节），而UTF-16可以表示辅助平面内的码点（采用4个字节）。

***我们可以抽象的认为UTF-16可看成是UCS-2的父集***。在没有辅助平面字符（surrogate code points）前，UTF-16与UCS-2所指的意思基本一致。但当引入辅助平面字符后，想要表示辅助平面字符时，就只能用UTF-16编码了。

4.2 UCS -4与 UTF-16的区别

在BMP上UTF-16采用2个字节表示，而在辅助平面上，UTF-16采用的是4个字节表示。对于UCS-4，不管在哪个平面都采用的是四个字节表示。

4.3 为什么UTF-8编码不需要BOM机制

　因为在UTF-8编码中，其自身已经带了控制信息，如1110xxxx 10xxxxxx 10xxxxxx 10xxxxxx，其中1110就起到了控制作用，所以不需要额外的BOM机制