ASCII：美国佬发现，只有一个字节的后7位，前面一位统一为0。0-127一共可以表示128位，这已经足够表示英文。

ISO：欧洲人发现，美国佬那套不能够完全表示我们的字符，所以他们在ASCII码基础上发现了一套新的编码，把闲置的最后一位也用上。

GBK：中国人发现美国佬和欧洲人那套根本不适合我们中国博大精深的中文库，所以采用两个字节来表示中文，而且GBK不是定长的，兼容ASCII。

Unicode：兼容所有语言，采用定长的三个字节。但没规定解码和编码

Utf-8：unicode的实现方式

Utf-16：unicode的实现方式

**1. ASCII码与非ASCII编码**

1.1 ASCII码

我们知道，在计算机内部，所有的信息最终都表示为一个二进制的字符串。每一个二进制位（bit）有0和1两种状态，因此八个二进制位就可以组合出256种状态，这被称为一个字节（byte）。也就是说，一个字节一共可以用来表示256种不同的状态，每一个状态对应一个符号，就是256个符号，从0000000到11111111。  
上个世纪60年代，美国制定了一套字符编码，对英语字符与二进制位之间的关系，做了统一规定。这被称为ASCII码，一直沿用至今。  
ASCII码一共规定了128个字符的编码，比如空格“SPACE”是32（二进制00100000），大写的字母A是65（二进制01000001）。这128个符号（包括32个不能打印出来的控制符号），只占用了一个字节的后面7位，最前面的1位统一规定为0。  
  
1.2、非ASCII编码  
英语用128个符号编码就够了，但是用来表示其他语言，128个符号是不够的。比如，在法语中，字母上方有注音符号，它就无法用ASCII码表示。于是，一些欧洲国家就决定，利用字节中闲置的最高位编入新的符号。比如，法语中的é的编码为130（二进制10000010）。这样一来，这些欧洲国家使用的编码体系，可以表示最多256个符号。  
但是，这里又出现了新的问题。不同的国家有不同的字母，因此，哪怕它们都使用256个符号的编码方式，代表的字母却不一样。比如，130在法语编码中代表了é，在希伯来语编码中却代表了字母Gimel (ג)，在俄语编码中又会代表另一个符号。但是不管怎样，所有这些编码方式中，0—127表示的符号是一样的，不一样的只是128—255的这一段。  
至于亚洲国家的文字，使用的符号就更多了，汉字就多达10万左右。一个字节只能表示256种符号，肯定是不够的，就必须使用多个字节表达一个符号。比如，简体中文常见的编码方式是GB2312，使用两个字节表示一个汉字，所以理论上最多可以表示256x256=65536个符号。  
中文编码的问题需要专文讨论，这篇笔记不涉及。这里只指出，虽然都是用多个字节表示一个符号，但是GB类的汉字编码与后文的Unicode和UTF-8是毫无关系的。

**2. 编码方式详述**

2.1 iso8859-1

属于单字节编码，最多能表示的字符范围是0-255，应用于英文系列。比如，字母'a'的编码为0x61=97。

很明显，iso8859-1编码表示的字符范围很窄，无法表示中文字符。但是，由于是单字节编码，和计算机最基础的表示单位一致，所以很多时候，仍旧使用iso8859-1编码来表示。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："d6 d0 ce c4"（事实上，在进行存储的时候，也是以字节为单位处理的）。而如果是UTF编码，则是6个字节"e4 b8 ad e6 96 87"。很明显，这种表示方法还需要以另一种编码为基础。

2.2 GB2312/GBK

这就是汉子的国标码，专门用来表示汉字，是双字节编码，而英文字母和iso8859-1一致（兼容iso8859-1编码）。其中gbk编码能够用来同时表示繁体字和简体字，而gb2312只能表示简体字，gbk是兼容gb2312编码的。

2.3 unicode

这是最统一的编码，可以用来表示所有语言的字符，而且是定长双字节（也有四字节的）编码，包括英文字母在内。所以可以说它是不兼容iso8859-1编码的，也不兼容任何编码。不过，相对于iso8859-1编码来说，uniocode编码只是在前面增加了一个0字节，比如字母'a'为"00 61"。

需要说明的是，定长编码便于计算机处理（注意GB2312/GBK不是定长编码），而unicode又可以用来表示所有字符，所以在很多软件内部是使用unicode编码来处理的，比如java。

2.4 UTF

考虑到unicode编码不兼容 iso8859-1编码，而且容易占用更多的空间：因为对于英文字母，unicode也需要两个字节来表示。所以unicode不便于传输和存储。因此而产生了utf编码，utf编码兼容iso8859-1编码，同时也可以用来表示所有语言的字符，不过，utf编码是不定长编码，每一个字符的长度从1-6 个字节不等。另外，utf编码自带简单的校验功能。一般来讲，英文字母都是用一个字节表示，而汉字使用三个字节。

注意，虽然说utf是为了使用更少的空间而使用的，但那只是相对于unicode编码来说，如果已经知道是汉字，则使用GB2312/GBK无疑是最节省的。不过另一方面，值得说明的是，虽然utf编码对汉字使用3个字节，但即使对于汉字网页，utf编码也会比unicode编码节省，因为网页中包含了很多的英文字符。

**3.  Unicode的问题**  
需要注意的是，Unicode只是一个符号集，它只规定了符号的二进制代码，却没有规定这个二进制代码应该如何存储。  
比如，汉字“严”的unicode是十六进制数4E25，转换成二进制数足足有15位（100111000100101），也就是说这个符号的表示至少需要2个字节。表示其他更大的符号，可能需要3个字节或者4个字节，甚至更多。  
这里就有两个严重的问题，第一个问题是，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，这是无法接受的。  
它们造成的结果是：1）出现了unicode的多种存储方式，也就是说有许多种不同的二进制格式，可以用来表示unicode。2）unicode在很长一段时间内无法推广，直到互联网的出现。

**4. UTF-8**  
互联网的普及，强烈要求出现一种统一的编码方式。UTF-8就是在互联网上使用最广的一种unicode的实现方式。其他实现方式还包括UTF-16和UTF-32，不过在互联网上基本不用。重复一遍，这里的关系是，UTF-8是Unicode的实现方式之一。  
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。  
UTF-8的编码规则很简单，只有二条：  
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。

下表总结了编码规则，字母x表示可用编码的位。  
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  
  
下面，还是以汉字“严”为例，演示如何实现UTF-8编码。  
已知“严”的unicode是4E25（100111000100101），根据上表，可以发现4E25处在第三行的范围内（0000 0800-0000 FFFF），因此“严”的UTF-8编码需要三个字节，即格式是“1110xxxx 10xxxxxx 10xxxxxx”。然后，从“严”的最后一个二进制位开始，依次从后向前填入格式中的x，多出的位补0。这样就得到了，“严”的UTF-8编码是“11100100 10111000 10100101”，转换成十六进制就是E4B8A5。

**5. little endian和big endian**  
Unicode码可以采用UCS-2格式直接存储。以汉字”严“为例，Unicode码是4E25，需要用两个字节存储，一个字节是4E，另一个字节是25。存储的时候，4E在前，25在后，就是Big endian方式；25在前，4E在后，就是Little endian方式。  
这两个古怪的名称来自英国作家斯威夫特的《格列佛游记》。在该书中，小人国里爆发了内战，战争起因是人们争论，吃鸡蛋时究竟是从大头(Big-Endian)敲开还是从小头(Little-Endian)敲开。为了这件事情，前后爆发了六次战争，一个皇帝送了命，另一个皇帝丢了王位。  
因此，第一个字节在前，就是”大头方式“（Big endian），第二个字节在前就是”小头方式“（Little endian）。  
那么很自然的，就会出现一个问题：计算机怎么知道某一个文件到底采用哪一种方式编码？  
Unicode规范中定义，每一个文件的最前面分别加入一个表示编码顺序的字符，这个字符的名字叫做”零宽度非换行空格“（ZERO WIDTH NO- BREAK SPACE），用FEFF表示。这正好是两个字节，而且FF比FE大1。  
如果一个文本文件的头两个字节是FE FF，就表示该文件采用大头方式；如果头两个字节是FF FE，就表示该文件采用小头方式。