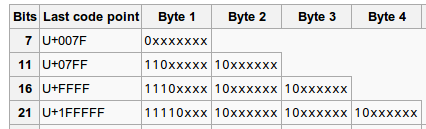
[UTF-8编码规则](http://blog.csdn.net/sandyen/article/details/1108168)

UTF-8 全称是：Universal Character Set Transformation Format--8-bit。UTF-8是表示Unicode字符集中的每个字符的一种编码方式，采用变长编码，1-4字节。和UFF-16以及UTF-32相比，没有endianness和字节顺序掩码引发的复杂性。  
  
UFT-8广泛用于web，email，现在越来越多的操作系统支持它。Linux默认就是UFT-8编码。

。2003年11月RFC 3629文档将UTF-8限制在[0, U+10FFFF]范围内，因此最多只能有4个字节。

下面的图来自wiki:



关于编码方法，首先要了解几个概念：

1. UTF-8的前128个字符都是用单字节表示，完全和ASCII码一样. 这就是上表的第一行

2. 2-4字节的情况下，第一个字节称为leading byte, 后面的字节叫做continuation bytes.

3. leading bytes的高位总是若干个1加上一个0组成，1的个数就是整个字节的个数，比如表格第二行就是110，第三行就是1110， 第四行就是11110.

4. continuation bytes的高位总是10

5.其余的bit用于编码，如果有用不到的bit，都设置为0

上表第一行用单字节表示128个字符，表示全部的ASCII码。

上表第二行用两字节表示1920个字符，包含了绝大多数拉丁字母。

上表第三行用三字节表示Unicode BMP中的余下字符。

上表第四行用四字节表示Unicode其他平面的字符，包括CJK中不常用的字符。

现在看一个来自wiki的例子演示如何对字符€进行UTF-8编码：

step 1：获取€的Unicode code point，是0xU+20AC

step 2：0xU+20AC范围在U+07FF和U+FFFF之间，因此用三字节表示。

step 3：0xU+20AC的二进制码是：10000010101100，14位长，要想表示3字节编码，必须凑成16 bits.因此高位补上两个0，变成2字节16位长:0010000010101100，我下面称为数值串。

step 4: 根据规则，添加一个leading byte，开头是1110，那么这个leading byte还有4个bit需要填充，从数值串高位取4个bit来，leading byte变成了：

11100010，而数值串值为000010101100

step 5: 第一个continuation byte高位应该是10，还缺少6 bits，从数值串中按高位取6 bit,这样第一个continuation byte为：

10000010，而数值串变为101100

step 6: 第二个continuation byte高位也应该是10，还缺少6 bits, 从数值串取6 bits,这样第二个continuation byte为：

10101100

最终编码形成的三字节：

11100010 10000010 10101100

写成16进制就是0xE282AC