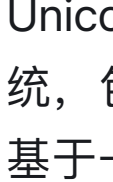


# 计算机组成原理 | Unicode 和 UTF-8是什么关系？

知乎 · 4 个回答 · 8 关注

 武辰  
数学话题下的优秀答主

## Unicode

Unicode 是一种在计算机中表示文本的标准编码系统，包括全球几乎所有语言的字符和符号。Unicode 基于一个唯一的数字代号 为每个字符或符号赋予明确且唯一的定义，比如，英文的“A”的 Unicode 码就是 65，而汉字的“中”的 Unicode 码就是20013。

要通过Python知道某个字符的Unicode编码，可以使用内置函数ord()。ord()函数接受一个字符作为参数，并返回该字符对应的Unicode编码（即码点）。chr()则执行相反的操作。

```
# 示例字符
char = 'A'

# 使用ord()函数获取Unicode编码
unicode_code_point = ord(char)
print(f"The Unicode code point of '{char}' is {unicode_code_point}")

char = '汉'
unicode_code_point = ord(char)
print(f"The Unicode code point of '{char}' is {unicode_code_point}")

unicode_code_point = 65
char = chr(unicode_code_point)
print(f"The character corresponding to Unicode code point {unicode_code_point} is '{char}'")
```

上面的代码是用十进制来表示编码点。也可以用16进制表示编码点。下面是代码：

```
char = '汉'
unicode_code_point = ord(char)
hex_code_point = hex(unicode_code_point)

print(f"The Unicode code point of '{char}' is {hex_code_point}")
```

## Unicode转义序列

Unicode就像一本巨大的全球字符字典，每个字符都有一个独一无二的“身份证号码”——这就是码点。码点是一个数字，用来唯一地识别每一个字符，不论它是英文、中文、日文，还是各种符号。

但是，直接在电脑程序或文本文件里写这些数字“身份证号码”来代表字符，既不好看也不方便。比如，你想在程序里写一个汉字“汉”，如果直接写它的Unicode码点数字（比如“28450”），别人很难一眼看出这是个汉字，而且打字也很麻烦。

这时，转义序列就像一个“暗号”，它用一种更易读、易写的格式来代替直接写数字。对于Unicode，这个“暗号”就是像\uXXXX这样的形式。比如，汉字“汉”的转义序列是\u6c49，一看就知道它代表一个特殊字符，而且比直接写数字好记多了。

使用转义序列有以下几个好处：

- 1.看得明白：当你在程序或文件里看到\u6c49，立刻知道它代表一个特定的字符（汉字“汉”），而不是一个普通的数字或乱码。
- 2.写得方便：打字时，输入\u6c49比输入一串长长的数字要快得多，也少出错。
- 3.电脑能懂：虽然我们看到的是\u6c49这个“暗号”，但电脑会把它自动翻译成对应的码点数字，然后正确显示或处理字符“汉”。这样，不管你的电脑设置、编程语言或文本格式怎么变，只要大家都遵守这个“暗号”规则，就能确保字符正确无误地传递和显示。

所以，尽管Unicode已经有了码点作为字符的唯一标识，但我们还需要转义序列这个“暗号”，来让字符在电脑程序和文本文件中既易读易写，又能被正确理解和处理。

Unicode转义序列允许以一种标准且易于理解的方式在字符串中表示任何Unicode字符。Unicode转义序列的格式通常是 \u 后面跟随四位十六进制数（'\u0000' 到 '\uFFFF'）。例如以下的代码：

```
print("汉字: \u6c49")
print("A: \u0041")
```

## ASCII

ASCII 是美国信息交换标准代码（American Standard Code for Information Interchange）的缩写，是计算机科学中最常用的字符编码标准之一。它主要用于显示现代英语，且也能处理其他一些在英语中常见的西欧语言。

ASCII标准包含了128个字符编码，其中包括了用来控制诸如回车、换行等的特殊字符，也包括了所有的英文大写和小写字母、数字、标点符号等。在ASCII码中，每个字符都对应一个从0到127的整数。

例如，大写的英文字母"A"在ASCII中的编码是65，小写的英文字母"a"在ASCII中的编码是97，数字"0"的编码是48等等。

ASCII在早期的电脑系统和网络中得到广泛应用，而现代的UTF-8编码等也是以ASCII为基础进行扩展的。

## UTF-8

UTF-8（8-bit Unicode Transformation Format）是一种实现 Unicode 字符集编码的可变长度编码方案。

假设文字是图画，Unicode就好比是一本“字典”，记录了每个图画（即每个字符）对应的唯一数字ID。任何语言的字符，不论是英文的“A”、“B”、“C”，还是中文的“中”、“国”，甚至是表情符号，在Unicode标准中都对应有一个唯一的数字。

这个Unicode的“字典”很好用，因为有了这个“字典”，世界各地的人都可以通过互相不认识的字符对应的Unicode编号（这个编号是公开并且全球统一的）来互相理解 and 交流。

然而，计算机其实并不懂这个“字典”，计算机只认识0和1。那么，如何把Unicode“字典”中的编号转化成计算机可以识别的0和1的序列呢？这就需要一种“翻译规则”，而UTF-8就是这样一种“翻译规则”。

UTF-8规定了怎么样把Unicode“字典”中的编号转化成计算机认识的0和1序列，它不止能表示Unicode范围内的所有字符，还能尽量节省存储空间。例如，UTF-8规定对于英文字母，就按照ASCII（另一种早期的字符编码，覆盖英文字符）的方式来编码，只用一个字节（0-255）。对于中文字符等，UTF-8则会用到3个或4个字节来存储。

总的来说，Unicode是为了统一世界上所有语言的文字而设立的一套全球统一的字符编号体系。而UTF-8则是一种常用的，把这些编号转化成计算机能理解的编码方式。

## Q&A

以下问题由GPT回答生成。

### 在unicode中，一个字符占用多少空间？

在Unicode中，一个字符（即一个Unicode码点）本身是一个抽象的概念，它并不直接占用物理存储空间。Unicode码点是一个唯一的数字标识符，用来唯一地表示一个字符。码点的表示范围超过110万个字符。每个码点通常用十六进制数表示，例如汉字“汉”的Unicode码点是U+6C49。

在Unicode中，一个字符的理论空间需求取决于所使用的编码方案。UTF-8编码下，单个字符分别可能占用1到4个字节。

### 请通俗解释utf-8和unicode转义字符形式的区别

UTF-8：想象一下，你要把全世界所有的字符（包括各国文字、符号、表情等）打包成一个个包裹，然后通过快递送到世界各地。UTF-8就是那个负责打包的快递员。他按照一套特别的规则，将每个字符装进大小不一的箱子（1到4个字节），并在箱子外面贴上标签，标明里面装的是哪个字符。这样，无论这些包裹走到哪里，只要按照标签上的信息，就可以准确地知道箱子里装的是哪个字符。在计算机的世界里，这些包裹就是存储在硬盘、网络传输、软件内部的字符数据，而UTF-8就是将字符转化为这些数据的标准方法。

Unicode转义字符形式（如\uXXXX）：现在，假设你在写一封信，里面需要用到一些特殊的字符，比如某个外国朋友的名字包含一个罕见的字母，或者你想在信中插入一个笑脸表情。直接把这些字符写在纸上，邮局的工作人员可能看不懂，甚至可能会弄丢。怎么办呢？这时，你可以使用“密码”来代替这些字符。比如，你可以写：“亲爱的朋友，你的名字是\uXXXX，我今天心情\uYYYY。”这里的\uXXXX和\uYYYY就是密码，分别代表那个罕见的字母和笑脸表情。当你朋友收到信后，他知道这些密码的含义，就能正确理解你的意思。在编程和文本文件中，Unicode转义字符形式就像这些密码，让我们在不能直接写出某些字符的情况下，也能明确告诉计算机我们要使用哪个字符。

总结一下：

UTF-8：是计算机用来实际存储和传输字符数据的“快递员”，它按照一套规则将字符打包成数据包（字节序列），确保字符能在各种环境中准确传递。

Unicode转义字符形式（如\uXXXX）：是在编程和文本文件中用来明确表示字符的一种“密码”，让我们在不能直接写出某些字符时，也能让计算机知道我们要使用哪个字符。