目录

1. 字符编码和乱码 3
1.1. 字符编码 3
1.2. UTF-8 3
1.3. 常见地区代码 4
1.3.1. 简体中文 4
1.3.2. 繁体中文 4
1.3.3. 日文 4
1.4. 避免乱码 4

1. 字符编码和乱码

> *手持两把锟斤拷，口中疾呼烫烫烫。脚踏千朵屯屯屯，笑看万物锘锘锘。*

1.1. 字符编码

字符编码可理解为字符和整数编号的一一对应。比如在广泛的ASCII编码中，大写字母A对应65。

Character	ASCII (dec)	ASCII (bin)	ASCII (hex)
A	65	01000001	41

ASCII是7位的，而一个字节有8位，故ASCII编码中，一个字符占一个字节，且最高位一定是0。

然而ASCII毕竟表示的字符有限，为了能表示特殊字符和非拉丁文字，各种编码纷纷出现。比如中文的GBK (CP936)、Big5 (CP950)、日文的Shift_JIS (CP932)等。这些编码互不兼容，即同一个整数在不同的编码中对应的字符不同。当选择错误的编码显示时，字符成了乱码。

于是Unicode出现了，Unicode对世界上大部分文字都作了统一编号，以适合多语种环境，解决乱码的问题。

需要强调的是，Unicode只是对每个字符指定了一个**数字编号**；系统的内部存储是二进制的，Unicode的具体实现才是指定了每个字符对应的**二进制序列**，Unicode的具体实现称为Unicode转换格式（UTF），UTF有多种，目前最广泛使用的是UTF-8。

1.2. UTF-8

UTF-8中8的含义是该编码方式的最小单元是8位，即一个字节。

与ASCII不同，UTF-8是变长的，一个字符可占用1～6个字节。第一个字节的前几位是描述部分，指定该字符占几个字节。格式举例如下：

Byte 1	Byte 2	Byte 3
0xxxxxxx		
110xxxxx	10xxxxxx	
1110xxxx	10xxxxxx	10xxxxxx

- 一个字节的第一位为**0**，则字符占1个字节。0之后的所有部分（7位）代表在Unicode中的编号。即**向下兼容ASCII**。
- 一个字节以**110**开头，则字符占2个字节。110之后的所有部分（5位）加上后一个字节的除开头10外的部分（6位）代表在Unicode中的编号。且第二个字节以10开头。
- 一个字节以**1110**开头，则字符占3个字节。110之后的所有部分（5位）加上后两个字节的除10外的部分（12位）代表在Unicode中的序号。且第二、第三个字节以10开头。
- 如果一个字节以**10**开头，则该字节是一个多字节字符的第二个或以后的字节。

UTF的其他编码如UTF-16不与UTF-8兼容，甚至不与ASCII兼容。
其最小单位是16位，即2个字节。另外，UTF-16区分大小端。故
为避免浪费空间以及造成混乱，目前Unicode编码中**只推
荐UTF-8**。

Windows记事本的Unicode编码其实是UTF-16LE。如果需要与其他系统交换文件，建议UTF-8。

1.3. 常见地区代码

以UTF-8为代表的Unicode编码方式已成为潮流，而因为节省空间等原因目前仍存在互不兼容的内码。简单介绍如下。

1.3.1. 简体中文

- GB2312
- GBK (CP936)
- GB18030 (CP54936)

三者的兼容关系大致为：GB2312⊂GBK⊂GB18030。

GB2312和GBK中，一个字符占2个字节；GB18030采用变长，一个字符占1、2或4个字节。

1.3.2. 繁体中文

- Big5 (CP950)
- CNS11643

两者的兼容关系为：Big5⊂CNS11643。

Big5为台湾最常用的编码，一个字符占2个字节；CNS11643完整收录Big5，但未被广泛使用。

1.3.3. 日文

- Shift_JIS
- ISO-2022-JP

1.4. 避免乱码

本文开头的“诗”即编码选择不当的例子。统一使用UTF-8是解决乱码的好方法。另外，当对应的字体不存在时，通常出现如黑色方块之类的符号，安装合适的字体可以解决。比较推荐的字体有：

- GNU Unifont
- Google Noto Fonts
- Adobe思源字体

然而遇到乱码时应如何解决？以下为部分场合时的解决方法。

1.4.1. 网页

部分网页的源代码于浏览器默认的编码不同，且网页未明确指出编码时，会出现乱码。此时在浏览器中选择正确的编码即可。如果您知道该网页的语言，可以尝试该语言所有的编码，直到显示正常。 当然，如果您是开发人员，最好指定编码，如：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

1.4.2. ZIP压缩文件

ZIP压缩文件中的文件名未指定编码，故在系统编码不同时直接解压可造成文件名乱码。如在Windows中文系统中压缩为ZIP文件，其编码为GBK□该文件在Linux□UTF-8□中正确的解压方式为：

```
LANG=C 7z x filename.zip
convmv -f gbk -t utf8 --notest -r .
```

您需要p7zip和convmv两个工具。

尽管ZIP被广泛支持，但考虑到文件名编码和压缩率等问题，建议采用其他方式压缩文件。

1.4.3. 文本文件

您可以用iconv工具转码。

1.4.4. Python

使用如下开头是个好习惯。

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

From:
https://irdya.top/ - **漂流記**

Permanent link:
https://irdya.top/zh/misc/encode

Last update: **2022/05/26 03:24**