1 NAME

perlunitut – Perl万国码指南

2 DESCRIPTION

编程语言只能处理简单字符的日子一去不复返了！现代程序必须能够和那些“有趣的”重音字母以及诸如欧洲符号一样的字符打交道。这意味着程序员要形成一种新的习性，编写能够处理 Unicode 的软件并不困难，但的确需要一些规范，才能把这件事情做好做对。

要了解字符集和字符编码，需要知道的很多。最好你能花一天时间来学习这些，但对于基础知识，几分钟时间就足够了。

然而这些知识也不算很“基础”。这里假设你已经懂得字节和字符之间的区别，知道有许多不同的字符集和编码，并且你的程序要很明确的区分它们。推荐的阅读物为 “The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets (No Excuses!)” 作者 Joel Spolsky, http://joelonsoftware.com/articles/Unicode.html.

这篇指南使用通用词汇，简单且有限介的绍了 Perl 提供的，与字符串相关的特性。对于大多数项目，这些信息应该已经足够了。

定义

在开始时将一些概念弄清楚非常重要。这也是本篇指南最重要的部分。这里的提供的信息可能和你之前在网络上找到的相冲突，最可能的原因是：它们大部分都错了。

你可能要重读这一整节几次

万国码(Unicode)

Unicode 是一个字符集，它提供了容纳字符的空间。字符的序数值叫做码位 (code point)。(实际上，码位和字符之间的区别已经很模糊了，这两个术语经常互用。)

这世界上有很多很多很多码位，但计算机以字节为单位工作，一个字节自只能提供 256 个值。万国码中包含的字符比这多多了，所以你必须使用某种方法来获得这些字符。

万国码可以使用许多种编码方式编码，其中 UTF-8 用得最为广泛。在一种万国码编码中，多字节的子序列可以用来存储一个码位，或者说：字符。

UTF-8

UTF-8 是一种万国码编码方式。很多人认为万国码和 UTF-8 是同一个东西，实际上并不是。编码方式还有更多，但大半个世界都在以 UTF-8 为标准。

UTF-8 的前128个码位和 ASCII 编码一样。一个字节容纳一个字符。所有的其他字符都使用了一种复杂的方法编码成两个或者更多（最高6）字节。幸运的是，Perl 替我们做了这一工作，所以我们不必担心这一点。

文本串（字符串）

文本串(Text strings)，或者字符串(character strings)由字符组成。它和字节没关系，和编码也没关系。字符就是字符而已。

对于一个文本串，你可以做一些操作，比如：

```
$text =~ s/foo/bar/;
if ($string =~ /^\d+$/) { ... }
$text = ucfirst $text;
my $character_count = length $text;
```

字符的值（ord，chr）是对应的万国码码位。

二进制串（字节串）

二进制串(Binary strings)或字节串(byte strings)由字节组成。这里没有 字符，只有字节。所有与外界（任何你当前 Perl 进程之外的东西）的交流都是通 过二进制串来实现。

在二进制串上，你可以做这些操作：

```
my (@length_content) = unpack "(V/a)*", $binary;
$binary =~ s/\x00\x0F/\xFF\xF0/; # 给胆子大的 :)
print {$fh} $binary;
my $byte_count = length $binary;
```

编码

编码（作动词）是从文本到二进制的转换。要给文本串编码，你必须给 出目标编码，比如 iso-8859 或 UTF-8。有些编码并不完全支持万国码规范 ，比如 iso-8859("latin") 范围。那么在编码过程中，会丢失那些万国码不能 代表的字符，

解码

解码是从二进制到文本的转换。解码时，你必须知道编码时用到的是哪种编码 方式。最重要的是，二进制串必须能被解码。试图把一张PNG图片解码成文本串是 不现实的。

内部格式

Perl有它自己的内部格式，它用这种内部格式来编码文本串，使其能保存到内 存里面。所有的文本串都是这种内部格式。事实上，它们从来没有其他格式。

你不需要知道这是什么格式，因为在你解码或者编码的时候，其间的转换会自动进 行。

你的新工具包

把下面几行加入到你的标准头部中：

```
use Encode qw(encode decode);
```

或则，如果你很懒，只需要：

```
use Encode;
```

I/O流（真正的5分钟指南）

一个程序典型的输入/输出流如下：

1. 收取和解码
2. 处理
3. 编码和输出

如果你的输入是二进制，而且应该保持为二进制，那么你当然不需要把它解码为文 本串。在其他所有情况下，你都应该将其解码。

如果你不知道数据是如何被编码的，解码也不会很可靠。你可以选择的话，使用 UTF-8 标准化是个好主意。

```
my $foo  = decode('UTF-8', get 'http://example.com/'); my $bar  =
decode('ISO-8859-1', readline STDIN);
my $xyzzy = decode('Windows-1251', $cgi->param('foo'));
```

处理过程就像你以前所知一样。唯一的不同是，你使用的是字符，而不是字节。当 你在使用如 `substr` 或 `length` 时，这一点非常有用。

在文本串中没有字节，知道这一点很重要。当然，Perl有它的内部格式，用来在内 存中存储字符串，但请你忽略这一点。如果你需要做一些关于字节数目的事情，你 最好把这些事情挪到步骤3，将字符串编码之后。这样你才能准确的知道目标串里 面有多少字节。

将文本串编码成二进制串的方法如下：

```
$body = encode('UTF-8', $body);
```

如果你需要知道字符串字节的长度，此时就是最好的时侯。因为 `$body` 现在是 字节串了，`length` 会报告字节的数目，而不是字符的数目。字符数现在是未知 的，因为字符只存在于文本串。

```
my $byte_count = length $body;
```

如果你所使用的协议支持设置你所使用的字符编码，那么请使用这个特性。比如 E-mail 和 HTTP 支持 MIME 头部，这样你可以使用 `Content-Type` 头部。它们 还可以有 `Content-Length`，用以表示字节的数目。如果已经知道了这个数 目，设置它永远是个好主意。

```
"Content-Type: text/plain; charset=UTF-8",
"Content-Length: $byte_count"
```

3 总结

将你收到的所有东西解码，将你发出的所有东西编码。（如果是文本数据的话。）

4 Q and A (or FAQ)

读完这篇文档之后，你应该继续阅读 perlunifaq

5 致谢

Thanks to Johan Vromans from Squirrel Consultancy. His UTF-8 rants during the Amsterdam Perl Mongers meetings got me interested and determined to find out how to use character encodings in Perl in ways that don't break easily.

Thanks to Gerard Goossen from TTY. His presentation "UTF-8 in the wild" (Dutch Perl Workshop 2006) inspired me to publish my thoughts and write this tutorial.

Thanks to the people who asked about this kind of stuff in several Perl IRC channels, and have constantly reminded me that a simpler explanation was needed.

Thanks to the people who reviewed this document for me, before it went public. They are: Benjamin Smith, Jan-Pieter Cornet, Johan Vromans, Lukas Mai, Nathan Gray.

6 AUTHOR

Juerd Waalboer <#####@juerd.nl>

7 SEE ALSO

perlunifaq, perlunicode, perluniintro, Encode

8 TRANSLATORS

Woosley Xu. woosley.xu@gmail.com