第十一章　正则表达式

如果你不懂正则表达式，而你还想进一步学习编程的话，那你应该停下手边的事情，先学学正则表达式。即使你不想学习编程，也不喜欢编程，学习一点正则表达式，也可能让你的文本编辑效率提高很多。

在这里，我不想详细介绍正则表达式，因为我觉得这类的文档已经很多了，比我写得好的文章多的是。如果你找不到一个好的入门教程，我建议你不妨看看 perlretut。我想说的是和 emacs 有关的正则表达式的内容，比如，和 Perl 正则表达式的差异、语法表格（ syntax table）和字符分类（ category）等。

11.1　与 Perl 正则表达式比较

Perl 是文本处理的首选语言。它内置强大而简洁的正则表达式，许多程序也都兼容 Perl 的正则表达式。说实话，就简洁而言，我对 emacs 的正则表达式是非常反感的，那么多的反斜线经常让我抓狂。首先，emacs 里的反斜线构成的特殊结构（ backslash construct）出现是相当频繁的。在 Perl 正则表达式里， “()[]—” 都是特殊字符，而 emacs 它们不是这样。所以它们匹配字符时是不用反斜线，而作为特殊结构时就要用反斜线。而事实上 “()—”作为字符来匹配的情形远远小于作为捕捉字符串和作或运算的情形概率小。而 emacs 的正则表达式又没有 Perl 那种简洁的记号，完全用字符串来表示，这样使得一个正则表达式常常一眼看去全是 “\\”。

到底要用多少个 \? 经常会记不住在 emacs 的正则表达式中应该用几个 \。有一个比较好的方法，首先想想没有引号引起时的正则表达式是怎样。比如对于特殊字符 \$ 要用\$，对于反斜线结构是 \(，\{，\| 等等。知道这个怎样写之后，再把所有 \ 替换成 \\，这就是最后写到双引号里形式。所以要匹配一个\，应该用\\，而写在引号里就应该用 \\\\ 来匹配。

emacs 里匹配的对象不仅包括字符串，还有 buffer，所以有一些对字符串和 buffer 有区分的结构。比如 \$ 对于字符串是匹配字符串的末尾，而在 buffer 里是行尾。而 \’ 匹配的是字符串和 buffer 的末尾。对应 ^ 和 \‘ 也是这样。

emacs 对字符有很多种分类方法，在正则表达式里也可以使用。比如按语法类型分类，可以用 \s 结构匹配一类语法分类的字符，最常用的比如匹配空格的 \s- 和匹配词的 \sw（等价于 \w）。这在 Perl 里是没有的。另外 emacs 里字符还对应一个或多个分类（ category），比如所有汉字在分类”c” 里。这样可以用 \cc 来匹配一个汉字。这在 Perl 里也有类似的分类。除此之外，还有一些预定义的字符分类，可以用 [:class:] 的形式，比如 [:digit:] 匹配 0-9 之间的数，[:ascii:] 匹配所有 ASCII 字符等等。在 Perl 里只定义几类最常用的字符集，比如 \d，\s，\w，但是我觉得非常实用。比 emacs 这样长的标记好用的多。

另外在用[] 表示一个字符集时，emacs 里不能用 \ 进行转义，事实上 \ 在这里不是一个特殊字符。所以 emacs 里的处理方法是，把特殊字符提前或放在后面，比如如果要在字符集里包括] 的话，要把] 放在第一位。而如果要包括 -，只能放在最后一位，如果要包括 ^ 不能放在第一位。如果只想匹配一个 ^，就只能用\^ 的形式。比较拗口，希望下面这个例子能帮你理解：

```
(let ((str "abc]-^]123"))
  (string-match "[]^0-9-]+" str)
  (match-string 0 str))                ; => "]-^]123"
```

最后提示一下，emacs 提供一个很好的正则表达式调试工具：M-x re-builder。它能显示 buffer 匹配正则表达式的字符串，并用不同颜色显示捕捉的字符串。

11.2 语法表格和分类表格简介

语法表格指的是 emacs 为每个字符都指定了语法功能，这为解析函数，复杂的移动命令等等提供了各种语法结构的起点和终点。语法表使用的数据结构是一种称为字符表（ char-table）的数组，它能以字符作为下标（还记得 emacs 里的字符就是整数吗）来得到对应的值。语法表里一个字符对应一个语法分类的列表。每一个分类都有一个助记字符（ mnemonic character）。一共有哪几类分类呢？

名称	助记符	说明
空白（ whitespace）	- 或␣	
词（ word）	w	
符号（ symbol）	_	这是除 word 之外其它用于变量和命令名的字符。
标点（ punctuation）	.	
open 和 close	(和)	一般是括号()[]
字符串引号（ string quote）	"	
转义符（ escape-syntax）	\	用于转义序列，比如 C 和 Lisp 字符串中的 \。
字符引号（ character quote）	/	
paired delimiter	$	只有 TeX 模式中使用
expression prefix	'	
注释开始和注释结束	< 和 >	
inherit standard syntax	@	
generic comment delimiter	!	

语法表格可以继承，所以基本上所有语法表格都是从 standard-syntax-table 继承而来，作少量修改，加上每个模式特有的语法构成就行了。一般来说记住几类重要的分类就行了，比如，空白包括空格，制表符，换行符，换页符。词包括所有的大小写字母，数字。符号一般按使用的模式而定，比如 C 中包含“_”，而 Lisp 中是 $&*+-_<>。可以用 M-x describe-syntax 来查看所有字符的语法分类。

字符分类（ category）是另一种分类方法，每个分类都有一个名字，对应一个从 到 ~ 的 ASCII 字符。可以用 M-x describe-categories 查看所有字符的分类。每一种分类都有说明，我就不详细解释了。

11.3 几个常用的函数

如果你要匹配的字符串中含有很多特殊字符，而你又想用正则表达式进行匹配，可以使用 regexp-quote 函数，它可以让字符串中的特殊字符自动转义。

一般多个可选词的匹配可以用或运算连接起来，但是这有两个不好的地方，一是要写很长的正则表达式，还含有很多反斜线，不好看，容易出错，也不好修改，二是效率很低。这时可以使用 regexp-opt 还产生一个更好的正则表达式：

```
(regexp-opt '("foobar" "foobaz" "foo")) ; => "foo\\(?:ba[rz]\\)?"
```

11.4 函数列表

```
(regexp-quote STRING)
(regexp-opt STRINGS &optional PAREN)
```

11.5 命令列表

```
describe-syntax
describe-categories
```