第六章 模式匹配

by flamephoenix

一、简介

模式指在字符串中寻找的特定序列的字符，由反斜线包含：/def/即模式def。其用法如结合函数split将字符串用某模式分成多个单词：@array = split(/ /, $line);

二、匹配操作符 =~、!~

=~检验匹配是否成功：$result = $var =~ /abc/;若在该字符串中找到了该模式，则返回非零值，即true，不匹配则返回0，即false。!~则相反。

这两个操作符适于条件控制中，如：

```
if ($question =~ /please/) {
  print ("Thank you for being polite!\n");
}
else {
  print ("That was not very polite!\n");
}
```

三、模式中的特殊字符

PERL在模式中支持一些特殊字符，可以起到一些特殊的作用。

1、字符 +

+意味着一个或多个相同的字符，如：/de+f/指def、deef、deeeeef等。它尽量匹配尽可能多的相同字符，如/ab+/在字符串abbc中匹配的将是abb，而不是ab。

当一行中各单词间的空格多于一个时，可以如下分割：

```
@array = split (/ +/, $line);
```

注：split函数每次遇到分割模式，总是开始一个新单词，因此若$line以空格打头，则@array的第一个元素即为空元素。但其可以区分是否真有单词，如若$line中只有空格，则@array则为空数组。且上例中TAB字符被当作一个单词。注意修正。

2、字符 []和[^]

[]意味着匹配一组字符中的一个，如/a[0123456789]c/将匹配a加数字加c的字符串。与+联合使用例：/d[eE]+f/匹配def、dEf、deef、dEdf、dEEEeeeEef等。^表示除其之外的所有字符，如：/d[^deE]f/匹配d加非e字符加f的字符串。

3、字符 *和?

它们与+类似，区别在于*匹配0个、1个或多个相同字符，?匹配0个或1个该字符。如/de*f/匹配df、def、deeeef等；/de?f/匹配df或def。

4、转义字符

如果你想在模式中包含通常被看作特殊意义的字符，须在其前加斜线"\"。如：/*+/中*即表示字符*，而不是上面提到的一个或多个字符的含义。斜线的表示为/\\/。在PERL5中可用字符对\Q和\E来转义。

5、匹配任意字母或数字

上面提到模式/a[0123456789]c/匹配字母a加任意数字加c的字符串，另一种表示方法为：/a[0-9]c/，类似的，[a-z]表示任意小写字母，[A-Z]表示任意大写字母。任意大小写字母、数字的表示方法为：/[0-9a-zA-Z]/。

6、锚模式

锚	描述
^ 或 \A	仅匹配串首
$ 或 \Z	仅匹配串尾
\b	匹配单词边界
\B	单词内部匹配

例1：/^def/只匹配以def打头的字符串，/$def/只匹配以def结尾的字符串，结合起来的/^def$/只匹配字符串def(?)。\A和\Z在多行匹配时与^和$不同。

例2：检验变量名的类型：

```
if ($varname =~ /^\$[A-Za-z][_0-9a-zA-Z]*$/) {
  print ("$varname is a legal scalar variable\n");
} elsif ($varname =~ /^@[A-Za-z][_0-9a-zA-Z]*$/) {
  print ("$varname is a legal array variable\n");
} elsif ($varname =~ /^[A-Za-z][_0-9a-zA-Z]*$/) {
  print ("$varname is a legal file variable\n");
} else {
  print ("I don't understand what $varname is.\n");
}
```

例3：\b在单词边界匹配：/\bdef/匹配def和defghi等以def打头的单词，但不匹配abcdef。/def\b/匹配def和abcdef等以def结尾的单词，但不匹配defghi，/\bdef\b/只匹配字符串def。注意：/\bdef/可匹配$defghi，因为$并不被看作是单词的部分。

例4：\B在单词内部匹配：/\Bdef/匹配abcdef等，但不匹配def；/def\B/匹配defghi等；/\Bdef\B/匹配cdefg、abcdefghi等，但不匹配def,defghi,abcdef。

7、模式中的变量替换

将句子分成单词：

```
$pattern = "[\\t ]+";
```

```
@words = split(/$pattern/, $line);
```

8、字符范围转义

转义字符	描述	范围
\d	任意数字	[0-9]
\D	除数字外的任意字符	[^0-9]
\w	任意单词字符	[_0-9a-zA-Z]
\W	任意非单词字符	[^_0-9a-zA-Z]
\s	空白	[\r\t\n\f]
\S	非空白	[^\r\t\n\f]

例：/[\da-z]/匹配任意数字或小写字母。

9、匹配任意字符

字符"."匹配除换行外的所有字符，通常与*合用。

10、匹配指定数目的字符

字符对{}指定所匹配字符的出现次数。如：/de{1,3}f/匹配def,deef和deeef；/de{3}f/匹配deeef；/de{3,}f/匹配不少于3个e在d和f之间；/de{0,3}f/匹配不多于3个e在d和f之间。

11、指定选项

字符"|"指定两个或多个选择来匹配模式。如：/def|ghi/匹配def或ghi。

例：检验数字表示合法性

```
if ($number =~ /^-?\d+$|^-?0[xX][\da-fa-F]+$/) {
  print ("$number is a legal integer.\n");
} else {
  print ("$number is not a legal integer.\n");
}
```

其中 ^-?\d+$ 匹配十进制数字，^-?0[xX][\da-fa-F]+$ 匹配十六进制数字。

12、模式的部分重用

当模式中匹配相同的部分出现多次时，可用括号括起来，用\n来多次引用，以简化表达式：

/\d{2}([\W])\d{2}\1\d{2}/ 匹配：

```
12-05-92
26.11.87
07 04 92等
```

注意：/\d{2}([\W])\d{2}\1\d{2}/ 不同于/(\d{2})([\W])\1\2\1/ ，后者只匹配形如17-17-17的字符串，而不匹配17-05-91等。

13、转义和特定字符的执行次序

象操作符一样，转义和特定字符也有执行次序：

特殊字符	描述
( )	模式内存
+ * ? { }	出现次数
^ $ \b \B	锚
|	选项

14、指定模式定界符

缺省的，模式定界符为反斜线/，但其可用字母m自行指定，如：

m!/u/jqpublic/perl/prog1! 等价于/\/u\/jqpublic\/perl\/prog1/

注：当用字母'作为定界符时，不做变量替换；当用特殊字符作为定界符时，其转义功能或特殊功能即不能使用。

15、模式次序变量

在模式匹配后调用重用部分的结果可用变量$n，全部的结果用变量$&。

```
$string = "This string contains the number 25.11.";
$string =~ /-?(\d+)\.?(\d+)/; # 匹配结果为25.11
$integerpart = $1; # now $integerpart = 25
$decimalpart = $2; # now $decimalpart = 11
$totalpart = $&; # now totalpart = 25.11
```

四、模式匹配选项

选项	描述
g	匹配所有可能的模式
i	忽略大小写
m	将串视为多行
o	只赋值一次
s	将串视为单行
x	忽略模式中的空白

1、匹配所有可能的模式(g选项)

```
@matches = "balata" =~ /.a/g; # now @matches = ("ba", "la", "ta")
```

匹配的循环：

```
while ("balata" =~ /.a/g) {
  $match = $&;
  print ("$match\n");
}
```

结果为：

```
ba
la
ta
```

当使用了选项g时，可用函数pos来控制下次匹配的偏移：

```
$offset = pos($string);
pos($string) = $newoffset;
```

2、忽略大小写(i选项)例

/de/i 匹配de,dE,De和DE。

3、将字符串看作多行(m选项)

在此情况下，^符号匹配字符串的起始或新的一行的起始；$符号匹配任意行的末尾。

4、只执行一次变量替换例

```
$var = 1;
$line = <STDIN>;
while ($var < 10) {
  $result = $line =~ /$var/o;
  $line = <STDIN>;
  $var++;
}
```

每次均匹配/1/。

5、将字符串看作单行例

/a.*bc/s匹配字符串axxxxx \nxxxxbc，但/a.*bc/则不匹配该字符串。

6、在模式中忽略空格

/\d{2} ([\W]) \d{2} \1 \d{2}/x等价于/\d{2}([\W])\d{2}\1\d{2}/。

五、替换操作符

语法为s/pattern/replacement/，其效果为将字符串中与pattern匹配的部分换成replacement。如：

```
$string = "abc123def";
$string =~ s/123/456/; # now $string = "abc456def";
```

在替换部分可使用模式次序变量$n，如s/(\d+)/[$1]/，但在替换部分不支持模式的特殊字符，如{},*,+等，如s/abc/[def]/将把abc替换为[def]。

替换操作符的选项如下表：

选项	描述
g	改变模式中的所有匹配
i	忽略模式中的大小写
e	替换字符串作为表达式
m	将待匹配串视为多行
o	仅赋值一次
s	将待匹配串视为单行
x	忽略模式中的空白

注：e选项把替换部分的字符串看作表达式，在替换之前先计算其值，如：

```
$string = "0abc1";
$string =~ s/[a-zA-Z]+/$& x 2/e; # now $string = "0abcabc1"
```

六、翻译操作符

这是另一种替换方式，语法如：tr/string1/string2/。同样，string2为替换部分，但其效果是把string1中的第一个字符替换为string2中的第一个字符，把string1中的第二个字符替换为string2中的第二个字符，依此类推。如：

```
$string = "abcdefghicba";
$string =~ tr/abc/def/; # now string = "defdefghifed"
```

当string1比string2长时，其多余字符替换为string2的最后一个字符；当string1中同一个字符出现多次时，将使用第一个替换字符。

翻译操作符的选项如下：

选项	描述
c	翻译所有未指定字符
d	删除所有指定字符
s	把多个相同的输出字符缩成一个

如$string =~ tr/\d/ /c;把所有非数字字符替换为空格。$string =~ tr/\t //d；删除tab和空格；$string =~ tr/0-9/ /cs；把数字间的其它字符替换为一个空格。

七、扩展模式匹配

PERL支持PERL4和标准UNIX模式匹配操作所没有的一些模式匹配能力。其语法为：(?<c>pattern)，其中c是一个字符，pattern是起作用的模式或子模式。

1、不存贮括号内的匹配内容

在PERL的模式中，括号内的子模式将存贮在内存中，此功能即取消存贮该括号内的匹配内容，如/(?:a|b|c)(d|e)f\1/中的\1表示已匹配的d或e，而不是a或b或c。

2、内嵌模式选项

通常模式选项置于其后，有四个选项：i、m、s、x可以内嵌使用，语法为：/(?option)pattern/，等价于/pattern/option。

3、肯定的和否定的预见匹配

肯定的预见匹配语法为/pattern(?=string)/，其意义为匹配后面为string的模式，相反的，(?!string)意义为匹配后面非string的模式，如：

```
$string = "25abc8";
$string =~ /abc(?=[0-9])/;
```

```
  $matched = $&; # $&为已匹配的模式，此处为abc，而不是abc8
```

4、模式注释

PERL5中可以在模式中用?#来加注释，如：

```
  if ($string =~ /(?i)[a-z]{2,3}(?# match two or three alphabetic characters)/ {
    ...
  }
```

上一章 下一章 目录