# 表达式模式

# 匹配操作符

### 匹配：m/<regexp>;/ （还可以简写为/<regexp>;/，略去m）

=~表示相匹配。

！=表示不相匹配。

=~检验匹配是否成功：$result=$var=~/abc/;若在该字符串中找到了该Perl正则表达式中模式，则返回非零值，即true，不匹配则返回0，即false。

!~则相反。

### 选项描述

g匹配所有可能的Perl正则表达式中模式

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Perl正则表达式中模式中的空白

# 替换操作符

### 替换：s/<pattern>;/<replacement>;/

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Perl正则表达式中模式次序变量$n，如s/(\d+)/[$1]/，但在替换部分不支持Perl正则表达式中模式的特殊字符，如{},\*,+等，如s/abc/[def]/将把abc替换为[def]。

### 选项描述

g改变Perl正则表达式中模式中的所有匹配

i忽略Perl正则表达式中模式中的大小写

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Perl正则表达式中模式中的空白

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";  
$string=~s/[a-zA-Z]+/$&x2/e;#now$string="0abcabc1"

# 翻译操作符

### 转化：tr/<pattern>;/<replacemnt>;/

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/;#nowstring="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。

# 反向引用

反向引用来引用圆括号中模式匹配的文字，成为捕获组(capture group)。反向引用的写法是在反斜线后面接上数字编号，比如\1,\2。相应的数字表示对应顺序的捕获组。

圆括号包括的点号匹配任意非换行符，反向引用\1来再次匹配圆括号内的任意字符：

If (/(.)\1/){

Print “It matche same char next to itself.\n”;

}

反向引用不必紧跟在对应的捕获组括号后面。

也可以用多个括号分成多个分组，每组都有自己的反向引用。

If (/a(.)(.)\2\1/){

Print "It matche after y.\n";

}

区分那个括号是第几组，可以依次点算做括号的序号。

\111反向引用会匹配尽可能大的组号。

Perl5.10后支持\g{N}的写法。N为反向引用的组号。