

正则表达式学习笔记

阿左¹ Nobody²

December 22, 2012

¹感谢档
²感谢郭嘉

Contents

I 基本概念	1
1 元字符 (Metacharacters)	2
1.1 基本元字符	2
1.1.1 任意字符	2
1.1.2 行开始与结束	2
1.1.3 单词分界符 (Word Boundaries)	2
1.2 字符范围 (Character Classes)	3
1.3 选择结构 (Alternation)	3
1.4 注意	3
1.5 重复控制	4
1.5.1 区间量次 (Interval Quantifier)	4
1.5.2 选项元素 (Optional Items)	4
1.5.3 其他量词: 重复出现 (Other Quantifier: Repetition)	4
1.6 括号与反向引用 (Parentheses and Backreferences)	4
1.7 转义元字符	4
2 拓展	5
2.1 引用匹配的内容	5
2.2 环视功能 (lookaround)	6
2.2.1 环视只匹配位置	6
2.2.2 利用环视来查找替换	7

II 语言与工具 8

3 egrep 9

3.1 基本使用 9

3.2 忽略大小写 9

3.3 反向引用 9

4 Perl 10

4.1 元字符 10

4.1.1 空白字符 10

4.2 基本使用 10

4.2.1 变量的声明与引用 11

4.2.2 控制结构 11

4.3 用正则匹配文本 11

4.4 取得用户输入 12

4.5 引用匹配的内容 12

4.6 修饰符 13

4.6.1 忽略大小写 13

4.6.2 全局匹配 14

4.6.3 宽松排列表达式 14

4.7 替换文本 14

4.7.1 使用 perl 自动替换文本 14

4.7.2 生成邮件回复的例子 15

4.7.3 读到文件文件 16

List of Figures

List of Tables

4.1 Perl 正则元字符	10
--------------------------	----

Abstract

Regex study note

摘要

Regex study note

Part I

基本概念

Chapter 1

元字符 (Metacharacters)

1.1 基本元字符

1.1.1 任意字符

点号 “.” 匹配任意一个字符。

1.1.2 行开始与结束

脱字符与美元符分别代表行的开始与结束位置。注意这两个元字符只表示两个特殊的位置，位置上是没有字符的。

1 ^This is a line.\$

匹配空白行：

1 ^\$

匹配所有的行（因为所有的行都有一个开头）：

1 ^

1.1.3 单词分界符 (Word Boundaries)

“\<”与“\>”匹配单词（包括字母和数字）的开始与结束。注意匹配的是位置，而不是字符。

1.2 字符范围 (Character Classes)

“[...]”可以定义一个位置上可以出现的字符的范围。“<H1>”、“<H2>”、“<H3>”可以用：“<H[123]>”来表示。

“[^...]”表示排除指定字符。没有列出来的任何字符都可以。

表达式 “q[^u]” 匹配不了单词 “Iraq”，因为表达式的意义不是 “q” 后面没有 u，而是 “q” 后面要 “有” 一个字符，这个字符不能是 “u”，其他的都行。

可以用连字符来表示连续的字符：“[0-9a-zA-Z_!?.]”；只有在也只有连字符是特殊字符。后面的下划线、问号、点号等都是普通字符。

如果连字符在开头，那也表示普通字符，不表示连续字符。

```
1 echo '-123456789' | egrep '[a-b]'    # not match
2 echo '-123456789' | egrep '[-ab]'   # match
```

1.3 选择结构 (Alternation)

括号构成子表达式，“|”表示逻辑“或”。

```
1 Jeffrey|Jeffery
2 Jeff(re|er)y
```

1.4 注意

比较下面二者的区别：

```
1 [ \t]*
2 ( *|t*)
```

第一个匹配的内容要么全是空格，要么全是 TAB；第二个可以匹配空格和 TAB 混合。

以下两行是相等的（不过字符范围速度更快）：

```
1 [ \t]*
2 ( |t)*
```

1.5 重复控制

1.5.1 区间量次 (Interval Quantifier)

“{min,max}” 规定重复出现的次数：

```
1 echo '1234567890' | egrep '[0-9]{8,15}'
```

1.5.2 选项元素 (Optional Items)

“?” 相当于 “{0,1}”。“July?” 可以匹配 “Jul” 或 “July”。

1.5.3 其他量词：重复出现 (Other Quantifier: Repetition)

“*” 相当于 “{0,n}”。

“+” 相当于 “{1,n}”；

1.6 括号与反向引用 (Parentheses and Backreferences)

在很多版本的正则表达式中，括号中的子表达式能“记住”匹配的内容。“verb|num|”可以代表第几个子表达式匹配的内容。如，要查找重复的单词：

```
1 echo 'that that' | egrep '\<([A-Za-z]+) +\1\>'
```

“([a-z])([0-9])\1\2” 这个表达式中，“\1” 表示第一个表达式 “[0-9]” 匹配的内容；“\2” 表示第二个表达式 “[0-9]” 匹配的内容。

1.7 转义元字符

反斜线 “\” 实现元字符的转义。大多数正则工具会把字符范围 “[...]” 中的 “\” 作为普通字符。

Chapter 2

拓展

2.1 引用匹配的内容

在 Perl 语言中通过\$num 取得匹配的表达式内容：

```
1 if("-111.222F" =~ m/^[+-]?[0-9]+(\.[0-9]*)?)([CF])$/) {
2     print "$1\n"; # -111.222
3     print "$2\n"; # .222
4     print "$3\n"; # F
5 }
```

通过\$(?:...) 只用来分组，但是不取得匹配内容：

```
1 if("-111.222F" =~ m/^[+-]?[0-9]+(?:\.[0-9]*)?)([CF])$/) {
2     print "$1\n"; # -111.222
3     print "$2\n"; # F
4 }
```

回到温度转换的例子，根据用户输入最后是 C 还是 F 来判断输入的类型：

```
1 print "Enter a temperature in input(e.g. 32.5F, 10.0C): \n";
2
3 $input = <STDIN>;
4 chomp($input); # remove \n at end of line
5
6 if($input =~ m/^[+-]?[0-9]+(\.[0-9]*)?)([CF])$/) {
7     $number = $1;
8     $type = $3;
9     if("C" eq $type){
10         $celsius = $number;
```

```
11     $fahrenheit = ($input*9/5)+32;
12 } else {
13     $fahrenheit = $number;
14     $celsius = ($input-32)*5/9;
15 }
16 printf "%.2f C is %.2f F.\n", $celsius, $fahrenheit ;
17 } else {
18     print "Expecting a number, don't understane \"$input\".\n";
19 }
```

2.2 环视功能 (lookaround)

环视具体有以下四种：

顺序肯定环视 “(?=...)”：某个位置的右边符合子表达式。

顺序否定环视 “(?!...)”：某个位置的右边不符合子表达式。

逆序肯定环视 “(?<=...)”：某个位置的左边符合子表达式。

逆序否定环视 “(?<!=...)”：某个位置的左边不符合子表达式。

2.2.1 环视只匹配位置

环视功能只匹配位置，而不匹配具体的字符（就像是行头“^”、字符分界符“\b”）。它匹配的是某一个位置前后的内容是否符合。

例如：表达式“Jeffrey”匹配的是一串文本：

```
by Jeffrey Friedl.
  ^-----^
```

而环视“(?=Jeffrey)”匹配的的两个字符之间的位置：

```
by Jeffrey Friedl.
-><-
```

再看一个例子，“(?=Jeffery)Jeff”和“Jeff(?=rey)”是等价的：

“(?=Jeffery)Jeff”：从“Jeffery”的开头位置开始找“Jeff”。

“Jeff(?=rey)”找到后面有“rey”的“Jeff”。

2.2.2 利用环视来查找替换

把“Jeffs”替换为“Jeff's”可以有很多种实现：

不用环视：“s/Jeffs/Jeff's/g”。

用单词分界锚点：“s/\bJeffs\b/Jeff's/g”。

用分组后再替换：“s/\b(Jeff)(s)\b/\$1'\$2/g”

通过环视：“s/\bJeff(?:s\b)/Jeff'/g”

在环视的例子中，环视的内容并不在最终匹配的文本中，因为环视只匹配位置而不包括任何字符。更进一步，我们可以把前面的“Jeff”也放入环视：

s/(?<=\bJeff)(?:s\b)/'/g

这样我们只要对应的位置插入了一个字符。

以格式化数字“123456789”为“123,456,789”为例，说明环视功能。

Part II

语言与工具

Chapter 3

egrep

3.1 基本使用

在邮件中查找发信人与主题的例子：

```
1 egrep '^(From|Subject):' ./*
```

3.2 忽略大小写

```
1 egrep -i '^(From|Subject):' ./*
```

3.3 反向引用

有些版本的 egrep 有个 bug：使用“-i”忽略大小写时会对反向引用无效，即可以查到“the the”但是查不到“The the”。

```
1 echo 'the the' | egrep '\<([A-Za-z]+) +\1\>'
2 the the
3 echo 'the The' | egrep '\<([A-Za-z]+) +\1\>'
```


Chapter 4

Perl

4.1 元字符

4.1.1 空白字符

Table 4.1: Perl 正则元字符

元字符	作用
\t	制表符
\n	换行
\b	一般情况表示下单词分界，但在字符范围中表示退格。
\s	空白字符（包括空格、制表符、换行）
\S	除了“\s”以外的任何字符
\w	“[A-Za-z0-9_]”
\W	除了“\w”以外的任何字符
\d	“[0-9]”
\D	除了“\d”以外的任何字符

4.2 基本使用

查找文件中接连重复出现的单词

```
1 $/ = ".\n";
2 while (<>) {
3     next if !s/\b([a-z]+)((?:\s|<[>]+>)+)(\1\b)/e[7m$1\ne[m$2\ne[7
        m$3\ne[m/ig;
```

```
4 s/^(?:[^\s\e]*\n)+//mg; # remove the unmarked line
5 s/^\$ARGV: /mg;      # add filename before the line
6 print;
7 }
```

4.2.1 变量的声明与引用

普通变量以美元符开头，而且可以在输出语句中直接使用。

以下是一个转摄氏度为华氏度的例子：

```
1 $celsius = 30;
2 $fahrenheit = ($celsius*9/5)+32;
3 print "$celsius C is $fahrenheit F.\n";
```

4.2.2 控制结构

```
1 $celsius = 20;
2 while($celsius <= 45) {
3     $fahrenheit = ($celsius*9/5)+32;
4     print "$celsius C is $fahrenheit F.\n";
5     $celsius = $celsius + 5;
6 }
```

运行时可以通过参数“-w”打开编译警告：

```
1 perl -w exp03.pl
```

4.3 用正则匹配文本

“=~”指定正则操作的对象。

“m/.../”表示通过正则进行的操作是匹配操作（可以省略 m，但是加上看起来更加清楚）。

“==”用来比较两个数字是否相等。

“eq”用来比较两个字符串是否相等。

查找是否是数字：

```

1 if($reply =~ m/^[0-9]+$/) {
2     print "Only digits\n";
3 } else {
4     print "Not only digits\n";
5 }

```

4.4 取得用户输入

增加能够处理小数部分；并通过函数“printf”格式化输出。

```

1 print "Enter a temperature in Celsius: \n";
2
3 $celsius = <STDIN>;
4 # remove \n at end of line
5 chomp($celsius);
6
7 if($celsius =~ m/^[+-]?[0-9]+(\.[0-9]*)?$/) {
8     $fahrenheit = ($celsius*9/5)+32;
9     # format output use printf
10    printf "%.2f C is %.2f F.\n", $celsius, $fahrenheit ;
11 } else {
12    print "Expecting a number, don't understane \"$celsius\".\n";
13 }

```

4.5 引用匹配的内容

通过\$num 取得匹配的表达式内容：

```

1 if("-111.222F" =~ m/^[+-]?[0-9]+(\.[0-9]*)?)([CF])$/) {
2     print "$1\n"; # -111.222
3     print "$2\n"; # .222
4     print "$3\n"; # F
5 }

```

通过\$(?:...) 只用来分组，但是不取得匹配内容：

```

1 if("-111.222F" =~ m/^[+-]?[0-9]+(?:\.[0-9]*)?)([CF])$/) {
2     print "$1\n"; # -111.222
3     print "$2\n"; # F
4 }

```

回到温度转换的例子，根据用户输入最后是 C 还是 F 来判断输入的类型：

```
1 print "Enter a temperature in input(e.g. 32.5F, 10.0C): \n";
2
3 $input = <STDIN>;
4 chomp($input); # remove \n at end of line
5
6 if($input =~ m/^[+-]?[0-9]+(\.[0-9]*)?)([CF])$/) {
7     $number = $1;
8     $type = $3;
9     if("C" eq $type){
10         $celsius = $number;
11         $fahrenheit = ($input*9/5)+32;
12     } else {
13         $fahrenheit = $number;
14         $celsius = ($input-32)*5/9;
15     }
16     printf "%.2f C is %.2f F.\n", $celsius, $fahrenheit ;
17 } else {
18     print "Expecting a number, don't understane \"$input\".\n";
19 }
```

4.6 修饰符

4.6.1 忽略大小写

在正则以后加个修饰符 “/i” 表示忽略大小写。

```
1 $input =~ m/aaa$/i;
```

更加完整的温度转换例子：

```
1 print "Enter a temperature in input(e.g. 32.5F, 10.0C): \n";
2
3 $input = <STDIN>;
4 chomp($input); # remove \n at end of line
5
6 if($input =~ m/^[+-]?[0-9]+(\.[0-9]*)?)([CF])$/i) {
7     $number = $1;
8     $type = $3;
9     if("C" eq $type){
10         $celsius = $number;
```

```
11     $fahrenheit = ($input*9/5)+32;
12 } else {
13     $fahrenheit = $number;
14     $celsius = ($input-32)*5/9;
15 }
16 printf "%.2f C is %.2f F.\n", $celsius, $fahrenheit ;
17 } else {
18     print "Expecting a number, don't understane \"$input\".\n";
19 }
```

4.6.2 全局匹配

修饰符 “/g” 表示全局匹配。就是在完成了一次匹配以后，再继续匹配剩下的内容。

```
1 $input =~ m/^aaa$/g;
```

4.6.3 宽松排列表达式

修饰符 “/x” 表示。

```
1 $input =~ m/^aaa$/x;
```

4.7 替换文本

“\$var =~ s/regex/replacement/” 以变量 “\$var” 为对象，把符合正则的内容替换掉。

例：无视大小写，把 “peter” 替换成 “Peter”。

```
1 $var =~ s/\bpeter\b/Peter/i;
2 print "$var";
```

4.7.1 使用 perl 自动替换文本

参数 “-p” 表示对目标文件每一行进行查找和替换；参数 “-i” 表示替换的结果写回文件；参数 “-e” 表示后面的字符串就是程序的代码；

```
1 perl -p -i -e 's/sysread/read/g' filename
```

可以合并参数为：

```
1 perl -pi -e 's/sysread/read/g' filename
```

4.7.2 生成邮件回复的例子

原始内容在文件“file.in”，通过程序“mkreply.pl”把结果存放在“file.out”。

file.in

```
1 From elvis Thu Feb 29 11:15 2007
2 Received: from elvis@localhost by tabloid.org (8.11.3) id KA8CMY
3 Received: from tabloid.org by gateway.net (8.12.5/2) id N8XBK
4 To: jfried@regex.info (Jeffrey Friedl)
5 From: elvis@tabloid.org (The King)
6 Date: Thu, Feb 29 2007 11:15
7 Message-Id: <200702239939.KA8CMY@tabloid.org>
8 Subject: Be seein' ya around
9 Reply-To: elvis@hh.tabloid.org
10 X-Mailer: Madam Zelda's Psychic Orb [version 3.7 PL92]
11
12 Sorry I haven't been around lately. A few years back I checked into
13     that ole heartbreak hotel in the sky, ifyaknowwhatImean.
14 The Duke says "hi".
15         Elvis
```

希望程序能自动生成回复的样式：

```
1 To: elvis@hh.tabloid.org (The King)
2 From: jfriedl@regex.info (Jeffery Friedl)
3 Subjet: Re: Be seein' ya around
4
5 On Thu, Feb 29 2007 11:15 The King wrote:
6 I> Sorry I haven't been around lately. A few years back i checked
7 I> into the ole heartbreak hotel in the sky, ifyaknowwhatImaen.
8 I> The Duke says "hi".
9 I>         Elvis
```

调用的方法：

```
1 perl -w mkreply.pl file.in > file.out
```

4.7.3 读到文件文件

Perl 提供了操作符 “<>” 把每一行读取到变量中，我们通过 “`^\s*$`” 检查空行（表示邮件 head 结束）。

```
1 while($line = <>) {  
2     # ... deal with $line ...  
3     if($line =~ m/^\s*$/) {  
4         last; # jump out of loop  
5     }  
6 }
```

从邮件头中提取信息的方法，以 Subject 为例：

```
1 if($line =~ m/^Subject: (.*)/i) {  
2     $subject = $1;  
3 }
```

分别取得回信地址和昵称：

```
1 # From: elvis@tabloid.org (The King)  
2 if($line =~ m/^From: (\S+) \((([^\()]*)\)/i) {  
3     $reply_address = $1;  
4     $from_name = $2;  
5 }
```

就连输出原文引用的部人也可以通过正则来实现：

```
1 $line =~ s/^\|> /;  
2 print $line
```