

正则表达式（7）：扩展正则表达式

在本博客中，“正则表达式”为一系列文章，如果你想要从头学习怎样在Linux中使用正则，可以参考此系列文章，直达链接如下：

[在Linux中使用正则表达式](#)

“正则”系列的每篇文章都建立在前文的基础之上，所以，**请按照顺序阅读这些文章，否则有可能在阅读中遇到障碍。**

前文中一直在说，在Linux中，正则表达式可以分为“基本正则表达式”和“扩展正则表达式”。

我们已经认识了“基本正则表达式”，现在，我们来认识一下“扩展正则表达式”。

有了之前的基础，学习“扩展正则表达式”简直不要太轻松。

之前说过，有些符号是通用的，不管是在“基本正则表达式”中，还是在“扩展正则表达式”中，这些通用的符号所表达的意思都是相同的。

那么，我们先来看看哪些符号是通用的，看完之后你会信心大增，如下字符都是通用的：

- 1 . 表示任意单个字符。
- 2 * 表示前面的字符连续出现任意次，包括0次。
- 3 .* 表示任意长度的任意字符，与通配符中的*的意思相同。
- 4 \ 表示转义符，当与正则表达式中的符号结合时表示符号本身。
- 5 []表示匹配指定范围内的任意单个字符。
- 6 [^]表示匹配指定范围外的任意单个字符。
- 7
- 8 [:alpha:] 表示任意大小写字母。
- 9 [:lower:] 表示任意小写字母。
- 10 [:upper:] 表示任意大写字母。
- 11 [:digit:] 表示0到9之间的任意单个数字（包括0和9）。
- 12 [:alnum:] 表示任意数字或字母。
- 13 [:space:] 表示任意空白字符，包括“空格”、“tab键”等。
- 14 [:punct:] 表示任意标点符号。
- 15 [^[:alpha:]] 表示单个非字母字符。
- 16 [^[:lower:]] 表示单个非小写字母字符。
- 17 [^[:upper:]] 表示单个非大写字母字符。
- 18 [^[:digit:]] 表示单个非数字字符。
- 19 [^[:alnum:]] 表示单个非数字非字母字符。
- 20 [^[:space:]] 表示单个非空白字符。
- 21 [^[:punct:]] 表示单个非标点符号字符。
- 22 [0-9]与[:digit:]等效。
- 23 [a-z]与[:lower:]等效。
- 24 [A-Z]与[:upper:]等效。
- 25 [a-zA-Z]与[:alpha:]等效。
- 26 [a-zA-Z0-9]与[:alnum:]等效。
- 27 [^0-9]与[^[:digit:]]等效。
- 28 [^a-z]与[^[:lower:]]等效。
- 29 [^A-Z]与[^[:upper:]]等效
- 30 [^a-zA-Z]与[^[:alpha:]]等效
- 31 [^a-zA-Z0-9]与[^[:alnum:]]等效
- 32
- 33 ^：表示锚定行首，此字符后面的任意内容必须出现在行首，才能匹配。
- 34 \$：表示锚定行尾，此字符前面的任意内容必须出现在行尾，才能匹配。
- 35 ^\$：表示匹配空行，这里所描述的空行表示“回车”，而“空格”或“tab”等都不能算作此处所描述的空行。
- 36 ^abc\$：表示abc独占一行时，会被匹配到。
- 37 \<或者\b：匹配单词边界，表示锚定词首，其后面的字符必须作为单词首部出现。
- 38 \>或者\b：匹配单词边界，表示锚定词尾，其前面的字符必须作为单词尾部出现。
- 39 \B：匹配非单词边界，与\b正好相反。

上述符号，在基本正则表达式中与扩展正则表达式中的用法完全相同。

有没有感觉，70%都是通用的，那么我们来动手试试。

在总结grep命令时，我们提到过，grep命令默认只支持基本正则表达式，如果想要让grep命令能够支持扩展的正则表达式，则需要使用"-E"选项，示例如下

```
[www.zsythink.net]#cat reg3
c3
cB
cC
cd
cE
c$
c#
[www.zsythink.net]#grep -E "c[A-Z]" reg3
cB
cC
cE
[www.zsythink.net]#
```

上图中，grep命令使用了"-E"选项，表示grep命令会把"正则表达式"中的符号当成"扩展正则表达式"去理解，而不再使用默认的"基本正则表达式"。但是由于"[A-Z]"是通用的，所以，不管是否使用扩展正则表达式，"[A-Z]"都表示单个大写字母。

刚才说过，70%的符号都是通用的，那么剩下的30%呢？

其实，剩下的30%也都差不多，与基本正则表达式相比，反而更加简单了，不信？我们就来看看。

在基本正则表达式中，`\{n\}` 表示前面的字符连续出现n次，将会被匹配到。

在扩展正则表达式中，`{n}` 表示前面的字符连续出现n次，将会被匹配到。

在基本正则表达式中，`\(\)` 表示分组，`\(ab\)` 表示将ab当做一个整体去处理。

在扩展正则表达式中，`()` 表示分组，`(ab)` 表示将ab当做一个整体去处理。

在写法上，"扩展正则表达式"的写法是不是更加简练呢？示例如下

```
[www.zsythink.net]#cat reg6
hello
helloo
hellooo
hellohello
[www.zsythink.net]#
[www.zsythink.net]#grep "\(hello\)\{2\}" reg6
hellohello
[www.zsythink.net]#
[www.zsythink.net]#grep -E "(hello){2}" reg6
hellohello
[www.zsythink.net]#
```

如上图所示，当使用"扩展正则表达式"时，在"书写"方面，反而省力不少，最终匹配到的文本却是相同的，是不是很方便？

看完了上述示例，我想你对扩展正则表达式应该已经有了一个初步的印象了。

那么，我们就来介绍一下，有哪些符号在"扩展正则表达式"中变得更加简练了。

在扩展正则表达式中：

`()` 表示分组

`(ab)` 表示将ab当做一个整体去处理。

`\1` 表示引用整个表达式中第1个分组中的正则匹配到的结果。

`\2` 表示引用整个表达式中第2个分组中的正则匹配到的结果。

`?` 表示匹配其前面的字符0或1次

`+` 表示匹配其前面的字符至少1次，或者连续多次，连续次数上不封顶。

`{n}` 表示前面的字符连续出现n次，将会被匹配到。

`{x,y}` 表示之前的字符至少连续出现x次，最多连续出现y次，都能被匹配到，换句话说，只要之前的字符连续出现的次数在x与y之间，即可被匹配到。

`{,n}` 表示之前的字符连续出现至多n次，最少0次，都会匹配到。

`{n,}`表示之前的字符连续出现至少n次，才会被匹配到。

看了上述总结以后，是不是已经想要放弃使用"基本正则表达式"了呢？因为与之相比，扩展正则表达式才更符合我们这些懒人的习惯，而且，扩展正则表达式的可读性就变强了。

扩展正则表达式中，还有一个常用的符号，是基本正则表达式中所没有的，它就是"`|`"

注：按住键盘的"shift键"和"\键"，就可以打出"`|`"

"|"在扩展正则表达式中，表示"或"，这样说不容易理解，我们来看个小例子，就能明白，示例文件内容如下。

```
[www.zsythink.net]#cat youxiang
zsythink@zsythink.net
l@2
zhushuangyin@zsythink.net
aaa.org
testregex@163.com
testregex@163zsy.net
testregex@zsythink.org
testregex@zsythink.edu
testregex@zsythink.ttt
a@1.com
testregex@163.cccom
```

zsythink.net 朱双印博客

如果，我们想要从上例文本中找到以"com"结尾的行，我们该怎么办呢？我们可以使用如下命令。

```
[www.zsythink.net]#grep -E "com$" youxiang
testregex@163.com
a@1.com
testregex@163.cccom
[www.zsythink.net]#
```

zsythink.net 朱双印博客

同理，如果我们想要从示例文本中找出以"net"结尾的行，可以使用如下命令。

```
[www.zsythink.net]#grep -E "net$" youxiang
zsythink@zsythink.net
zhushuangyin@zsythink.net
testregex@163zsy.net
[www.zsythink.net]#
```

zsythink.net 朱双印博客

那么，如果我们想要从示例文本找出，以"com"结尾，或者以"net"结尾的行，我们该怎么办呢？

这时候，我们就需要用到"|"

"|"在扩展正则表达式中表示"或者"，所以，我们可以使用如下表达式

```
[www.zsythink.net]#grep -E "(com|net)$" youxiang
zsythink@zsythink.net
zhushuangyin@zsythink.net
testregex@163.com
testregex@163zsy.net
a@1.com
testregex@163.cccom
[www.zsythink.net]#
```

zsythink.net 朱双印博客

上图中的扩展正则使用了分组符号"()"，"(com|net)"表示将括号内的内容看做一个整体，而括号内的内容为"com|net"，它表示"com或者net"，所以，"(com|net)om或者net"结尾的行。是不是很简单？

那么，我们就趁热打铁，通过实际练习，来熟悉一下"扩展正则表达式"吧。

仍然以刚才的示例文件作为测试文件，假设，我们想要查找出测试文本中的"合法邮箱"，我们应该怎么做呢？

既然是要找出"合法邮箱"，那么，我们则必须事先定义，满足哪些条件的邮箱才属于合法邮箱。

所以，我们规定，如果一个邮箱属于合法邮箱，那么必须满足如下条件。

- 1、邮箱字符串中必须包含"@"符。
- 2、"@"符前面的字符只能是小写字母或数字，不能包含特殊符号。
- 3、"@"符前面的字符数量至少需要4个，至多为16个。
- 4、邮箱必须以"com"、"net"、"org"、"edu"等顶级域名结尾（此处为了方便演示，不判断更多的域名）。
- 5、顶级域名之前必须包含一个"点"，换句话说就是，邮箱必须以".com"、".net"、".org"、".edu"结尾。
- 6、"@"与"."之间的字符数量不能超过12个，不能低于2个。
- 7、"@"与"."之间的字符只能是小写字母或数字，不能包含特殊符号。

好了，了解了合法邮箱的规则以后，我们就可以开始编写正则表达式了，我们可以使用如下正则，查找文本中的合法邮箱。

```
[www.zsythink.net]#grep --color -E "[a-z0-9]{4,16}@[a-z0-9]{2,12}\.(\<(com|net|org|edu))$" youxiar
zsythink@zsythink.net
zhushuangyin@zsythink.net
testregex@163.com
testregex@163zsy.net
testregex@zsythink.org
testregex@zsythink.edu
[www.zsythink.net]#
```

zsythink.net 朱双印博客

如果你觉得上述正则稍微有些复杂，不容易理解，那么可以将其拆分成几个部分去理解，拆分后的每一部分，可以与之前的"合法邮箱条件"——对应，如下图所示。

`grep --color -E "[a-z0-9]{4,16}@[a-z0-9]{2,12}\.(\<(com|net|org|edu))$" youxiang`

条件2 条件3 条件1 条件7 条件6 条件5 条件4

zsythink.net 朱双印博客

这样看，是不是容易理解多了，好了，赶快自己动手实验一下吧。

小结

我认为，有了之前基础的你，搞定"扩展正则表达式"，肯定是分分钟的事情，所以，我们就对"扩展正则表达式"进行一下总结吧。

- 1 常用符号
- 2 . 表示任意单个字符。
- 3 * 表示前面的字符连续出现任意次，包括0次。
- 4 .* 表示任意长度的任意字符，与通配符中的*的意思相同。
- 5 \ 表示转义符，当与正则表达式中的符号结合时表示符号本身。
- 6 | 表示"或者"之意
- 7 []表示匹配指定范围内的任意单个字符。
- 8 [^]表示匹配指定范围外的任意单个字符。
- 9
- 10 单个字符匹配相关
- 11 [[:alpha:]] 表示任意大小写字母。
- 12 [[:lower:]] 表示任意小写字母。
- 13 [[:upper:]] 表示任意大写字母。
- 14 [[:digit:]] 表示0到9之间的任意单个数字（包括0和9）。
- 15 [[:alnum:]] 表示任意数字或字母。
- 16 [[:space:]] 表示任意空白字符，包括"空格"、"tab键"等。
- 17 [[:punct:]] 表示任意标点符号。
- 18 [^[[:alpha:]] 表示单个非字母字符。
- 19 [^[[:lower:]] 表示单个非小写字母字符。
- 20 [^[[:upper:]] 表示单个非大写字母字符。
- 21 [^[[:digit:]] 表示单个非数字字符。
- 22 [^[[:alnum:]] 表示单个非数字非字母字符。
- 23 [^[[:space:]] 表示单个非空白字符。
- 24 [^[[:punct:]] 表示单个非标点符号字符。
- 25 [0-9]与[[:digit:]]等效。
- 26 [a-z]与[[:lower:]]等效。
- 27 [A-Z]与[[:upper:]]等效。
- 28 [a-zA-Z]与[[:alpha:]]等效。
- 29 [a-zA-Z0-9]与[[:alnum:]]等效。
- 30 [^0-9]与[^[[:digit:]]等效。
- 31 [^a-z]与[^[[:lower:]]等效。
- 32 [^A-Z]与[^[[:upper:]]等效
- 33 [^a-zA-Z]与[^[[:alpha:]]等效
- 34 [^a-zA-Z0-9]与[^[[:alnum:]]等效
- 35
- 36 次数匹配相关
- 37 ? 表示匹配其前面的字符0或1次
- 38 + 表示匹配其前面的字符至少1次，或者连续多次，连续次数上不封顶。
- 39 {n} 表示前面的字符连续出现n次，将会被匹配到。
- 40 {x,y} 表示之前的字符至少连续出现x次，最多连续出现y次，都能被匹配到，换句话说，只要之前的字符连续出现的次数在x与y之间，即可被匹配到。
- 41 {,n} 表示之前的字符连续出现至多n次，最少0次，都会匹配到。
- 42 {n,} 表示之前的字符连续出现至少n次，才会被匹配到。
- 43
- 44 位置边界匹配相关
- 45 ^：表示锚定行首，此字符后面的任意内容必须出现在行首，才能匹配。

