

## 正则表达式 ( 1 ) : 入门

从这篇文章开始, 我们将介绍怎样在Linux中使用"正则表达式", 如果你想要学习怎样在Linux中使用正则表达式, 这些文章就是你所需要的。

在认识"正则表达式"之前, 请先阅读如下两篇文章, 如下两篇文章是学习"正则表达式"的基础。

[扫盲: 什么是正则表达式](#)

[详解grep命令](#)

阅读完上述两篇文章以后, 你肯定会明白, grep命令是支持正则表达式的。

所以, 我们可以通过grep命令学习正则表达式(下文中简称为"正则")。

当grep与正则结合在一起时, grep就会根据"正则的含义"在文本中查找符合条件的字符串。

什么是正则? 什么是grep? 前文介绍过了, 我们就不再废话了, 直接切入正题

我们通过grep命令来实践一下正则, 仍然以前文中提到的例子作为切入点, 示例如下:

```
[www.zsythink.net]#cat regex
hello world
hi      hello
hello ,zsy
```

我们在系统中创建了一个文件, 用于测试正则, 文件名为regex

如上图所示, 文件中有三行文本, 每行都包含"hello"这个单词, 如果我们想要利用grep在此文本中搜索包含"hello"的行, 则可以使用如下命令。

```
[www.zsythink.net]#grep "hello" regex
hello world
hi      hello
hello ,zsy
[www.zsythink.net]#
```

没错, 由于regex文件中的每一行都包含hello, 所以, 所有行都被打印出来了。

如果, 我们只想要打印出"以hello开头的行" (hello位于行首的行), 该怎么办呢? 没错, 使用正则表达式即可, 示例如下

```
[www.zsythink.net]#grep "^hello" regex
hello world
hello ,zsy
[www.zsythink.net]#
```

前文中已经介绍过, 在正则表达式中, "^"表示"锚定行首" (符号"^"是数字键6对应的符号), 所以"^hello"表示只匹配位于行首的hello字符串。

由于regex文本中的第二行的hello位于行尾, 所以, 第二行并不符合条件, 于是, 只有第一行与第三行被打印了出来。

"^"在正则中表示锚定行首, 那么, 什么符号在正则中表示锚定行尾呢?

"\$"在正则中表示锚定行尾, 符号"\$"为数字键4对应的符号, 那么我们来看看怎样锚定行尾, 仍然以之前的regex文本为例, regex文本中的第二行中, 单词hello位

所以, 我们可以使用"hello\$", 去匹配"位于行尾的hello"字符串, 示例如下:

```
[www.zsythink.net]#grep "hello$" regex
hi hello
[www.zsythink.net]#
```

如上所示, "hello\$"表示匹配位于行尾的hello字符串, 只有第二行满足条件, 所以, 只有第二行被输出了。

我们已经学会了"^"与"\$", 我们知道, 它们在正则表达式中分别代表锚定行首与锚定行尾, 那么, 我们将它们结合在一起使用, "^hello\$"表示什么意思呢? 聪明如了, "^hello\$"表示hello既位于行首, 同时也位于行尾, 换句话说, 就是整行中只有一个单词hello, 没有其他单词, 那么是这样吗, 我们来试一试, 在regex中新加hello一个单词, 如下。

```
[www.zsythink.net]#cat -n regex
1 hello world
2 hi hello
3 hello ,zsy
4 hello
```

现在, 我们使用正则表达式"^hello\$", 看看能不能匹配到文本中的第四行。

```
[www.zsythink.net]#grep -n --color "^hello$" regex
4:hello
[www.zsythink.net]#
```

如上图所示, 我们成功匹配到了regex文本中的第四行, 并且将第四行打印了出来。

看到这里, 我想你应该已经学会举一反三了, "^hello\$"表示hello即位于行首也位于行尾, 那么"^\$"表示什么意思呢? 没错, "^\$"表示行首与行尾相连, 换句话说"行", 我们在regex中添加一行"空行", 看看能不能匹配到, 示例如下:

```
[www.zsythink.net]#cat -n regex
1  hello world
2  hi hello
3
4  hello ,zsy
5  hell
```

我们直接在第二行后按回车键，于是第三行变成了“空行”，注意，“空行”表示当前行不包含任何字符，包含“空格”的行不能被当做“空行”。

现在，我们来使用正则表达式“^\$”，试试能不能匹配到文本中的第三行，如下图所示。

```
[www.zsythink.net]#grep "^$" regex

[www.zsythink.net]#grep -n "^$" regex
3:
[www.zsythink.net]#
```

可以看到，文本中的“空行”被匹配到了。

现在，我们已经能够灵活的锚定“行首”与“行尾”了，那么，我们能不能锚定“词首”或“词尾”呢？

必须能啊，正则表达式中，“\<”表示锚定词首，“\>”表示锚定词尾。

为了方便示例，我们再准备另外一个测试文件REG，文件内容如下。

```
[www.zsythink.net]#cat REG
abchello world
abc helloabc abc
abc abchelloabc abc
[www.zsythink.net]#
```

上图中，“abchello”中包含“hello”，但是“hello”位于“abchello”这个单词的词尾，同理，“helloabc”中也包含“hello”，但是“hello”位于“helloabc”这个单词的词首。刚才提到过，正则表达式中，“\<”表示锚定词首，“\>”表示锚定词尾，现在我们就来实验一下。

```
[www.zsythink.net]#cat -n REG
1  abchello world
2  abc helloabc abc
3  abc abchelloabc abc
[www.zsythink.net]#
[www.zsythink.net]#grep --color "\<hello" REG
abc helloabc abc
[www.zsythink.net]#grep --color "hello\>" REG
abchello world
[www.zsythink.net]#
```

如上图所示，“\<hello”表示以hello作为词首的单词将会被匹配到，“hello\>”表示以hello作为词尾的单词将会被匹配到。

同理，我们也可以将“\<”与“\>”结合在一起使用，示例如下。

为了测试，我们在REG文件中又添加了一行，内容如下

```
[www.zsythink.net]#cat -n REG
1  abchello world
2  abc helloabc abc
3  abc abchelloabc abc
4  abchello helloabc hello ahelloa
[www.zsythink.net]#
[www.zsythink.net]#grep --color "\<hello\>" REG
abchello helloabc hello ahelloa
[www.zsythink.net]#
```

上图中，“\<hello\>”表示当hello既是词首又是词尾时则会被匹配到，换句话说，就是当hello作为一个独立的单词时，则会被匹配到，如上图所示，REG文本中第4行了，因为只有第4行中才包含了一个独立的hello单词。

其实，正则表达式中，除了“\<”与“\>”能够表示锚定词首与锚定词尾以外，我们还可以使用“\b”去代替“\<”和“\>”，“\b”既能锚定词首，也能锚定词尾，示例如下。

```
[www.zsythink.net]#cat -n REG
1  abchello world
2  abc helloabc abc
3  abc abchelloabc abc
4  abchello helloabc hello ahelloa
[www.zsythink.net]#
[www.zsythink.net]#grep --color "\bhello" REG
abc helloabc abc
abchello helloabc hello ahelloa
[www.zsythink.net]#
[www.zsythink.net]#grep --color "hello\b" REG
abchello world
abchello helloabc hello ahelloa
[www.zsythink.net]#
[www.zsythink.net]#grep --color "\bhello\b" REG
abchello helloabc hello ahelloa
[www.zsythink.net]#
[www.zsythink.net]#
```

zsythink.net 朱双印博客

聪明如你，只要懂得了"\<"与"\>"，再结合上述示例理解"\b"，绝对不是事儿。

"\b"还有一个孪生兄弟，"\B"，虽然它们长得很像，但是它们的功能完全不一样。

"\b"是用来锚定词首、锚定词尾的，换句话说，"\b"是用来匹配"单词边界"的，而"\B"则正好相反。

"\B"是用来匹配"非单词边界"的，这样说并不容易理解，看了示例就会秒懂，示例如下。

```
[www.zsythink.net]#cat REG
abchello world
abc helloabc abc
abc abchelloabc abc
abchello helloabc hello ahelloa
[www.zsythink.net]#
[www.zsythink.net]#grep --color "\Bhello" REG
abchello world
abc abchelloabc abc
abchello helloabc hello ahelloa
[www.zsythink.net]#
```

zsythink.net 朱双印博客

上例中的"\Bhello"表示，只要hello不是词首，就会被匹配到，如上图所示。

而"\bhello"表示，只要hello是词首，就会被匹配到，所以，"\B"与"\b"所要表达的意思正好相反。

"hello\b"与"hello\B"同理，此处不再赘述，快动手试试吧。

在正则表达式中，又有"基础正则表达式"和"扩展正则表达式"之分（此处不用纠结，后面会专门对扩展正则表达式进行总结，我们现在所展示的都是基本正则表达式正则表达式"，再看"扩展正则表达式"，绝对分分钟搞定）。

有些符号在基础正则表达式中和扩展正则表达式中是通用的，有些则不然。

比如我们今天学习到的这些符号，就是通用的，不管是在基础正则还是扩展正则中，它们表示的含义都是相同的。

细心如你一定发现了，今天所使用的正则表达式都与"位置"有关，比如"行首"、"行尾"、"词首"、"词尾"等，我们可以把这些符号理解为与"位置匹配"有关的正则表。我们今天所认识的符号只是正则表达式中的一部分，之后的文章我们会继续总结正则表达式，只要坚持看完它们，你肯定会掌握正则表达式的。

## 小结

为了方便以后回顾。我们一起来总结一下上文中提到过的这些符号。

^：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。

\$：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。

^\$：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。

^abc\$：表示abc独占一行时，会被匹配到。

\<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。

\>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。


\B：匹配非单词边界，与\b正好相反。

这篇文章中所涉及到的只是正则表达式中的一部分。

本博客会对正则表达式进行系统的总结，直达链接如下：

[正则表达式详解系列](#)

不知道这些文章能不能对你有所帮助？  
如果能够帮到你，希望你能够给我一些回应（比如留言、评论、点赞）。  
让我确定写这些文章是有价值的，我会继续写下去。



**我的微信公众号**

关注"实用运维笔记"微信公众号，当博客中有新文章时，可第一时间得知哦~