A+

# 正则表达式（3）：常用符号

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。

之前已经总结了怎样利用正则表达式去"匹配位置"或者"匹配连续次数"，此处，我们来总结一下正则中其他的一些常用符号。

在开始学习新知识之前，我们先回顾一下之前使用过的一个符号，它就是"."

之前说过，在正则表达式中，"."表示匹配任意单个字符（换行符除外），示例如下。

```
[www.zsythink.net]#cat reg1
a
a1
a6d
a89&
a7idai8
[www.zsythink.net]#grep --color "a..." reg1
a89&
a7idai8
[www.zsythink.net]#
```



示例中的正则表示，只要a字母后面跟随任意3个字符，即可被正则表达式匹配到。

正如上图所示，字母a后面跟随的3个字符可以是"数字"，或者是"字母"，再或者是"符号"，都可以，因为"."表示任意单个字符，"任意"就体现在这里了。

如果我们想要更加"细致"一些呢？

比如，我们仍然想要从文本中找出a字母后面跟随3个字符的字符串，但是，我们对后面跟随的3个字符有要求，并不能是任意3个字符，而必须是三个字母，我们该怎

没错，这个问题会引出我们将要认识的新符号，它就是 "[[:alpha:]]"

在正则表达式中，[[:alpha:]] 表示"任意字母"（不区分大小写）

[[:alpha:]] 这个符号看上去略微有点复杂，但是不要害怕，习惯了就好，其实，"[[:alpha:]]" 可以拆开成几部分去理解，我们后面再聊。

我们先来实验一下，示例如下。

```
[www.zsythink.net]#cat reg1
a
a6d
a89&
a7idai8
abcd
aBdC
aBCD
a123
a1a3
a&@%
[www.zsythink.net]#
[www.zsythink.net]#grep --color "a[[:alpha:]]\{3\}" reg1
abcd
aBdC
aBCD
[www.zsythink.net]#
```



上例中，"[[:alpha:]]\{3\}"表示3个连续的任意字母，此处结合了之前的知识，其中"\{3\}" 表示其前面的字符连续出现3次（如果你没有看懂，请回顾前文），所以，式整体的含义就是， 只有a字母后面跟随了3个字母的字符串才会被匹配到，如果a字母后面跟随的3个字符中包含非字母（数字或符号），就不会被匹配到，正如上

所以，使用[[:alpha:]]可以匹配到不区分大小写的字母，没错，alpha的读音你应该很熟悉了，就是"阿尔法狗"的"阿尔法"。

那么，我们再"细化"一点，我们不仅要字母a后面跟随的3个字符是字母，我们还要求，这3个字符必须是小写字母，我们该怎么办呢？

我们可以使用另外一个符号，它就是"[[:lower:]]"

[[:lower:]]表示任意小写字母，我们来试试。

```
[www.zsythink.net]#cat reg1
a
a6d
a89&
a7idai8
abcd
aBdC
aBCD
a123
a1a3
a&@%
[www.zsythink.net]#
[www.zsythink.net]#grep --color "a[[:lower:]]\{3\}" reg1
abcd
[www.zsythink.net]#
```



可以看到，只有当a后面的3个字符均为小写字母时，才会被匹配到。

我们已经学会了怎样表示"不区分大小写的字母"和"小写字母"，那么怎样表示"大写字母"呢？

我们可以使用[[:upper:]]表示任意大写字母，示例如下。

```
[www.zsythink.net]#cat reg1
a
a6d
a89&
a7idai8
abcd
aBdC
aBCD
a123
a1a3
a&@%
[www.zsythink.net]#
[www.zsythink.net]#grep --color "a[[:upper:]]\{3\}" reg1
aBCD
[www.zsythink.net]#
```



聪明如你，一定已经发现了一些规律，规律就是，我们替换"[[: :]]"中的单词，即可表示不同的含义。

那么我们来看看一些常用的符号都表示什么含义。

[[:alpha:]] 表示任意大小写字母

[[:lower:]] 表示任意小写字母

[[:upper:]] 表示任意大写字母

[[:digit:]] 表示0到9之间的任意单个数字（包括0和9）

[[:alnum:]] 表示任意数字或字母

[[:space:]] 表示任意空白字符，包括"空格"、"tab键"等。

[[:punct:]] 表示任意标点符号

好了，了解了上述符号的含义后，你可以自己创建一个测试文件，进行测试，快点动手试试吧，我相信亲自实验获得的理解肯定更加深刻。

之前，我们使用"[[:lower:]]"表示任意一个小写字母，其实，还有另外一种方法，也能够表示"任意单个小写字母"。

除了"[[:lower:]]"，"[a-z]"也能表示任意一个小写字母，你一定猜出来了，没错，"[a-z]"所表示的意思就是，从a到z的26个小写英文字母中的任意一个。

所以，[a-z]与[[:lower:]]是等价的。

同理，[A-Z]也能表示任意一个大写字母，[A-Z]与[[:upper:]]是等价的，示例如下。

```
[www.zsythink.net]#cat reg1
a
a6d
a89&
a7idai8
abcd
aBdC
aBCD
a123
a1a3
a&@%
[www.zsythink.net]#
[www.zsythink.net]#grep --color "a[[:lower:]]\{3\}" reg1
abcd
[www.zsythink.net]#grep --color "a[a-z]\{3\}" reg1
abcd
[www.zsythink.net]#grep --color "a[[:upper:]]\{3\}" reg1
aBCD
[www.zsythink.net]#grep --color "a[A-Z]\{3\}" reg1
aBCD
[www.zsythink.net]#
```



有了之前的基础，你猜猜，"[a-zA-Z]"表示什么意思？

没错，"[a-zA-Z]"表示任意字母，不区分大小写。

[a-zA-Z]与[[:alpha:]]等效。

同理，[0-9]与[[:digit:]]等效，都表示0到9之间的任意单个数字，示例如下。

```
[www.zsythink.net]#cat reg1
a
a6d
a89&
a7idai8
abcd
aBdC
aBCD
a123
a1a3
a&@%
[www.zsythink.net]#grep --color "a[0-9]\{3\}" reg1
a123
[www.zsythink.net]#
```



我们已经了解到，[a-z]表示任意一个小写字母，其实，"[a-z]"外侧的方括号有特殊的含义。

方括号在正则中代表什么意思呢？

"[ ]"表示匹配指定范围内的任意单个字符，这样说可能不容易理解，我们来动手实验一下，就能秒懂，示例如下。

```
[www.zsythink.net]#cat reg2
bc
bd
be
bf
bg
[www.zsythink.net]#
[www.zsythink.net]#grep --color "b[ceg]" reg2
bc
be
bg
[www.zsythink.net]#
```



可以看到，字母b后面跟随字母c、或者跟随字母e、或者跟随字母g，都可以被匹配到，"[ceg]"表示c或者e或者g中的任何一个字母都能被匹配到。

那么活学活用，"[Bd#3]"表示什么意思呢？

[Bd#3]表示字符是大写B、或者是小写d、或者是符号#、再或者是数字3，都可以被匹配到，示例如下

```
[www.zsythink.net]#cat reg3
c3
cB
cC
cd
cE
c$
c#
[www.zsythink.net]#
[www.zsythink.net]#grep --color "c[Bd#3]" reg3
c3
cB
cd
c#
[www.zsythink.net]#
```



"[ ]"表示匹配指定范围内的任意单个字符，换句话说，就是字符与方括号"[ ]"内的任意一个字符相同，就可以被匹配到。

我们了解了方括号的含义以后，再回过头看之前的符号，会有新发现。

上文说过，[0-9]表示0到9之间的任意一个数字，其实，[0-9]就相当于[0123456789]

同理，[a-z]表示a到z之间的任意一个字母，其实，[a-z]就相当于[abcdefghijklmnopqrstuvwxyz]

之前说过，[[:alpha:]]代表单个任意的字母，前文也提到过，[[:alpha:]]可以拆开去理解，聪颖如你一定想到了，我们可以把[[:alpha:]]拆成两部分理解。

第一部分：最外层的[ ]，表示指定范围内的任意单个字符

第二部分：最内层的[:alpha:]，表示不区分大小写的字母

所以，当两部分结合在一起时，就变成了[[:alpha:]]，就表示任意单个字母（不区分大小写），[[:digit:]]等其他类似符号也可以这样拆开来理解。

我们已经理解了方括号"[ ]"的含义，我们再来认识一个它的孪生兄弟，它就是 "[^ ]"

"[^ ]"表示匹配指定范围外的任意单个字符，注意，它与"[ ]"的含义正好相反。

"[ ]"表示匹配指定范围内的任意单个字符。

如果你觉得不好理解，可以先看示例，示例如下：

```
[www.zsythink.net]#cat reg4
fa
fb
fc
fd
fe
ff
fg
[www.zsythink.net]#
[www.zsythink.net]#grep --color "f[^aceg]" reg4
fb
fd
ff
[www.zsythink.net]#
```



如上图所示，字母f后面跟随的字母只要不是a、c、e、g中的任何一个，即可被匹配到，相当于排除了a、c、e、g这些字母。

所以，"[^ ]"表示匹配指定范围外的任意单个字符

我们之前说过，"^"符号的含义为锚定行首，但是，当它与"[ ]"结合在一起的时候，则没有锚定行首之意，只能把"[^ ]"当做一个整体去看待，可以把此处的 "^"理解

既然"[ ]"与"[^ ]"是相对的，那么，能不能把[0-9]改写成[^0-9]呢？必须能啊。

"[^0-9]"表示匹配单个非数字字符，与[0-9]的含义这正好相反，示例如下。

```
[www.zsythink.net]#cat reg5
e1
ea
e7
eY
e$
e8
[www.zsythink.net]#grep --color "e[^0-9]" reg5
ea
eY
e$
[www.zsythink.net]#
```



如上图所示，只要字母e后面跟随的字符不是数字，就可以被匹配到。

同理：

[^a-z]表示非小写字母的单个字符可以被匹配到。

[^A-Z]表示非大写字母的单个字符可以被匹配到。

[^a-zA-Z]表示非字母的单个字符可以被匹配到,比如数字或符号。

[^a-zA-Z0-9]表示非字母、非数字的单个字符可以被匹配到，比如符号。

结合之前的理论，你一定想到了，既然[0-9]与[[:digit:]]等效，那么[^0-9]与[^[:digit:]]等效吗？

试试就知道了，如下图所示，的确是等效的。

```
[www.zsythink.net]#cat reg5
e1
ea
e7
eY
e$
e8
[www.zsythink.net]#grep --color "e[^0-9]" reg5
ea
eY
e$
[www.zsythink.net]#grep --color "e[^[:digit:]]" reg5
ea
eY
e$
[www.zsythink.net]#
```



举一反三

[^0-9]与[^[:digit:]]等效

[^a-z]与[^[:lower:]]等效

[^A-Z]与[^[:upper:]]等效

[^a-zA-Z]与[^[:alpha:]]等效

[^a-zA-Z0-9]与[^[:alnum:]]等效

其实，不仅[0-9]与[[:digit:]]能够表示数字，还有一些简写格式的符号也能表示数字，比如"\d"

**但是**，并不是所有的正则表达式处理器都能够识别这些简写格式

示例如下

```
[www.zsythink.net]#cat reg5
e1
ea
e7
eY
e$
e8
e8
[www.zsythink.net]#grep --color "e[0-9]" reg5
e1
e7
e8
e8
[www.zsythink.net]#grep --color "e[[:digit:]]" reg5
e1
e7
e8
e8
[www.zsythink.net]#grep --color "e\d" reg5
[www.zsythink.net]#
[www.zsythink.net]#
```



如上图所示，默认情况下，grep就无法识别"\d"这种简短格式，所以上图中，没有匹配到任何结果。

如果我们想要让grep能够识别这种简短格式，可以使用-P选项，表示grep使用兼容perl的正则表达式引擎，示例如下。

```
[www.zsythink.net]#cat reg5
e1
ea
e7
eY
e$
e8
e8
[www.zsythink.net]#
[www.zsythink.net]#grep --color "e\d" reg5
[www.zsythink.net]#
[www.zsythink.net]#grep -P --color "e\d" reg5
e1
e7
e8
e8
[www.zsythink.net]#
```



我想，有了前面的基础，再理解这些简写格式，应该相对容易了。

所以，此处直接列出一些常用的简写格式的符号，不再赘述了，大家可以动手实验一下。

\d 表示任意单个0到9的数字

\D 表示任意单个非数字字符

\t 表示匹配单个横向制表符（相当于一个tab键）

\s表示匹配单个空白字符，包括"空格"，"tab制表符"等。

\S表示匹配单个非空白字符

## 小结

为了方便以后回顾，我们将上述知识点总结一下。

如果你不明白下述描述，请回顾上述示例。

```
. 表示匹配任意单个字符
* 表示匹配前面的字符任意次，包括0次

[ ] 表示匹配指定范围内的任意单个字符
[^ ] 表示匹配指定范围外的任意单个字符

[[:alpha:]] 表示任意大小写字母
[[:lower:]] 表示任意小写字母
[[:upper:]] 表示任意大写字母
[[:digit:]] 表示0到9之间的任意单个数字（包括0和9）
[[:alnum:]] 表示任意数字或字母
[[:space:]] 表示任意空白字符，包括"空格"、"tab键"等。
[[:punct:]] 表示任意标点符号

[0-9]与[[:digit:]]等效
[a-z]与[[:lower:]]等效
[A-Z]与[[:upper:]]等效
[a-zA-Z]与[[:alpha:]]等效
[a-zA-Z0-9]与[[:alnum:]]等效

[^0-9]与[^[:digit:]]等效
[^a-z]与[^[:lower:]]等效
[^A-Z]与[^[:upper:]]等效
[^a-zA-Z]与[^[:alpha:]]等效
[^a-zA-Z0-9]与[^[:alnum:]]等效

#简短格式并非所有正则表达式解析器都可以识别
\d 表示任意单个0到9的数字
\D 表示任意单个非数字字符
```

```
\t 表示匹配单个横向制表符（相当于一个tab键）
\s表示匹配单个空白字符，包括"空格"，"tab制表符"等
\S表示匹配单个非空白字符
```

这篇文章就总结到这里，希望能够帮助到你~~

**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~