A+

正则表达式（6）：基本正则表达式小结

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。

写这篇文章的目的就是总结前文中所介绍的"基本正则表达式"，并且结合一些实例进行练习，以便我们能够在练习中完全掌握它们。

首先，我们对前文中提到的符号进行总结，总结如下

```
##################常用符号#################
.  表示任意单个字符。
*  表示前面的字符连续出现任意次，包括0次。
.* 表示任意长度的任意字符，与通配符中的*的意思相同。
\  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。
[ ]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。

##################单个字符匹配相关#################
[[: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。
[^[:alpha:]] 表示单个非字母字符。
[^[:lower:]] 表示单个非小写字母字符。
[^[:upper:]] 表示单个非大写字母字符。
[^[:digit:]] 表示单个非数字字符。
[^[:alnum:]] 表示单个非数字非字母字符。
[^[:space:]] 表示单个非空白字符。
[^[:punct:]] 表示单个非标点符号字符。
[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。

##################次数匹配相关#################
\?  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，都会陪匹配到。
\{n,\}表示之前的字符连续出现至少n次，才会被匹配到。

##################位置边界匹配相关#################
```

^：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。

##################分组与后向引用#################
\(\) 表示分组，我们可以将其中的内容当做一个整体，分组可以嵌套。
\(ab\) 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。

回头看看，似乎我们已经掌握了不少符号，那么，我们能够通过这些符号干嘛呢？我们来动手试试？

比如，我想要从如下文本中找出，哪些人的手机号是以136开头的，我们该怎样做呢？

```
[www.zsythink.net]#cat shoujihaoma
朱双印：13688888888
葫芦娃：15505872487
金刚狼：13697429376
余小二：15712340987
擎天柱：13892652730

错误号码：1369807238738729879
```


我们可以使用grep命令，配合如下正则表达式。

```
[www.zsythink.net]#grep --color "\b136[[:digit:]]\{8\}\b" shoujihaoma
朱双印：13688888888
金刚狼：13697429376
[www.zsythink.net]#
```


可以看到，我们通过上述正则，找到了手机号以136开头的用户，我和金刚狼的手机号都是136打头的。
上述正则中，"136[[:digit:]]\{8\}"表示136后面跟随了8个连续的任意数字，所以，"136[[:digit:]]\{8\}"就表示一个以136开头的11位数字，也就是我们想要找到的"
但是，如果仅仅使用"136[[:digit:]]\{8\}"这个正则表达式，那么文本中的错误号码也会被匹配到，所以，我们在需要在正则的两侧加上"\b"。
两端的"\b"表示锚定词首与锚定词尾，所以，"\b136[[:digit:]]\{8\}\b"表示一个以136开头的11位数字，并且这11个数字作为一个单独的单词存在。
如果之前每一篇关于正则表达式的文章你都阅读过，并且理解了，那么看懂上述正则应该不是什么难事。这里就不再赘述了。

那么，如果我们想要从ifconfig命令的结果中找出IPV4格式的IP地址，应该怎么办呢？

```
[www.zsythink.net]#ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:B7:F4:C7
          inet addr:10.1.0.2  Bcast:10.1.255.255  Mask:255.255.0.0
          inet6 addr: fe80::20c:29ff:feb7:f4c7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:18 errors:0 dropped:0 overruns:0 frame:0
          TX packets:16 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1656 (1.6 KiB)  TX bytes:1128 (1.1 KiB)

eth4      Link encap:Ethernet  HWaddr 00:0C:29:B7:F4:D1
          inet addr:192.168.1.139  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:feb7:f4d1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:17770 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4548 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1415763 (1.3 MiB)  TX bytes:576535 (563.0 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:160 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:11292 (11.0 KiB)  TX bytes:11292 (11.0 KiB)

[www.zsythink.net]#
```

我们可以使用如下正则表达式。

```
[www.zsythink.net]#ifconfig | grep --color "\([0-9]\{1,3\}\.\)\{3\}[0-9]\{1,3\}"
          inet addr:10.1.0.2  Bcast:10.1.255.255  Mask:255.255.0.0
          inet addr:192.168.1.139  Bcast:192.168.1.255  Mask:255.255.255.0
          inet addr:127.0.0.1  Mask:255.0.0.0
[www.zsythink.net]#
```


为了方便理解，我们可以将上述正则表达式拆分成3段去理解，没错，我们把上述正则拆分成红色标注部分，蓝色标注部分，绿色标注部分。

红色部分的正则为"\([0-9]\{1,3\}\.\)"，它表示一个最少为1位数字，最多为3位数字的字符串，并且这个字符串后面跟随了一个"点"，我们把这个带有点的数字字符串整体。

蓝色部分的正则为"\{3\}"，它表示之前的字符需要连续出现3次，当它与红色部分的正则结合在一起时，表示符合红色部分正则的字符串需要连续出现3次。

绿色部分的正则为"[0-9]\{1,3\}"，它表示一个最少为1位数字，最多为3位数字的字符串。

当上述三部分正则结合在一起时，就能表示一个类似IPV4地址的字符串（此处暂不考虑1到254的取值范围）。

其实怎样去写正则表达式，没有一个固定的方法，只要能够正确的排列组合，表达出我们想要表达的意思，匹配到我们想要匹配的字符串，就是正确的写法，你也可以发挥想法，写出对应正则表达式。

关于怎样在Linux中使用"基本正则表达式"，就总结到这里，之后，我们会介绍怎样在Linux中使用"扩展正则表达式"。

有了基本正则表达式的基础，再去理解扩展正则表达式，绝对很轻松，好了，今天就到这里，希望这篇文章能够帮到你~~~

正则表达式