

在本博客中，AWK是一个系列文章，本人会尽量以通俗易懂的方式递进的总结awk命令的相关知识点。

awk系列博文直达链接：[AWK命令总结之从放弃到入门（通俗易懂，快进来看）](#)



这篇文章中的知识点是建立在前文的基础上的，如果你还没有掌握前文中的知识，请先参考之前的文章。

在前文中，我们已经认识了awk的模式，而且，我们已经介绍了awk中的3中模式

- 1、空模式
- 2、关系运算模式
- 3、BEGIN/END模式

那么今天，我们就来介绍一下awk的另外两种常用模式，正则模式与行范围模式，别着急，我们一个一个慢慢聊。

正则模式

先说说什么是正则模式。

见名知义，“正则模式”肯定与“正则表达式”有关，所以，如果想要使用这种模式，则必须先学会在Linux中使用正则表达式，如果你对正则表达式还不是特别熟悉，我的系列文章：[在Linux中使用正则表达式](#)

前文中提到过，“模式”可以理解为“条件”，当不指定模式时，文本中的每一行都会执行对应的动作，当指定模式时，只有被模式匹配到的、符合条件的行才会执行对应动作。那么什么是正则模式呢？正则模式可以理解为，把“正则表达式”当做“条件”，能与正则匹配的行，就算满足条件，满足条件的行才会执行对应的动作，不能被正则匹配的行不会执行对应的动作。如果你觉得我说的不明白，来看一个小例子，就能理解。

不过在进行示例之前，我们先来思考一个小问题。

我们知道，在Linux中，/etc/passwd文件中存放了用户信息，那么假设，我们想要从/etc/passwd文件中找出“用户名以zsy开头”的用户，我们该怎么办呢？

没错，我们可以使用grep命令，配合正则表达式，找出对应的信息，示例如下。

注：如果你还不了解grep命令和正则表达式，请参考博客中的文章，此处不再赘述。

如上例所示，我们通过grep命令，配合正则表达式，找出了我们需要的信息，那么，使用awk命令，能否完成上述需求呢？

答案是肯定的，那么我们一起来看看，使用awk命令，怎样从/etc/passwd文件中找出用户名以zsy开头的用户，示例如下

聪明如你一定看出来，不管是使用grep命令，还是使用awk命令，都使用了相同的正则表达式“^zsy”

唯一的区别就是，在grep命令中，直接使用了正则表达式，而在awk命令中，正则表达式被放入了两个斜线中。

这样说可能不容易理解，看图说话似乎更加容易理解。

上图中，awk命令在使用正则表达式时，将正则表达式放入了“/ /”中。

其实，这就是我们今天要介绍的“正则模式”，在使用“正则模式”时，文本行如果能够被正则表达式匹配到，就会执行对应的动作，如果没有被正则匹配到，则不会执行动作，而上例中，对应的动作就是{print \$0}，也就是打印整行，所以，上例中的grep命令与awk命令所实现的效果是完全相同的，那么你可能会问，既然效果完全相同，为什么要使用awk呢？似乎grep更加简单一些，没错，上例中，grep是更加简单一些，但是不要忘了，awk有自己的优势，就是格式化能力，那么，我们换一个场景，可能更加实用了，示例如下。

猛然一看，上例似乎非常复杂，但是如果你已经掌握了前文中的知识，那么你一定能够看明白，上例中蓝线标注的部分使用了BEGIN模式，并且格式化输出了一行文头，上例中红线标注的部分使用了正则模式，并且格式化输出了/etc/passwd文件中的第一列与第三列（用户名字段与用户ID字段），上例中，只使用了awk一条命令就完成了多项工作。

- 1、从/etc/passwd文件中找出符合条件的行（用户名以zsy开头的用户）。
- 2、找出符合条件的文本行以后，以":"作为分隔符，将文本行分段。
- 3、取出我们需要的字段，格式化输出。
- 4、结合BEGIN模式，输出一个格式化以后的文本，提高可读性。

因为我们在处理文本时，往往需要用到正则表达式，所以，awk的正则模式应该会经常用到。

但是需要注意，在使用正则模式时，如果正则中包含"/"，则需要进行转义，这样说可能不容易理解，我们来看个例子。

仍然使用/etc/passwd进行测试，我们知道，/etc/passwd中保存了用户信息，其中每行的最后一个字段为用户使用的登录shell，假设，我们想要从passwd文件中bash作为登录shell的用户，我们该怎么办呢？

没错，我们可以使用grep命令，配合正则表达式完成我们的需求，示例如下。

如上图所示，使用"/bin/bash"作为登录shell的用户都被我们找了出来。

同理，我们使用awk命令，同样能够实现与grep相同的效果。

于是，按照套路，你可能会尝试如下命令。

正如上图所示，按照套路，我们将正则部分放入了两个斜线中，但是运行命令时报错。

这是因为正则中包含"/"，而当使用正则模式时，又需要把正则放入到两个"/"中。

所以，我们需要对正则中的"/"进行转义，转义后即可正常运行命令，示例如下

如上图所示，经过转义后，awk命令即可正常的匹配到符合正则条件的行，并执行了相应的动作。

除此之外，还要注意以下两点

- 1、当在awk命令中使用正则模式时，使用到的正则用法属于"扩展正则表达式"（如果不理解，请参考博客中的"正则表达式"系列文章）。
- 2、当使用 {x,y} 这种次数匹配的正则表达式时，需要配合--posix选项或者--re-interval选项。

示例如下

上例中，正则模式中的正则表达式为"he{2,3}y"，此表达式表示"hey"中的字母e最少需要连续出现2次，最多只能连续出现3次，才能被正则表达式匹配到，但是正如有使用--posix选项或者--re-interval选项时，awk无法根据正则表达式对文本进行处理，因为上例的正则中包含类似"x,y"这样的次数匹配字符，所以，在使用正则对应的正则表达式中包含类似"x,y"这样的次数匹配字符，则需要使用--posix选项或者--re-interval选项。

好了，正则模式我们已经说明白了，赶快动手试试吧。

行范围模式

现在聊聊行范围模式。

其实，只要理解了正则模式，再理解行范围模式，就容易多了。

在介绍行范围模式之前，先来思考一个小问题，有一个文本文件，文件内容如下。

如上图所示，Lee这个名字出现了两次，第一次出现是在第2行，Kevin这个名字也出现了两次，第一次出现是在第5行。

假设我想从上述文本中找出，从Lee第一次出现的行，到Kevin第一次出现的行之间的所有行，我该怎么办呢？

使用awk的行范围模式，即可完成上述要求，示例如下。

我们来解释一下，上例中的行范围模式的语法是什么意思。

我们可以把上述行范围模式的语法与正则模式的语法对比着理解，可能更加方便我们理解。

上图中第一种语法是正则模式的语法，表示被正则表达式匹配到的行，将会执行对应的动作。

上图中第二种语法是行范围模式的语法，它表示，从被正则1匹配到的行开始，到被正则2匹配到的行结束，之间的所有行都会执行对应的动作，所以，这种模式被称式，因为它对应的是一个范围以内的所有行，但是需要注意的是，在行范围模式中，不管是正则1，还是正则2，都以第一次匹配到的行为准，就像上述示例中，即与第3行中都出现了，但是由于正则1先匹配到第2行中的lee，所以，最终打印出的内容从第2行开始，即使Kevin在第5行与第7行中都出现了，但是由于Kevin第一次行，所以最终打印出的内容到第5行结束，也就是说，最终打印出了第2行到第5行以内的所有行。

但是，你可能会有这样的需求，你不想依靠正则表达式去匹配行的特征，你只是想单纯的打印出从X行到Y行之间的所有行。

比如，我们有一个文本文件，这个文件中一共有7行文本，你想要打印出从第3行到第6行之间的所有行，该怎么做呢？

其实，使用之前学习到的"关系运算符模式"，即可满足我们的需求，示例如下。

上图中，NR为awk的内置变量，表示行号，"NR>=3 && NR<=6"表示行号大于等于3，并且行号小于等于6时，执行对应的动作，而对应的动作就是打印整行，所表示打印出文本中从第3行到第6行之间的所有行。

你是不是和我一样，学了新知识，忘了旧知识呢？就像上述示例一样，你是不是总在尝试使用"正则模式"解决问题，而忘记了使用"关系表达式模式"解决问题呢？

其他

其实，在学习"关系表达式模式"时，有一个"关系运算符"需要与"正则模式"配合使用，它就是"~"
还记得我们之前总结的一些常用的关系运算符吗，我们来回顾一下。

关系运算符	含义	用法示例
<	小于	x < y
<=	小于等于	x <= y
==	等于	x == y
!=	不等于	x != y
>=	大于等于	x >= y
>	大于	x > y
~	与对应的正则匹配则为真	x ~ /正则/
!~	与对应的正则不匹配则为真	x !~ /正则/

没错，细心如你一定发现了，关系运算符"~"与关系运算符"!~"都需要配合"正则模式"使用。
我们来看一个小示例，就更容易理解了。
比如，我想要从如下文本中找出，网卡1的IP地址在192.168.0.0/16网段内的主机，该怎么办呢？

我们可以使用如下命令，利用关系运算符与正则模式，达到我们的目的。


上述示例中，\$2为awk的内置变量，表示文本中的第2列，"\$2~/正则/"表示文本中的第2列如果与正则匹配，则执行对应的动作，对应的动作为"{print \$1,\$2}"，表的第1列与第2列，上例中的正则表达式我就不再赘述了，如果你还不太了解正则表达式，可以参考博客中的正则系列文章。

是不是很简单？我想你应该明白了。
到目前为止，我们已经认识了awk的模式，模式可以总结为如下5种。

- 1、空模式
- 2、关系运算模式
- 3、正则模式
- 4、行范围模式
- 5、BEGIN/END模式

如果你还有没有搞明白的地方，就再把之前的文章看一遍吧…

好了，今天就总结到这里，希望这篇文章可以帮助你~~~~



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~