URL: http://www.webtechina.com/zone/showcon4.php?id=1560

UNIX下的正则表达式（Regular Expression）

来自：蓝飞鸟
加入时间：2003-11-12 10:42:53
点击数：120

UNIX下的正则表达式（Regular Expression）
（初2002年3月12日，最后修改：2002年3月26日 ，修改次数：21 蓝飞鸟）

让我们来看一个命令，用ls命令列出目录：
ls -l |grep ^d
初学Linux的朋友，明白ls和DOS的DIR是意思相差不多，但是如果没弄明白Regular Expression，就只会觉得UNIX下的命令太复杂，而不会享受到UNIX的命令的强大。象这个命令：ls -l |grep ^d 就运用了|（管道）、grep（搜索）和正则表达式，而要看懂这个命令，关键的还在正则表达式。
正则表达式就是regular expression，从英文翻译过来常有多种译法，因此regular expression还有别的叫法，如规则表达式、文字匹配模式，一般来说是在使用grep搜索，sed和awk语言时列出或者匹配字符串的一种办法。值得注意的正则表达式看起来颇象一些命令的文件名匹配模式，如find，实际上，正则表达式与文件名匹配模式不同。在UNIX中，正则表达式是由一个或多个字符和meta字符组成的字符串，把正则表达式与数据源进行匹配之前，程序自动把它展开变为规定的模式，然后一个字符一个字符的进行比较。
我查了一下帮助资料，目前regular expression有两种形式，一种是现代的（POSIX 1003.2叫这种为扩展的regular expression），另一种是老的（基本regular expression）形式，我这里讲的完全针对现代形式，如egrep的regular expression（以下简称RE），因为以前的基本RE只保留在那些向后兼容老程序中。
UNIX有很多地方会用到正则表达式，在大文件中查找符合条件的字符串或SHELL编程中，要想把自己的想法贯彻给UNIX，非得熟练运用RE不可，搞不清楚RE，在UNIX中，无异于背着肥马爬行，既觉得处处不方便，做甚麼事都碍手碍脚，又浪费时间体力。想想，本来敲一行命令然后去喝杯茶就能完成的工作，非得守在计算机旁边敲了命令等着结果输出，再接着输入数据，眼睛还得盯着屏幕，费神又费力。如果弄明白了RE，许多时候就能得心应手。
在我以前的学习过程中，随着对Linux系统的熟悉，我发现学习UNIX第一步就是要弄清RE，这样，更容易看懂复杂的命令组合。正则表达式的具体应用，要结合实际情况，但首先得搞懂几个RE的基本观念。
RE由以下几个基本部分组成：
字符集，指匹配一个或多个指定的字符。字符集可以指定范围或一一列出所有指定的字符。
定位标识符，指明在一行的行首或是行尾。
匹配次数。一般指某个字符连续重复出现多次。
meta字符。必须结合meta字符才能实现。
总的来说，meta字符是关键，明白了meta字符的含义，就容易搞清楚字符集、定位标识符以及匹配次数，通常有以下meta字符，在列出meta字符之前，先约定几个术语（借用POSIX中的说明文档）。
每个RE由一个或多个 分支 构成，每个分支之间由|符号分隔。如在file中查找匹配字符串string1或string2的行：
egrep "string1|string2" file
每个分支由一个或多个 块 连接而成，依次匹配每一个块。
每一个块由一个 原子 后面跟*、+、-、?符号或者{}（表示范围）组成，原子可以是用()括号包围起来的RE。
使用中，通常要把RE用""（引号）引起来，否则，{}和()必须加\（转义符），即\{原子\}，\(原子\)。值得注意的是，grep和egrep(扩展grep)存在着诸多不同，其中有一点

就是，在egrep中的{}、()、|等符号，只要在""(引号)中，不加\（斜杠），而grep中的这些符号，在""（引号）中也必须加\（斜杠），如egrep "string1|string2"用grep表示就是 grep "string1\|string2"
$ 从行尾开始匹配
^ 从行首开始匹配
. 匹配任一字符，当然，除了换行符
[] 匹配[]内列出的字符，如果[]内以^符号开始，则表明匹配[]内列出的所有字符以外的字符，如果要匹配一个]符号，则]符号必须在[]的开始（在^后面），如果要匹配一个-（减号），-最好放在首位或末位。注意：在[]内的meta字符，没有特殊含义。
[]内的字符可以一一列出或指定范围。对于一一列出的字符集，可以用“，”（逗号）隔开以便于阅读，但并不强制要求如此。
如要表示所有数字
[0123456789]
也可以用[0-9]表示
如果要表示所有字母
[a-zA-Z]
如仅仅表示所有小写字母
[a-z]
当我要找一个单词，这个单词包括4个字母，以b开头，以d结尾，因为这是个单词，所有b和d之间必须是字母，当然无论大小写都行。用模式表示就是：
b[a-zA-Z][a-zA-Z]d
它可以是bird、band或者是bind。
原子{n1,n2} 匹配原子出现次数，如果只有n1则表明精确出现n1次，如果是n1,（n1加逗号）则表明出现至少n1次，如果是n1,n2则表示出现最少n1次，最多n2次，即在n1和n2之间。注意：n1<n2
(原子) 用括号注明原子，括号内可以是字符、字符串甚至RE。在随后的RE中可以用\跟一个小于10的数字来依次存取(原子)
* 原子后跟*，匹配0个或多个该原子
+ 原子后跟+，匹配至少1个或多个该原子
? 原子后跟?，匹配0个或1个该原子
\< 即使RE加了""引号，也应当包括\（下同），使用中在放原子的前面，表示该原子的前面是完整单词，如\<and匹配and、android而不会匹配hand。
\> 使用中放在原子的后面，表示该原子的后面部分是完成单词，如and\>匹配and、hand而不会匹配android，如果要表示一个完整的单词，\<and\>
\ 转义符，用来表示特殊符号，即让meta字符失去他们的特殊含义：^.

[$()|*+?{\，便
于搜索这些字符。
让我们用例子来说明：
我有一个文件叫myfile，先看看它的内容：
[flaunt@yy example]$ cat myfile
blueflybird is me.
BLUEFLYBIRD IS ME.
Yes,I'm blueflybird.
of course,Blueflybird is me.
Of Course,BLUEFLYBIRD IS ME.
[BLUEBLUE BLUE by BLUE blue bird bird]
分支、块、原子
[flaunt@yy example]$ egrep "^blueflybird|BLUEFLYBIRD.*E.$" myfile
blueflybird is me.
BLUEFLYBIRD IS ME.
Of Course,BLUEFLYBIRD IS ME.
上面的命令包括两个分支，用|分开，意思是查找myfile文件中，以小写blueflybird开头的行，或者是含有大写BLUEFLYBIRD单词并且以E.结尾的行。
匹配行末$
如果要查找myfile文件中以e.结尾的行（注意加了转义符\）：
[flaunt@yy example]$ egrep "e\.$" myfile
blueflybird is me.

of course,Blueflybird is me.
或者匹配以bird.结尾的行（注意$号放在命令后面）：
[flaunt@yy example]$ egrep "bird\.$" myfile
Yes,I'm blueflybird.
匹配行首^
如果要查找myfile文件中以小写b开头的行：
[flaunt@yy example]$ egrep "^b" myfile
blueflybird is me.
如想查找以小写b或者大写B开始的行：
[flaunt@yy example]$ egrep "^[bB]" myfile
blueflybird is me.
BLUEFLYBIRD IS ME.
匹配任一字符.
.通常与其他meta字符联合使用，因为.代表任一字符，如下面的命令表示查找以任一字符开头，第二个字符是 f 的行
[flaunt@yy example]$ egrep "^.f" myfile
of course,Blueflybird is me.
Of Course,BLUEFLYBIRD IS ME.
匹配[]内列出的字符
查找包括有小写o与大写O的行
[flaunt@yy example]$ egrep "[oO]f" myfile
of course,Blueflybird is me.
Of Course,BLUEFLYBIRD IS ME.
查找不包括小写o与大写O的行
[flaunt@yy example]$ egrep "[^oO]f" myfile
blueflybird is me.
Yes,I'm blueflybird.
of course,Blueflybird is me.
查找包括符号]和大写Y的行
[flaunt@yy example]$ egrep "[]Y]" myfile
BLUEFLYBIRD IS ME.
Yes,I'm blueflybird.
Of Course,BLUEFLYBIRD IS ME.
[BLUEBLUE BLUE by BLUE blue bird bird]
查找所有以大写字母开始的行
[flaunt@yy example]$ egrep "^[A-Z]" myfile
BLUEFLYBIRD IS ME.
Yes,I'm blueflybird.
Of Course,BLUEFLYBIRD IS ME.
\<完整单词\>
这里的完整单词并不是说真实的英语单词，而是指连续的字母组合中不含有数字、空格、换行符及制表符，如myfile1中完整单词是myfile
同样"my name1eye"中，my、name、eye各自是完整单词，而"bbeekks"无序的组合也是完整单词
（原子）
这里要提一下sed命令，sed是一种流编辑器，还是用例子来说明，下面的命令把myfile中的大写BLUEFLYBIRD替换成--------
[flaunt@yy example]$ sed s/BLUEFLYBIRD/--------/ myfile
blueflybird is me.
-------- IS ME.
Yes,I'm blueflybird.
of course,Blueflybird is me.
Of Course,-------- IS ME.
[BLUEBLUE BLUE by BLUE blue bird bird]
如果我们要在myfile中大写BLUEFLYBIRD后面加上--------就要用到（原子）的引用。即前面讲的在\后面跟小于10的数字依次引用
[flaunt@yy example]$ sed "s/\(BLUEFLYBIRD\)/\1--------/" myfile
blueflybird is me.
BLUEFLYBIRD-------- IS ME.
Yes,I'm blueflybird.

of course,Blueflybird is me.
Of Course,BLUEFLYBIRD-------- IS ME.
[BLUEBLUE BLUE by BLUE blue bird bird]
上面的命令中，（BLUEFLYBIRD)增加了转义符\(BLUEFLYBIRD\)，注意使用了\1来引用BLUEFLYBIRD
匹配次数{}、+、?、*
为了继续说明匹配次数{}、+、?、*，我们新建一个文件叫myfile1，内容如下：
[flaunt@yy example]$ cat myfile1
blue
blueblue
blueblueblue
BLUE
BLUEBLUEBLUEBLUE
BBLLUUEE
BLUEEE
如果我要查找两个（精确的两个）连续BLUE相连的行
[flaunt@yy example]$ egrep "(BLUE){2}" myfile1
BLUEBLUE
BLUEBLUEBLUEBLUE
注意：为了查找BLUEBLUE，用RE表示则是(BLUE){2}，而不是BLUE{2},如果是BLUE{2}，则表示BLUEE。
警告：你说我们本来要查BLUEBLUE，为甚麽上面还列出了三个连续的BLUE呢？因为BLUE BLUEBLUE包括了BLUEBLUE。
如果想查找BLUEBLUE那一行，则必须使用定位符，如：
[flaunt@yy example]$ egrep "^(BLUE){2}\>" myfile1
BLUEBLUE
上面的例子查找以两个连续BLUE开始的完整单词的行
如要查找以两个或者两个以上连续BLUE开始的完整单词，则在2后面加上逗号2,
[flaunt@yy example]$ egrep "^(BLUE){2,}\>" myfile1
BLUEBLUE
BLUEBLUEBLUEBLUE
如查找以一至三个连续BLUE开始的完整单词的行
[flaunt@yy example]$ egrep "(BLUE){1,3}" myfile1
BLUE
BLUEBLUE
BLUEBLUEBLUEBLUE
BLUEEE
如果要查找以至少1个或多个BLUE开始的完整单词行
[flaunt@yy example]$ egrep "^(BLUE)+\>" myfile1
BLUE
BLUEBLUE
BLUEBLUEBLUEBLUE
如果要查找以0个或多个BLUE开始的完整单词行
[flaunt@yy example]$ egrep "^(BLUE)*\>" myfile1
BLUE
BLUEBLUE
BLUEBLUEBLUEBLUE
如果要查找0个或1个以BLUE开始的完整单词行
[flaunt@yy example]$ egrep "^(BLUE)?\>" myfile1
BLUE
非常值得注意的是：上面的命令 (BLUE)?\> 中增加了\>是否完整单词的测试，如果去掉\>，那麽
[flaunt@yy example]$ egrep "^(BLUE)?" myfile1
blue
blueblue
blueblueblue
BLUE
BLUEBLUE
BLUEBLUEBLUEBLUE
BBLLUUEE

BLUEEE
由于^(BLUE)?" 表示以0个或1个BLUE开始的行，没有测试BLUE是否完整单词，所以可以匹配任何行，而加了完整单词测试的命令，
只能匹配有1个BLUE开始并是完整单词的行匹配。
现在以例子来说明如果在一个目录中筛选需要的文件名并列出：
我们在home 目录中新建一个文件名example，里面有如下文件
[flaunt@yy example]$ pwd
/home/flaunt/example
[flaunt@yy example]$ ls -l
total 12
drwxrwxr-x 2 flaunt flaunt 4096 3月 12 10:41 子目录
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 11:41 blueflybird
-rw-rw-r-- 1 flaunt flaunt 62 3月 11 11:43 head
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 11:41 flaunt
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 14:59 flaunt1
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 14:59 flaunt2
-rw-rw-r-- 1 flaunt flaunt 601 3月 12 10:41 yy
本文以后的的例子要求你至少明白ls、grep命令和|(管道)。
ls 指列出文件目录，类似与DOS命令的DIR，但ls有太多参数，我们这儿仅用到的 -l 参数，即使用宽列格式列出。
ls -l 参数列出的宽列格式含义如下：
grep 是UNIX下的常用命令，在标准输出或文件中查找匹配的字符串（或模式）
| 管道简单的说就是把|左边命令的输出传到|右边作为命令的接受的输入数据。
$ 匹配行尾
bird$ 表示匹配以bird结尾的一行。
[flaunt@yy example]$ ls -l | grep bird$
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 11:41 blueflybird
上面的命令找到文件 blueflybird。
d$ 表示匹配以d 结尾的行
[flaunt@yy example]$ ls -l |grep d$
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 11:41 blueflybird
-rw-rw-r-- 1 flaunt flaunt 62 3月 11 11:43 head
上面的命令查找到以d 结尾的行的两个文件，blueflybird 和 head
[flaunt@yy example]$ ls -l |grep 录$
drwxrwxr-x 2 flaunt flaunt 4096 3月 12 10:41 子目录
上面的命令查找到以“录”结尾的行的文件，子目录
^ 与$相反，^表示匹配行首，即从每行的开始比较。 这在使用ls命令时查看目录时很有用，如：
查看目录^d
[flaunt@yy example]$ ls -l |grep ^d
drwxrwxr-x 2 flaunt flaunt 4096 3月 12 10:41 子目录
由于用ls -l 命令时会列出文件的各种特性，其中drwxrwxr-x是该文件的属性，说明此文件
目录，并且d 在行首，所以ls -l |grep ^d 就是列出目录
当然也可以只查看普通文件 ^-
[flaunt@yy example]$ ls -l |grep ^-
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 11:41 blueflybird
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 11:41 flaunt
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 14:59 flaunt1
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 14:59 flaunt2
-rw-rw-r-- 1 flaunt flaunt 62 3月 11 11:43 head
-rw-rw-r-- 1 flaunt flaunt 1066 3月 12 10:41 yy
用ls 命令时，不带 -l 参数，
[flaunt@yy example]$ ls
子目录 blueflybird flaunt flaunt1 flaunt2 head yy
如果要查看以 flau 开头的文件，可用命令 ^flau
[flaunt@yy example]$ ls |grep ^flau
flaunt
flaunt1
flaunt2

或 f 开头的文件，^f
[flaunt@yy example]$ ls |grep ^f
flaunt
flaunt1
flaunt2
特别的，表示匹配空行（即^后面和$前面没有任何字符）
^$
. .表示匹配任一字符
如：
[flaunt@yy example]$ ls -l |grep .
total 12
drwxrwxr-x 2 flaunt flaunt 4096 3月 12 10:41 子目录
-r--r--r-- 1 flaunt flaunt 0 3月 11 11:41 blueflybird
-rwxrwxr-x 1 flaunt flaunt 0 3月 11 11:41 flaunt
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 14:59 flaunt1
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 14:59 flaunt2
-rw-rw-r-- 1 flaunt flaunt 62 3月 11 11:43 head
-rwxrw-r-- 1 flaunt flaunt 1066 3月 12 10:41 yy
可以把.和^ $符号结合起来用。
如要查看该目录下，文件所有者有执行权限的文件 ^...x
[flaunt@yy example]$ ls -l |grep ^...x
drwxrwxr-x 2 flaunt flaunt 4096 3月 12 10:41 子目录
-rwxrwxr-x 1 flaunt flaunt 0 3月 11 11:41 flaunt
-rwxrw-r-- 1 flaunt flaunt 1066 3月 12 10:41 yy
我们来分析^...x，其中“^”表示从行首的开始，“.”表示可以是任一符号，“x”
就是小写字母x，那麼这几个符号组合在一起的规则表达是甚麼意思呢？把ls -l 输出的
每行行首与之比较，前三个是任一字符，第四个字符是小写字母x的行列出来。如下：
[flaunt@yy example]$ ls -l |grep ^...x
drwxrwxr-x 2 flaunt flaunt 4096 3月 12 10:41 子目录
-rwxrwxr-x 1 flaunt flaunt 0 3月 11 11:41 flaunt
-rwxrw-r-- 1 flaunt flaunt 1066 3月 12 10:41 yy
* *在正则表达式中与文件名替换中的 * 表示的意思不同，在文件名替换中，*表示任意
字符，但正则表达式中，*表示在某字符后匹配任意个或者0个该字符，注意是该字符（
一般来说必须跟在某字符后面）。
如 a* 就是在a后面跟随0个或多个a，那麼：
a = a*
aaaa = a*
aaaaaaaaaaaaaaaaaaaa 同样等于a*
再如，bcd*
bcd = bcd*
bcddddddd = bcd*
而要匹配0个或多个任意字符就要用“.*”才行。
还是实例最能说明问题，比如说我们现在要查一个文件，必须满足以下条件：
1. 是普通文件，而不是目录
2. 该文件的所有者具有写权限
3. 文件名的最后一个字符是数字2
[flaunt@yy example]$ ls -l |grep ^-.w.*2$
-rw-rw-r-- 1 flaunt flaunt 0 3月 11 14:59 flaunt2
具体应用中最好把 ^-.w.*2$ 放在引号中如： grep "^-.w.*2$"
[] 匹配[]内的特定字符或字符集。
例如：列出当前目录下以数字结尾的所有文件
[flaunt@yy example]$ ls |grep [0-9]$
flaunt1
flaunt2
或
[flaunt@yy example]$ ls |grep [0,1,2,3,4,5,6,789]$
flaunt1
flaunt2
如果要找字符串，同样以b开头，d结尾，中间是两个字符，注意是字符，可以是任何符
号，用模式表示：

b..d
而不是b[.][.]d
因为[.] 是指匹配符号.
pattern\{\} 匹配pattern出现的次数。这里有几种情况：
1．pattern\{n\} pattern出现n次
2．pattern\{n,\}pattern至少出现n次
3．pattern\{n,m\} pattern出现的范围在n-m之间，n,m为0－255任意整数
\ 转义符
蓝色飞鸟☆1997-2002
欢迎个人自由传播 禁止任何商业媒体转载本站文章