sed文档

**元字符**

**. 匹配除换行符之外的任意单个字符，awk中可以匹配换行符**

例:

abc adc aec

a.c

**\* 匹配任意一个（包括零个）在它前面的字符**

\*匹配0到多个前边字符

a

ab

ac

abb

abc

abbc

abbbbbbbbbbbc

sed 's/ab\*/x/' a.txt 匹配a开头b出现0到多次

asdfsdkjfhskc

ac

asdfjslkfjslfjslfjsfljc

a23113131c

aacc

aaaaaaa1ccccccccc2aaaaaaaaaaa3c4

sed 's/a.\*c/x/' a.txt 匹配a0-多个c结尾的字段 \*的贪婪性尽可能长的c

**[...] 匹配方括号中的任意一个字符，^为否定匹配， -表示字符的范围**

acc

a.c

abc

a2c

a6c

a7c

a8c

sed 's/a.c/x/' a.txt

sed 's/a[67c89]c/x/' a.txt

sed 's/a[1-9]c/x/' a.txt

sed 's/a[^1-9]c/x/' a.txt

1+2

1-2

1\*2

1/2

sed 's/1[+-\*/]2/x/' a.txt -号报错

sed 's/1[-+\*/]2/x/' a.txt

**^ 作为正则表达式的第一个字符，匹配行的开始。在awk中可以嵌入**

**换行符**

**$ 作为正则表达式的最后一个字符，匹配行的结尾。在awk中可以嵌**

**入换行符**

root

aroot

roota

sed 's/root/x/' a.txt

sed 's/^root/x/' a.txt

sed 's/root$/x/' a.txt

sed 's/^root$/x/' a.txt

sed 's/^...$/x/' a.txt 只包含3个字符

sed 's/.../x/' a.txt 最少包含3个字符

sed 's/^$/x/' a.txt sed不能匹配换行符 匹配空行

sed 's/^ \*$/x/' a.txt 多个空格的行

**\{n,m\} 匹配出现的n到m次数， \{n\}匹配出现n次。\{n,\}匹配至少出现**

**n次(看扩展)**

**\ 转义字符**

**如**

**a/b**

sed 's/a\/b/x\/y/' tt.txt

我们可以替换分隔符

sed 's@a/b@x/y@' tt.txt

sed 's;a/b;x/y;' tt.txt

任意的3个字符 都可以 避免产生冲突

date +%Y/%m/%d | sed ‘s@/@:@g’ g全局替换

**扩展元字符**

**+ 匹配前面的正则表达式的一次出现或多次出现**

a

ab

abb

abbb

sed 's/abb\*/x/' a.txt

sed -r 's/ab+/x/' a.txt

**? 匹配前面的正则表达式的零次出现或一次出现**

y

yes

Y

Yes

sed -r 's/[yY](es)?/x/' a.txt

**| 可以匹配前面的或后面的正则表达式（替代方案）**

**() 替换方案 对正则表达式分组**

company

companies

compan(y|ies)

[abc] (a|b|c) 某些情况下一样 还有分组功能后边会用到

**{n,m} 匹配出现的n到m次数， {n}匹配出现n次。{n,}匹配至少出现n次**

sed -r 's/a/!/' a.txt 只替换第一次匹配到的

sed -r 's/a{4}/!/' a.txt

sed -r 's/a{4,6}/!/' a.txt

sed -r 's/a{,4}/!/' a.txt

sed -r 's/a{5,}/!/' a.txt

匹配一个字符出现0-1次 基本正则也可以实现

.\{,1\}

sed ‘s/元字符在这里起作用匹配/+-|则不再起元字符作用/’

编写sed脚本 要求如下

原始 完成后

apple banana

banana orange

一条命令完成

是不是我们想的结果?

为什么产生上边的问题:

首先sed在运行中会维护一段内存空间,这段内存空间称为模式空间:

模式空间存储被替换的文件,默认是按行为单位

如上文件读取过程

首先进入模式空间s-----apple 替换为banana

在执行s-----banana 替换为orange

结果apple被替换成orange 没有-n参数 输出终端上

第二条进入模式空间 第一条命令匹配不上,第二条匹配 banana

被替换为orange

寻址上的全局透视（定址）

sed将命令应用于每个输入行，它可以指定零个、一个或两个地址。每个地址

都是一个描述模式、行号或者行寻址符号的正则表达式。(默认全文寻址)

范例file2.txt

.TS

Beijing,CN

.TE

Shanghai,CN

guangzhou,CN

shenyang,CN

sed ‘2s/CN/China/’file2.txt

sed '/Beijing/s/CN/China/' file2.txt

sed ‘$s/CN/China/’file2.txt

sed ‘/^\.TS/,/^\.TE/s/CN/China/’file2.txt(避免中间出现)

sed '4,6s/CN/China/' file2.txt

sed '4,$s/CN/China/' file2.txt

sed '4,/guangzhou/s/CN/China/' file2.txt

删除命令d

删除所有的行

d

只删除第一行

1d

使用寻址符号$，删除最后一行

$d

删除空行，正则表达式必须封闭在斜杠//当中

/^$/d

删除.TS 和.TE 标记的tbl 输入

/^\.TS/,/^\.TE/d

删除第五行到结尾所有的行

5,$d

混合使用行地址和模式地址

$ sed '1,/^$/d' file2.txt

删除除了那些行以外的行

1，5!d

分组命令

.TS

Beijing,CN

.TE

Shanghai,CN

guangzhou,CN

shenyang,CN

sed ‘/^\.TS/,/^\.TE/s/,/:/ ; /^\.TS/,/^\.TE/s/CN/China/ file2.txt

使用分组

sed '/^\.TS/,/^\.TE/{ s/,/:/ ; s/CN/china/ }' file2.txt

练习:

文件:datafile  
Steve Blenheim:238-923-7366:95 Latham Lane, Easton, PA 83755:11/12/56:20300  
Betty Boop:245-836-8357:635 Cutesy Lane, Hollywood, CA 91464:6/23/23:14500  
Igor Chevsky:385-375-8395:3567 Populus Place, Caldwell, NJ 23875:6/18/68:23400  
Norma Corder:397-857-2735:74 Pine Street, Dearborn, MI 23874:3/28/45:245700  
Jennifer Cowan:548-834-2348:583 Laurel Ave., Kingsville, TX 83745:10/1/35:58900  
Jon DeLoach:408-253-3122:123 Park St., San Jose, CA 04086:7/25/53:85100  
Karen Evich:284-758-2857:23 Edgecliff Place, Lincoln, NB 92086:7/25/53:85100  
Karen Evich:284-758-2867:23 Edgecliff Place, Lincoln, NB 92743:11/3/35:58200  
Karen Evich:284-758-2867:23 Edgecliff Place, Lincoln, NB 92743:11/3/35:58200  
Fred Fardbarkle:674-843-1385:20 Parak Lane, DeLuth, MN 23850:4/12/23:780900  
Fred Fardbarkle:674-843-1385:20 Parak Lane, DeLuth, MN 23850:4/12/23:780900  
Lori Gortz:327-832-5728:3465 Mirlo Street, Peabody, MA 34756:10/2/65:35200  
Paco Gutierrez:835-365-1284:454 Easy Street, Decatur, IL 75732:2/28/53:123500  
Ephram Hardy:293-259-5395:235 CarltonLane, Joliet, IL 73858:8/12/20:56700  
James Ikeda:834-938-8376:23445 Aster Ave., Allentown, NJ 83745:12/1/38:45000  
Barbara Kertz:385-573-8326:832 Ponce Drive, Gary, IN 83756:12/1/46:268500  
Lesley Kirstin:408-456-1234:4 Harvard Square, Boston, MA 02133:4/22/62:52600  
William Kopf:846-836-2837:6937 Ware Road, Milton, PA 93756:9/21/46:43500  
Sir Lancelot:837-835-8257:474 Camelot Boulevard, Bath, WY 28356:5/13/69:24500  
Jesse Neal:408-233-8971:45 Rose Terrace, San Francisco, CA 92303:2/3/36:25000  
Zippy Pinhead:834-823-8319:2356 Bizarro Ave., Farmount, IL 84357:1/1/67:89500  
Arthur Putie:923-835-8745:23 Wimp Lane, Kensington, DL 38758:8/31/69:126000  
Popeye Sailor:156-454-3322:945 Bluto Street, Anywhere, USA 29358:3/19/35:22350  
Jose Santiago:385-898-8357:38 Fife Way, Abilene, TX 39673:1/5/58:95600  
Tommy Savage:408-724-0140:1222 Oxbow Court, Sunnyvale, CA 94087:5/19/66:34200  
Yukio Takeshida:387-827-1095:13 Uno Lane, Ashville, NC 23556:7/1/29:57000  
Vinh Tranh:438-910-7449:8235 Maple Street, Wilmington, VM 29085:9/23/63:68900

1.把Jon's的名字改成Jonathan.  
2.删除头三行  
3.显示5-10行  
4.删除包含Lane的行.  
5.显示所有生日在November-December之间的行  
6.把三个星号(\*\*\*)添加到也Fred开头的行  
7.用JOSE HAS RETIRED取代包含Jose的行  
8.把Popeye的生日改成11/14/46  
9.删除所有空白行

替换命令

**n 可以是1-512，表示第n次出现的情况进行替换**

如:

ababab

sed ‘s/ab/oo/2’file

**g 全局更改**

如:

apple apple apple

sed ‘s/apple/ooo/g’file

**p 打印模式空间的内容(完成替换后)**

如

apple apple apple

123 123 123

sed ‘s/apple/123/p’file.txt

只打印apple行 所以能用-n参数

**w file 写入到一个文件file中(只保存替换后行)**

如:

sed 's/ab/OOO/w a.txt' test

**& 用正则表达式匹配的内容进行替换**

**\n 回调参数 (前边的分组再拿回来)**

**如:**

apple

123

apple

sed ‘s/apple/&&/’a.txt &表示前边的正则表达式 2个&代表输出2次

**比如:**

this is linux

sed ‘s/linux/& redhat/’file.txt

**\n如:**

$ cat test1

first:second

one:two

$ sed 's/\(.\*\):\(.\*\)/\2:\1/' test1

second:first

two:one

匹配交换第一第二个字符

sed -r‘s/(.)(.)/\2\1/’/etc/passwd

sed –r ‘s/(.)(.)(.\*)/\2\1\3/’/etc/passwd

交换前两个单词

如文件

apple banana egg

1. apple banana egg

apple.Banana egg

apple. Banana egg

apple? Banana egg

**删除**

**[address]d**

**删除模式空间的内容，同时改编脚本的控制流，执行这个命令后，在“空**

**的”模式空间不再有命令执行。删除命令会导致读取新的输入行**

sed ‘d;s’ 如果执行了d后边所有命令就不执行了,那么s做什么用?

sed ‘/a/d;s 这样如果匹配了a字符 就删除,否则就执行s 这样就构成了一个判断

sed ‘=’a.txt =显示行号

aaaaa

bbbbb

ccccc

dddd

sed '/c/d;=' test.txt

**追加、插入和更改**

**[line-address]a tex**t

例:

aaaaa

bb xx bbb

ccccc

dddd

sed '/xx/a hello' test.txt

**[line-address]i text**

aaaaa

bb xx bbb

ccccc

dddd

sed '/xx/i hello' test.txt

[address]c text

aaaaa

bb xx bbb

ccccc

dddd

**sed '/xx/c hello' test.txt**

以上的操作都是将结果显示在屏幕上:怎么写入到文件

sed –i '/xx/c hello' test.txt

-i 直接对源文件做改变 注意文件inode会变化

**转换**

**[address]y/abc/xyz/**

sed ‘y/abcdefghijklmnopqrstu…/ABCDEFGHIJKLMNOPQRS…/’a.txt

**打印**

**[address]p 用于输出模式空间的内容**

sed ‘’a.txt

sed ‘3p’a.txt

**打印行号**

**[line-address]=**

apple

123

apple

sed -n '/apple/{=;p}' test

**下一步**

**[address]n**

**n命令输出模式空间的内容，然后读取输入的下一行，而不用返回脚本的顶端**

test aa bb

test aa bb

sed ‘/test/{n;s/aa/xx}’test.txt

**读和写文件 范例文件 file4.txt**

**[line-address]r file**

**[address]w file**

sed -e '/<mail list>/r maillist' -e '/<mail list>/d' file4.txt

/MA$/w region.MA

/VA$/w region.VA

/CA$/w region.CA

如:

test aa bb

test aa bb

apple

banana

abc

sed '/banana/r aa.txt' tt.txt

**退出(速度更快)**

**[line-address]q**

$ sed '100q' test

测试:

for i in {1..100000}

do

echo $i >> num.txt

done

time sed –n ‘1,100p’num.txt

time sed ‘100q’num.txt

**高级sed 命令**

高级命令分成3个组：

1 处理多行模式空间（N、D、P）

2 采用保持空间来保存模式空间的内容并使它可用于后续的命令

（H、h、G、g、x）

3 编写使用分支和条件指令的脚本来更改控制流（:、b、t）

高级脚本都做一件共同的事，那就是他们改变了执行或控制的流程顺序。

**多行模式空间**

追加下一行

多行Next（N）命令通过读取新的输入行，并将它添加到模式空间的现有内容

之后来创建多行模式。

user1

123

user2

1234

user3

12345

将该文件 用户名,密码追加到一起

每两行合并sed ‘N;s/\n/:/’

**多行删除**

D命令删除模式空间中直到第一个换行符的内容。它不会导致读入新的输入行，

相反，它返回到脚本的顶端，将这些指令应用与模式空间剩余的内容。

user1

123

M

user2

1234

W

user3

12345

M

sed '/^$/{N;/^\n$/D}' user.txt

/^$/{

N

/^\n$/D

}

多行删除命令完成工作的原因是，当遇到两个空行时，D命令只删除两个空行

中的第一个。下一次遍历该脚本时，这个空行将导致另一行被读入模式空间。

如果那行不为空，那么两行都输出，因此确保了输出一个空行。换句话说，当

模式空间中有两个空行时，只有第一个空行被删除。当一个空行后面跟有文本

时，模式空间可以正常输出。

多行

多行打印

P命令输出多行模式空间的第一部分，直到第一个嵌入的换行符为止。在执行

完脚本的最后一个命令之后，模式空间的内容自动输出。

P命令经常出现在N命令之后和D命令之前。

这三个命令能建立一个输入、输出循环，用来维护两行模式空间，但是一次只

输出一行。

这个循环的目的是只输出模式空间的第一行，然后返回到脚本的顶端将所有的

命令应用于模式空间的第二行。

没有这个循环，当执行脚本中的最后一个命令时，模式空间中的这两行都将被

输出。

删除文件倒数第二行

sed 'N;$!P;D' a.txt

删除文件最后两行

sed 'N;$!P;$!D;$d' a.txt

包含

**包含那一行**

模式空间是容纳当前输入行的缓冲区。还有一个成为保持空间（hode space）

的预留（set-aside）缓冲区。模式空间的内容可以复制到保持空间，而且保持

空间的内容也可以复制到模式空间。有一组命令用于在保持空间和模式空间之

间移动数据。保持空间用于临时存储。单独的命令不能寻址保持空间或者更改

它的内容。

保持空间最常见的用途是，当改变模式空间中的原始内容时，用于保留当

前输入行的副本。

模式空间 保持空间

默认换行符

h—覆盖🡪 全部覆盖

H—追加-> 追加到换行符之后

g🡨覆盖取回--

G🡨追加取回—

例:

1111111

2222222

利用空间输出

2222222

11111111

sed ‘1h;1d;2G’test.txt

每行加一个空行?

**大写转换**

样本文件 file6.txt

find the Match statement

Consult the Get statement.

using the Read statement to retrieve data

将 the 和statement之间的单词转换成大写

脚本：changsrc

# capitalize statement names

/the .\* statement/{

h

s/.\*the \(.\*\) statement.\*/\1/

y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/

G

s/\(.\*\)\n\(.\*the \).\*\( statement.\*\)/\2\1\3/

}

执行过程：

h 将当前输入行复制到保持空间

Pattern Space: find the Match statement

Hold Space: find the Match statement

s/.\*the \(.\*\) statement.\*/\1/ 取出将被替换的语句

Pattern Space: Match

Hold Space: find the Match statement

y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/

Pattern Space: MATCH

Hold Space: find the Match statement

G 将保持空间中的内容追加到模式空间

Pattern Space: MATCH\nfind the Match statement

Hold Space: find the Match statement

s/\(.\*\)\n\(.\*the \).\*\( statement.\*\)/\2\1\3/ 替换并排序

Pattern Space: find the MATCH statement

Hold Space: find the Match statement