# sed

功能强大的流式文本编辑器

## 补充说明

**sed** 是一种流编辑器，它是文本处理中非常重要的工具，能够完美的配合正则表达式使用，功能不同凡响。处理时，把当前处理的行存储在临时缓冲区中，称为“模式空间”（pattern space），接着用sed命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。文件内容并没有 改变，除非你使用重定向存储输出。Sed主要用来自动编辑一个或多个文件；简化对文件的反复操作；编写转换程序等。

## sed的选项、命令、替换标记

**命令格式**

sed [options] 'command' file(s)  
sed [options] -f scriptfile file(s)

### 选项

-e<script>或--expression=<script>：以选项中的指定的script来处理输入的文本文件；  
-f<script文件>或--file=<script文件>：以选项中指定的script文件来处理输入的文本文件；  
-h或--help：显示帮助；  
-n或--quiet或——silent：仅显示script处理后的结果；  
-V或--version：显示版本信息。

### 参数

文件：指定待处理的文本文件列表。

### sed命令

a\ # 在当前行下面插入文本。  
i\ # 在当前行上面插入文本。  
c\ # 把选定的行改为新的文本。  
d # 删除，删除选择的行。  
D # 删除模板块的第一行。  
s # 替换指定字符  
h # 拷贝模板块的内容到内存中的缓冲区。  
H # 追加模板块的内容到内存中的缓冲区。  
g # 获得内存缓冲区的内容，并替代当前模板块中的文本。  
G # 获得内存缓冲区的内容，并追加到当前模板块文本的后面。  
l # 列表不能打印字符的清单。  
n # 读取下一个输入行，用下一个命令处理新的行而不是用第一个命令。  
N # 追加下一个输入行到模板块后面并在二者间嵌入一个新行，改变当前行号码。  
p # 打印模板块的行。  
P # (大写) 打印模板块的第一行。  
q # 退出Sed。  
b lable # 分支到脚本中带有标记的地方，如果分支不存在则分支到脚本的末尾。  
r file # 从file中读行。  
t label # if分支，从最后一行开始，条件一旦满足或者T，t命令，将导致分支到带有标号的命令处，或者到脚本的末尾。  
T label # 错误分支，从最后一行开始，一旦发生错误或者T，t命令，将导致分支到带有标号的命令处，或者到脚本的末尾。  
w file # 写并追加模板块到file末尾。   
W file # 写并追加模板块的第一行到file末尾。   
! # 表示后面的命令对所有没有被选定的行发生作用。   
= # 打印当前行号码。   
# # 把注释扩展到下一个换行符以前。

### sed替换标记

g # 表示行内全面替换。   
p # 表示打印行。   
w # 表示把行写入一个文件。   
x # 表示互换模板块中的文本和缓冲区中的文本。   
y # 表示把一个字符翻译为另外的字符（但是不用于正则表达式）  
\1 # 子串匹配标记  
& # 已匹配字符串标记

### sed元字符集

^ # 匹配行开始，如：/^sed/匹配所有以sed开头的行。  
$ # 匹配行结束，如：/sed$/匹配所有以sed结尾的行。  
. # 匹配一个非换行符的任意字符，如：/s.d/匹配s后接一个任意字符，最后是d。  
\* # 匹配0个或多个字符，如：/\*sed/匹配所有模板是一个或多个空格后紧跟sed的行。  
[] # 匹配一个指定范围内的字符，如/[sS]ed/匹配sed和Sed。   
[^] # 匹配一个不在指定范围内的字符，如：/[^A-RT-Z]ed/匹配不包含A-R和T-Z的一个字母开头，紧跟ed的行。  
\(..\) # 匹配子串，保存匹配的字符，如s/\(love\)able/\1rs，loveable被替换成lovers。  
& # 保存搜索字符用来替换其他字符，如s/love/ \*\*&\*\* /，love这成 \*\*love\*\* 。  
\< # 匹配单词的开始，如:/\<love/匹配包含以love开头的单词的行。  
\> # 匹配单词的结束，如/love\>/匹配包含以love结尾的单词的行。  
x\{m\} # 重复字符x，m次，如：/0\{5\}/匹配包含5个0的行。  
x\{m,\} # 重复字符x，至少m次，如：/0\{5,\}/匹配至少有5个0的行。  
x\{m,n\} # 重复字符x，至少m次，不多于n次，如：/0\{5,10\}/匹配5~10个0的行。

## sed用法实例

### 替换操作：s命令

替换文本中的字符串：

sed 's/book/books/' file

**-n选项** 和 **p命令** 一起使用表示只打印那些发生替换的行：

sed -n ‘s/test/TEST/p’ file

直接编辑文件 **选项-i** ，会匹配file文件中每一行的所有book替换为books：

sed -i 's/book/books/g' file

### 全面替换标记g

使用后缀 /g 标记会替换每一行中的所有匹配：

sed 's/book/books/g' file

当需要从第N处匹配开始替换时，可以使用 /Ng：

echo sksksksksksk | sed 's/sk/SK/2g'  
skSKSKSKSKSK  
  
echo sksksksksksk | sed 's/sk/SK/3g'  
skskSKSKSKSK  
  
echo sksksksksksk | sed 's/sk/SK/4g'  
skskskSKSKSK

### 定界符

以上命令中字符 / 在sed中作为定界符使用，也可以使用任意的定界符：

sed 's:test:TEXT:g'  
sed 's|test|TEXT|g'

定界符出现在样式内部时，需要进行转义：

sed 's/\/bin/\/usr\/local\/bin/g'

### 删除操作：d命令

删除空白行：

sed '/^$/d' file

删除文件的第2行：

sed '2d' file

删除文件的第2行到末尾所有行：

sed '2,$d' file

删除文件最后一行：

sed '$d' file

删除文件中所有开头是test的行：

sed '/^test/'d file

### 已匹配字符串标记&

正则表达式 + 匹配每一个单词，使用 [&] 替换它，& 对应于之前所匹配到的单词：

echo this is a test line | sed 's/\w\+/[&]/g'  
[this] [is] [a] [test] [line]

所有以192.168.0.1开头的行都会被替换成它自已加localhost：

sed 's/^192.168.0.1/&localhost/' file  
192.168.0.1localhost

### 子串匹配标记\1

匹配给定样式的其中一部分：

echo this is digit 7 in a number | sed 's/digit \([0-9]\)/\1/'  
this is 7 in a number

命令中 digit 7，被替换成了 7。样式匹配到的子串是 7，(..) 用于匹配子串，对于匹配到的第一个子串就标记为 **\1** ，依此类推匹配到的第二个结果就是 **\2** ，例如：

echo aaa BBB | sed 's/\([a-z]\+\) \([A-Z]\+\)/\2 \1/'  
BBB aaa

love被标记为1，所有loveable会被替换成lovers，并打印出来：

sed -n 's/\(love\)able/\1rs/p' file

通过替换获取ip：

ifconfig ens32 | sed -n '/inet /p' | sed 's/inet \([0-9.]\+\).\*/\1/'  
192.168.75.126

### 大小写转换U/L

\u： 首字母转换为大写  
\U： 全部转换为大写  
\l： 首字母转换为小写  
\L： 全部转换为小写

首字母转换为大写：

[root@node6 ~]# sed 's/^[a-z]\+/\u&/' passwd   
Root:x:0:0:root:/root:/bin/bash  
Bin:x:1:1:bin:/bin:/sbin/nologin  
Daemon:x:2:2:daemon:/sbin:/sbin/nologin  
Adm:x:3:4:adm:/var/adm:/sbin/nologin  
Lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin  
Sync:x:5:0:sync:/sbin:/bin/sync

匹配到的字符全部转换为大写：

[root@node6 ~]# sed 's/^[a-z]\+/\U&/' passwd   
ROOT:x:0:0:root:/root:/bin/bash  
BIN:x:1:1:bin:/bin:/sbin/nologin

### 组合多个表达式

1. 替换文本中的多个字符串：

sed -e 's/old\_string/new\_string/g' -e 's/another\_old\_string/another\_new\_string/g' file.txt

1. 删除文本中的多个行：

sed -e '1d' -e '/pattern/d' file.txt

1. 在文本中插入多个行：

sed -e '1i\inserted\_line1' -e '2i\inserted\_line2' file.txt

其中，-e 表示指定一个表达式，多个表达式之间用 -e 分隔。每个表达式可以是一个 sed 命令，例如 s、d、i 等。

### 引用

sed表达式可以使用单引号来引用，但是如果表达式内部包含变量字符串，就需要使用双引号。

test=hello  
echo hello WORLD | sed "s/$test/HELLO"  
HELLO WORLD

### 选定行的范围：,（逗号）

所有在模板test和check所确定的范围内的行都被打印：

sed -n '/test/,/check/p' file

打印从第5行开始到第一个包含以test开始的行之间的所有行：

sed -n '5,/^test/p' file

对于模板test和west之间的行，每行的末尾用字符串aaa bbb替换：

sed '/test/,/west/s/$/aaa bbb/' file

### 多点编辑：e命令

-e选项允许在同一行里执行多条命令：

sed -e '1,5d' -e 's/test/check/' file

上面sed表达式的第一条命令删除1至5行，第二条命令用check替换test。命令的执行顺序对结果有影响。如果两个命令都是替换命令，那么第一个替换命令将影响第二个替换命令的结果。

和 -e 等价的命令是 –expression：

sed --expression='s/test/check/' --expression='/love/d' file

### 从文件读入：r命令

file里的内容被读进来，显示在与test匹配的行后面，如果匹配多行，则file的内容将显示在所有匹配行的下面：

sed '/test/r file' filename

### 写入文件：w命令

在example中所有包含test的行都被写入file里：

sed -n '/test/w file' example

### 追加（行下）：a

将 this is a test line 追加到 以test 开头的行后面：

sed '/^test/a\this is a test line' file

在 test.conf 文件第2行之后插入 this is a test line：

sed -i '2a\this is a test line' test.conf

### 插入（行上）：i

将 this is a test line 追加到以test开头的行前面：

sed '/^test/i\this is a test line' file

在test.conf文件第5行之前插入this is a test line：

sed -i '5i\this is a test line' test.conf

### 替换指定行：c

把root开头的行替换新内容：

[root@node6 ~]# sed '/^root/c this is new line!' passwd   
this is new line!  
bin:x:1:1:bin:/bin:/sbin/nologin

如果是指定范围替换，需要注意，sed不是每行进行替换，而是把整个范围作为整体替换：

[root@node6 ~]# nl passwd | sed '1,5c\ this is dangerous!'  
 this is dangerous!  
 6 sync:x:5:0:sync:/sbin:/bin/sync  
 7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown

如果想实现对第一行到第五行统一替换为相同内容，可以用下面的命令实现：

[root@node5 ~]# sed '1{:a;s/.\*/lutxixia/;n;6!ba}' passwd   
lutxixia  
lutxixia  
lutxixia  
lutxixia  
lutxixia  
sync:x:5:0:sync:/sbin:/bin/sync  
  
其中：  
:a 是设置一个循环标签  
s/.\*/lutixia/ 是用lutixia字符替换匹配到的每行内容  
n 是读取下一行  
6! 是读到第六行退出循环，终止操作,如果没有，则继续循环。  
ba 是如果没有到第六行就跳转到a继续循环

### 下一个：n命令

如果test被匹配，则移动到匹配行的下一行，替换这一行的aa，变为bb，并打印该行，然后继续：

sed '/test/{ n; s/aa/bb/; }' file

### 变形：y命令

把1~10行内所有abcde转变为大写，注意，正则表达式元字符不能使用这个命令：

sed '1,10y/abcde/ABCDE/' file

### 退出：q命令

打印完前10行后，退出sed:

sed '10q' file

直到找到第一个匹配项，退出sed：

[root@node4 ~]# sed '/nginx/q' nginx.yml   
---  
- hosts: nginx

### 保持和获取：h命令和G命令

在sed处理文件的时候，每一行都被保存在一个叫模式空间的临时缓冲区中，除非行被删除或者输出被取消，否则所有被处理的行都将 打印在屏幕上。接着模式空间被清空，并存入新的一行等待处理。

sed -e '/test/h' -e '$G' file

在这个例子里，匹配test的行被找到后，将存入模式空间，h命令将其复制并存入一个称为保持缓存区的特殊缓冲区内。第二条语句的意思是，当到达最后一行后，G命令取出保持缓冲区的行，然后把它放回模式空间中，且追加到现在已经存在于模式空间中的行的末尾。在这个例子中就是追加到最后一行。简单来说，任何包含test的行都被复制并追加到该文件的末尾。

### 保持和互换：h命令和x命令

互换模式空间和保持缓冲区的内容。也就是把包含test与check的行互换：

sed -e '/test/h' -e '/check/x' file

### 脚本scriptfile

sed脚本是一个sed的命令清单，启动Sed时以-f选项引导脚本文件名。Sed对于脚本中输入的命令非常挑剔，在命令的末尾不能有任何空白或文本，如果在一行中有多个命令，要用分号分隔。以#开头的行为注释行，且不能跨行。

sed [options] -f scriptfile file(s)

### 打印奇数行或偶数行

方法1：

sed -n 'p;n' test.txt #奇数行  
sed -n 'n;p' test.txt #偶数行

方法2：

sed -n '1~2p' test.txt #奇数行  
sed -n '2~2p' test.txt #偶数行

### 打印匹配字符串的下一行

grep -A 1 SCC URFILE  
sed -n '/SCC/{n;p}' URFILE  
awk '/SCC/{getline; print}' URFILE