Linux sed 命令：功能强大的流式文本编辑器

> sed: 功能强大的流式文本编辑器 – 最专业的 Linux 命令大全，内容包含 Linux 命令手册、详解、学习，值得收藏的 Linux 命令速查手册。

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\
i\
c\
d
D
s
h
```

```
H
g
G
l
n
N
p
P
q
b lable
r file
t label
T label
w file
W file
!
=
```

sed 替换标记

```
g
p
w
x
y
\1
&
```

sed 元字符集

```
^
$
.
*
[]
[^]
\(..\)
&
\<
\>
x\{m\}
x\{m,\}
x\{m,n\}
```

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 (d 写外边效果一样)：

```
sed '/^test/d' file
sed '/^test/'d file
```

已匹配字符串标记 &

正则表达式 \w+ 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己加 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 ~]
```

```
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 ~]
ROOT:x:0:0:root:/root:/bin/bash
BIN:x:1:1:bin:/bin:/sbin/nologin
```

组合多个表达式

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
```

删除文本中的多个行：

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

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 \ 命令

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 \ 命令

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 \ 命令

把 root 开头的行替换新内容：

```
[root@node6 ~]
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 ~]
    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 ~]
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 ~]
---
- 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
sed -n 'n;p' test.txt
```

方法 2：

```
sed -n '1~2p' test.txt
sed -n '2~2p' test.txt
```

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
```