Linux sed命令完全攻略（超级详细）

< 上一节 下一节 >

我们知道，Vim 采用的是交互式文本编辑模式，你可以用键盘命令来交互性地插入、删除或替换数据中的文本。但本节要讲的 sed 命令不同，它采用的是流编辑模式，最明显的特点是，在 sed 处理数据之前，需要预先提供一组规则，sed 会按照此规则来编辑数据。

sed 会根据脚本命令来处理文本文件中的数据，这些命令要么从命令行中输入，要么存储在一个文本文件中，此命令执行数据的顺序如下：

1. 每次仅读取一行内容；
2. 根据提供的规则命令匹配并修改数据。注意，sed 默认不会直接修改源文件数据，而是会将数据复制到缓冲区中，修改也仅限于缓冲区中的数据；
3. 将执行结果输出。

当一行数据匹配完成后，它会继续读取下一行数据，并重复这个过程，直到将文件中所有数据处理完毕。

sed 命令的基本格式如下：

```
[root@localhost ~]# sed [选项] [脚本命令] 文件名
```

该命令常用的选项及含义，如表 1 所示。

表 1 sed 命令常用选项及含义

选项	含义
-e 脚本命令	该选项会将其后跟的脚本命令添加到已有的命令中。
-f 脚本命令文件	该选项会将其后文件中的脚本命令添加到已有的命令中。
-n	默认情况下，sed 会在所有的脚本指定执行完毕后，会自动输出处理后的内容，而该选项会屏蔽启动输出，需使用 print 命令来完成输出。
-i	此选项会直接修改源文件，要慎用。

成功使用 sed 命令的关键在于掌握各式各样的脚本命令及格式，它能帮你定制编辑文件的规则。

sed脚本命令

sed s 替换脚本命令

此命令的基本格式为：

```
[address]s/pattern/replacement/flags
```

其中，address 表示指定要操作的具体行，pattern 指的是需要替换的内容，replacement 指的是要替换的新内容。

> 关于指定具体操作行（address）的用法，这里先不做解释，文章后续会对其做详细介绍。

此命令中常用的 flags 标记如表 2 所示。

表 2 sed s命令flags标记及功能

flags 标记	功能

n	1~512 之间的数字，表示指定要替换的字符串出现第几次时才进行替换，例如，一行中有 3 个 A，但用户只想替换第二个 A，这是就用到这个标记；
g	对数据中所有匹配到的内容进行替换，如果没有 g，则只会在第一次匹配成功时做替换操作。例如，一行数据中有 3 个 A，则只会替换第一个 A；
p	会打印与替换命令中指定的模式匹配的行。此标记通常与 -n 选项一起使用。
w file	将缓冲区中的内容写到指定的 file 文件中；
&	用正则表达式匹配的内容进行替换；
\n	匹配第 n 个子串，该子串之前在 pattern 中用 \(\) 指定。
\	转义（转义替换部分包含：&、\ 等）。

比如，可以指定 sed 用新文本替换第几处模式匹配的地方：

```
[root@localhost ~]# sed 's/test/trial/2' data4.txt
This is a test of the trial script.
This is the second test of the trial script.
```

可以看到，使用数字 2 作为标记的结果就是，sed 编辑器只替换每行中第 2 次出现的匹配模式。

如果要用新文件替换所有匹配的字符串，可以使用 g 标记：

```
[root@localhost ~]# sed 's/test/trial/g' data4.txt
This is a trial of the trial script.
This is the second trial of the trial script.
```

我们知道，-n 选项会禁止 sed 输出，但 p 标记会输出修改过的行，将二者匹配使用的效果就是只输出被替换命令修改过的行，例如：

```
[root@localhost ~]# cat data5.txt
This is a test line.
This is a different line.
[root@localhost ~]# sed -n 's/test/trial/p' data5.txt
This is a trial line.
```

w 标记会将匹配后的结果保存到指定文件中，比如：

```
[root@localhost ~]# sed 's/test/trial/w test.txt' data5.txt
This is a trial line.
This is a different line.
[root@localhost ~]#cat test.txt
This is a trial line.
```

在使用 s 脚本命令时，替换类似文件路径的字符串会比较麻烦，需要将路径中的正斜线进行转义，例如：

```
[root@localhost ~]# sed 's/\/bin\/bash/\/bin\/csh/' /etc/passwd
```

sed d 替换脚本命令

此命令的基本格式为：

```
[address]d
```

如果需要删除文本中的特定行，可以用 d 脚本命令，它会删除指定行中的所有内容。但使用该命令时要特别小心，如果你忘记指定具体行的话，文件中的所有内容都会被删除，举个例子：

```
[root@localhost ~]# cat data1.txt
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
```

```
[root@localhost ~]# sed 'd' data1.txt
#什么也不输出，证明成了空文件
```

当和指定地址一起使用时，删除命令显然能发挥出大的功用。可以从数据流中删除特定的文本行。

address 的具体写法后续会做详细介绍，这里只给大家举几个简单的例子：

- 通过行号指定，比如删除 data6.txt 文件内容中的第 3 行：

```
[root@localhost ~]# cat data6.txt
This is line number 1.
This is line number 2.
This is line number 3.
This is line number 4.
[root@localhost ~]# sed '3d' data6.txt
This is line number 1.
This is line number 2.
This is line number 4.
```

- 或者通过特定行区间指定，比如删除 data6.txt 文件内容中的第 2、3行：

```
[root@localhost ~]# sed '2,3d' data6.txt
This is line number 1.
This is line number 4.
```

- 也可以使用两个文本模式来删除某个区间内的行，但这么做时要小心，你指定的第一个模式会“打开”行删除功能，第二个模式会“关闭”行删除功能，因此，sed 会删除两个指定行之间的所有行（包括指定的行），例如：

```
[root@localhost ~]#sed '/1/,/3/d' data6.txt
#删除第 1~3 行的文本数据
This is line number 4.
```

- 或者通过特殊的文件结尾字符，比如删除 data6.txt 文件内容中第 3 行开始的所有的内容：

```
[root@localhost ~]# sed '3,$d' data6.txt
This is line number 1.
This is line number 2.
```

在此强调，在默认情况下 sed 并不会修改原始文件，这里被删除的行只是从 sed 的输出中消失了，原始文件没做任何改变。

sed a 和 i 脚本命令

a 命令表示在指定行的后面附加一行，i 命令表示在指定行的前面插入一行，这里之所以要同时介绍这 2 个脚本命令，因为它们的基本格式完全相同，如下所示：

```
[address]a（或 i）\新文本内容
```

下面分别就这 2 个命令，给读者举几个例子。比如说，将一个新行插入到数据流第三行前，执行命令如下：

```
[root@localhost ~]# sed '3i\
> This is an inserted line.' data6.txt
This is line number 1.
This is line number 2.
This is an inserted line.
This is line number 3.
This is line number 4.
```

再比如说，将一个新行附加到数据流中第三行后，执行命令如下：

```
[root@localhost ~]# sed '3a\
> This is an appended line.' data6.txt
This is line number 1.
This is line number 2.
This is line number 3.
This is an appended line.
This is line number 4.
```

↑

如果你想将一个多行数据添加到数据流中，只需对要插入或附加的文本中的每一行末尾（除最后一行）添加反斜线即可，例如：

```
[root@localhost ~]# sed '1i\
> This is one line of new text.\
> This is another line of new text.' data6.txt
This is one line of new text.
This is another line of new text.
This is line number 1.
This is line number 2.
This is line number 3.
This is line number 4.
```

可以看到，指定的两行都会被添加到数据流中。

sed c 替换脚本命令

c 命令表示将指定行中的所有内容，替换成该选项后面的字符串。该命令的基本格式为：

```
[address]c\用于替换的新文本
```

举个例子：

```
[root@localhost ~]# sed '3c\
> This is a changed line of text.' data6.txt
This is line number 1.
This is line number 2.
This is a changed line of text.
This is line number 4.
在这个例子中，sed 编辑器会修改第三行中的文本，其实，下面的写法也可以实现此目的：
[root@localhost ~]# sed '/number 3/c\
> This is a changed line of text.' data6.txt
This is line number 1.
This is line number 2.
This is a changed line of text.
This is line number 4.
```

sed y 转换脚本命令

y 转换命令是唯一可以处理单个字符的 sed 脚本命令，其基本格式如下：

```
[address]y/inchars/outchars/
```

转换命令会对 inchars 和 outchars 值进行一对一的映射，即 inchars 中的第一个字符会被转换为 outchars 中的第一个字符，第二个字符会被转换成 outchars 中的第二个字符...这个映射过程会一直持续到处理完指定字符。如果 inchars 和 outchars 的长度不同，则 sed 会产生一条错误消息。

举个简单例子：

```
[root@localhost ~]# sed 'y/123/789/' data8.txt
This is line number 7.
This is line number 8.
This is line number 9.
This is line number 4.
This is line number 7 again.
This is yet another line.
This is the last line in the file.
```

可以看到，inchars 模式中指定字符的每个实例都会被替换成 outchars 模式中相同位置的那个字符。

转换命令是一个全局命令，也就是说，它会文本行中找到的所有指定字符自动进行转换，而不会考虑它们出现的位置，再打个比方：

```
[root@localhost ~]# echo "This 1 is a test of 1 try." | sed 'y/123/456/'
This 4 is a test of 4 try.
```

↑

sed 转换了在文本行中匹配到的字符 1 的两个实例，我们无法限定只转换在特定地方出现的字符。

sed p 打印脚本命令

p 命令表示搜索符号条件的行，并输出该行的内容，此命令的基本格式为：

```
[address]p
```

p 命令常见的用法是打印包含匹配文本模式的行，例如：

```
[root@localhost ~]# cat data6.txt
This is line number 1.
This is line number 2.
This is line number 3.
This is line number 4.
[root@localhost ~]# sed -n '/number 3/p' data6.txt
This is line number 3.
```

可以看到，用 -n 选项和 p 命令配合使用，我们可以禁止输出其他行，只打印包含匹配文本模式的行。

如果需要在修改之前查看行，也可以使用打印命令，比如与替换或修改命令一起使用。可以创建一个脚本在修改行之前显示该行，如下所示：

```
[root@localhost ~]# sed -n '/3/{
> p
> s/line/test/p
> }' data6.txt
This is line number 3.
This is test number 3.
```

sed 命令会查找包含数字 3 的行，然后执行两条命令。首先，脚本用 p 命令来打印出原始行；然后它用 s 命令替换文本，并用 p 标记打印出替换结果。输出同时显示了原来的行文本和新的行文本。

sed w 脚本命令

w 命令用来将文本中指定行的内容写入文件中，此命令的基本格式如下：

```
[address]w filename
```

这里的 filename 表示文件名，可以使用相对路径或绝对路径，但不管是哪种，运行 sed 命令的用户都必须有文件的写权限。

下面的例子是将数据流中的前两行打印到一个文本文件中：

```
[root@localhost ~]# sed '1,2w test.txt' data6.txt
This is line number 1.
This is line number 2.
This is line number 3.
This is line number 4.
[root@localhost ~]# cat test.txt
This is line number 1.
This is line number 2.
```

当然，如果不想让行直接输出，可以用 -n 选项，再举个例子：

```
[root@localhost ~]# cat data11.txt
Blum, R       Browncoat
McGuiness, A  Alliance
Bresnahan, C  Browncoat
Harken, C     Alliance
[root@localhost ~]# sed -n '/Browncoat/w Browncoats.txt' data11.txt
cat Browncoats.txt
Blum, R       Browncoat
Bresnahan, C  Browncoat
```

可以看到，通过使用 w 脚本命令，sed 可以实现将包含文本模式的数据行写入目标文件。

sed r 脚本命令

r 命令用于将一个独立文件的数据插入到当前数据流的指定位置，该命令的基本格式为：

```
[address]r filename
```

sed 命令会将 filename 文件中的内容插入到 address 指定行的后面，比如说：

```
[root@localhost ~]# cat data12.txt
This is an added line.
This is the second added line.
[root@localhost ~]# sed '3r data12.txt' data6.txt
This is line number 1.
This is line number 2.
This is line number 3.
This is an added line.
This is the second added line.
This is line number 4.
```

如果你想将指定文件中的数据插入到数据流的末尾，可以使用 $ 地址符，例如：

```
[root@localhost ~]# sed '$r data12.txt' data6.txt
This is line number 1.
This is line number 2.
This is line number 3.
This is line number 4.
This is an added line.
This is the second added line.
```

sed q 退出脚本命令

q 命令的作用是使 sed 命令在第一次匹配任务结束后，退出 sed 程序，不再进行对后续数据的处理。

比如：

```
[root@localhost ~]# sed '2q' test.txt
This is line number 1.
This is line number 2.
```

可以看到，sed 命令在打印输出第 2 行之后，就停止了，是 q 命令造成的，再比如：

```
[root@localhost ~]# sed '/number 1/{ s/number 1/number 0/;q; }' test.txt
This is line number 0.
```

使用 q 命令之后，sed 命令会在匹配到 number 1 时，将其替换成 number 0，然后直接退出。

sed 脚本命令的寻址方式

前面在介绍各个脚本命令时，我们一直忽略了对 address 部分的介绍。对各个脚本命令来说，address 用来表明该脚本命令作用到文本中的具体行。

默认情况下，sed 命令会作用于文本数据的所有行。如果只想将命令作用于特定行或某些行，则必须写明 address 部分，表示的方法有以下 2 种：

1. 以数字形式指定行区间；
2. 用文本模式指定具体行区间。

以上两种形式都可以使用如下这 2 种格式，分别是：

```
[address]脚本命令

或者

address {
    多个脚本命令
}
```

以上两种形式在前面例子中都有具体实例，因此这里不再做过多赘述。

以数字形式指定行区间

当使用数字方式的行寻址时，可以用行在文本流中的行位置来引用。sed 会将文本流中的第一行编号为 1，然后继续按顺序为接下来的行分配行号。

在脚本命令中，指定的地址可以是单个行号，或是用起始行号、逗号以及结尾行号指定的一定区间范围内的行。这里举一个 sed 命令作用到指定行号的例子：

```
[root@localhost ~]#sed '2s/dog/cat/' data1.txt
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy cat
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy dog
```

可以看到，sed 只修改地址指定的第二行的文本。下面的例子中使用了行地址区间：

```
[root@localhost ~]# sed '2,3s/dog/cat/' data1.txt
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy cat
The quick brown fox jumps over the lazy cat
The quick brown fox jumps over the lazy dog
```

在此基础上，如果想将命令作用到文本中从某行开始的所有行，可以用特殊地址——美元符（$）：

```
[root@localhost ~]# sed '2,$s/dog/cat/' data1.txt
The quick brown fox jumps over the lazy dog
The quick brown fox jumps over the lazy cat
The quick brown fox jumps over the lazy cat
The quick brown fox jumps over the lazy cat
```

用文本模式指定行区间

sed 允许指定文本模式来过滤出命令要作用的行，格式如下：

```
/pattern/command
```

注意，必须用正斜线将要指定的 pattern 封起来，sed 会将该命令作用到包含指定文本模式的行上。

举个例子，如果你想只修改用户 demo 的默认 shell，可以使用 sed 命令，执行命令如下：

```
[root@localhost ~]# grep demo /etc/passwd
demo:x:502:502::/home/Samantha:/bin/bash
[root@localhost ~]# sed '/demo/s/bash/csh/' /etc/passwd
root:x:0:0:root:/root:/bin/bash
...
demo:x:502:502::/home/demo:/bin/csh
...
```

虽然使用固定文本模式能帮你过滤出特定的值，就跟上面这个用户名的例子一样，但其作用难免有限，因此，sed 允许在文本模式使用正则表达式指明作用的具体行。正则表达式允许创建高级文本模式匹配表达式来匹配各种数据。这些表达式结合了一系列通配符、特殊字符以及固定文本字符来生成能够匹配几乎任何形式文本的简练模式。

关于正则表达式，本节不做过多介绍，有兴趣的读者可阅读《正则表达式入门教程》一文，这里仅给读者提供一个简单示例：

```
[root@localhost ~]# cat test.txt
<html>
<title>First Wed</title>
<body>
h1Helloh1
h2Helloh2
h3Helloh3
```

```
</body>
</html>
#使用正则表示式给所有第一个的h1、h2、h3添加<>，给第二个h1、h2、h3添加</>
[root@localhost ~]# cat sed.sh
/h[0-9]/{
   s//\<&\>/1
   s//\<\/&\>/2
}
[root@localhost ~]# sed -f sed.sh test.txt
<h1>Hello</h1>
<h2>Hello</h2>
<h3>Hello</h3>
```

收到篇幅的限制，本节仅介绍了 sed 命令每次只读取一行内容并处理，有关 sed 命令如何一次处理多行文本内容，放到下节继续讲解，读者可点击《Linux sed命令高级用法》继续学习。

< 上一节　　下一节 >

优秀文章

C++二进制文件读写（read和write）详解
串的定长顺序存储结构
Linux Shell trap命令捕获信号实例演示
Linux Shell移除（重置）信号捕获
Java注释：单行、多行和文档注释
Eclipse创建JSP项目（图解）
终止线程执行，千万别踩这个坑！
Redis INCR数值操作命令
AOP面向切面编程
Pandas是什么